Universidad de Valladolid

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

DPTO. DE TEORÍA DE LA SEÑAL Y COMUNICACIONES
E INGENIERÍA TELEMÁTICA

TESIS DOCTORAL:

Rendimiento de TCP y Cálculo de Rutas en Redes de Conmutación Óptica de Ráfagas

Presentada por Óscar González de Dios para optar al grado de doctor por la Universidad de Valladolid

Dirigida por:
Ignacio de Miguel Jiménez

RESUMEN

La tecnología de conmutación óptica de ráfagas (*Optical Burst Switching*, OBS) es una alternativa prometedora para la próxima generación de redes ópticas. Esta tesis estudia el comportamiento de flujos de datos que empleen el protocolo de transporte TCP (*Transmission Control Protocol*) sobre una red basada en la mencionada tecnología OBS, propone una técnica de encaminamiento adaptativa y multicamino para OBS, y diseña e implementa un elemento de cálculo de rutas basado en PCE (*Path Computation Element*) para redes de conmutación de ráfagas OBS con encaminamiento por longitud de onda, conocidas como WR-OBS (*Wavelength-Routed* OBS).

Uno de los aspectos clave a la hora de investigar una nueva tecnología de red es comprobar cómo se comporta el tráfico que circulará por la misma y cómo la red responde ante dicho tráfico. El tráfico que circula por las redes de comunicaciones está principalmente controlado por el protocolo de transporte TCP. En este sentido, con el fin de profundizar en su funcionamiento, se realiza en primer lugar una revisión detallada del protocolo de transporte TCP y se estudia su comportamiento en distintos tipos de redes. En concreto, las investigacines llevadas a cabo en esta tesis se centran en el comportamiento de dicho protocolo en una red que emplea la tecnología OBS. Muchos parámetros del protocolo TCP son configurables en las distintas implementaciones del mismo, que varían según el sistema operativo, y su elección puede llegar a tener mayor o menor impacto en el rendimiento de una transferencia de datos con TCP según la tecnología de red que se emplee. Así, se estudia el impacto de un aspecto concreto de TCP, el asentimiento retardado, que se evalúa mediante simulación. Por otro lado, se realiza una importante aportación al estado del arte estudiando TCP, no de manera completamente aislada, sino con múltiples fuentes de tráfico. Mediante simulación, y posteriormente de manera analítica, se estudia el comportamiento de un flujo TCP que se ensambla en un ensamblador OBS junto con tráfico perteneciente a otras conexiones, que denominaremos tráfico de fondo, y se analiza qué impacto tiene en el rendimiento de TCP en redes OBS.

Con el objetivo de obtener una relación sencilla entre los principales parámetros de diseño de OBS y TCP, se propone un modelo teórico que captura el funcionamiento de TCP sobre OBS con múltiples flujos. Para ello, se estudian con detalle las técnicas de modelado de TCP. El modelo propuesto se valida mediante simulación con una gran variedad de escenarios. Para el caso de múltiples flujos, un parámetro con gran influencia es el número de segmentos TCP por ráfaga, que solamente puede conocerse con certeza en escenarios muy acotados. La tesis propone y valida mediante simulación unas cotas superiores e inferiores del *throughput* de TCP sobre OBS con múltiples flujos. La validación se ha llevado a cabo mediante simulación tanto en OPNET Modeler como en OMNeT++.

La tesis también estudia el efecto de la sincronización de flujos TCP en una red OBS. La sincronización de flujos TCP aparece cuando varias conexiones TCP comparten un enlace y los mecanismos de control de congestión de los clientes y servidores TCP involucrados en la transmisión reaccionan al mismo tiempo. Este hecho lleva a un uso ineficiente del ancho de banda disponible, por lo que si la sincronización es excesiva, el ancho de banda necesario para ofrecer el mismo rendimiento a las conexiones TCP es mayor que en un caso sin sincronización. La tesis cuantifica el efecto de sincronización en los mecanismos clásicos de ensamblado de OBS, aplicando distintas métricas. Se evalúan mediante simulación distintos escenarios, desde un caso peor de sincronización total, hasta un escenario pragmático con variedad de RTTs (*Round Trip Times*), en los que se calcula cuál es la capacidad que se ha de reservar en un enlace OBS para cursar un tráfico determinado. Con el objetivo de aumentar la eficiencia del uso del ancho de banda

disponible, se propone un nuevo esquema de ensamblado que de manera estática o dinámica asigna flujos a un conjunto de colas para controlar el proceso de agregación de flujos. Se evalúa la bondad de este esquema, mostrando una mejora significativa en la utilización del ancho de banda del enlace óptico.

También se propone una técnica de encaminamiento multicamino, adaptativa y con histéresis para OBS con reserva sin confirmación. El objetivo de la técnica propuesta es reducir la probabilidad de bloqueo de ráfagas repartiendo la carga de tráfico, balanceándola de tal forma que se reduzca el tráfico en los enlaces más cargados y se utilicen enlaces poco cargados con muchos recursos vacantes. Para realizar el reparto, se emplea únicamente información obtenible localmente en el nodo, sin necesidad de enviar información de estado adicional en los protocolos de encaminamiento. Mediante simulación con ns-2 se evalúan las prestaciones de la técnica propuesta en cuanto a probabilidad de bloqueo y retardo.

En último lugar, la tesis propone una arquitectura de WR-OBS con elementos definidos por los estándares actuales en el IETF (*Internet Engineering Task Force*), destacando el elemento de cálculo de rutas, denominado PCE. Se diseña e implementa un prototipo de PCE para redes de conmutación de ráfagas OBS con encaminamiento por longitud de onda (WR-OBS). La tesis realiza una importante contribución mediante la propuesta, implementación y validación de las extensiones del protocolo PCEP (protocolo de comunicación con el PCE) necesarias para las redes WR-OBS. Se lleva a cabo una implementación de un prototipo del PCE diseñado con las extensiones propuestas, que responde en tiempo real a peticiones de rutas para ráfagas ópticas. Se implementa un emulador de WR-OBS con el que se valida experimentalmente el prototipo de PCE. Se evalúan las prestaciones del PCE en cuanto a rapidez de cálculo y se evalúa el rendimiento de la arquitectura WR-OBS con PCE en cuanto a retardo de borde, número medio de *lightpaths* establecidos y probabilidad de bloqueo.

ABSTRACT

Optical Burst Switching (OBS) is a promising alternative for next-generation optical networks. This PhD thesis studies the behavior of data flows that use TCP (Transmission Control Protocol) as transport protocol over a network based on the mentioned OBS technology, proposes an adaptive multipath routing technique for OBS, and designs and implements a Path Computation Element (PCE) for Wavelength-Routed Optical Burst-Switched Networks (WR-OBS).

One of the key aspects in order to research in a new networking technology is to understand how the traffic behaves in the network and how the network responds to such traffic. The traffic that runs through modern communication networks is mainly controlled by TCP (Transmission Control Protocol), the most important transport protocol. This thesis performs a detailed revision of the TCP protocol and its behavior is analyzed in several types of networks, finally focusing on studying the performance in OBS.

There are many TCP parameters that can be tuned in TCP implementations and may have an impact on the throughput depending on the networking technology used. In particular, the impact of a specific TCP feature, the delayed acknowledgement, is studied and evaluated by means of simulation. Also, a significant contribution to the state of the art is made by studying TCP over OBS, not in an isolated manner, but with multiple traffic sources. By means of simulation, and also analytically, it is studied the behavior of TCP flows assembled with traffic belonging to other connections, so called background traffic, and it is analyzed its impact on OBS Networks.

A theoretical model that captures the behavior of TCP over OBS with multiple flows is proposed with the aim of obtaining a simple expression that relates the most important parameters of OBS and TCP. To that end, different TCP modeling techniques are studied in detail. The proposed model is validated through simulation in a variety of scenarios. In the case where multiple TCP flows are assembled, the number of TCP segments per burst has a great influence, and can only be exactly known in few cases. The thesis proposes and validates by means of simulation upper and lower bounds of the throughput of TCP over OBS with multiple flows. The validation is performed by simulation in both OMNeT++ and OPNET Modeler.

The thesis also studies the effect of synchronization of TCP flows in an OBS network. The synchronization of TCP flows appears when several TCP connections share a link and the congestion control mechanisms of TCP clients and servers involved in the transmissions react at the same time. This fact leads to an inefficient use of the available bandwidth. Thus, if the synchronization is excessive, the bandwidth needed for several TCP connections is higher than in a case without synchronization, maintaining the performance. The thesis quantifies the effect of synchronization in the classical burst assembly techniques in OBS, applying different metrics. By means of simulation, several scenarios are evaluated, from a worst case of total synchronization to a pragmatic scenario with a variety of RTTs. In such scenarios, based on the simulation results, the total capacity that needs to be guaranteed in an OBS link to carry certain traffic is determined. With the goal of increasing the efficiency in the use of available bandwidth, it is proposed a new assembly scheme in which, in a static or dynamic way, the flows are assigned to a set of queues to control the aggregation process. The performance of the scheme is evaluated, showing a significant improvement in optical bandwidth utilization.

A multipath, adaptive routing technique with hysteresis for OBS is proposed. The goal of the proposed technique is to reduce burst blocking probability by traffic load balancing,

in such a way that the traffic in the most used link is reduced, while the very lightly loaded links with free resources are used. The traffic distribution is based on information available locally at the node where the routing decisions are made, without the need of sending additional state information is the routing protocols. By means of simulations with ns-2, the performance of the proposed technique is evaluated in terms of delay and blocking probability.

Finally, the thesis proposes a WR-OBS architecture based on elements defined by current IETF standards, like the Path Computation Element (PCE). Moreover, a prototype of PCE of WR-OBS networks is designed and implemented. The thesis makes a significant contribution proposing, implementing and validating PCPE protocol extensions for WR-OBS. The designed prototype implements the extensions and is able to respond in real time path computation requests for optical bursts. A WR-OBS network emulator is implemented and used to validate experimentally the designed PCE. The performance of the PCE in terms of speed and the performance of the PCE based WR-OBS architecture in terms of edge delay, mean number of established lightpaths and blocking probability are experimentally evaluated.

AGRADECIMIENTOS

El trabajo realizado en esta tesis comenzó en 2003 cuando estaba trabajando en Telefónica I+D en el Parque Tecnológico de Boecillo y acababa de cambiar de división. Notaba que el nuevo puesto trabajo no me llenaba, era demasiado rutinario y no sacaba todo el partido a mis neuronas, por lo que necesitaba un buen reto. Así que decidí volver por la escuela y visitar a los distintos profesores para conocer qué temas se investigaban y ver si había alguno en concreto que me atrajera lo suficiente como para embarcarme en una tesis. En una de estas visitas, tuve la suerte de contactar con Nacho de Miguel, mi director de tesis, que me enganchó al tema de las redes ópticas, "al OBS", "al TCP" (y un gran sinfín de siglas más) y al mundillo de la investigación y las publicaciones. A Nacho debo agradecerle ser mi mentor en el mundo de la investigación y posibilitar que gracias a conocer y entrar en este mundo, haya podido orientar mi carrera profesional hacia la investigación en redes dentro de Telefónica y de paso, conocer medio mundo :-).

Así que mientras trabajaba en el Parque Tecnológico de Boecillo, por las noches me dedicaba a los trabajos que me iba indicando Nacho, adentrándome poco a poco en el mundo de la investigación. Pronto obtuve la suficiencia investigadora, muy ilusionado, y sucedió un hecho que cambió mi carrera profesional. Se lanzó en Telefónica el programa "JAP", Jóvenes de Alto Potencial, en el cual, tras interminables pruebas y entrevistas logré participar. En él, en un curso en 2005, coincidí con Jesús Felipe Lobo. En una charla de ascensor, hablando de lo que hacíamos le mencioné que, aparte de lo que estaba haciendo en ese momento en TID (pruebas de sistema, poco apasionantes), me dedicaba a investigar en OBS. Y se le encendió la bombilla, me dijo… pues nosotros estamos en un proyecto en el que se trabaja en OBS y redes ópticas, el proyecto NOBEL, pero es en Madrid… ¿te quieres venir? Así que aprovechando el programa JAP, empecé una estancia en Madrid en el grupo de Jesús. Mi segundo agradecimiento va para Jesús Lobo, al que agradezco eternamente la oportunidad que me dio y supuso un giro en mi carrera profesional. En los primeros meses estuve yendo y viniendo muchísimo a Valladolid, donde Estela acababa de terminar la carrera y se preparaba las oposiciones. Mi gran agradecimiento especial es a Estela, ahora mi mujer tras casarnos en Las Vegas, que ha sufrido, aguantado y apoyado todas mis horas dedicadas a realizar esta tesis. No hay que olvidar que esta tesis se ha realizado con gran esfuerzo cada vez que salía del trabajo, por lo que según la época hubo períodos de gran actividad y a veces varios meses sin avance alguno.

Mi otro gran agradecimiento especial es a mis padres, que me han apoyado siempre en el estudio, lo han sacrificado todo por mí y mis hermanos para que tengamos la mejor educación, y han presionado incansablemente para que no me rinda en la tesis (da igual de lo que habláramos, que al final siempre preguntaban por la tesis), que siga para adelante y la termine.

Aunque me fui con ganas a Madrid, guardo un grato recuerdo del gran grupo de gente de Boecillo, que me ayudaron y enseñaron un montón (agradecimientos al *pavo team*, en especial a Juan Calero y Carlos Rodríguez que me enseñaron cómo debe ser un buen jefe).

Cuando me uní a un grupo de Jesús Lobo en Madrid conocí a un montón de gente de TID que me han enseñado muchísimo, de las cuales no he parado de aprender en el mundo de las redes. Espero no olvidarme de nadie de TID… En primer lugar una gran persona, Juan Fernández Palacios, actualmente el líder de la Iniciativa *Core Network Evolution* en Telefónica I+D en la que trabajo actualmente. También a Javier Jiménez Chico, actualmente en China; Jorge Andrés Colas, mi primer compañero de mesa en Madrid, que me enseñó los secretos del *routing*, y a quien lamentablemente un cáncer se llevó sin compasión; Manuel Villén Altamirano, antiguo gerente; Carlos García Argos, que se lleva

un agradecimiento especial por haberme dado el honor de dirigir su proyecto fin de carrera, y que ahora está en el CERN a por su tesis; Victor López, a quien conocí cuando él era becario, y ahora es todo un señor doctor, de vuelta con nosotros en I+D; Marian Callejo, por tierras mexicanas; Francisco Javier Salguero, la cara que veo enfrente todos los días en Don Ramón de la Cruz, Gerardo García de Blas, amo del tráfico; Antonio Elizondo; maestro en la gestión de proyectos europeos y ahora estratega; Juan Rodríguez, el rey del laboratorio; Luis Peréz, felizmente recuperado; Maria Luisa Garcia, ya corporativa… Con todos ellos compartimos buenos ratos en Emilio Vargas en el sitio de la antigua biblioteca de TID.

Mi primer gran recuerdo de la tesis fue escribir mi primer artículo y asistir a mi primer congreso donde presenté en público un artículo (una investigación mía, que ilusión) por primera vez, mi bautizo en el mundo de los *papers*. Fue el NOC 2005, al que asistí con mi compañero de TID Antonio Elizondo y con Ramón Durán y Nacho de Miguel de la Universidad de Valladolid. Justo antes de mi presentación… se oyó un gran boom. Decían que había explotado una tubería de gas. Aunque me dijeran que el mundo había desaparecido, a mi me daba igual, yo iba a presentar mi artículo. Antonio no estaba, yo ya pensaba, qué tío… le veo a él presentando, y cuando me toca a mí se queda dormido… Pasó el artículo y los nervios se fueron… y alejé mi mente del TCP y el OBS para volver a la realidad y empezar a enterarme de lo que pasaba. Habían puesto una tele… resulta que había habido un atentado en Londres… en el metro… en un autobús… Era el 7J… Entonces empecé a preocuparme… nada de cobertura… no podía localizar a Antonio… Nos dejaron entonces usar la línea fija, educadamente, por turnos… Llamé a casa para decir que estaba bien, al trabajo, donde estaban preocupadísimos… Antonio estuvo una hora en el metro atrapado cerca de donde la explosión. El autobús que yo había cogido los días anteriores había explotado. Ese día, como presentaba, cogí el metro para llegar antes… Ya me sorprendió que en una dirección el metro no iba… Y llamé a Estela que, como estaba en USA, se acababa de levantar… Dije a Estela… estoy bien… y ella... ya… ¿¿y?? :-)

Por esto, tengo un gran aprecio al NOC, donde he aprendido mucho y pasado grandes ratos. Y los proyectos europeos, NOBEL, NOBEL II, ePhoton ONe, STRONGEST, AGAVE, BONE, ONE y pronto… IDEALIST. Ahí he conocido a muchísima gente de los cuales he aprendido mucho y he podido hablar de todas las cosas de la tesis.

Un agradecimiento especial al Grupo de Comunicaciones Ópticas de la Universidad de Valladolid, con Evaristo incansable en insistir en que no cese en el intento de la tesis, Ramón Durán, Noemi Merayo, Natalia Fernández, Tamara Jiménez. Con ellos he compartido charlas, técnicas y menos técnicas y asistido a conferencias donde hemos aprendido y además nos lo hemos pasado fenomenal.

Mis agradecimientos a Kyriakos Vlachos y Kostas Ramantas, de la Universidad de Patras, con los que he colaborado en temas de TCP sobre OBS. Mis agradecimientos también a Carla Raffaeli y Ana Guidotti, que pasó 3 meses en I+D, de la Universidad de Bolonia.

Mis agradecimientos a todo el grupo del CTTC, de los cuales no paro de aprender nunca, en especial a Ramón Casellas, al que admiro profundamente, gran sabio del GMPLS, Raul Muñoz y Ricardo Martínez, compañeros de batallas en muchos proyectos europeos.

Mis agradecimientos a la gente de la UPC con la que he tenido la oportunidad de colaborar y aprender, en especial a Davide Careglio, con quien colaboré en los temas de routing en OBS, Salvatore Spadaro, Xavi Masip, gran elaborador de propuestas y siempre activo, buscando nuevos retos, Marcelo Yannuzzi, eternas discusiones en el ONE, Luis Velasco, maestro de la optimización y Josep Solé Pareta, también gran animador a que finalice mi tesis.

Un gran agradecimiento también a Javier Aracil, de la UAM, un genio, con quien he tenido el honor de colaborar en una gran variedad de temas.

Hace unos años, en 2010, en un proyecto de colaboración con Nokia Siemens Networks, surgió la oportunidad de empezar a colaborar en el IETF. Agradezco eternamente esa oportunidad, que me permitió expandir el ámbito de mi trabajo y mis investigaciones, haciendo que las ideas, no sólo se escriban en un papel o presenten en un congreso, sino que se conviertan en un estándar de referencia mundial. Mi gran agradecimiento a toda la gente que he conocido en el IETF especialmente a Cyril Margaria y Fatai Zhang.

Esta tesis se ha realizado en el marco de diversos proyectos de investigación financiados por la Comisión Europea, tales como ICT-BONE y FP7-STRONGEST (Ref 247674), y por el Ministerio de Ciencia e Innovación, tales como INSTINCT y la red temática FIERRO (Refs., TEC2010-21178-C02-02 y TEC2010-12250-E, respectivamente). Mis agradecimientos a la Comisión Europea y al Ministerio de Ciencia e Innovación.

Asimismo, quiero dedicar un gran agradecimiento a Telefónica I+D, que ha depositado su confianza en mí durante más de diez años y ha ayudado a la realización de esta tesis.

También quiero agradecer a todas las personas que siempre que me han visto me han insistido en que termine la tesis.

Finalmente, un agradecimiento a la Universidad de Valladolid y los profesores que me han transmitido sus conocimientos para llegar hasta aquí.

Ha llevado mucho tiempo y trabajo… pero aquí está la tesis. ¡GRACIAS A TODOS!

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1

Introducción

El volumen de tráfico que circula en las redes de comunicaciones ha crecido de manera continua en la última década, a un ritmo en torno al 50% anual [1]. A medida que el número de usuarios de Internet ha ido aumentando, los anchos de banda no han parado de subir y las aplicaciones han demandado cada vez más un mayor ancho de banda. Este crecimiento de tráfico está aún lejos de su fin y hay expectativas de que este aumento continuará, motivadas principalmente por la irrupción del tráfico multimedia proveniente de nuevos dispositivos móviles avanzados, como *smartphones*, *tablets*, etc, que se espera que generen, ya solo en 2015, unos 6 exabytes de datos adicionales al mes [2]. A este tráfico hay que sumar todo el tráfico de Internet de los accesos fijos tradicionales, que demandarán vídeos de mayor definición, transferencias de fotos a alta calidad, videoconferencias, etc, por los que, según Cisco, en 2013, el equivalente a diez mil millones de DVDs circulará en Internet cada mes [3]. Incluso se espera que nuevos servicios basados en la red, como el *cloud computing* o servicios de almacenamiento masivo añadan un tráfico masivo y muy dinámico en las redes. Asimismo, se esperan despliegues generalizados de fibra hasta el hogar, que proporcionarán a los hogares un ancho de banda de acceso muy superior a las conexiones típicas actuales, lo que supondrá un aumento en la presión de tráfico en las redes metropolitanas (MANs, *Metropolitan Area Networks*), que agregan el tráfico de los usuarios, y en las redes de núcleo o troncales (*backbone networks*), que transportan el tráfico hacia Internet. Estas redes, metropolitanas y troncales, han de actualizarse para poder soportar esta presión de tráfico de una manera económicamente y energéticamente viable.

Para absorber la gran cantidad de tráfico esperada se han planteado distintas alternativas. Por un lado, las redes de conmutación óptica de circuitos (OCS, *Optical Circuit Switching*), también conocidas como redes ópticas de conmutación de longitud de onda (WSON, *Wavelength Switching Optical Networks*) [4], son la alternativa a corto plazo. De hecho, los operadores ya han realizado despliegues de redes basadas en este paradigma, principalmente debido a la popularidad y madurez de su principal elemento, el ROADM (*Reconfigurable Add-Drop Multiplexer*) [5], un equipo capaz de conmutar longitudes de onda entre fibras e introducir y extraer canales ópticos. Sin embargo, esta tecnología tiene el inconveniente de su baja flexibilidad y su escasa granularidad, ya que los circuitos establecidos son de velocidades fijas muy altas, por ejemplo 10, 40 ó 100 Gbps en la actualidad. Estos canales ópticos ocupan una porción fija del espectro, independientemente del volumen de datos que estén transmitiendo en un momento puntual.

Como alternativa, se plantean tecnologías ópticas que sean capaces de aprovechar las longitudes de onda, bien sea mediante ráfagas, *slots* de tiempo fijo, o paquetes. Debido a la madurez de los láseres sintonizables rápidos, y a la disponibilidad de elementos de conmutación rápida, se contempla como alternativa a corto y medio plazo la conmutación óptica de ráfagas, conocida como OBS (*Optical Burst Switching*) [6]. A corto plazo se prevé la aparición de estas tecnologías en el entorno metropolitano, en topologías en anillo, donde los requisitos tecnológicos son menores, y el estado del arte de los componentes permite ya los primeros productos comerciales, mientras que a medio plazo, OBS se puede expandir hacia entornos mallados y de distancias mayores.

Como alternativa a largo plazo, se busca llegar a la conmutación óptica de paquetes (OPS, *Optical Packet Switching*) [7], que se diferencia de la ráfagas principalmente en el tamaño del elemento conmutado, que en el caso de los paquetes es menor que en el de las ráfagas, por lo que los requisitos tecnológicos son mayores. En muchos casos, los términos OBS y OPS son intercambiables y se habla de OBS/OPS (*Optical Burst/Packet Switching*). A pesar de su inmadurez, cada año aparecen nuevos prototipos y demostradores de OPS, acercándose poco a poco al ideal.

Dada la madurez de las tecnologías ópticas que hacen posible OBS, y de los potenciales beneficios de la misma, la tesis se centrará en completar el estado del arte de las redes de conmutación óptica de ráfagas.

En concreto, uno de los aspectos clave a la hora de investigar una nueva tecnología de red es comprobar cómo se comporta el tráfico que circulará por la misma y cómo la red responde ante dicho tráfico. Desde hace más de dos décadas, el tráfico que circula por las redes de comunicaciones está principalmente controlado por el protocolo de transporte TCP (*Transmission Control Protocol*) [8]. De hecho, más del 90% de los bytes que circulan en Internet son paquetes TCP. Su uso no ha dejado de crecer, y hoy se usa para transportar todo tipo de aplicaciones, desde correos hasta vídeos, que tradicionalmente empleaban otros protocolos como UDP, por lo que se espera que el tráfico siga basándose en TCP durante las próximas décadas. Por ello, la tesis va a estudiar el comportamiento de TCP en una red basada en OBS. La tesis, completando los estudios de la literatura, estudiará distintos aspectos de TCP, como el asentimiento retardado, y modelará el comportamiento cuando se ensamblan una gran cantidad de flujos. Se tendrá especial hincapié en un aspecto poco estudiado para OBS, pero conocido para las redes de paquetes tradicionales, como es la sincronización de flujos TCP.

Uno de los aspectos importantes en una red es el encaminamiento, que decide por dónde se envía el tráfico. En el caso de OBS, el encaminamiento decide por dónde se transmiten las ráfagas ópticas y si se realiza de manera adecuada ayuda a mejorar el rendimiento de red. En la literatura se han realizado numerosas propuestas para conseguir el conjunto de rutas óptimo dada una determinada demanda de tráfico. Sin embargo, la distribución y volumen del tráfico en la red cambia con el tiempo. Por ello, hacen falta técnicas que, en tiempo real, sean capaces de encontrar rutas que se adapten a la nueva situación en la red. Si bien se han propuesto algunas técnicas en la literatura, su aplicación práctica es compleja. En la tesis se evaluará una técnica que, empleando solamente información local, pueda repartir las ráfagas en múltiples caminos cuando la red esté saturada, para mejorar la probabilidad de bloqueo y con mecanismos de histéresis, para lograr un comportamiento estable.

Por otro lado, el organismo IETF (*Internet Engineering Task Force*) [9] ha definido un elemento funcional denominado PCE (*Path Computation Element*) [10], especializado en el cálculo de rutas, válido para todo tipo de tecnologías de red, con un protocolo de comunicación estándar, denominado PCEP (*Path Computation Element Protocol*) [11]. La tecnología OBS puede verse beneficiada por la inclusión de este elemento, donde se podrá incluir una gran variedad de algoritmos. Sin embargo, hasta la fecha, no se tiene conocimiento del uso de PCE en OBS. Esta tesis estudiará la aplicación del concepto de PCE en una red basada en conmutación de ráfagas ópticas. Asimismo se propondrán las extensiones protocolarias necesarias y realizará una implementación de un prototipo software del mismo.

1.1 Objetivos de la tesis

El objetivo de esta tesis es estudiar el comportamiento de flujos de datos que empleen el protocolo de transporte TCP sobre una red OBS, proponer una técnica de encaminamiento para OBS y diseñar e implementar un elemento de cálculo de rutas basado en PCE para redes de conmutación de ráfagas OBS con encaminamiento por longitud de onda.

En concreto, se propone:

- Realizar una revisión del estado del arte de la tecnología de redes basadas en conmutación óptica de ráfagas, centrándose en las distintas técnicas de señalización, el ensamblado y el encaminamiento.
- Realizar una revisión detallada del protocolo de transporte TCP y su comportamiento en distintos tipos de redes.
- Modelar el comportamiento de TCP sobre una red OBS en entornos de simulación (OPNET Modeler [12] y OMNeT++ [13])
- Estudiar el rendimiento de TCP en redes OBS teniendo en cuenta aspectos como la existencia de múltiples flujos, la coexistencia con tráfico adicional y el impacto de la técnica del asentimiento retardado en TCP aprovechando los modelos de simulación.
- Proponer un modelo teórico que capture el funcionamiento de TCP sobre OBS con múltiples flujos alimentando un único ensamblador y valide los modelos de simulación.
- Estudiar el efecto de sincronización de la ventana de múltiples flujos TCP cuando comparten un ensamblador y su impacto en el ancho de banda agregado.
- Proponer una técnica de ensamblado que reduzca la sincronización y las necesidades de ancho de banda.
- Proponer una técnica de encaminamiento multicamino para OBS con reserva sin confirmación y evaluar sus prestaciones.
- Diseñar un PCE (*Path Computation Element*) para redes de conmutación de ráfagas OBS con encaminamiento por longitud de onda y proponer las extensiones protocolares necesarias.
- Realizar una implementación de un prototipo del PCE diseñado con las extensiones propuestas y que permita, en tiempo real, responder a peticiones de rutas para ráfagas ópticas.

1.2 Organización de la tesis

Esta tesis se inicia presentando en el Capítulo 2 los fundamentos de la tecnología OBS. En primer lugar, se aborda una justificación de dicha tecnología y después se describe la arquitectura de una red basada en OBS con sus aspectos principales, como son el ensamblado de ráfagas, la reserva de camino, las técnicas de resolución de contienda, el encaminamiento y la calidad de servicio. Al final del capítulo, se realiza un recorrido histórico por los experimentos de OBS, terminando por los primeros productos comerciales que están apareciendo en la actualidad en el mercado.

Dado que la mayor parte del tráfico en las redes emplea actualmente TCP y las previsiones indican que seguirá empleándose durante mucho tiempo, el diseño de nuevas

tecnologías de redes de comunicaciones, como es el caso de OBS, ha de tener en cuenta de manera significativa la interacción entre TCP y la tecnología de red. En primer lugar, en el Capítulo 3 se realiza un estudio detallado del estado del arte del protocolo de transporte TCP, con énfasis en los mecanismos de control de flujo y congestión. Después, se analiza el comportamiento básico de TCP en redes OBS desde un punto de vista cualitativo, y se realizan simulaciones de TCP en OBS mediante la herramienta OPNET Modeler para ilustrar dicho comportamiento. El análisis se centrará en las versiones de TCP Reno, por ser la versión que ilustra el comportamiento general de TCP (el resto de versiones son modificaciones más o menos agresivas) y SACK (asentimiento selectivo, que permite obtener información adicional en caso de pérdidas múltiples), por ser una opción ampliamente utilizada en todos los sistemas operativos modernos. Finalmente, se realiza un recorrido por los estudios de la literatura con respecto a TCP sobre OBS y se analizan los aspectos abiertos que se estudian en esta tesis.

Con el objetivo de profundizar en el estudio de TCP sobre OBS, en el Capítulo 4 se estudia en primer lugar el impacto de un aspecto concreto de TCP, el asentimiento retardado, una técnica empleada ampliamente en redes asimétricas. Mediante simulación con OPNET Modeler se analiza el *throughput* de TCP sobre OBS con y sin la utilización del asentimiento retardado. Después, mediante simulación, se estudia el comportamiento de un flujo TCP que se ensambla en un ensamblador OBS junto con tráfico perteneciente a otras conexiones, que denominaremos tráfico de fondo, y se analiza qué impacto tiene en el rendimiento de TCP en redes OBS. No solo se presentan resultados cuantitativos, sino explicaciones cualitativas del impacto de diferentes escenarios y parámetros, como la utilización de las distintas versiones de TCP, como Reno y SACK y el asentimiento retardado, complementando el estudio comenzado al principio del capítulo.

Tras estudiar TCP sobre OBS empleando la técnica de la simulación, en el Capítulo 5, con el objetivo por un lado de profundizar en el entendimiento de la dinámica de TCP sobre OBS, y por otro lado de obtener una expresión sencilla que pueda relacionar los principales parámetros del diseño de OBS con el rendimiento de una conexión TCP, se ha diseñado un modelo teórico para el rendimiento de TCP en redes OBS. En primer lugar, se estudian las técnicas de modelado de TCP de la literatura y los trabajos de modelado de TCP sobre OBS. Después, se realiza la propuesta del modelo de TCP sobre OBS, que tiene en cuenta aspectos clave como la limitación de ventana, el asentimiento retardado estudiado en el capítulo anterior y el número de segmentos por ráfaga, parámetro que tienen gran impacto en el rendimiento. Se obtienen cotas analíticas para el número de segmentos por ráfaga, que se validan mediante simulación con OMNeT++. Finalmente, se valida el modelo propuesto con simulaciones de TCP sobre OBS adicionales.

En el Capítulo 6 se estudia el efecto de la sincronización de flujos TCP en una red OBS. La sincronización de TCP aparece cuando varios flujos TCP comparten un enlace y los mecanismos de control de congestión de TCP reaccionan al mismo tiempo, provocando un uso ineficiente del ancho de banda. En OBS esta sincronización se produce cuando se pierde una ráfaga con segmentos de varios flujos. En este capítulo se estudia la cuantía de este efecto y se analiza cuánto es el ancho de banda mínimo necesario para transportar una serie de flujos con varios mecanismos de ensamblado. Finalmente, se propone un nuevo esquema de ensamblado que asigna flujos a varias colas para reducir la sincronización y se evalúa la bondad del esquema.

El Capítulo 7 se dedica al encaminamiento en OBS, que es el proceso por el cual se decide el conjunto de enlaces y nodos que empleará una ráfaga de datos para ir desde el nodo origen al nodo destino. Se realiza una propuesta de algoritmo multicamino que ayude a reducir la probabilidad de bloqueo en redes OBS con reserva de camino sin confirmación. En este capítulo, se detalla en primer lugar el estado del arte de las técnicas

de encaminamiento en OBS. Se realiza un recorrido por la técnica MRDV (*Multipath Routing with Dynamic Variance*), un técnica multicamino para redes IP que se empleará como base. A continuación se describe cómo se han extraído los principales conceptos y se ha diseñado el algoritmo adaptado a una red OBS con señalización en un único sentido, por ejemplo mediante JET (*Just Enough Time*). La técnica de dedica a balancear el tráfico en la red para que la probabilidad de bloqueo global se reduzca. Finalmente se realiza una evaluación del mismo mediante ns2 en un escenario de red de tamaño nacional.

En el Capítulo 8 se presenta una arquitectura de WR-OBS empleando elementos definidos por los estándares, como son el elemento de cálculo de caminos, PCE y el protocolo de establecimiento de caminos RSVP. En primer lugar se realiza un breve repaso al estado del arte del PCE. Se proponen unas extensiones al protocolo PCEP (el protocolo estándar para comunicarse con el PCE) para emplearse en redes WR-OBS. Finalmente, se realiza el diseño de un PCE para WR-OBS y se implementa un prototipo del mismo.

En el Capítulo 9 se realiza una evaluación experimental del rendimiento de la arquitectura de PCE con WR-OBS en un entorno emulado, con los mecanismos de construcción de ráfagas LBS (*Limited Burst Size*) y FBAT (*Fixed Burst Assembly Timer*), midiendo parámetros tales como el retardo de borde, la probabilidad de bloqueo o el tiempo de cálculo. Cabe destacar que no se trata de una simulación, ya que las peticiones al PCE se generan y responden en tiempo real. Con el objetivo de complementar y contrastar los resultados experimentales, se obtienen expresiones analíticas para el retardo de borde y el número medio de *lightpaths* establecidos.

Las conclusiones principales de los estudios llevados a cabo en esta tesis, junto con las futuras líneas de investigación, se explican en el Capítulo 10, en el que también se incluye un listado de las publicaciones generadas a partir de los resultados de esta tesis y un resumen de las contribuciones a los organismos de estandarización, IETF e ITU-T, que se han logrado con el trabajo realizado en la tesis.

Debido al gran número de acrónimos empleados en la tesis, se incluye con el fin de facilitar su lectura un glosario de términos en el Capítulo 11. Finalmente, en el Capítulo 12, se incluye un listado de las referencias bibliográficas empleadas.

Capítulo 2

Fundamentos de OBS

Esta tesis se centra en las redes basadas en la tecnología de conmutación óptica de ráfagas. En este capítulo se realiza una revisión del estado del arte de los aspectos principales de esta tecnología. En concreto, se describe la arquitectura de una red OBS, se estudia el tipo de nodos y se profundiza en las distintas funciones, como son las técnicas de ensamblado de ráfagas, señalización, encaminamiento y resolución de contienda.

2.1 Motivación de OBS

El continuo incremento en la demanda de tráfico impone la necesidad de tecnologías cada vez más eficientes en el transporte del tráfico. La tecnología óptica ha ayudado desde sus inicios a realizar la transmisión de grandes volúmenes de datos. El componente clave es la fibra óptica, un medio guiado fácil de fabricar y que ofrece un ancho de banda y unas cualidades de transmisión excelentes. Un ejemplo de la magnitud del potencial del ancho de banda y la distancia que se puede alcanzar mediante la transmisión en fibra óptica, se encuentra en el experimento realizado por Salsi *et al.* [14] quienes han logrado realizar una transmisión de 8 Tbps en una sola fibra a 9000 km de distancia. Este ancho de banda se puede aumentar a costa de reducir la distancia, como se muestra en el experimento llevado a cabo por Yu *el al.* [15] en el que se consigue ofrecer una transmisión de 11,2 Tbps a 640 km de distancia.

Aprovechando la capacidad de transmisión sobre fibra óptica, las primeras redes de comunicaciones ópticas consistían en conectar equipos mediante enlaces ópticos punto a punto de gran capacidad [16]. Como ejemplo se puede mencionar la arquitectura de una red troncal actual típica, que está formada por potentes *routers* que procesan paquetes IP (*Internet Protocol*) o MPLS (*Multiprotocol Label Switching*) interconectados mediante enlaces punto a punto de alta capacidad, principalmente basados en tecnología de multiplexación por división en longitud de onda densa, DWDM (*Dense Wavelength Division Multiplexing*). Aunque la transmisión de datos es puramente óptica, en esta primera alternativa de redes de comunicaciones ópticas, la conmutación se realiza de manera electrónica, bien mediante conmutación de datagramas IP, etiquetas en el caso de *routers* MPLS, conmutación de tramas Ethernet o bien mediante conmutación de *slots* de tiempo, por ejemplo, en SDH (*Synchronous Digital Hierarchy*, Jerarquía Digital Síncrona).

Por tanto, en los nodos de la red, los procesos por los cuales se decide qué hacer con los datos que llegan al nodo y la conmutación física de una entrada a otra de los mismos se realizan con un procesamiento electrónico. Gracias a este procesamiento electrónico, se consigue una elevada flexibilidad, y se puede incluir una gran variedad de funciones avanzadas. Sin embargo, el procesamiento electrónico tiene unas necesidades de consumo de potencia, que aumentan con la velocidad de procesamiento (según Pickavet *et al.* [17] la potencia consumida en un *router* es proporcional a $C^{2/3}$ donde C la capacidad del *router* en Mbps). El problema se debe a que, mientras que el rendimiento de los semiconductores, pieza clave de la electrónica se dobla cada 18 meses (por la conocida Ley de Moore), el

tráfico se dobla anualmente. Por otro lado, en un estudio realizado por Cisco [18], se muestra que hasta 2008 aproximadamente el consumo energético por bit ha ido descendiendo a medida que la tecnología avanza. Sin embargo, esta mejora se está estancando y el tráfico sigue en aumento. Este aumento excesivo del consumo energético a altas velocidades provoca que la integración de los dispositivos electrónicos sea mucho más compleja.

La tecnología óptica, en concreto la aparición de los conmutadores ópticos, ha supuesto una alternativa mucho más eficiente que la electrónica para la conmutación de datos. Conmutar una longitud de onda consume mucha menos energía que conmutar la misma cantidad de datos en un *router* paquete a paquete [19].

Por tanto, la aparición de equipos ópticos capaces de conmutar de manera totalmente óptica canales ópticos (denominados *lightpaths*, o, de manera simplificada, lambdas o longitudes de onda), ha marcado la siguiente alternativa en cuanto a redes de comunicaciones óptica, que se denomina conmutación óptica de circuitos (*optical circuit switching*, OCS) y permite, además de transmitir en fibra óptica, la posibilidad de conmutar los datos en el dominio óptico, sin necesidad de emplear el dominio electrónico. El componente clave de esta segunda generación es el ROADM (*Reconfigurable Add Drop Multiplexer*), un equipo capaz de insertar y extraer longitudes de onda, conmutar longitudes de onda entre varios puertos y reconfigurarse [20] [21]. Cabe decir que aunque tradicionalmente al equipo capaz de conmutar longitudes de onda entre varios puertos se le ha denominado OXC (*optical crossconnect*), actualmente se emplea el término ROADM por razones de *marketing*. Este equipo, el ROADM, ya está disponible comercialmente, y su despliegue masivo en las distintas redes de transporte de los operadores se realizará en los próximos años.

Sin embargo, la principal limitación de esta tecnología es su granularidad, ya que ofrece conexiones ópticas con unos anchos de banda concretos, que equivalen a una longitud de onda, típicamente 10 Gbps, 40 Gbps o incluso 100 Gbps. De hecho, la anchura de los canales ópticos está especificada por el organismo ITU (*International Telecommunication Union*) [22], en su norma ITU-T G.694.1 [23]. La granularidad mayor (conexiones de diferentes anchos de banda), y la agregación (combinación de conexiones de menor velocidad en una de mayor velocidad) se proporcionan típicamente mediante tecnología electrónica, que adolece de los problemas de escalabilidad mencionados anteriormente.

Por ello, la investigación actual en redes ópticas apunta hacia alternativas en las que los paquetes IP/MPLS se conmuten a nivel totalmente óptico, con el objetivo de obtener una tecnología eficiente, en cuanto a coste y consumo energético por bit. La visión a largo plazo, es llegar a la conmutación óptica de paquetes ópticos (*Optical Packet Switching*, OPS) [Yoo06]. Sin embargo, a pesar de los continuos avances, el grado de madurez tecnológico de OPS es aún bajo, y aún queda mucho camino que recorrer [7].

Por tanto, se ha planteado, como alternativa a medio camino entre la conmutación de circuitos y de paquetes la conmutación óptica de ráfagas (*Optical Burst Switching*, OBS). El fundamento de esta tecnología, propuesta inicialmente por Qiao [6] y Turner [24], es agrupar los paquetes electrónicos en un "superpaquete" o ráfaga, y que éste sea transmitido y conmutado de manera totalmente óptica en la red. Los requisitos tecnológicos para conmutar ráfagas de manera totalmente óptica son menos exigentes que para conmutar paquetes. Esto se debe principalmente al hecho de que al tener las ráfagas una longitud mayor que un simple paquete, la frecuencia con la que hay que conmutar y procesar cabeceras es menor, alcanzable con la tecnología actual, como se verá al estudiar los componentes ópticos habilitadores de OBS en la Sección 2.3.

Hoy en día, la frontera entre OBS y OPS es cada vez más difusa, y se emplea habitualmente el término *burst/packet switching* u OBS/OPS [7]. Asimismo, se habla de

tecnologías *subwavelength* como término genérico para todas las tecnologías que permiten compartir el uso de una longitud de onda [25] [26] [27].

A modo de resumen final, se muestra en la Tabla 2-1 una comparativa entre las principales alternativas de redes de comunicaciones ópticas mencionadas anteriormente, OCS, OBS y OPS, en cuanto a los requisitos tecnológicos y rendimiento obtenido por cada una de ellas.

	OCS	OBS	OPS
Velocidad de conmutación necesaria	ms	µs-ms	ns-µs
Señalización	Fuera de banda	Fuera de banda/en banda	En banda
Complejidad del hardware	Muy baja	Media	Alta
Problemas y algoritmos de control	*Routing and Wavelength Assignment* (RWA) estático y dinámico	Algoritmos de ensamblado. Planificación de ráfagas Resolución de contienda	Resolución de contienda. *Schedulers* de paquetes de alta velocidad
Complejidad de los algoritmos de control	Media	Muy alta	Alta
Complejidad del encaminamiento	Media/alta	Alta para mejorar rendimiento	Alta
Utilización del ancho de banda	Baja	Alta (multiplexación estadística de ráfagas)	Alta (multiplexación estadística de paquetes)
Eficiencia	Baja	Alta	Alta
Throughput	Alto	Pérdidas de ráfagas pueden bajar el *throughput*	Alto
Retardo	Bajo	Retardo extra por la generación de ráfaga	Retardo extra por encolado de paquetes

Tabla 2-1 Comparativa OCS, OBS, OPS

A continuación se describe en qué tipos de redes se puede emplear OBS, con el objetivo de saber para qué se va a utilizar OBS. Después se analizan brevemente las tecnologías ópticas que hacen que construir una red basada en OBS sea factible. Posteriormente se centra el estudio en la arquitectura propia de OBS y se analizan las funciones principales, que después se analizan una a una, viendo el estado del arte y las carencias del mismo para cada función. Por último, se realiza un recorrido por los *testbeds* de OBS que han servido de prueba de concepto y demostrado su viabilidad, finalizando con los primero productos comerciales basados en la conmutación óptica de ráfagas.

2.2 Tipos de red donde aplicar OBS

La tecnología de conmutación óptica de ráfagas se puede emplear en redes de transporte tanto metropolitanas o regionales como de núcleo. Según la Unión Internacional de Telecomunicaciones (ITU) [22], el propósito de una red de transporte es el de proveer una infraestructura para transportar cualquier tráfico cliente y agregar este tráfico de una manera fiable [28]. Para ello, una red de transporte transfiere información de un punto a otro de la red. En este contexto, OBS sería la tecnología encarga de realizar esta transferencia de información entre dos puntos de una red de transporte.

Las redes de transporte metropolitanas o regionales se caracterizan en primer lugar por su alcance. Este no suele ser mayor de 200 kilómetros, por lo que la transmisión de datos no tiene unos requisitos fuertes (al menos si se compara con la transmisión a larga distancia). Las redes metropolitanas/regionales tienen, por un lado, una misión agregadora. Se utilizan para recolectar todo el tráfico proveniente de las redes de acceso que llegan a los

hogares, por ejemplo recogiendo el tráfico de los DSLAM (*Digital Subscriber Line Access Multiplexer*, Multiplexor de acceso a la línea digital de abonado) o de las OLT (*Optical Line Terminal*) y llevarlo al núcleo de la red. Otra de las misiones de estas redes metropolitanas regionales es la de conectar sedes cercanas de empresas o campus universitarios, transfiriendo el tráfico entre ellas.

Cuando la red cumple la misión de recoger el tráfico de los hogares, el tráfico que se tiene que transportar no se distribuye apenas entre todos los nodos, sino más bien se canaliza hacia un punto (o dos si hay redundancia). En el caso de la conectividad entre empresas, el tráfico que se genera es más dinámico y está más distribuido, al haber muchas conexiones entre sedes.

Finalmente, las redes metropolitanas-regionales tienen una misión de dar soporte a nuevos servicios. Por ejemplo, en el entorno metropolitano se espera una emergencia de servicios basados en red, como el Vídeo bajo demanda (VoD, *video on demand*), servicios de almacenamiento (SAN, *Storage Area Network*,), servicios de computación remota distribuida (*cloud computing*) o el PC en red. Estos servicios demandarán conexiones dinámicas de corta duración (segundos, minutos) de gran ancho de banda. Además, algunos servicios demandarán diferentes garantías estrictas de calidad de servicio (retardo, *jitter* y disponibilidad), por ejemplo en aplicaciones de telemedicina (bajo retardo y elevada disponibilidad) o juegos en red (bajo retardo y *jitter*).

Finalmente, cabe destacar que, por razones históricas, las redes metropolitanas-regionales se han desplegado en topologías de anillo, estableciendo la ubicación de las canalizaciones de fibra. Por tanto, en el futuro es probable que, al menos en parte, se sigan respetando estas topologías de anillo, principalmente por la disposición física de las fibras. Sin embargo, por motivos de redundancia, estas se irán mallando progresivamente.

Resumiendo, OBS en el contexto de las redes metropolitanas regionales tendrá que ser capaz de transmitir datos hasta 200 km y soportar un tráfico dinámico. A corto plazo, OBS puede emplearse en redes en anillo, por lo que la conmutación en los nodos se simplifica. A medio/largo plazo, tendrá que migrar hacia redes malladas para ofrecer una mayor supervivencia ante fallos.

Por otro lado, las redes de núcleo (también conocidas como redes troncales, o por su terminología inglesa *core* o *backbone networks*) son las encargadas de transferir el tráfico agregado entre las distintas regiones y hacia Internet. Con respecto al alcance, los enlaces entre los nodos suele ser de cientos de kilómetros. Dependiendo del tamaño del país, pueden existir rutas entre dos nodos de hasta 1000 km en el caso de España, o de 5000 km en el caso de países extensos como Estados Unidos.

Al transferir el tráfico de regiones enteras, las conexiones que se demandan en una red de transporte de núcleo son de un gran ancho de banda, de varios Gigabit/segundo. Además, el tráfico transportado en estas conexiones ha sido agregado previamente de miles de usuarios. La distribución de la demanda de tráfico en este caso viene marcada por el reparto de hogares, empresas y nodos de servicio en el país en cuestión. Por ejemplo, en España, el tráfico se distribuye de una manera no uniforme y la mayor parte de la demanda de tráfico está concentrada en conexiones hacia Madrid, Barcelona y la salida internacional hacia Internet, mientras que en países como Alemania, en los que hay un mayor número de ciudades de tamaño intermedio repartidas por el territorio, o Estados Unidos, donde los principales servidores de internet están repartidos por todo el territorio, el tráfico está mucho más distribuido. Independientemente de su distribución, el tráfico cursado por las redes de transporte de núcleo es relativamente estable y presenta un menor dinamismo que el cursado por las redes metropolitanas, debido principalmente al hecho de trabajar con tráfico previamente agregado.

Con respecto a la topología, esta es muy mallada, al tener que cubrir un extenso territorio y tener una fiabilidad alta, por lo que el corte de un enlace o el mal funcionamiento de un nodo no han de impedir a la red su misión. Al transportar el tráfico de un gran número de usuarios y servicios, cualquier corte o degradación de la calidad en los servicios transportados, tiene un gran impacto, que puede dar lugar a cuantiosas pérdidas económicas.

Por tanto, en el caso de OBS para la red de núcleo va a tener que transmitir grandes volúmenes de datos a cientos de kilómetros de distancia. A diferencia de las redes metropolitanas, al tratar con tráfico agregado previamente, el tráfico de las conexiones que tiene que transportar es más estable y no tan dinámico.

2.3 Tecnologías ópticas habilitadoras

La conmutación óptica de ráfagas OBS es viable gracias a la aparición, con mejoras año a año, de un conjunto de tecnologías ópticas, como son láseres sintonizables ultra-rápidos [29], que permiten el uso compartido de una longitud de onda y los conmutadores ópticos rápidos [30] [31], que permiten reconfigurar en muy poco tiempo una matriz de conmutación óptica.

2.3.1 Láseres sintonizables ultra-rápidos

Los láseres sintonizables convencionales empleados en la transmisión WDM tienen unos tipos de sintonización en el orden de segundos, siendo adecuados para la conmutación de circuitos, ya que una vez sintonizado el mismo, su configuración no cambia en el tiempo. Sin embargo, para poder compartir una longitud de onda entre varias fuentes, si se quiere que haya un alto aprovechamiento del canal, la sintonización del láser ha de realizarse de manera mucho más rápida. Los láseres sintonizables ultra-rápidos que emplean la banda C de la ITU-T pueden cambiar de frecuencia en nanosegundos [29].

Hay dos aproximaciones principales para los láseres ultra-rápidos, unos se basan en la selección entre múltiples cavidades, como por ejemplo el láser multi-frecuencia, MFL (*Multi-Frequency Laser*) [32] y otros en sintonizar elementos de una cavidad, como los *gratings* o resonadores de anillo, como los SG-DBR (*Sampled Grating Distributed Bragg Reflector*) [29].

Con respecto a los láseres MFL, se ha reportado tiempos por debajo del nanosegundo con dispositivos capaces de sintonizar entre 32 y 56 canales [32]. En cuanto a los láseres SG-DBR, se han reportado tiempos de sintonización de 45 nanosegundos entre 32 longitudes de onda en una rejilla de 100 GHz, [33], reduciéndose a 5 nanosegundos con técnicas de preénfasis [34].

Tipo de laser	Tiempo de sintonización	Referencias
MFL	< 1 ns	[32]
SG-DBR	45 ns	[33]
SG-DBR con preénfasis	5 ns	[34]

Tabla 2-2 Láseres sintonizables ultra-rápidos

2.3.2 Conmutadores ópticos

Los conmutadores ópticos son la pieza clave para poder realizar la conmutación óptica de ráfagas (y paquetes). Su rapidez, es decir, el tiempo que transcurre desde que se indica a un conmutador óptico que cambie su matriz de conmutación de una configuración a otra hasta que se realiza dicho cambio, determinará la viabilidad o no de OBS/OPS.

A continuación se realiza un resumen de las principales ventajas de cada una de las tecnologías, y se muestra en qué demostradores se han utilizado. Una recopilación del detalle de las distintas tecnologías de conmutación puede encontrarse en el estudio de Vlachos [35].

Opto-mecánicos

La tecnología opto-mecánica fue la primera tecnología propuesta comercialmente para conmutación óptica [36]. La conmutación se realiza por medios mecánicos exclusivamente, como prismas, espejos, y acopladores direccionales. Tiene un buen comportamiento en cuanto a *crosstalk*, polarización y pérdidas, y su coste de fabricación es bajo. Su tiempo de conmutación está en el orden de ms. Su principal desventaja es la escalabilidad.

MEMS

Los MEMS (*Micro-electro-mechanical systems*) son elementos de conmutación hechos a partir de obleas de silicio y micro espejos. El principio de funcionamiento es sencillo. La luz se conmuta de una fibra a otra encaminando la luz con la ayuda de un espejo [37].

Al tratarse de un espejo altamente reflectante, es prácticamente independiente de la longitud de onda, por lo que hay que añadir filtros adicionales si se quieren eliminar longitudes de onda. Hay dos tipos de MEMS, planos, de dos dimensiones (2-D) o espaciales, en tres dimensiones (3-D) [38].

La tecnología MEMS está muy madura, y se emplea en conmutadores comerciales, por ejemplo de JDSU [39].

Termo-ópticos

Los conmutadores termo-ópticos se basan en que un cambio en la temperatura del material cambia sus propiedades, en concreto el índice de refracción [37]. La velocidad y estabilización de un flujo térmico para cambiar la temperatura de un dispositivo es relativamente lenta, y depende del tipo de material. Los tiempos de conmutación están en el orden de decenas de milisegundos en el caso del silicio, y unos pocos milisegundos en el caso de polímeros. Los conmutadores termo-ópticos funcionan aplicando más potencia a un calentador, que cambiará el estado del conmutador. Uno de los principales problemas es la disipación de calor, ya que el comportamiento del conmutador se ve afectado por el calor disipado por dispositivos cercanos.

Electro-ópticos

En algunos cristales, el índice de refracción es proporcional a la intensidad del campo eléctrico que se le aplica. Esto se conoce como el efecto lineal electro-óptico [40]. De esta forma, se puede controlar la intensidad o la fase de la luz que se propaga por dicho material. Hay varios tipos de materiales donde se manifiesta el efecto mencionado, por ejemplo materiales de estado sólido ($LiNbO_3$), polímeros o *Lead Zinconate Titane* (PLZT). Los basados en $LiNbO_3$ se usan para moduladores rápidos por su tiempo de respuesta (ps), mientras que los basados en PLZT obtienen tiempos de conmutación de ns. Sin embargo, el material más popular en la actualidad para conmutadores es el PLZT, debido a su facilidad de fabricación y coste [37] [35].

Cristal líquido (LCOS)

Los conmutadores basados en cristal líquido (LC, *Liquid Crystal*, aunque el tipo más popular es LCOS, *Liquid Crystal on Silicon*) emplean la óptica de espacio abierto para conmutar [35]. El tipo de conmutador de cristal líquido más común se basa en el principio por el cual la luz, al reflejarse en un cristal líquido, sufre una rotación en su polarización. Para realizar un conmutador, se utiliza un par de celdas de cristal líquido junto con un par

de *splitters* de haz de polarización (PBS, *Polarization Beam Splitter*) [41], que son unos elementos que dividen la luz en dos componentes polarizadas ortogonalmente entre sí. Un ejemplo de funcionamiento consiste en dirigir el haz luz hacia el mencionado *splitter* de polarización PBS, que enviará cada una de las componentes en una dirección diferente. En la trayectoria de cada uno de los componentes se sitúa una celda de cristal líquido, controlada mediante un voltaje para decidir si se rota o no la polarización de la luz que refleja. La luz reflejada por ambas celdas de cristal líquido se cruza a la entrada de otro *splitter* de haz de polarización, el cual combinará las dos componentes en una dirección u otra, según la polarización de los haces de luz que le lleguen, realizando así la conmutación. Por tanto, lo que hace falta para manejar un conmutador de este tipo es un voltaje que cambie cuando se quiera conmutar. Al no tener partes movibles, como los MEMS, son más sencillos y duraderos. La tecnología LCOS está muy madura y hoy en día se emplean en ROADMs comerciales, por ejemplo de Finisar [42].

Semiconductor Optical Amplifier (SOA)

Los amplificadores de semiconductor ópticos (SOA, *Semiconductor Optical Amplifiers*), pueden funcionar como un conmutador [35]. Pueden conseguir un alto nivel de integración, con lo que se pueden combinar fácilmente con *splitters* o multiplexores, para poder realizar funciones de inserción/extracción de longitudes de onda. Las principales desventajas son su alto factor de ruido y el *crosstalk*. Se pueden conseguir tiempos de conmutación por debajo del nanosegundo.

Comparativa de las tecnologías

Las tecnologías mencionadas se pueden emplear en *routers* OBS/OPS. Zervas *et al.* [30] han realizado un estudio sobre los elementos de conmutación ópticos más adecuados para un *router* OBS/OPS comparando los tiempos de conmutación y su escalabilidad. La Tabla 2-3 muestra un resumen de los resultados de Zervas *et al.* [30] completados con datos de la recopilación de tecnologías de conmutación realizada por Vlachos [35].

	Tiempo de conmutación	Escalabilidad
Conmutador opto-mecánico	4 ms	16×16
MEMS ópticos	3D ~ 10 ms, 2D: ~3 ms	3D: 1000×100, 2D: 32×32
PLC termo-ópticos	~ 3 ms	4×4
Electro-ópticos (PLZT)	~ 20 ns	4×4
Cristal líquido (LCOS)	~ 3-10 ms	32×32
Semiconductor *Optical Phase Array*	~ 30 ns	64×64
SOA *Broadcast and select*	~ 1 ns	32×32
SOA *Cross-point*	~ 1 ns	4×4

Tabla 2-3 Comparativa de tecnologías de conmutación fotónicas

Conmutadores ópticos usados en experimentos de OBS

Algunas de estas tecnologías se han usado en *testbeds* de OBS (en el apartado 2.11 y la Tabla 2-11 se muestra una recopilación de los *testbeds*). Por ejemplo, Baldine *et al.* [43] realizaron un demostrador de OBS con MEMS. NTT y Fujitsu emplearon PLCs y MEMS en su *testbed* de OBS [44] [45]. La universidad de Tokio empleó conmutadores PLCs comerciales para construir una matriz de conmutación 16×16 con un tiempo de conmutación de 3 ms [46]. Por último, cabe destacar el *testbed* de OBS del proyecto OITDA en el que se ha empleado la tecnología PLZT y se ha conseguido un tiempo de conmutación de 3,5 ns [47].

2.4 Arquitectura de OBS

Actualmente, el concepto de OBS es muy amplio, ya que engloba las tecnologías capaces de emplear fracciones de una longitud de onda y conmutar de manera totalmente óptica. A pesar de haber distintas alternativas y variantes específicas, todas siguen una misma filosofía: agregar el tráfico en el borde, transmitir ráfagas ópticas y evitar la conversión opto-electrónica en los nodos intermedios. La arquitectura de una red OBS genérica (Figura 2-1) consta de dos tipos de nodos funcionales: los nodos frontera (*edge router*) y los nodos de transporte (*core router*).

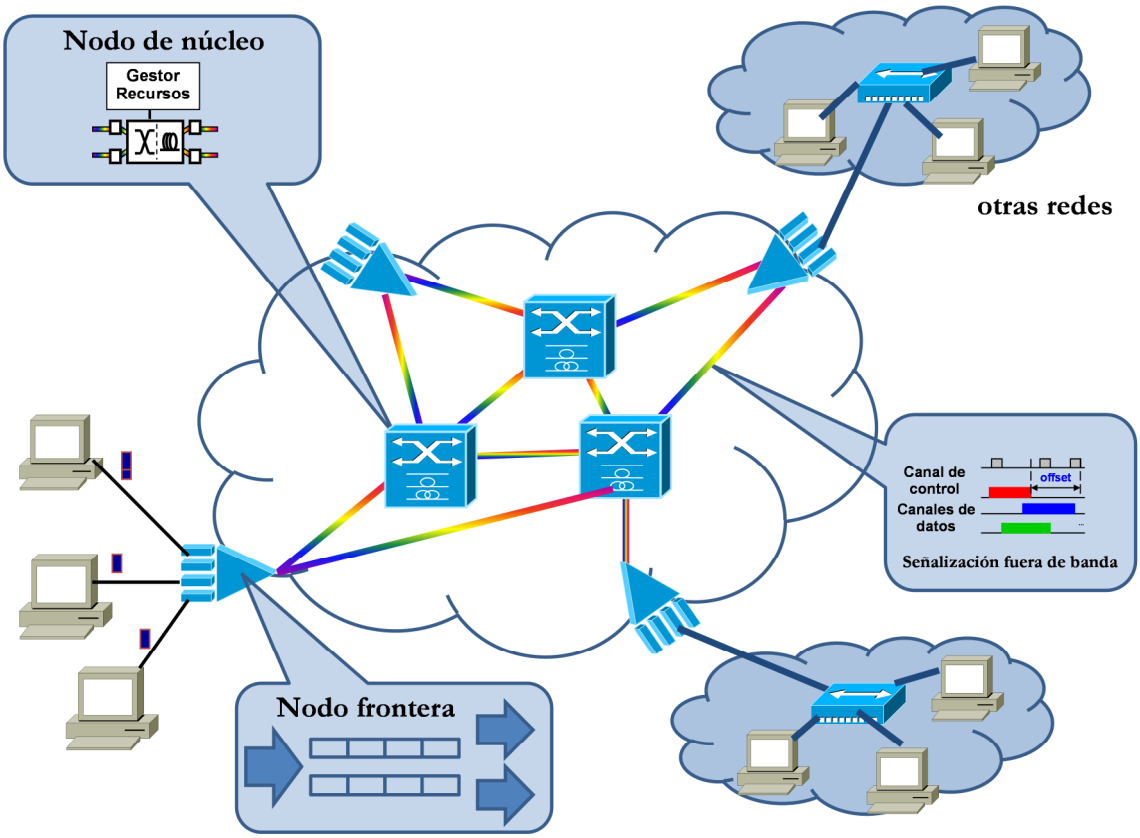


Figura 2-1 Arquitectura genérica de una red OBS

Nodo frontera

Los nodos frontera son los encargados de agregar los paquetes del exterior, clasificarlos, enviarlos al ensamblador adecuado, y generar las ráfagas. Los paquetes se reciben en el nodo a través de unas interfaces clientes, que implementan protocolos de enlace como por ejemplo los basados en Ethernet. Una vez ha llegado el paquete al nodo, se observa su cabecera, y en base a su contenido, se decide a qué clase de servicio pertenece y cuál es el nodo frontera de la red OBS al que hay que enviar el paquete.

Este primer encaminamiento es similar al que ocurre en el caso de la tecnología de paquetes MPLS (*Multi Protocol Label Switching*) en el *router* de borde, denominado LER (*Label Edge Router*, también conocido como *Edge Label Switch Router*) [48], donde se procesa toda la cabecera del datagrama IP, se emplea su información de encaminamiento para añadir la cabecera MPLS y enviar el paquete (datagrama IP + cabecera MPLS) por el puerto correspondiente.

Ensamblado de ráfagas en el nodo frontera

En cada nodo frontera OBS hay un ensamblador por cada uno del resto de destinos de la red OBS (resto de nodos frontera) y clase de servicio, por lo que, una vez clasificado, se

envía el paquete al ensamblador adecuado. Cada ensamblador puede tener internamente una o más colas, cada una de las cuales generará ráfagas ópticas.

El caso trivial es en el que hay una cola por cada destino y clase de servicio [6], es decir, una cola por ensamblador. Por tanto, todos los flujos hacia un mismo nodo frontera en la red OBS comparten dicha cola. Se denomina a esta estrategia como MF (*Mixed Flow,* flujos mezclados) [49], ya que múltiples flujos se mezclan en una cola. El problema de una sola cola es que si se pierde una ráfaga, esta contiene segmentos de muchos flujos diferentes. En el caso de que los flujos estén controlados por un protocolo de transporte como TCP que reacciona ante las pérdidas, habrá una respuesta sincronizada de dichos mecanismos, que se estudian con detalle en la tesis.

Una alternativa es la estrategia PF (*Per Flow,* por flujo). En esta estrategia cada flujo se envía a una cola. El principal problema de esta técnica es su escalabilidad, ya que en un entorno real el número de colas sería muy elevado [49]. Por tanto, esta estrategia tiene un interés meramente teórico. Sin embargo, se ha considerado adecuado mencionarla, ya que muchos estudios de TCP sobre OBS consideran un único flujo TCP por cola [50] [51] [52].

Finalmente, con la estrategia MQ (*Multi Queue,* multicola), se tienen varias colas por ensamblador para un mismo par origen destino y clase de servicio [49]. La estrategia MQ se propone y estudia en esta tesis, con el objetivo de reducir la sincronización de flujos TCP y aprovechar mejor el ancho de banda.

En cada cola del ensamblador se van acumulando paquetes, y periódicamente se generan ráfagas con varios de estos paquetes. Hay varias estrategias para generar estas ráfagas, que se describen con detalle en la sección 2.5 de este capítulo. A modo de resumen, una estrategia de ensamblado típica consiste en tener un temporizador, fijo o adaptable (a través de la información recibida de la red como podría ser la carga) a cuyo vencimiento finalice la construcción de la misma.

Una vez que se ha determinado, por el algoritmo de ensamblado, que se ha finalizado la creación de la ráfaga, todos los segmentos en la cola se unen para formar una ráfaga óptica, que está lista para ser enviada al nodo frontera destino.

Configuración del camino óptico

Sin embargo, para poder enviar la ráfaga, se ha de configurar primero el camino óptico. Es decir, todos los nodos intermedios (los nodos OBS de transporte), han de ajustar su configuración interna para que, al menos durante el intervalo de tiempo en el que la ráfaga atraviese cada nodo, la ráfaga pueda entrar por un puerto y salir por otro hasta llegar al nodo frontera destino, que ajustará su receptor a la longitud de onda de la ráfaga óptica, pudiendo extraer de nuevo todos los paquetes originales que formaron la ráfaga.

Las distintas alternativas para abordar la configuración del camino óptico dan lugar a las diferentes variantes de OBS. En la Sección 2.6 de este capítulo se tratan con detalle cada una de las alternativas, que resumimos a continuación.

Por ejemplo, en la alternativa WR-OBS (*Wavelength-Routed OBS*), se solicita la reserva de un *lightpath*, es decir, de un circuito óptico extremo a extremo, y hasta que no llegue la confirmación del mismo, no se envía la ráfaga óptica [53]. La reserva puede realizarse tanto de manera coordinada por un elemento central (que calcularía el camino e informaría a cada nodo involucrado de la configuración necesaria a realizar), como de manera totalmente distribuida (por ejemplo empleando un protocolo como RSVP-TE para GMPLS [54]). En esta tesis, se va a diseñar e implementar un elemento central para el cálculo de los caminos para las ráfagas ópticas en WR-OBS, con una comunicación basada en el protocolo PCEP [11], que se procederá a estudiar y ampliar en la medida que sea necesario.

Una alternativa muy estudiada se conoce como reserva en un sentido o "*one-way reservation*" [55], y se basa en transmitir la ráfaga antes de recibir la respuesta de la reserva del camino. Hay varias aproximaciones para la reserva de un solo sentido, siendo el protocolo más popular el denominado OBS-JET (J*ust Enough Time*). En este protocolo, se genera en el nodo frontera un paquete de control, denominado BCP (*Burst Control Packet*), que se envía al siguiente nodo en el camino óptico y así sucesivamente hasta llegar al destino, informando del instante de la llegada y salida de la ráfaga a cada nodo. Esta reserva se hace con el mínimo tiempo posible, de tal forma que justo después de configurarse el nodo destino, llegue la ráfaga (justo a tiempo) al destino. Para poder llegar justo a tiempo, entre el instante de tiempo en el que se envía el paquete de control y el tiempo en el que se envía la ráfaga se deja un tiempo de *offset*.

Como el tiempo de *offset* no siempre es fácil de calcular (si antes de enviar la ráfaga se sabe el camino, el cálculo del *offset* es fácil, pero en caso contario solo se puede estimar), Klinkowski *et al.* [56] proponen la alternativa de OBS sin *offset*, en el que en cada nodo, la ráfaga se espera en un FDL (*Fiber Delay Line*) mientras se resuelven los potenciales conflictos.

Nodos de núcleo

Los nodos OBS de núcleo han de participar en la configuración del camino óptico y han de realizar la configuración de la matriz de conmutación del equipo para que cuando llegue la ráfaga, esta pueda salir por el puerto adecuado. Según la alternativa de OBS empleada, recibirá unos mensajes de señalización u otro.

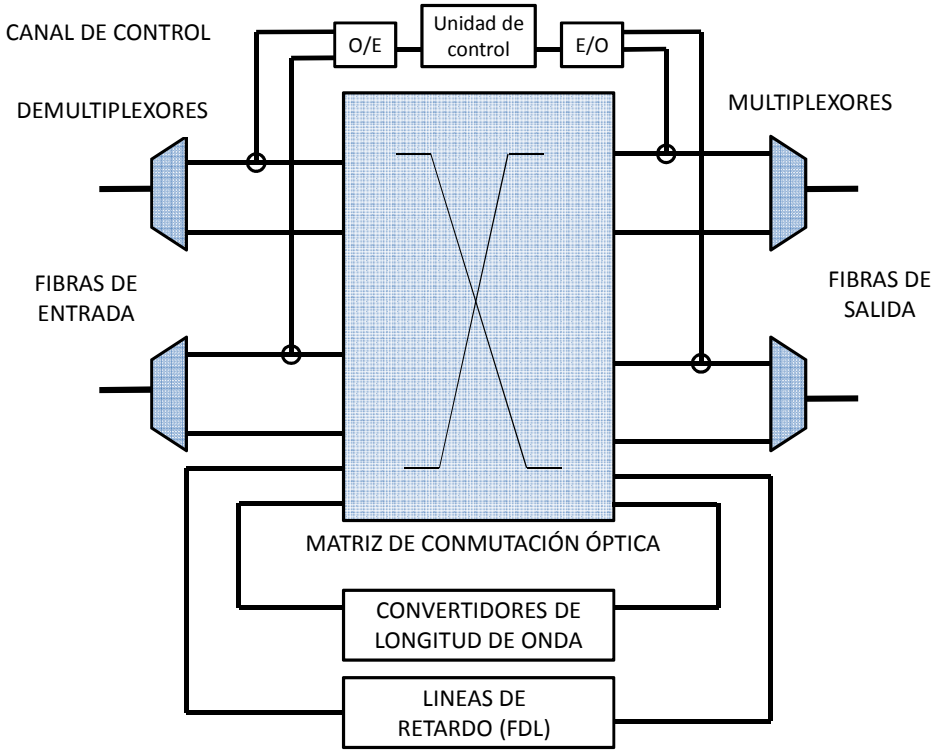


Figura 2-2 Arquitectura genérica de un nodo OBS de núcleo

Un ejemplo de la arquitectura de un nodo de núcleo típico se muestra en la Figura 2-2. Tal y como se ha mencionado, en OBS el control y los datos viajan por separado. Por tanto, una opción para el envío de los mensajes de señalización es emplear un canal de la fibra. Para poder recibir los mensajes de señalización por un canal de la fibra, el nodo ha de disponer de convertidores optoelectrónicos. Asimismo, para procesar el mensaje de señalización ha de disponer de una unidad de control. En esta unidad de control se realiza planificación de las ráfagas, se resuelven las situaciones de contienda y se crea un nuevo

mensaje de señalización para otro nodo si fuera necesario. Las técnicas para resolver las contiendas, como se verá en las Secciones 2.8 y 2.9, en muchos casos implicarán ligeras modificaciones en la arquitectura del nodo, ya que pueden implicar componentes adicionales, por ejemplo las líneas de retardo que almacenan temporalmente una ráfaga óptica en caso de contienda y convertidores de longitud de onda, para poder emplear canales libres.

Matriz de conmutación óptica

Una parte importante del nodo de núcleo es la matriz de conmutación óptica, que permitirá conmutar las ráfagas de una fibra de entrada a otra de salida. Este es un elemento, reconfigurable que se controla automáticamente desde la unidad de control, y estará realizado a partir de una de las tecnologías explicadas en la sección 2.3.2. La rapidez de la misma es clave para lograr una buena utilización del canal. Esto se debe a que mientras se está cambiando la configuración de la matriz de conmutación de un determinado canal no pueden llegar datos por el mismo. Por tanto, a la hora de realizar la planificación del canal óptico se ha de tener en cuenta la velocidad de la misma.

2.5 Ensamblado de ráfagas

Como se ha comentado al mostrar la arquitectura general de OBS, uno de los aspectos clave es el proceso de agregación de paquetes en los nodos frontera para formar ráfagas. Conviene recordar que se definió ráfaga como un conjunto de paquetes que se agrupan para transmitirse posteriormente como un único bloque. Por tanto, cuando los paquetes llegan al nodo frontera, se distribuyen en los distintos ensambladores según su destino y clase de servicio. Dentro de los ensambladores, se enviará el paquete a una cola determinada, pudiendo haber una o varias colas por ensamblador. En estas colas se implementa un algoritmo de ensamblado, que determina cuándo se ha completado la formación de una ráfaga. En la literatura se han propuesto distintos algoritmos, que se recopilan en la Tabla 2-4 y se explican en esta sección.

Tipo de algoritmo de ensamblado	Algoritmos propuestos
Basados en Temporizador fijo	[57] [58] [59] [60] [61]
Basados en Tamaño de ráfaga máxima	[62] [63] [60] [64]
Híbrido (temporizador y tamaño)	[65] [66]
Selección aleatoria	[60]
Adaptativo (el temporizador depende de la carga, predicción o características del tráfico)	[59] [67] [68] [69] [70]

Tabla 2-4 Tipos de algoritmo de ensamblado

2.5.1 Algoritmos basados en temporizador

Los algoritmos más populares son los basados en temporizador [57] [58] [59] [60]. Cuando llega un paquete al ensamblador, y no se ha empezado aún a formar una ráfaga, se comienza una nueva ráfaga y se activa un temporizador. Los paquetes que llegan posteriormente se van añadiendo a la ráfaga hasta el instante en el que vence el temporizador. Para evitar ráfagas excesivamente pequeñas, Ge *et al.* proponen que las ráfagas tengan un tamaño mínimo, por lo que si el número de bytes acumulado no llega al mínimo, se rellena la ráfaga (*padding*) [57]. En situaciones de carga moderada de tráfico, un algoritmo de ensamblado basado en temporizador provoca que se envíen ráfagas en intervalos de tiempo periódicos (con periodicidad igual al temporizador) con longitud variable.

Ensamblado basado en temporizador en WR-OBS

En el caso particular de WR-OBS, la técnica de ensamblado FBAT (*Fixed Burst Assembly Timer*) [61] está basada en un temporizador de ensamblado fijo. Empleando dicha técnica, tras el vencimiento del temporizador, todos los paquetes en la cola de ensamblado forman una ráfaga. El siguiente paquete que llegue a la cola de ensamblado iniciará el proceso de ensamblado de una nueva ráfaga. Sin embargo, en WR-OBS, con la técnica LBS (*Limited Burst Size*) [61], cuando vence el temporizador no se finaliza el ensamblado de la ráfaga, sino que comienza el proceso del establecimiento del camino óptico, y se continúan ensamblando paquetes en la misma ráfaga. En el instante en el que el camino óptico esté establecido y se pueda comenzar a transmitir, se termina la formación de la ráfaga, y se inicia un nuevo temporizador. Todos los segmentos que lleguen posteriormente se ensamblarán en una nueva ráfaga. Como alternativa, con la técnica UBS (*Unlimited Burst Size*), el proceso de formación de ráfaga no se termina cuando se ha establecido el camino. En UBS, todos los paquetes que lleguen a la cola mientras se está transmitiendo la ráfaga se añaden a la misma. Se habla de UBS (ráfagas de tamaño infinito) ya que potencialmente podría ser una transmisión sin fin. El proceso de formación de ráfaga se termina cuando no quedan datos en la cola de ensamblado. En ese instante, se activa de nuevo el temporizador.

2.5.2 Algoritmos basados en tamaño de la ráfaga fijo

También existen algoritmos de ensamblado basados en fijar la longitud de ráfaga. En estos algoritmos, el proceso de formación de ráfaga finaliza cuando se alcanza un tamaño de ráfaga fijado. Hay dos alternativas propuestas, o bien se alcanza un determinado tamaño en paquetes [62] [63] [60], o bien se alcanza un número de bytes determinado [60] [64]. En cualquiera de los casos, el comportamiento es que se envían ráfagas en intervalos de tiempo variables, con longitud de la ráfaga fija (en paquetes o bytes).

2.5.3 Algoritmos híbridos

El impacto de los algoritmos de ensamblados en el tamaño de las ráfagas y el tiempo de llegada entre ráfagas ha sido estudiado en varias publicaciones [71] [72]. Ambos algoritmos (temporizador y tamaño fijo) tienen sus ventajas y desventajas. La principal ventaja de algoritmo basado en temporizador es que establece un límite superior al retardo de los paquetes. En los algoritmos con temporizador puede ocurrir que el valor de temporizador sea muy bajo, en cuyo caso se incrementa la carga de los paquetes de control. Si por el contrario se emplea un temporizador muy alto, el retardo introducido por el proceso de formación de la ráfaga puede ser intolerable. Por otro lado, los algoritmos basados en longitud de ráfaga no controlan el tiempo de finalización de la ráfaga, con lo que el retardo de los paquetes no está acotado, de modo que no es adecuado para muchas aplicaciones. Debido a los pros y contras de los algoritmos de temporizador y tamaño de ráfaga fijo, se han propuesto versiones híbridas de ambas técnicas [65] [66], en los que finaliza una ráfaga bien por temporizador o bien por tamaño, buscando un equilibrio entre ambos.

2.5.4 Algoritmo de selección aleatoria

El algoritmo de selección aleatoria, propuesto por Vega *et al.* [60], se basa en asignar a cada paquete que entra en el ensamblador un cero o un uno. El nodo OBS tiene un generador de números aleatorios de Bernoulli para obtener esta secuencia de 1's y 0's con cierta probabilidad p para los 1's y 1-p para los 0's. Cada vez que se asigna un 0 a un paquete, se añade a la ráfaga que se está formando, mientras que si se asigna un 1, se termina la ráfaga y se comienza una nueva. El parámetro p del generador de números aleatorios de Bernoulli tiene un significado muy intuitivo, representa la inversa del número de paquetes IP por ráfaga.

Vega *et al.* [60], en su estudio del comportamiento de los distintos algoritmos de ensamblado, muestran que la propiedad más importante del algoritmo de selección aleatoria es que el proceso de generación de ráfagas que se obtiene tras aplicar este algoritmo es un proceso de *Poisson* puro, algo que no se consigue con el resto de algoritmos. En la literatura es frecuente asumir que el proceso de generación de ráfagas es un proceso de *Poisson*, ya que facilita en gran medida obtener expresiones analíticas.

2.5.5 Algoritmos adaptativos

El funcionamiento de los algoritmos basados en temporizador o tamaño de ráfaga es independiente del comportamiento del tráfico, es decir, de la frecuencia con la que llegan los datos, el tamaño de los paquetes, etc. Es posible mejorar el comportamiento de los algoritmos anteriores adaptando su funcionamiento a las características del tráfico existente en cada momento. Estos algoritmos se conocen como algoritmos adaptativos, y en ellos los valores del temporizador y/o del tamaño de las ráfagas se ajustan continuamente.

***Adaptive Assembly Period* (AAP)**

Cao *et al.* [59] proponen un mecanismo adaptativo denominado *Adaptive Assembly Period* (AAP) para mejorar el rendimiento, por el cual, para cada nueva ráfaga se calcula el valor del temporizador. Este valor depende del tamaño que han alcanzado las ráfagas anteriores. En concreto, el valor del temporizador, denominado según Cao *et al.* "AP" (*assembly period*), se calcula de manera independiente para cada cola de transmisión y tiene en cuenta el ancho de banda disponible para transmitir y el tamaño medio de las ráfagas en la cola correspondiente. El valor de este tamaño medio es una ponderación entre el último valor medido y la media hasta este momento. Los mismos autores proponen dar un mayor peso al valor de la última ráfaga frente a la media resultante hasta ese instante, sugiriendo una relación de 3/4 frente a 1/4. La razón principal de dar más peso a la última ráfaga es que después de una ráfaga muy grande, es probable que se continúe con un elevado ritmo de transmisión. En cambio, si hay pocos paquetes es probable que haya habido algún tipo de reducción en la tasa de transferencia (por ejemplo si se trata de tráfico TCP), por lo que se espera que lleguen pocos paquetes, y para que tengan que esperar poco tiempo, se reduce su temporizador.

Temporizador de ensamblado dinámico (DT-BA) para reducir consumo energético

Kang *et al.* [67] proponen un algoritmo de ensamblado basado en temporizador dinámico (DT-BA, *Dynamic Time based Burst Assembly*) con el objetivo de minimizar el consumo energético del nodo y ofrecer garantías de retardo. El algoritmo calcula el temporizador de ensamblado de la ráfaga acorde al estado de la cola y las tendencias en el tráfico. El algoritmo asume que mientras se está ensamblando la ráfaga, se opera en modo de bajo consumo, y mientras se transmite la ráfaga, es cuando el consumo es elevado. Bajo estas premisas, el algoritmo mantiene dos umbrales, el umbral alto y el umbral bajo. Si el tamaño de la última ráfaga está por debajo del umbral bajo, se entiende que hay poco tráfico en la red. Por tanto, se escoge un valor de temporizador alto, para que el sistema esté más tiempo en modo de bajo consumo. En caso contrario, si está por encima del umbral alto, se reduce el temporizador, para reducir cuanto antes el tamaño de la cola y reducir el retardo de los paquetes. Para calcular el valor del temporizador se tiene en cuenta también las características del tráfico hasta ese instante. Para ello, se calcula de manera continua una media móvil del tráfico. Al final, el valor del temporizador tiene que llegar a un compromiso entre un menor consumo de energía (valores altos de temporizador, con lo que se minimiza el número de veces que se transmite), y un menor retardo en los paquetes (valores bajos del temporizador, pero que aumentan el número de veces que se transmite y por tanto el consumo).

Ensamblado mejorando (*Enhanced Burst Assembly*)

Nakkeeran *et al.* [68] proponen un método adaptativo en el que se escoge el tamaño de la ráfaga en función a una predicción del tráfico basada en la observación de la variación de la tasa de llegada de paquetes. En este algoritmo se establece un tamaño mínimo de ráfaga. Cuando se supera dicho tamaño mínimo, se escoge el tamaño de la ráfaga y el *offset* en el que se va a enviar la ráfaga. Si el tráfico es muy intenso y llegan más paquetes, estos se dejan para otra ráfaga. Para realizar el cálculo del tamaño de la ráfaga se propone un algoritmo de predicción. En primer lugar se mide la tasa media de llegada de paquetes de la ráfaga anterior. Se mide la tasa de llegada en la ráfaga actual hasta el momento en el que se supera el umbral de tamaño mínimo. A partir de estos dos valores se hace una media ponderada, y se usa este valor para estimar el número de paquetes que vendrán en un tiempo de *offset* fijo. La gran ventaja de este algoritmo es que permite saber con antelación el tamaño de la ráfaga.

Ensamblado adaptativo según el desbordamiento en los *buffers*

Düser *et al.* [69] proponen una estrategia de ensamblado que escoge dinámicamente el tiempo de ensamblado con el objetivo de lograr un determinado nivel de desbordamiento en el *buffer* de entrada a la red. Durante el proceso de formación de la ráfaga, se mide el tráfico medio y la varianza, que se introducen a un bucle de control donde se calcula, dado un determinado tamaño de *buffer* a la entrada de la red OBS, una estimación del nivel de desbordamiento en dicho *buffer*. Cuando el tráfico crece, el retardo en el *buffer* de entrada a la red aumenta, pero tras unas cuantas iteraciones en el bucle de control, el tiempo de ensamblado se modifica, de forma que el retardo vuelve a estar en los tiempos objetivos que se tenían previamente al incremento del tráfico. Cuando vuelva a decrecer el tráfico, el tiempo de ensamblado volverá a subir. La principal aportación de esta técnica de ensamblado es que se adapta a cualquier tipo de tráfico y consigue mantener el retardo controlado.

Algoritmo de ensamblado adaptativo con la ventana de TCP.

Los algoritmos anteriores solo tienen en cuenta el tráfico anterior en cuanto a número de datos o tasa de los mismos para poder realizar la estimación de qué temporizador es el más conveniente. La tasa de transmisión de un flujo TCP depende del valor de la ventana de transmisión. Ramantas *et al.* [70] proponen un algoritmo de ensamblado adaptativo en el que se asignan distintos valores del temporizador de retransmisión a los flujos según la ventana de congestión del flujo TCP. De esta forma, si la ventana es pequeña, se sabe que el número de segmentos que quedan por transmitir es bajo, con lo que el valor del temporizador será asimismo bajo. En cambio, si la ventana es grande, se sabe que el flujo transmitirá más segmentos, con lo que el valor del temporizador se puede alargar para intentar transmitir el mayor número de segmentos de dicho flujo.

2.5.6 Predicción del tamaño de la ráfaga

Cuando se emplea un algoritmo basado en temporizador, el tiempo de finalización de la ráfaga es conocido, no así su tamaño, que sólo se conoce una vez vencido el temporizador. Poder conocer con anterioridad el tamaño de la ráfaga, permite adelantar la reserva del canal para su transmisión. Hashiguchi *et al.* [73] proponen un método para estimar el tamaño de la ráfaga. La estrategia consiste en establecer un tiempo fijo desde el inicio de formación de la ráfaga durante el cual se realiza la estimación del tamaño de la ráfaga. Una vez hecha la predicción, se procede al envío del paquete de control. Nada más terminar la formación de la ráfaga, esta puede enviarse.

En el caso de que la predicción sea errónea, si la predicción es superior al tamaño real de la ráfaga, se desperdiciará la ocupación del canal, con lo que el rendimiento de la red

OBS es menor. En caso de que la predicción sea inferior, se habrá reservado menos tamaño del necesario, con lo que se perderán parte de los paquetes de la ráfaga.

Como se ha comentado anteriormente, Nakkeearn *et al.* [68] proponen no sólo estimar el tamaño, sino realizar una predicción para estimar el tamaño en un temporizador y fijarlo. De esta forma, si una vez que se tiene que enviar la ráfaga no se ha llegado al tamaño estimado, se desperdiciará igualmente la ocupación del canal. Sin embargo, nunca se superará el tamaño estimado, ya que si se llega a este tamaño, se empieza a crear una nueva ráfaga.

2.6 Estrategias de señalización

Las diferencias principales en OBS vienen de las distintas aproximaciones para realizar el proceso de señalización mediante el cual se reserva el camino óptico por el que se va a transmitir la ráfaga. Como se mencionó a la hora de describir la arquitectura y el comportamiento general de OBS, este proceso consiste en el envío de mensajes de señalización a cada uno de los nodos intermedios de la ruta para que estos se configuren de tal forma que, en un instante de tiempo determinado (por ejemplo, el instante en el que llegue la ráfaga), y por un determinado intervalo de tiempo (por ejemplo, la duración de la ráfaga), la matriz de conmutación del nodo esté programada para realizar la conmutación deseada (por ejemplo, fibra y longitud de onda de entrada, a fibra y longitud de onda de salida).

A grandes rasgos, hay dos aproximaciones principales en OBS, una en la que se realiza primero una reserva del camino, y una vez esté confirmado el mismo, se puede proceder al envío de los datos (ráfaga), y otra en la que se envían los datos sin esperar a recibir una confirmación de la reserva del camino óptico.

Düser y Bayvel [53] proponen la variante de OBS denominada *Wavelength Routed Optical Burst-Switched Network* (WR-OBS), o red de conmutación de ráfagas con encaminamiento por longitud de onda. WR-OBS realiza la suposición de que los nodos de núcleo pueden estar basados en conmutadores de longitud de onda "tradicionales", cuya implementación es factible con componentes del estado del arte. En una WR-OBS, cuando se quiere transmitir una ráfaga, se solicita la reserva de una longitud de onda, y hasta que no se reciba una confirmación de que se ha realizado con éxito la reserva de la longitud de onda, no se puede proceder a enviar la ráfaga.

La señalización del camino óptico puede ser a su vez totalmente distribuida, en la que la reserva se realiza nodo a nodo, o bien realizarse de una manera parcialmente centralizada. En este caso, será un elemento central, con conocimiento de toda la red, el encargado de enviar las solicitudes de reserva de recursos a cada uno de los nodos involucrados. Este elemento, debido al conocimiento de las reservas en todos los nodos, presenta un potencial mayor para obtener una mayor eficiencia en las reservas. En la aproximación centralizada, el proceso de reserva ha de finalizar informando a cada nodo particular de la decisión de la misma, ya que son cada uno de los nodos particulares los han de realizar la configuración física necesaria.

La aproximación de la reserva confirmada puede implementarse, por ejemplo, mediante un plano de control basado en GMPLS (*Generalized Multi-Protocol Label Switching*), cuya arquitectura está descrita en la RFC 3945 [74], aunque se sugiere al lector acudir al libro de Farrel y Bryskin para profundizar en el conocimiento de GMPLS [75]. En la tesis, se va a emplear uno de los elementos de la arquitectura de GMPLS, el PCE (*Path Computation Element*) [10], para WR-OBS. Por un lado, se proponen las extensiones protocolares necesarias, y por otro lado se implementa para poder estudiar su viabilidad.

La confirmación de la reserva puede llevar un tiempo excesivo, y aumentar el retardo de los paquetes de la ráfaga. Por ello, las estrategias de señalización en las que se espera a la confirmación de la reserva son la aproximación favorita para redes de tamaño pequeño/medio, como las redes metropolitanas, en las que el retardo no es un factor clave [76].

Para reducir este retardo, se plantea como alternativa la señalización del camino óptico sin confirmación, que permite el envío de la ráfaga sin tener que esperar a la confirmación de si el camino podrá ser establecido o no. Empleando señalización sin confirmación, se corre el riesgo de que la ráfaga que se envíe no llegue nunca a su destino si la reserva no se pudo realizar con éxito, pero por otro lado, reduce drásticamente el retardo de los paquetes.

La señalización con reserva no confirmada se conoce en la literatura como "*one-way reservation*", y es la aproximación con más aceptación dentro de la comunidad científica, principalmente para redes de gran tamaño. Se han propuesto en la literatura varios esquemas, como el "*Tell and go*" (TAG), empleados anteriormente para ATM [77], *Just in Time* (JIT) [78] y *Just Enough Time* (JET) [6].

La señalización para la reserva del camino está íntimamente relacionada con cómo se decide cuál es el conjunto de nodos (ruta) del camino en cuestión. Aunque en la literatura se estudia el encaminamiento y las estrategias de señalización por separado, ambas están muy relacionadas, ya que según el mecanismo específico que se emplee para la señalización se puede necesitar información de la ruta en un determinado instante de tiempo, que no es posible obtener en todos las técnicas de encaminamiento, lo que obliga a realizar modificaciones en los nodos, o bien se puede reducir el rendimiento. Por tanto, mientras se estudian las distintas técnicas de reserva, se realizaran unas pequeñas consideraciones sobre el proceso de encaminamiento cuando se emplee esa técnica.

En la Tabla 2-5 se muestra una tabla resumen de las técnicas mencionadas, que se describen con más detalle en lo que resta de sección.

Tipo de reserva	Técnica	Referencias
Con confirmación	WR-OBS	[53]
	GMPLS	[75]
	Tell and Wait	[79]
Sin confirmación	*Tell and Go*	[6]
	JET	[6]
	JIT	[78]
	E-JIT	[80]
	Offset-time emulated	[56]

Tabla 2-5 Alternativas de OBS para reserva del camino

2.6.1 WR-OBS

Düser y Bayvel proponen una arquitectura de OBS denominada *Wavelength Routed Optical Burst-Switched Network* (WR-OBS) [53], o red de conmutación de ráfagas con encaminamiento por longitud de onda.

En una red WR-OBS, como ya se ha comentado, una vez que se decide que datos suficientes para enviar una ráfaga, se procede a realizar la reserva del camino. Cuando se ha recibido la confirmación, se envía la ráfaga y posteriormente se libera el camino. El tiempo total de establecimiento y las garantías de retardo se han estudiado con detalle por de Miguel *et al.* [61] [81].

Se puede emplear un modelo centralizado, en el que la petición del camino se envía a un nodo central, que cuenta con la ventaja de conocer todo el estado de la red, por lo que evitará situaciones de bloqueo. Una vez ha calculado la ruta del camino, informa a cada uno de los nodos involucrados de la reserva, que procederán a realizar la misma. Otra aproximación es realizar el cálculo y reserva de manera completamente distribuida. El principal problema en este caso, es que el tiempo de establecimiento puede alargarse debido a los bloqueos en los establecimientos.

La gran ventaja de esta técnica es que sus requisitos tecnológicos son factibles a corto/medio plazo, requiriendo únicamente de conmutadores suficientemente rápidos. Su desventaja está en que, debido a la necesidad de la confirmación de la reserva, el tiempo que hay que esperar antes de poder transmitir una ráfaga depende de la longitud del camino. Por tanto, para redes de entorno regional no supone un problema, pero para redes troncales con grandes distancias, los retardos introducidos pueden ser excesivos para ciertas aplicaciones. De hecho, un estudio de Zapata *et al.* llega a la conclusión de que WR-OBS es la mejor alternativa para redes metropolitanas, por encima de otros tipos de OBS [76]. Un estudio posterior de Zapata y Bayvel analiza las ventajas del empleo de WR-OBS en distintas redes troncales, determinando que su uso deriva en ahorros de longitud de onda comparado con las redes de conmutación de longitud de onda tradicionales [82].

2.6.2 GMPLS

Una posible forma de implementar la señalización necesaria en OBS, y en particular WR-OBS, es empleando los conceptos y protocolos del plano de control GMPLS definido por el IETF [75]. En GMPLS se reserva un LSP (*Label Switched Path*), que es una asociación lógica de un conjunto de conexiones físicas, bien sean conexiones en una matriz de conmutación óptica, o *slots* de tiempo en una matriz de conmutación TDM, que permiten enviar datos a través de ese conjunto de conexiones. El tipo de conexión en concreto dependerá de la tecnología, y en GMPLS se conoce como tipo de conmutación (*switching type*). En el caso de OBS con confirmación, se puede trabajar con LSPs del tipo de conmutación *lambda* (es decir, longitud de onda).

Mediante el protocolo RSVP-TE, especificado en [83], se puede realizar la señalización para la reserva del camino óptico empleando las extensiones para GMPLS, definidas en [54]. Para enviar los mensajes de señalización de dicho protocolo es necesario emplear un canal separado. Esta red de señalización es una red IP convencional, por lo que cada nodo OBS tendrá una dirección de control para poder enviar la señalización. Los mensajes de RSVP-TE consisten en un mensaje PATH para realizar la reserva, y en un mensaje RESV para confirmar la misma. A la hora de hacer la reserva, se puede especificar en los mensajes de señalización la ruta concreta, e incluso la longitud de onda concreta.

En el caso de OBS, a la hora de establecer un camino, hay que calcular la ruta y longitud de onda específicas. Para ello se puede emplear el PCE (*Path Computation Element*) [10], un elemento funcional definido por el IETF especializado en realizar cálculos complejos de rutas, que puede estar embebido en los nodos, o bien en un elemento separado. Cuando se desea establecer un camino óptico, en primer lugar se consulta al PCE, que devuelve en su respuesta un tipo de objeto especial, denominado ERO (*Explicit Route Object*) [11], que contiene la ruta del camino, y que se puede incluir directamente en los mensajes de señalización de RSVP-TE.

Como se ha mencionado, una de las posibilidades de WR-OBS es la de disponer un elemento central para el cálculo de las rutas de las ráfagas. De esta forma, el PCE se estila como un elemento adecuado para implementar esta funcionalidad. Sin embargo, el PCE no se ha particularizado aún para OBS, por lo que en la tesis se definirán las extensiones, tanto

arquitecturales como de protocolo, necesarias para implementar WR-OBS con un *Path Computation Element*.

El empleo del protocolo RSVP-TE en OBS es adecuado siempre y cuando se estén reservando ráfagas de un tamaño considerablemente grande (como mínimo en el orden de varias decenas de milisegundos) [25], ya que para reservas de menor tiempo, la señalización de GMPLS no es suficientemente rápida, y hará falta recurrir a otros mecanismos específicos.

La gran ventaja del empleo de GMPLS es que una gran parte de los mensajes de señalización necesarios están definidos y estandarizados, a diferencia del resto de propuestas, que no especifican los detalles de los mensajes de protocolo.

2.6.3 *Tell and wait* (TAW)

El esquema *Tell and Wait (TAW)* [79] es una variante de un esquema de reserva de ráfaga de ATM denominado *ATM block transfer scheme* (ABT) [84] [85]. Se trata de un mecanismo de reserva de ráfaga en dos sentidos. Antes de transmitir la ráfaga, se envía un paquete de control de origen a destino para reservar los recursos en el camino. En cada nodo, se reservan los recursos si están disponibles. Si en el camino completo están disponibles los recursos, entonces se envía un mensaje de asentimiento (ACK) desde el destino al origen, en el sentido inverso del camino. En el caso de que algún recurso no esté disponible, se envía un mensaje de asentimiento negativo (NACK) desde el nodo en el que falló la asignación de recursos al origen. Cuando llega el ACK al origen, éste transmite la ráfaga. Cuando se termina de transmitir la ráfaga se liberan los recursos.

2.6.4 *Tell and Go* (TAG)

El esquema *Tell and Go* (TAG) se inspira en una variante del protocolo de reserva rápida en ATM denominado *"fast reservation protocol"* (FRPP) o "*ATM block transfer with Immediate Transmission*" (ABT-IT) [6]. En este mecanismo, el nodo origen envía las ráfagas sin realizar ninguna reserva previa, guardándose una copia de la ráfaga en el nodo fuente, hasta que el destino confirme la recepción de la ráfaga. En los nodos intermedios la ráfaga ha de almacenarse para que se pueda procesar el paquete de control y configurarse el nodo. Por tanto, hace falta que el nodo esté equipado con algún tipo de *buffer* (por ejemplo FDLs) para almacenar temporalmente las ráfagas entrantes. En el caso de que la reserva de recursos falle en algún nodo intermedio, se envía un mensaje de confirmación negativa (NACK) al nodo origen. Entonces, el nodo origen procederá un tiempo después a la retransmisión de la ráfaga. Si la ráfaga ha llegado con éxito al destino, este envía un mensaje de confirmación positiva (ACK).

2.6.5 *Just in Time* (JIT)

En el protocolo JIT [78] [86] se hace una reserva previa de los recursos, de forma que los nodos intermedios serán configurados (se realizan físicamente las conexiones internas en el nodo) en cuanto llegue el paquete de control, que se envía antes de que se transmita la ráfaga. De hecho el protocolo se denomina *Just in time*, es decir, justo a tiempo, ya que al llegar al último nodo, los recursos se han reservado (y terminado de configurar) en el último instante antes de que llegue la ráfaga.

En el protocolo JIT, hay dos mensajes, uno denominado *SETUP*, que indica que se va a transmitir la ráfaga, y uno denominado *RELEASE*, que indica que se ha terminado la ráfaga y hay que liberar los recursos. Una vez ha finalizado la creación de la ráfaga a transmitir, se envía desde el nodo origen un paquete de control con el mensaje de *SETUP*. Este paquete de control se envía al siguiente nodo del camino, que tras recibirlo, pone en marcha la configuración del nodo, y lo reenvía al nodo siguiente en el camino. Por tanto, la

configuración (realización de las conexiones físicas en el conmutador óptico) se realiza en cuanto se procesa el paquete de control. Un tiempo de *offset* después del envío del paquete de control, se procede al envío de la ráfaga. Este *offset* es el suficiente para que de tiempo a que se reconfiguren todos los nodos del camino antes de que llegue la ráfaga a cada uno de ellos. El canal (longitud de onda) permanece reservado desde que llega el paquete de control con el mensaje SETUP hasta llega el siguiente paquete de control con el mensaje *RELEASE* del nodo origen indicando que la ráfaga se ha terminado y que se pueden liberar los recursos. Por tanto, JIT es un protocolo que funciona con reserva inmediata y liberación explícita. En el caso de que cuando llegue un mensaje de SETUP a un nodo la configuración de éste no pueda realizarse con éxito por la falta de recursos libres, el mensaje de SETUP no se reenviará, y cuando llegue la ráfaga al nodo, ésta se descartará, y por tanto se perderán todos los datos en su interior.

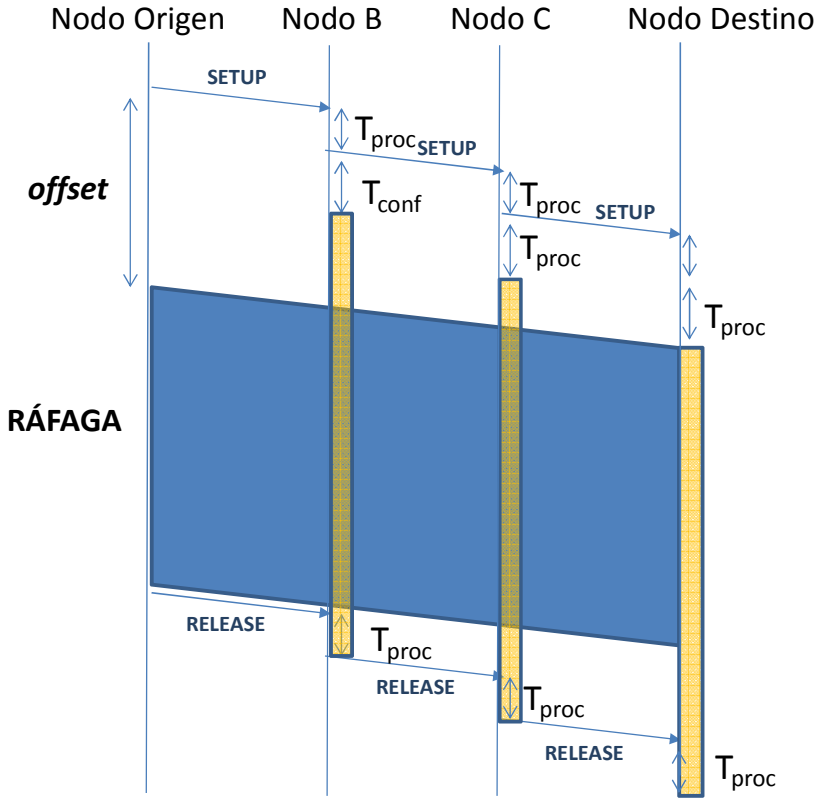


Figura 2-3 Funcionamiento de JIT

El funcionamiento puede verse en la Figura 2-3. Cuando llega el mensaje SETUP a los nodos intermedios, tarde en procesarse un tiempo T_{proc}. Una vez procesado, se envía un nuevo mensaje SETUP al siguiente nodo del camino. Asimismo, en cuanto se procesa el SETUP, se procede a configurar el nodo, que lleva un tiempo T_{conf}. Este tiempo incluye, por ejemplo, el tiempo en realizar la conmutación física en la matriz de conmutación. Se puede observar en la Figura 2-3 todo el tiempo que permanece el canal del nodo reservado. El principal problema de este mecanismo es que la reserva se realiza durante más tiempo que la duración de la ráfaga, por lo que la eficiencia de uso del canal es baja.

Rodrigues y Freire han propuesto una versión mejorada de JIT, denominada *Enhanced* JIT (E-JIT) [80]. E-JIT realiza una estimación de la duración de la ráfaga para liberar los recursos. En el mensaje de SETUP de E-JIT, se envía la información de la estimación de la longitud de la ráfaga y el *offset*. Al igual que en JIT, el canal se reserva en cuanto llega el paquete de control, pero se libera antes, sin esperar a que llegue el mensaje de liberación (RELEASE). De esta forma, el canal permanece reservado menos tiempo, por lo que E-JIT es más eficiente que JIT, y ofrece un mejor comportamiento en cuanto a probabilidad

de bloqueo. El principal inconveniente de esta técnica es que si falla la estimación de la longitud de la ráfaga, o bien se reserva menos tiempo del necesario, con lo que se perderán datos, o bien se reserva más tiempo del necesario, con lo que se reduce la eficiencia.

2.6.6 *Just-Enough Time* (JET)

El protocolo más popular de reserva sin confirmación se denomina JET (*Just-Enough Time*) [6]. El protocolo JET se basa en que, antes de la transmisión de la ráfaga, el nodo frontera origen envía un paquete de control, que se denomina BCP (*Burst Control Packet*) a los nodos troncales, indicándoles que se va a proceder a transmisión de una ráfaga, así como el momento en el que se va a producir y la longitud esperada de la misma. Un cierto tiempo después (denominado tiempo de *offset*), se envía la ráfaga. Durante ese tiempo de *offset*, los nodos intermedios tienen tiempo para reservar recursos para el transporte de la ráfaga que se va a transmitir, pero la reserva se realiza únicamente para el intervalo de tiempo en que la ráfaga pasará por el nodo.

El funcionamiento detallado del protocolo JET puede verse en la Figura 2-4. Cuando llega el mensaje BCP a los nodos intermedios, éste se procesa, pero no se procede a realizar la reconfiguración del conmutador del nodo inmediatamente. En el mensaje se incluye la información del *offset*, que es el intervalo de tiempo en el que va a llegar la ráfaga. Por tanto, se reserva únicamente el intervalo de tiempo durante el cual va a estar transmitiéndose la ráfaga en ese nodo, y cuando llegue el momento, el nodo ha de proceder a realizar las acciones previstas. Por tanto, el empleo de JET obliga al nodo a tener un mecanismo para planificar las configuraciones del mismo. Al igual que en el caso del JIT, hay que tener en cuenta el tiempo de configuración del equipo, que dependerá de la velocidad de los componentes del nodo. Este tiempo, que a veces es denominado tiempo de guarda, impide el aprovechamiento al 100% del canal.

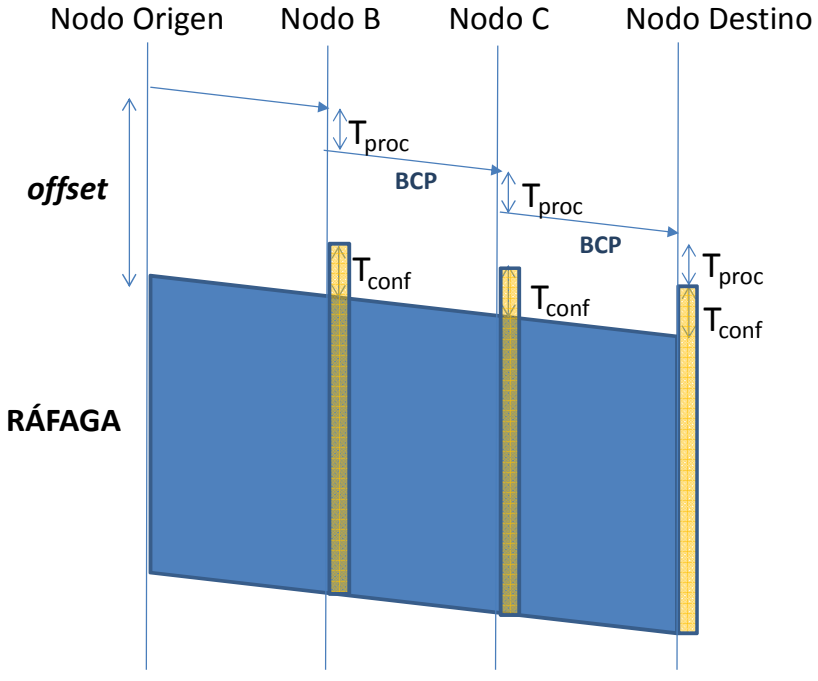


Figura 2-4 Funcionamiento de JET

A diferencia de JIT, no hace falta un mensaje de liberación explícita de los recursos, ya que en el mensaje BCP se envía la información de la longitud de la ráfaga. Por tanto, JET es un protocolo de reserva retardada y liberación implícita, a diferencia de JIT en el que la reserva es inmediata y la liberación explícita.

2.6.7 Tiempo de *offset* emulado

En los protocolos tradicionales de reserva de ráfaga sin confirmación (JET, JIT) mencionados, es necesario conocer a priori el tiempo del *offset*. Este hecho obliga a conocer la ruta de antemano, o bien, si se quiere calcular la ruta de la ráfaga sobre la marcha para un encaminamiento más avanzado, hay que proporcionar un *offset* extra, incrementando el retardo de los paquetes. Por ello, Klinkowski *et al.* [56] proponen la técnica denominada E-OBS, (*offset-time emulated OBS*), en la que el nodo frontera envía el paquete de control y la ráfaga de manera consecutiva, sin esperar un tiempo de *offset*. Para dar tiempo a procesar el paquete de control y configurar el equipo, se introduce un retardo salto a salto mediante un FDL.

2.7 Encaminamiento

El encaminamiento en OBS es el proceso por el cual se decide el conjunto de enlaces y nodos que empleará una ráfaga de datos para ir desde el nodo origen al nodo destino. Por tanto, el encaminamiento está estrechamente ligado al proceso de reserva del camino (de hecho, puede considerarse una parte clave del mismo). Dentro del proceso de encaminamiento hay dos tareas claramente diferenciadas. Una de ellas es el proceso de obtención de la información topológica y de ingeniería de tráfico (por ejemplo, ocupación de los enlaces), y otra es el proceso por el cual, una vez conocido el estado de la red, se obtiene la ruta del camino óptico.

La obtención de la información topológica puede realizarse mediante protocolos tipo IGP (*Interior Gateway Protocol*), como por ejemplo *Routing Information Protocol* (RIP) [87], *Open Shortest Path First* (OSPF) [88] o *Intermediate System-Intermediate System* IS-IS [89]. Si además de la información puramente topológica quiere emplearse en el proceso de encaminamiento información de ingeniería de tráfico, por ejemplo el grado de ocupación de los enlaces, pueden emplearse las extensiones de ingeniería de tráfico, como por ejemplo OSPF-TE [90]. Mediante este protocolo, fuera de banda, cada nodo de la red disemina mediante inundación la información relativa a qué nodos son sus vecinos, qué enlaces parten de los nodos y su ocupación. Si además de la información topológica, se quieren incluir información de ingeniería de tráfico de OBS, es necesario realizar extensiones específicas para tener en cuenta las particularidades de OBS. Un punto de partida es el trabajo que está realizando el IETF con las extensiones a OSPF para el soporte al encaminamiento y asignación de longitud de onda redes de conmutación de circuitos ópticos (WSON) [91]. Sin embargo, en el caso de una WR-OBS en las que las peticiones de longitud de onda llegan en intervalos de tiempo muy cortos, el empleo de un protocolo como OSPF-TE se torna inviable, debido al tiempo de actualización de información inherente a dicho protocolo. En ese caso, se propone que el elemento encargado de realizar el cálculo de la ruta asuma de manera temporal que el camino que ha calculado se establecerá con éxito, por lo que para la siguiente ráfaga no se le dará en ningún caso el mismo recurso.

En cuanto al cálculo de la ruta (y en el caso de WR-OBS de la longitud de onda), si se atiende a un criterio de dónde se realiza el proceso de encaminamiento, se puede distinguir entre encaminamiento de fuente, encaminamiento salto a salto o encaminamiento centralizado.

Encaminamiento de fuente

El camino es calculado en su totalidad por el nodo origen, en este caso, por el nodo de ingreso de la red OBS. Este es el tipo de encaminamiento habitual, por ejemplo, en redes orientadas a conexión que emplean MPLS o GMPLS [74].

La principal ventaja del encaminamiento de fuente se debe a que al poder conocer en el nodo de ingreso la ruta completa de la ráfaga, es posible determinar con exactitud el *offset* necesario cuando se emplea el protocolo JET. Por otro lado tiene la desventaja de no poder reaccionar ante problemas puntuales en el camino, por ejemplo las situaciones de contienda en las que dos ráfagas han de salir por el mismo canal en el mismo instante de tiempo, y ofrecer caminos alternativos.

En una WR-OBS, realizar el encaminamiento de fuente tiene el riesgo de aumentar las situaciones de contienda. Cada fuente toma las decisiones de encaminamiento de manera independiente. Por tanto, cuando hay que realizar dos encaminamientos de fuente en nodos diferentes en paralelo, o en intervalos de tiempo tales que la reserva de la primera ráfaga no ha terminado antes de que empiece el encaminamiento de la segunda, cada proceso de encaminamiento de fuente puede escoger tramos comunes en los que no haya longitudes de onda libres para satisfacer a dos conexiones, sino únicamente a una de las dos reservas de ráfaga. En ese caso, una de las reservas se realiza con éxito y la otra reserva falla. Entonces, se tiene que volver a realizar el proceso de encaminamiento de la ráfaga cuya reserva falló y posteriormente realizar un nuevo intento de reserva.

Encaminamiento salto a salto

En este caso no se conoce a priori la ruta del camino, sino que cada nodo toma la decisión independiente de cuál será el siguiente nodo en la ruta de la ráfaga. Éste es el tipo de encaminamiento empleado en redes IP para los datagramas. Asimismo puede verse como un tipo de encaminamiento salto a salto la reserva de un camino mediante RSVP-TE en el que no se especifica la ruta. En este caso, salto a salto se toma la decisión de la ruta del circuito.

La principal ventaja de este mecanismo es la flexibilidad en el encaminamiento, ya que permite tomar decisiones en tiempo real en cada nodo. Como contrapartida, en OBS sin confirmación como JET, se imposibilita el conocer con exactitud a priori el tiempo de *offset*. Por lo tanto, o bien se sobreestima el *offset*, o se emplean esquemas en los que el *offset* se introduce nodo a nodo como *Emulated offset* [56], que se comentó en la sección anterior.

Encaminamiento centralizado

El IETF ha definido el PCE (*Path computation Element*) [10] como un elemento funcional dedicado al específicamente al cálculo de rutas. Las principales ventajas del encaminamiento basado en PCE son la posibilidad de emplear algoritmos complejos, la flexibilidad en la elección del algoritmo y su empleo en redes con múltiples dominios y capas. Gracias a poder realizar cálculos complejos, se pueden realizar cálculos en los que se realice la asignación de longitud de onda y se tengan en cuenta las restricciones físicas de la transmisión. Como desventaja, se introduce un retardo adicional en el cálculo de la ruta por el hecho de consultar un elemento externo.

Pre-cómputo de rutas

Una última opción, empleada en redes MPLS, es calcular las rutas externamente, bien mediante un PCE, bien mediante el sistema de gestión, y asignar a cada flujo (identificado por ejemplo por el par origen destino) una determinada ruta. En el caso de OBS, de la misma manera, todas las ráfagas que se envían entre un mismo par origen-destino seguirían la misma ruta. El pre-cómputo de rutas es especialmente útil para caminos de protección.

Encaminamiento estático vs. dinámico

Por otro lado, se puede clasificar las técnicas de encaminamiento según los parámetros que se emplean a la hora de calcular los caminos. Se denomina encaminamiento estático al empleo de algoritmos en los que los parámetros empleados para el cálculo de rutas se establecen en el momento de operar la red y no cambian en el tiempo. Únicamente, se actualiza la información con el grafo de la red en el caso de caída de enlace o nodo. Por

otro lado, se denomina encaminamiento dinámico al proceso de encaminamiento que emplea parámetros que se actualizan dinámicamente (por ejemplo el coste de los caminos o la carga de los enlaces). Estos parámetros, o bien se diseminan en las extensiones de ingeniería de tráfico, o bien se calculan en los nodos localmente. La gran ventaja de las técnicas dinámicas es que pueden adaptarse al estado de la red, por ejemplo evitar rutas saturadas, o balancear la carga. El principal inconveniente es que un excesivo dinamismo puede provocar fluctuaciones e inestabilidades en la red.

Importancia del encaminamiento

El algoritmo de encaminamiento es muy importante en el comportamiento de la red. Por un lado, un buen encaminamiento puede hacer que el rendimiento global de la red sea mayor. En el caso concreto de una red de conmutación óptica de ráfagas, con una buena distribución de las ráfagas, se pueden evitar sensiblemente las situaciones de contienda entre ráfagas, por lo que ayuda a reducir la probabilidad de bloqueo global. Por otro lado, empleando los recursos disponibles en la red de una manera óptima, se reducen las necesidades de ampliación de capacidad en los equipos y pueden posponerse las inversiones necesarias.

2.7.1 Algoritmos de encaminamiento

La solución típicamente empleada en el encaminamiento es emplear un algoritmo de camino de menor coste. Sin embargo, el principal problema de las técnicas de encaminamiento que escogen un único camino fijo para un par origen destino es que pueden saturar en exceso determinados enlaces, que pueden convertirse en cuellos de botella, mientras que otros enlaces de la red pueden estar desocupados. Para mejorar el comportamiento, se puede calcular dinámicamente el mejor camino para cada par origen destino. En la Tabla 2-6 se presenta un conjunto de técnicas de encaminamiento para OBS representativos, que muestran las distintas filosofías y se describen posteriormente. Para un listado completo de todas las técnicas propuestas en la literatura, se puede acudir a los estudios de Gonzalez *et al.* [92] y Klinkowski *et al.* [93]. Hay que destacar un tipo particular de encaminamiento, el encaminamiento por deflexión, tratado en detalle en la sección 2.8.4 que se emplea cuando hay una situación de contienda en el camino inicialmente escogido, y hay que buscar una alternativa en tiempo real.

Tipo de encaminamiento	Cálculo de ruta	Información	Referencias
Centralizado	Optimizado	Demandas de tráfico	[94] [95] [96]
Centralizado	Heurística	Demandas de tráfico	[97]
Salto a salto	Heurística	Broadcast	[98]
De fuente	Heurística	Broadcast	[99] [100]

Tabla 2-6 Resumen de algoritmos de encaminamiento

Cálculo de rutas optimizado

Una aproximación al encaminamiento es realizar ingeniería de tráfico y, mediante técnicas de optimización, por ejemplo programación lineal, dada una determinada matriz de tráfico, encontrar el conjunto de rutas óptimo para cada par origen destino. En el caso de OBS, la meta es conseguir reducir la probabilidad de bloqueo global [95]. El principal problema de esta aproximación es que la solución es únicamente óptima para el tráfico con el que se han calculado. Si el tráfico cambia o difiere del previsto, hay que recalcular las rutas. Las técnicas de optimización suelen emplear algoritmos de cómputo intensivo y escalar de una manera no lineal con el número de nodos de la topología, por lo que su ejecución puede llevar un tiempo elevado.

Una de las primeras optimizaciones para OBS fue realizada por Hyytia y Nieminen, quienes proponen una formulación MILP (*Mixed Integer Linear Programming*) para el cálculo de las rutas [94]. En dicha formulación, se calcula una ruta y una longitud de onda para cada par origen destino dada una matriz de tráfico, empleando como criterio de optimización la carga total de los enlaces. En cuanto a las particularidades de OBS, la formulación incluye una restricción para que varios pares origen destino puedan emplear la misma longitud de onda un determinado número de veces. Esta es una forma de limitar la contienda en ráfagas, que ocurre cuando las ráfagas que comparten longitud de onda coinciden en el tiempo.

Teng y Rouskas mejoran la formulación, ya que consiguen introducir la probabilidad de bloqueo en la misma [95]. En una red OBS con conversión de longitud de onda, proponen calcular el conjunto de rutas para cada par origen destino que minimiza la probabilidad de bloqueo global. La probabilidad de bloqueo se calcula típicamente empleando la fórmula de *Erlang B*. El problema de dicha fórmula es que no es una ecuación lineal. Para poder introducirla en la formulación, se hace una interpolación lineal por tramos, de forma que se obtiene un conjunto de ecuaciones lineales. En el estudio de Teng y Rouskas, se demuestra que para conseguir la solución óptima hay que repartir las ráfagas que van de un mismo origen a un mismo destino entre varias rutas.

Barradas y Medeiros continúan con la aproximación de la optimización dada una matriz de tráfico y proponen una técnica en dos pasos [96]. En primer lugar, se calcula mediante un ILP (*Integer Linear Programming*) los *k* caminos más cortos con menor solape y, en una segunda fase, se escoge entre ellos, también mediante un ILP, el conjunto de caminos que minimicen un determinado criterio. Se proponen dos criterios. Por un lado, escoger los caminos que minimicen la carga del enlace más cargado de la red (MCL, *maximum congested link*). Por otro lado, el criterio es escoger los caminos que minimicen la probabilidad de bloqueo del camino con mayor probabilidad de bloqueo estimada (MEC, *maximum end to end congested path*).

Técnicas heurísticas para optimizar las rutas estáticas

Para reducir el tiempo de cálculo de los algoritmos optimizados basados en ILP en redes grandes, se proponen heurísticas cuyo comportamiento se acerque al óptimo [95] [97].

Por ejemplo, Teng y Rouskas proponen dividir el problema en dos partes, en primer lugar emplear una optimización basada en LP (*linear programming*) para calcular un conjunto de rutas [95]. La ventaja de este paso es que se trata de programación lineal (y no entera), por lo que se resuelve más rápido, pero hay valores que no indican claramente un resultado, por lo que hay pares origen destino cuya solución oscila entre varias posibilidades. En una segunda parte de la técnica heurística de Teng y Rouskas, entre los pares origen destino con varias posibilidades, se ordenan los caminos por número de saltos, y se prueban varias combinaciones, evaluando la probabilidad de bloqueo de cada solución.

Du *et al.* [97] proponen emplear directamente, en vez de un ILP una modificación del algoritmo de Dijskstra, denominada ALB-Dijkstra (*Adaptive Load Balancing-Dijkstra*), mediante la cual calculan un conjunto de rutas (una por cada par origen destino) que balancean la carga en la red con la que logran reducir la probabilidad de bloqueo global. La mayor ventaja es la rapidez con la que se ejecuta el algoritmo.

Heurísticas adaptativas

Las técnicas anteriores proporcionan un camino óptimo a partir de una determinada distribución de tráfico conocida. Sin embargo, este dato no siempre es posible conocerlo, y, aún en caso de conocerlo, el tráfico real puede diferenciarse del previsto. Por ello, las

técnicas de encaminamiento adaptativas buscan reaccionar ante las distintas situaciones en la red.

Hay algoritmos reactivos, que reaccionan ante situaciones en la red. Por ejemplo, Thodime *et al.* [99] proponen la técnica denominada *Congestion-Based Static-Route Calculation*. Esta técnica realiza encaminamiento de fuente, por lo que el nodo origen de las ráfagas es el único encargado de tomar las decisiones en cuanto a la ruta de la ráfaga. Se calcula de forma estática un conjunto de caminos disjuntos alternativos y se selecciona uno de ellos de forma dinámica en base a la información de congestión recolectada de los nodos del núcleo. En los nodos se toman medidas de carga de manera periódica en sus enlaces de salida. Si se supera un umbral de carga, se indica que ese nodo está congestionado y se disemina esta información. A la hora de escoger el camino, si el camino más corto atraviesa algún enlace congestionado, se escoge el camino alternativo con menor número de tramos congestionados.

Klinkowski *et al.* [98] [92] proponen la técnica *Bypass Path* (BP), en la que se selecciona un camino en el nodo de origen del paquete como función del estado de sus colas de salida, por lo que emplea únicamente información local. La ruta sólo puede modificarse cuando el paquete se encuentra con un enlace saturado. En ese caso, el nodo intenta hacer un "bypass" de ese enlace saturado, utilizando un nodo intermedio para alcanzar al próximo salto.

Klinkowski *et al.* [98] también proponen la técnica *Distributed Path* (DP), con variantes con 1 o varias alternativas (DP-1, DP-*k*). Es una estrategia de encaminamiento adaptativa distribuida que emplea información global para la toma de las decisiones. En esta técnica, cada nodo pre-calcula las *k* rutas más cortas hacia cualquier destino. Posteriormente, el algoritmo elige un camino entre todos ellos en función de un coste compuesto a base de tres parámetros, la media de ráfagas perdidas de cada enlace (ABL, *Average Burst Loss*), la utilización media del enlace (ALU, *Average Link Utilization*) y la ocupación media de *buffer* (ABO, *Average Buffer Occupancy*). Cada LSP de la red tiene diferente coste, calculado como la suma de las contribuciones de cada enlace que forma el camino. Para que los nodos tengan conocimiento de esta información de los demás, cada nodo mide el estado de sus parámetros (ABL, ALU y ABO) en un intervalo de tiempo determinado, y a continuación inunda la red con la información recolectada para que llegue a los demás nodos y estos puedan recalcular el coste de sus caminos.

Problemática de las técnicas de la literatura

Resumiendo, el principal problema de las técnicas descritas es que para conseguir buenos resultados (incluyendo el óptimo) es necesario conocer la demanda exacta y requiere de una continua diseminación de la misma.

Por ello, esta tesis realiza una propuesta de un algoritmo heurístico multicamino que ayude a reducir la probabilidad de bloqueo y que se pueda realizar empleando únicamente información local del nodo. De esta forma, al utilizar información disponible en el propio equipo, se reducen los requisitos de intercambio de información con el resto de la red OBS.

Otro de los problemas de los que adolecen las técnicas de la literatura es la estabilidad. En el caso de que una variación en el patrón de tráfico provoque cambios en el encaminamiento, dichos cambios puede dar lugar a inestabilidades y fluctuaciones. Para evitarlo, la elección del conjunto de rutas a emplear para un par origen-destino tiene un mecanismo de histéresis, por lo que hay unos umbrales cuando se utilizan más rutas y otros cuando se reducen. Así, las fluctuaciones de carga habituales no provocan movimientos de tráfico en la red.

2.8 Técnicas de resolución de contienda

En una red OBS puede darse el caso de que dos o más ráfagas necesiten el mismo recurso en el mismo instante de tiempo. En este caso se produce una contienda. En una red de conmutación de paquetes clásica, cuando dos paquetes necesitan emplear el mismo puerto de salida, mientras uno de ellos se transmite, el otro espera en una cola, hasta que el recurso quede libre y pueda transmitirse. Por tanto, el problema se solventa mediante el almacenamiento electrónico temporal en una memoria, habitualmente denominada *buffer*. Sin embargo, cuando la transmisión se realiza por medios ópticos, el almacenamiento se complica. Actualmente la tecnología de almacenamiento óptico está inmadura, ya que no existe aún el equivalente de una memoria RAM (*Random-Access Memory*) óptica de capacidad similar a las memorias RAM electrónicas actuales [101].

Por tanto, en caso de que se produzca una contienda entre dos o más ráfagas, solamente una de ellas podrá emplear el canal de salida. La aproximación más evidente es emplear prioridades, de tal forma que sean las ráfagas de menor prioridad las que se descarten en caso de contienda. Sin embargo, de esta forma no se soluciona el problema, sino que simplemente se da prioridad a un tráfico frente a otro.

Para resolver las situaciones de contienda se han propuesto varias técnicas, resumidas en la Tabla 2-7. La primera de ellas consiste en emplear las facilidades de la conversión de longitud de onda y realizar una planificación de los canales de salida, logrando un aprovechamiento eficiente de todos los canales. El gran inconveniente es la disponibilidad de elementos que realicen de manera totalmente óptica la conversión, así como el coste adicional que introducirán.

También se puede emplear la posibilidad de almacenar temporalmente las ráfagas en fibras de retardo. Actualmente, no existe el equivalente óptico de la memoria RAM. Sin embargo, se han desarrollado elementos que son capaces de "almacenar" un paquete/ráfaga óptico un tiempo fijo, denominados bucles de fibra óptica (FDL, *Fiber Delay Lines*) y que realmente retrasan la ráfaga un tiempo fijo determinado.

Otra técnica se basa en la segmentación de las ráfagas. Una ráfaga se divide internamente en dos o más segmentos. En caso de contienda, se descarta únicamente el segmento de la ráfaga en el que hay solape temporal. El control de este mecanismo es bastante complejo.

Por último, se puede realizar un encaminamiento por deflexión, es decir, en caso de contienda enviar la ráfaga por un puerto que esté libre, pero diferente al calculado originalmente.

Mecanismo	Referencias
Conversión de longitud de onda	[24] [102] [103] [104]
Almacenamiento	[105] [106], [107]. [108]
Segmentación	[109] [110]
Deflexión	[111] [112] [72] [113] [114] [115].

Tabla 2-7 Mecanismos de resolución de contienda

2.8.1 Algoritmos de planificación con conversión de longitud de onda

En el caso de disponer de **conversión de longitud** de onda en los nodos OBS, se puede planificar una ráfaga en cualquiera de las longitudes de onda disponibles en el puerto de salida escogido. El "planificador de ráfagas" escoge la longitud de onda a utilizar teniendo en cuenta las reservas realizadas hasta ese momento. En la literatura se han

propuesto diversos algoritmos de planificación con distinto nivel de complejidad y rendimiento que se describen en esta sección.

Con el fin de ilustrar las diferencias de comportamiento entre los distintos algoritmos de planificación y ayudar a su comprensión, se ha incluido la Figura 2-5 en la que se representa la planificación de una nueva ráfaga. En dicha figura se muestra la planificación temporal de cada uno de los canales ópticos disponibles, del C1 al C5. Para cada canal, se muestran las ráfagas que hay planificadas, avanzando el tiempo de izquierda a derecha. Mediante la línea de puntos se indica el intervalo de tiempo en el que va a pasar la ráfaga por el enlace. Junto al nombre de cada canal óptico (C1, C2, C3, C4 y C5), se incluye el nombre del algoritmo que, dada la situación de planificación mostrada, escogería dicho canal para planificar la ráfaga.

Horizon [24], también denominado LAUC [102] (*Latest Available Unscheduled Channel*): Para cada longitud de onda se almacena un único valor, denominado *horizonte de planificación*. Este horizonte se define como el instante temporal hasta el cual hay alguna ráfaga planificadas en el canal (canal = longitud de onda). Por lo tanto, los canales disponibles son aquellos cuyo horizonte temporal sea anterior al instante en el que está prevista la llegada de la ráfaga. El algoritmo escoge, entre los canales disponibles el que tenga el horizonte mayor. Una vez escogido el canal, se vuelve a calcular el horizonte de dicho canal.

El principal problema del Horizon/LAUC es el bajo aprovechamiento del canal, ya que el algoritmo genera huecos en la planificación que no es posible utiliza. Para aprovechar los huecos en la planificación se han propuesto técnicas con rellenado de huecos:

LAUC-VF [103] (*Latest Available Unscheduled Channel with Void Filling*): La idea de este algoritmo es minimizar los huecos que se generan durante el proceso de planificación. Para ello, cuando tiene que planificar una nueva ráfaga, el algoritmo busca en primer lugar los canales en los que la ráfaga no se solape con el resto de ráfagas planificadas, que se denominarán canales disponibles. Por tanto, LAUC-VF ha de almacenar la información de inicio y fin de cada una de las ráfagas planificadas. A continuación, entre los canales disponibles, elige aquel en el que el último instante planificado antes del instante previsto de llegada de la ráfaga sea mayor, es decir, aquel que genere un menor hueco entre la ráfaga y la última ráfaga planificada.

Este algoritmo presenta un mayor aprovechamiento de los canales que Horizon/LAUC. Sin embargo, el mayor aprovechamiento sólo es significativo cuando se combina con el uso de FDLs, ya que el número de huecos generados aumenta considerablemente [103] No obstante, el número de variables que ha de almacenar es muy superior (dos instantes de tiempo por cada ráfaga planificada), y el número de comprobaciones aumenta, requiriendo un mayor tiempo de procesado.

Min-SV/Min-EV [104]: Xu *et al.* proponen el empleo de otros criterios para seleccionar el canal de una ráfaga. El primer criterio es, dado un *offset*, encontrar el intervalo que minimice el tiempo entre el comienzo de la ráfaga a planificar y el último instante de cada una de las ráfagas ya planificadas. Este algoritmo se conoce como Min-SV (*minimum starting void*), y su objetivo coincide con el de LAUC-VF. Xu *et al.* proponen de manera alternativa minimizar el tiempo entre el fin de la ráfaga a planificar y el comienzo de las posteriores ráfagas planificadas. Este último algoritmo se denomina Min-EV (*minimum ending void*).

Best Fit [104]: Xu *et al.* proponen finalmente planificar la ráfaga en el canal en el que se minimice la suma del tiempo que hay entre el final de la última ráfaga planificada en el canal y el comienzo de la ráfaga a planificar, y el tiempo entre el fin de la ráfaga a planificar y el comienzo de la siguiente ráfaga planificada en dicho canal.

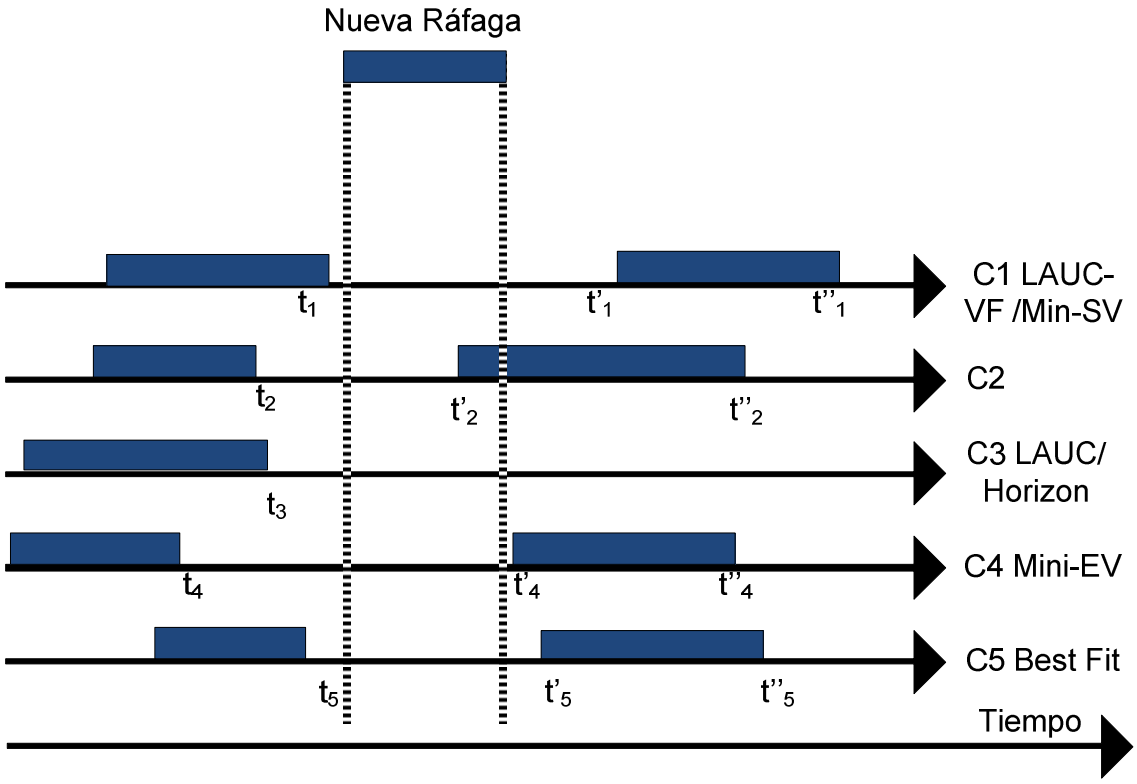


Figura 2-5 Comportamiento algoritmos de planificación

El rendimiento de los algoritmos se compara en términos de probabilidad de bloqueo en un estudio de Xu *et al.* [104]. Por probabilidad de bloqueo se entiende la probabilidad de que una ráfaga no pueda planificarse y por tanto, tenga que ser descartada. En el mencionado estudio, Min-SV, Min-EV y *Best Fit* obtienen los mejores resultados en cuanto a probabilidad de bloqueo. Su principal problema radica en su mayor complejidad y cantidad de información a mantener en memoria.

2.8.2 Resolución espacial de contienda (líneas de retardo, *buffers*)

Un componente clave en los *routers* de paquetes electrónicos es un *buffer* donde almacenar los paquetes de manera temporal mientras el puerto de salida está ocupado. Este *buffer* está compuesto de memoria electrónica de acceso aleatorio (RAM). En el caso de un nodo OBS, se puede aplicar el mismo principio para resolver la contienda, y almacenar la ráfaga temporalmente.

Empleo de RAM electrónica

Una primera solución posible es equipar al nodo con convertidores opto-electrónicos y almacenar la ráfaga en una RAM electrónica. En esta solución habría un conjunto de receptores, (convertidores O/E), memoria RAM electrónica (similar a la de cualquier *router IP*), y un conjunto de transmisores (convertidores E/O), de las mismas características que los presentes en los nodos frontera. El gran inconveniente de esta solución es el coste añadido de la conversión optoelectrónica. Otra limitación es la velocidad de escritura en la memoria. Según Takashi *et al.* [116] es posible escribir paquetes ópticos a 40 Gbps, aunque su escalabilidad, es decir, la capacidad para poder escribir a velocidades superiores, está muy limitada por la electrónica.

Para conseguir una solución que no incremente excesivamente el coste, Yang *et al.* [108] proponen emplear los propios receptores y transmisores del nodo OBS junto con una memoria. Yang *et al.* basan su propuesta, denominada *"edge buffer based contention resolution scheme"*, en que la funcionalidad de nodo de núcleo y nodo de ingreso esté en el mismo

equipo. Cuando llega una ráfaga a un nodo OBS, y no puede planificarse correctamente por el canal previsto de salida ya que hay una contienda con otra ráfaga, esta se desvía a uno de los receptores del nodo, siempre y cuando esté libre. Una vez recibida la ráfaga, se almacena en el *buffer* electrónico y se planifica su salida cuando el canal esté libre. Para evitar el bloqueo de ráfagas en el destino, las ráfagas que lleguen a un nodo y este sea su destino final, tienen prioridad sobre las ráfagas que necesitasen emplear el receptor para almacenar la ráfaga En el estudio realizado por Yang *et al.*, se consiguen mejoras significativas en cuanto a probabilidad de bloqueo, especialmente en cargas altas, cuando se compara con una solución de OBS con resolución de contienda por conversión de longitud de onda.

Empleo de RAM totalmente óptica

La solución deseable sería emplear una memoria RAM óptica equivalente, sin necesidad de conversiones opto-electrónicas. En la actualidad, el componente que está más desarrollado es el *buffer* basado en líneas de retardo, que retrasa los paquetes una cantidad fija de tiempo. La búsqueda de una verdadera memoria óptica de acceso totalmente aleatorio está en sus primeros pasos. Por ejemplo, Pleros *et al.* [117] han desarrollado una celda RAM estática de 1 bit a partir de dos conmutadores SOAs y un *flip-flop* óptico SOA-MZI (*SOA-Match Zender Interferometer*). En Japón, desde 2006 a 2011 se ha desarrollado un programa para la búsqueda de una memoria RAM óptica [101]. Kitayama *et al.* [118] en el marco de dicho programa han logrado desarrollar un sistema completo de memoria RAM, a partir de un *array* de celdas de 1 bit construidas mediante una nanocavidad, con una velocidad de acceso de 10 GHz (la velocidad de las memorias se mide actualmente en hertzios, con lo que se obtendría un tiempo de acceso de 1 ps [119]). Para construir las nanocavidades de las celdas de memoria, se han empleado distintos cristales fotónicos, empezándose por una prueba de concepto empleando Si [120], mejorándose drásticamente el consumo energético y el rendimiento empleando InGaAsP [121]. Por tanto, el grado de avance experimentado en los últimos años permite pensar que pueda emplearse RAM óptica en los próximos años.

Empleo de FDLs

La aproximación principal en cuanto a memoria óptica es el empleo de *buffers* basados en líneas de retardo (*Fiber Delay Lines,* FDL) [122]. Se trata de trozos de fibra óptica de longitud fija que incrementan el tiempo de transmisión de los datos ópticos. Para lograr un determinado tiempo total de transmisión, la principal solución es variar la longitud física de la fibra. Otra alternativa es incrementar el tiempo de transmisión reduciendo la velocidad de la luz (en concreto, la velocidad de grupo) en dicha fibra con materiales especiales. Esta solución se conoce como *slow light* [123]. Esta tecnología aún no está madura, y es objeto de continua investigación, ya que potencialmente puede ser una solución de tamaño muy pequeño y que proporciona tiempos de almacenamiento variable, que la hace ideal para paquetes de tamaño variable. Sin embargo, sus principales limitaciones son la dispersión y las pérdidas, que penalizan excesivamente su rendimiento. Por tanto, la alternativa más empleada es la de variar la longitud de la fibra, en la que la señal no sufre tanta degradación. Aunque una de las limitación de esta solución es su tamaño (para conseguir 1 ms de retardo hacen falta 200 km de fibra aproximadamente), se han logrado avances significativos en su miniaturización.

Con el objetivo de reducir el tamaño y aumentar la granularidad, es decir, poder seleccionar distintos retardos, las fibras de retardo se combinan con conmutadores ópticos. De esta forma, las arquitecturas de *buffers* ópticos basadas en FDL y conmutadores se pueden clasificar, a su vez, en dos tipos, *feed-forward* (FF) y realimentadas (*feedback)* (FB)

[107]. Estos *buffers* ópticos pueden estar dedicados a unos puertos específicos o compartirse entre varios puertos.

Además de la clasificación en cuanto a FF y FB, se pueden distinguir dos tipos principales de arquitecturas de FDL, las arquitecturas de etapa única, en las que el retardo de la ráfaga se debe a un único FDL de longitud fija (la longitud determinará el retardo introducido) y las arquitecturas de etapas múltiples, en las que hay un conjunto de FDLs en cascada que se combinan con elementos de conmutación.

Arquitectura de FDL de tipo *feed-forward*

En la arquitectura de *buffer* de tipo FF, los datos entran en la línea de retardo a la salida del nodo. Las arquitecturas *feed-forward* empleadas en nodos OBS se basan en acoplar a un conjunto de canales de salida del conmutador una serie de líneas de retardo, que posteriormente se multiplexarán a la salida del nodo. Por tanto, se necesitan recursos dedicados en la salida de la matriz de conmutación del nodo.

Un ejemplo de nodo de núcleo con una configuración de FDL *feed-forward* es el diseño propuesto por Gauger [107] que se muestra en la Figura 2-6. En la arquitectura propuesta por Gauger, tras la salida de la matriz de conmutación, hay diversas líneas. Una de ellas es la línea directa, en las que no se introduce un retardo adicional y que es el camino habitual de las ráfagas que se planifican sin problemas. El resto son líneas de retardo, cada una de ellas con un valor de retardo diferente.

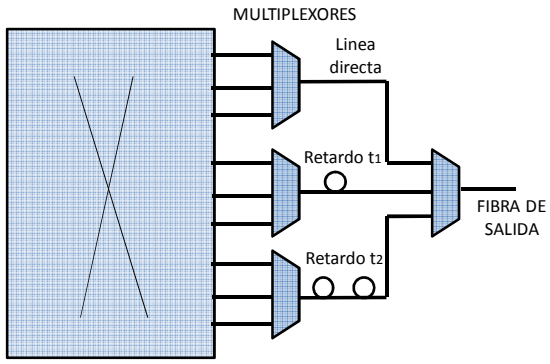


Figura 2-6 Arquitectura de nodo OBS con FDLs de tipo *feed-forward*

Nodo OBS con FDL *feedback*

En las arquitecturas de nodos OBS con FDLs de tipo *feedback*, los datos, tras salir del nodo se retrasan y vuelve a una etapa anterior del nodo, existiendo una realimentación. Un ejemplo de nodo con *feedback* se muestra en la Figura 2-7 [107]. El principal problema de las arquitecturas de FDL con *feedback* son la atenuación adicional y la degradación que sufre de la señal óptica en cada realimentación. Aunque la atenuación se compensa introduciendo etapas de amplificación, la degradación es más complicada de resolver, y es la que limita las recirculaciones posibles.

Beheshti *et al.* [124] reportan un experimento con un *buffer* basado en FDL con realimentación en que se consigue un retardo de hasta 184 ns con un 98% de tasa de recuperación de paquetes. Cabe destacar el prototipo presentado por Burmeister *et al.* [125], en el que se consigue un retardo de 64 ns con 5 recirculaciones, que se ha realizado en un circuito integrado, del tamaño de una moneda.

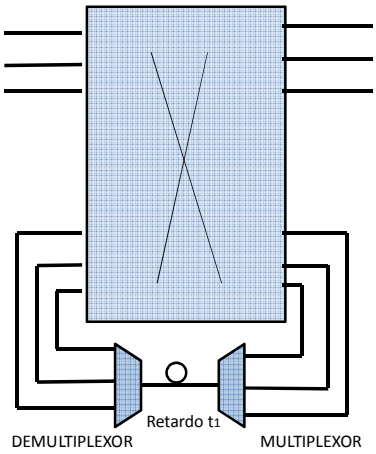


Figura 2-7 Arquitectura de nodo OBS con FDLs de tipo *feedback*

Estrategias de reserva con FDLs

Hay varias alternativas para realizar la reserva de recursos con FDLs [107]. Una primera estrategia se denomina *PreRes* (Reserva Previa). En esta estrategia, asumiendo el empleo se señalización de ráfaga JET, cuando llega el paquete de control BCP y el canal y longitud de onda de salida elegidos están ocupados en el tiempo previsto de llegada de la ráfaga, se realiza una búsqueda de un FDL libre para el instante de llegada de la ráfaga. En caso positivo de encontrar un FDL libre, se comprueba si, en el instante de tiempo resultante de sumar al instante de llegada de la ráfaga el tiempo de retardo, el canal está libre (no está reservado para ninguna otra ráfaga). Si el canal está libre en ese instante, se reservan el FDL y el canal en los instantes de tiempo necesarios. Por tanto, se realiza la reserva del canal de salida antes de enviar la ráfaga al *buffer* de FDL.

En cambio, en el caso de la estrategia *PostRes* (Reserva Posterior), sólo se reserva el FDL. Al igual que en la estrategia anterior, cuando el canal de salida está reservado, se busca el menor FDL disponible, y se reserva. Una vez la ráfaga se envía al FDL, se comprueba si la salida está libre, y en el caso de estarlo, se envía por el canal deseado.

La principal diferencia entre ambas estrategias es que la estrategia *PreRes* otorga una cierta ventaja a las ráfagas que se envían a *buffer*, mientras que se perjudica a otras ráfagas que pueden sufrir bloqueos. Sin embargo, empleando la estrategia *PostRes*, el comportamiento en cuanto a equidad es similar al caso sin FDLs. En este caso, las ráfagas que se han enviado al FDL no interfieren con el resto de ráfagas "normales", ya que sólo ocupan el canal de salida si este está libre, es decir, no se han aprovechado de la ventaja de reservar con mucho tiempo de antelación.

2.8.3 Segmentación de ráfagas

La segmentación de ráfagas fue propuesta por Vokkarane *et al.* [109] para reducir las pérdidas de ráfagas. En esta técnica, la ráfaga de datos se divide en múltiples segmentos, que pueden contener uno o varios paquetes. Empleándose en combinación con otras técnicas, como por ejemplo el encaminamiento por deflexión, Vokkarane *et al.* [109] demostraron que la segmentación de ráfagas reduce la probabilidad de pérdida de paquetes (no confundir con la de ráfagas) en comparación con la técnica de descartar la ráfaga completa. Los mismos autores proponen dos mecanismos para resolver la contienda de dos ráfagas con segmentación:

Segment-first. Se realiza el supuesto de que hay una ráfaga ya planificada y llega la petición de una nueva ráfaga que se solapa con la anterior. En este caso, se compara el número de segmentos que quedarían por transmitir de la ráfaga original cuando lleguese la nueva ráfaga con la longitud de la ráfaga nueva. Si la longitud de la ráfaga nueva es menor que la

cantidad anterior, se envía la ráfaga nueva por un puerto libre (deflexión), manteniéndose la planificación de la ráfaga original. En caso contrario, se segmenta la ráfaga original, y los segmentos que se solapan se envían por un puerto alternativo, o se descartan si no hay ninguna salida disponible.

Deflection-first. Si hay puertos libres para enviar la nueva ráfaga, se realiza en primer lugar un encaminamiento por deflexión de la ráfaga. En caso contrario, se sigue un proceso similar al *segment first*, y se comparan los segmentos a descartar de las dos ráfagas. El más corto es el que se descarta (o se encamina por un puerto alternativo si lo hay).

Sin embargo, la implementación de las técnicas de segmentación no es trivial [110]. Hay unos cuantos aspectos que hay que considerar para determinar la eficiencia y viabilidad:

Tiempo de conmutación: Es el tiempo necesario para configurar el conmutador. Este tiempo depende del tipo de conmutador (MEMs, LCOS, etc.), y puede variar de un nodo a otro. Entre dos ráfagas o segmentos hay que dejar suficiente tiempo para reconfigurar el conmutador.

Tamaño de las ráfagas: En el proceso de ensamblado, se indicó que era conveniente que las ráfagas tuvieran un tamaño mínimo, para evitar saturar el canal de control con señalización de ráfagas prácticamente vacías. Lo mismo ocurre con las ráfagas truncadas, es decir, aquellas ráfagas que han perdido parte o los segmentos que se han desprendido de la ráfaga original. Hay que evitar un particionamiento excesivo y transmitir muchas ráfagas con segmentos pequeños.

Actualización del paquete de control: En caso de truncar una ráfaga en varios segmentos, hay que generar nuevos paquetes de control, que contengan la información del nuevo tamaño de las ráfagas.

FDLs: Habrá que emplear FDLs para retrasar las ráfagas tanto como tarden en actualizarse los paquetes de control.

2.8.4 Encaminamiento por deflexión

El encaminamiento por deflexión (*Deflection routing*) es una técnica para resolver la contienda cuando dos ráfagas necesitan emplear el mismo canal de salida. El encaminamiento por deflexión es una técnica conocida desde hace mucho tiempo para otras tecnologías, como se puede comprobar en los estudios de Bannister *et al.* [126] y Bononi *et al.* [127]. Esta técnica puede emplearse tanto en arquitecturas sin *buffer* (*bufferless*) como en arquitecturas con memoria o *FDLs*. La forma más simple de encaminamiento por deflexión se conoce como "encaminamiento de la patata caliente" (*hot potato routing*) y consiste enviar el paquete o ráfaga por el puerto libre que haya en ese mismo instante. Si se combina con pequeño número de *buffers*, el rendimiento mejora sustancialmente.

Problemas del encaminamiento por deflexión

El encaminamiento por deflexión en OBS-JET tiene un problema muy importante, el tiempo de *offset* insuficiente [128]. Cuando se emplea OBS-JET, la ráfaga se envía un tiempo de *offset* después del envío del paquete de control. Este tiempo de *offset* está ajustado para un camino determinado. Si este camino se cambia en tiempo real, como puede ocurrir cuando la salida prevista está ocupada y es necesario emplear otro camino alternativo, el nuevo camino alternativo puede tener un número de saltos diferente al original, con lo que el tiempo de *offset* puede no ser suficiente y la ráfaga llegaría al destino antes que el paquete de control. Hay varias propuestas para solucionar este problema:

1. **Tiempo de *offset* adicional**

Debido al hecho de que si no hay suficiente *offset*, la deflexión no puede realizarse con éxito, una posibilidad es incrementar el tiempo de *offset* al inicio. De esta manera, hay

margen para estar cubierto ante posibles situaciones de deflexión. Sin embargo, el incrementar el *offset* al inicio puede afectar a otras ráfagas.

2. **Retrasar la ráfaga en el nodo congestionado**

Si la ráfaga que se tiene que enviar por un camino alternativo no tiene suficiente *offset*, hay que enviar la ráfaga a un FDL del propio nodo, donde se retrasará lo suficiente para poder incrementar el tiempo de *offset*. El principal problema de esta solución es su implementación práctica, ya que depende de la arquitectura del nodo y cómo estén conectados los FDLs (si están asociados a puertos de entrada o salida).

3. **Retrasar la ráfaga en el nodo siguiente**

Otra solución es, ya que va a quedar al menos un salto hasta el destino, es enviar la ráfaga a un FDL en el nodo siguiente. Esta aproximación tiene una implementación más fácil, ya que no tiene dependencia de cómo estén conectados los FDLs.

Por otro lado, aunque el encaminamiento por deflexión salva temporalmente una situación de contienda tiene varios efectos secundarios. Por un lado, se incrementa el retardo extremo a extremo en los paquetes de las ráfagas que se han encaminado por deflexión. Por otro, puede producir el adelantamiento de ráfagas, por lo que los segmentos llegarán en desorden, provocando una bajada de rendimiento en protocolos de capas superiores, como TCP. Por último, un encaminamiento por deflexión excesivo aumenta la carga global en la red, con lo que la probabilidad de bloqueo global se incrementa, en vez de disminuir.

Algoritmos de encaminamiento por deflexión

Hay diversas propuestas para mejorar el encaminamiento por deflexión, resumidas en la Tabla 2-8, que se explicarán a continuación. Las técnicas de deflexión varían principalmente en la forma de calcular y escoger el camino de deflexión [111]. La aproximación más simple es calcular los caminos más cortos, y escoger el camino libre más corto. A partir de ahí, se pueden aplicar técnicas más complejas y calcular dinámicamente un conjunto de caminos con más criterios, como la carga o una estimación de la probabilidad de contienda. Una vez calculadas las posibles rutas de deflexión, la elección puede ser tan simple como elegir la más corta de las calculadas, elegirlas al azar o distribuir las ráfagas entre varias de las rutas.

Técnica	Tipo camino de deflexión	Referencias
SP-DR	Estático	[72]
SP-PRDR	Estático con prioridades	[113]
CLDR	Optimizado periódicamente con ILP	[114]
LLN	Decisión según la carga de los enlaces (medida cada 10ms).	[115]
U-TURN	Ida y vuela a un nodo vecino	[115]
Balanceo de carga	Reparto de ráfagas entre múltiples caminos	[115]
DR- probabilidad de contienda	Decisión en función de la estimación de contienda de cada ruta de deflexión	[129]
RLDRS	Dinámico basado en aprendizaje reforzado	[112]

Tabla 2-8 Técnicas de encaminamiento por deflexión

Encaminamiento por deflexión al camino más corto (SP-DR)

El método natural cuando se realiza un encaminamiento por deflexión es escoger como camino alternativo el siguiente camino más corto al destino. En la literatura se conoce

como "*Shortest Path Deflection Routing*"(SP-DR) La principal ventaja de esta técnica, a parte de su sencillez, es que, al minimizarse el número de saltos hacia el destino, se minimiza el número de potenciales puntos de bloqueo de la ráfaga.

Zalesky *et al.* [72] realizaron un estudio del rendimiento de una red OBS con deflexión por el camino más corto. Los resultados de dicho estudio muestran una reducción significativa de la probabilidad de bloqueo, especialmente en cargas de tráfico bajas. Comparando con otras técnicas de resolución de contienda, por ejemplo, la reducción de probabilidad de bloqueo es ligeramente mejor a la que se obtiene introduciendo un convertidor de longitud de onda, solución que necesita de un hardware adicional. Por tanto, acorde al citado estudio, se demuestra que, por lo menos a cargas bajas y fijándonos en la probabilidad de bloqueo, es una alternativa válida.

Encaminamiento por deflexión priorizado (SP-PRDR)

Cameron *et al.* [113] proponen la técnica de deflexión denominada "*Shortest Path Prioritized Random Deflection Routing"* (SP-PRDR). Esta técnica se basa en que, en caso de contienda, se envía la ráfaga a un vecino aleatoriamente. A la ráfaga desviada se le asigna una prioridad baja. Al resto de ráfagas (las que no se han desviado) se les asigna una prioridad alta. De esta forma, en caso de contienda, siempre vence la ráfaga que no ha sufrido deflexión sobre la que ha sido desviada. Con esta técnica, se minimiza el impacto de la deflexión en el resto del tráfico de la red, ya que la ráfaga desviada solo llega al destino si no afecta al resto de ráfagas.

SP-PRDR y SP-DR se consideran las técnicas 'clásicas' de encaminamiento por deflexión en OBS. El resto de las técnicas, explicadas a continuación, se centran en mejorar el proceso de deflexión para reducir la probabilidad de bloqueo.

***Contention-based Limited Deflection Routing* (CLDR)**

Lee *et al.* [114].proponen un mecanismo de deflexión denominado *Contention-based Limited Deflection Routing* (CLDR) Este mecanismo se fundamente en dos pilares. Por un lado, una técnica para decidir en tiempo real, en caso de contienda, cuándo conviene emplear deflexión, y cuando conviene descartar la ráfaga y esperar a que esta se retransmita. Por otro lado, propone para el caso de deflexión, un encaminamiento optimizado que escogerá un camino óptimo que minimice la distancia y las pérdidas por contención. Este camino óptimo se calcula de forma periódica mediante un modelo de optimización.

Cuando un nodo recibe un paquete de control y ha de reencaminarlo por un camino no previsto, emplea normalmente únicamente la información local disponible. Este hecho limita la eficiencia del encaminamiento por deflexión clásico. Los autores de CLDR proponen realizar periódicamente una optimización global de los caminos de deflexión que minimicen la probabilidad de bloqueo global. Por tanto, los caminos alternativos de deflexión están pre-calculados con este mecanismo. Cada nodo mantiene una lista ordenada de los caminos de deflexión. La optimización es un ILP, en el que se parte de una demanda conocida, caracterizada por un tiempo medio entre ráfagas y un grafo de red con un determinado número de longitudes de onda por enlace. Para cada demanda se calcula si el camino de deflexión desde un nodo congestionado atraviesa un determinado enlace. La función de coste consta de dos partes, cada una con su peso correspondiente. La primera hace referencia al retardo medio y la segunda a la probabilidad de bloqueo. De esta forma, se puede evitar que la solución se base en caminos excesivamente largos.

En los estudios llevados a cabo por Lee *et al.* [114], para condiciones de carga media, CLDR mejora en un orden de magnitud la probabilidad de bloqueo frente al caso en que se emplea deflexión por el camino más corto, en vez del mecanismo óptimo propuesto.

Estrategia de deflexión Least Load Node (LLN)

Countelen *et al.* [115] proponen una técnica alternativa para calcular el camino de deflexión denominada LLN (*Least Loaded Node*) basada en medidas, de forma que se reduzca de manera significativa la probabilidad de bloqueo.

En primer lugar, los caminos alternativos se ordenan por número de saltos. A continuación, entre los caminos de igual número de saltos, se ordenan por la carga del siguiente nodo en el camino. Esta carga se calcula a partir de medidas de tráfico en intervalos de 10 milisegundos. Después se normaliza según el número de longitudes de onda, y la capacidad por longitud de onda.

Deflexión de ida y vuelta (U-TURN)

Coutelen *et al.* [115] también proponen, como último recurso, en caso de contienda, enviar la ráfaga a un nodo por un camino de deflexión libre, y que el siguiente nodo la mande de vuelta, con la esperanza de que la situación de contienda se haya resuelto.

Balanceo de carga en la deflexión

Coutelen *et al.* [115] sugieren realizar un balanceado de carga a la hora de escoger los caminos de deflexión, evitando saturar las mejores rutas de deflexión. Para obtener el balance de carga óptimo se emplea una formulación ILP, basada en las formulaciones propuestas por Teng y Rouskas [95]. En dicha formulación se fija como función objetivo minimizar la capacidad residual en las fibras y se obtiene el conjunto de caminos de óptimos para cada par origen-destino. Una vez obtenidas las rutas, en cada nodo se examina, de entre todos los caminos que pasan por dicho nodo, cuales son los caminos escogidos para un determinado destino. El número de caminos que sean iguales para un mismo destino determinarán la probabilidad de ese camino como camino de deflexión, y será escogido ese porcentaje de veces cuando haya que escoger un camino de deflexión hacia ese destino.

La función objetivo planteada por Coutelen *et al.* obtiene, en el escenario concreto estudiado, muy buenos resultados en cuanto a balanceo de carga, pero obtiene caminos excesivamente largos. Los mismos autores plantean una función objetivo alternativa, en la que se minimiza el número de saltos totales, incluyendo una restricción para la capacidad residual. Mediante simulación, muestran la mejora de rendimiento en cuanto a probabilidad de bloqueo de la estrategia de balanceo de carga.

Deflexión teniendo en cuenta la probabilidad de contienda

Cuando se encamina una ráfaga por deflexión, en los siguientes nodos del camino la ráfaga se puede encontrar en nuevas situaciones de contienda, y por tanto, o bien acabar descartada, o bien acabar siguiendo una ruta excesivamente larga. Para minimizar estos riesgos, Ogino *et al.* [129] proponen un mecanismo de deflexión en el que se tiene en cuenta la probabilidad de contienda en los nodos de los caminos. Se define una métrica dinámica denominada "distancia de ruta esperada" (*expected route distance, ERD*) como alternativa a una métrica estática como el número de saltos. Esta métrica se calcula midiendo las probabilidades de bloqueo en cada uno de los nodos del camino. La elección del camino de deflexión se realiza dinámicamente basándose en la métrica ERD. De esta forma, al incluir la probabilidad de bloqueo en el encaminamiento, la probabilidad de pérdida de ráfaga global se mejora. Esta mejora es significativa cuando se carga la red con un tráfico no balanceado (es decir, el tráfico difiere de manera significativa al tráfico para el que fue diseñada originalmente la red).

***Reinforcement Learning-based Deflection Routing Scheme* (RLDRS)**

Belbekkouche *et al.* [112] proponen un esquema dinámico para elegir el camino de deflexión basado en técnicas de aprendizaje reforzado (*reinforcement learning*), con el objetivo de mejorar el rendimiento, en cuanto a probabilidad de bloqueo y retardo, del resto de técnicas de deflexión. El mecanismo propuesto se denomina *Reinforcement Learning-based Deflection Routing Scheme* (RLDRS). La métrica para escoger el camino se obtiene a partir de la probabilidad de pérdida y el retardo.

La técnica de deflexión RLDRS propone además limitar el número de deflexiones a las que se puede someter una ráfaga. De esta manera, se reduce el impacto de las deflexiones en el resto de la red, al introducir menor tráfico, y por otro lado se limita el retardo, que puede llegar a ser muy elevado si una ráfaga se reencamina en numerosas ocasiones.

En las simulaciones llevadas a cabo por Belbekkouche *et al.* [112], el mecanismo RDLDRS mejora en un orden de magnitud el comportamiento del SP-DR, para todas las condiciones de carga.

2.8.5 Resolución de contienda del paquete de control

En esta sección se ha estudiado cómo resolver la contienda de ráfagas. Sin embargo, no solamente puede haber contienda en las ráfagas, sino que también es posible que haya contienda en los mensajes de señalización, que se envían típicamente por un canal de control dedicado. En este canal de control se transmite, por ejemplo en el caso de OBS-JET, el paquete de control (BCP) que indica cuando va a llegar la próxima ráfaga.

Por tanto, si es necesario enviar dos paquetes de control por el mismo puerto de salida, uno de ellos se enviará normalmente y el otro se almacenará temporalmente hasta que se transmita. El problema es que la información de *offset* del paquete ya no es válida, ya que el paquete se transmite más tarde de lo previsto.

Para solucionar este problema, Lin *et al.* [130] proponen el mecanismo SCORE (*Store-and-forward COtention-Resolution*). Para ello, se propone retrasar la ráfaga de datos en un FDL tanto tiempo como haya tenido que esperar el paquete de control para transmitirse.

2.9 Mecanismos para recuperación de pérdida de ráfagas

En las dos secciones anteriores se ha visto cómo se puede proceder para solucionar el problema de contienda de ráfagas, o cómo se puede evitar que ocurran estas contiendas. Sin embargo, en el supuesto de que haya una contienda de ráfagas, y que ésta no se haya podido resolver satisfactoriamente, se tiene que proceder al descarte de una de las ráfagas. Ante este evento, se tienen dos posibilidades, o bien son las capas superiores las encargadas de asegurar la fiabilidad de la transmisión (por ejemplo, el protocolo de transporte TCP se encarga de retransmitir los segmentos perdidos), actuando de manera independiente para cada flujo de tráfico, o bien es la propia red OBS la que proporciona mecanismos para asegurar que los datos no se pierdan.

La Tabla 2-9 resume los principales mecanismos de recuperación en la red OBS para que no sea necesario recurrir a capas superiores. El problema de recurrir a las capas superiores es que las pérdidas pueden interpretarse como signos de congestión, y tomar decisiones erróneas en base a ello. En otros casos, puede implicar retardos elevados en la aplicación hasta que se recupera del fallo, o bien, por ejemplo en el caso de servicios de vídeo en tiempo real, se degradaría el vídeo.

Mecanismo	Referencias
Retransmisión	[131]
FEC	[132]
Clonado de ráfagas	[133], [134]

Tabla 2-9 Mecanismos de fiabilidad ante pérdida de ráfagas

2.9.1 Retransmisión de ráfagas

El propósito de la retransmisión de ráfagas en OBS es el de permitir que ráfagas que se hayan tenido que descartar en alguna contienda puedan ser retransmitidas, reduciéndose así la probabilidad de bloqueo global [131].

En el esquema de retransmisión, cuando el nodo origen envía el paquete de control informando de la próxima transmisión de la ráfaga, éste guarda una copia de la ráfaga, a la que se asignará un identificador único de secuencia. En el caso de que un nodo intermedio al procesar el paquete de control decida que hay que descartar la ráfaga, informará con un mensaje ARQ (*Automatic Retransmission Request*) al nodo origen del fallo en la reserva de recursos. En ese momento, el nodo origen procederá al envío del duplicado de la ráfaga, precedido de un nuevo paquete de control.

En caso de que la red esté poco cargada, hay bastantes posibilidades de que el duplicado llegue al destino, mientras que si la red está cargada, las probabilidades son escasas. El mecanismo de retransmisión puede permitir que una ráfaga se retransmita tantas veces como sea necesario. Sin embargo, para evitar un retardo excesivo, es conveniente limitar el número de retransmisiones. El principal inconveniente de la retransmisión es que introduce más carga en la red. A pesar de ello, la retransmisión reduce la probabilidad de bloqueo, especialmente a niveles bajos de carga de la red.

Zhang *et al.* [131] han comparado mediante simulación el rendimiento de la retransmisión frente a una red OBS sin este mecanismo, y frente a una red OBS con deflexión. En el escenario empleado en dicho estudio, se obtiene 4 veces menos probabilidad de bloqueo comparando con el caso con deflexión, y hasta 2 órdenes de magnitud, siempre en casos de carga media-baja. En casos de carga alta, la diferencia es inapreciable.

2.9.2 Corrección de pérdida de ráfagas con FEC

Una alternativa para recuperar ráfagas perdidas es el empleo de técnicas basadas en FEC (*Forward Error Correction*). Este tipo de técnicas consiste en enviar junto con los datos información redundante, de tal forma que en el destino, aunque falten una parte de los datos, estos puedan recuperarse en su totalidad. El codificado del FEC se hace en la fuente y el decodificado en destino. El FEC es una técnica comúnmente utilizada en todo tipo de sistemas de comunicaciones, especialmente en aquellos en los que la retransmisión es muy costosa (por ejemplo, transmisiones por satélite).

Arima *et al.* [132] proponen dos técnicas basadas en FEC para recuperarse de las pérdidas de ráfagas. En ambos métodos, los datos redundantes se generan a partir de un conjunto de ráfagas. En el método OBG (*Out of Burst Generation*) se genera una ráfaga únicamente con información redundante. Cada cierto número de ráfagas de datos, se transmite una de estas ráfagas redundantes. Si se pierde una ráfaga por el camino, entre los datos de cada una de las ráfagas de datos que han llegado bien y la ráfaga redundante, se recupera la ráfaga perdida. El principal inconveniente de este método es que genera mayor número de ráfagas, y por tanto aumenta la posibilidad de que se pierdan ráfagas. En el otro método, denominado IBG (*In-Burst Generation*), el número de ráfagas permanece constante. Los datos redundantes se añaden al final de cada ráfaga. Estos datos redundantes

corresponden a las ráfagas anteriores. En este método, el inconveniente es el incremento de tamaño de las ráfagas. Mediante simulación, Arima *et al.* comprueban la bondad del método propuesto y la habilidad para recuperarse ante pérdidas de ráfagas, consiguiendo resultados similares a otras técnicas como por ejemplo el *offset* extra, pero sin aumentar el retardo de los paquetes transmitidos.

2.9.3 Clonado de ráfagas

Huang *et al.* [133] proponen un esquema proactivo denominado clonado de ráfagas para minimizar las pérdidas por contienda. La idea consiste en replicar una ráfaga y enviar las copias a la red simultáneamente. Si la ráfaga original se pierde, las ráfagas clonadas pueden llegar al destino. Para evitar interferencias con el resto de ráfagas, se otorga una baja prioridad a las ráfagas clonadas, de tal forma que en caso de contienda con una ráfaga original, éstas se descarten. Los principales problemas en el diseño del clonado en una red OBS son escoger cuántas ráfagas se clonan, en qué nodos se hace el clonado y el encaminamiento de las ráfagas originales y clonadas. Huang *et al.* proponen que el nodo sea diferente para cada par origen-destino. Una vez clonada la ráfaga, para maximizar las probabilidades de éxito, el camino de las ráfagas clonadas y las originales debe ser disjunto.

Sullivan *et al.* [134] evaluaron el rendimiento de una red OBS con tráfico TCP cuando se empleaba clonado de ráfagas. Los resultados muestran el incremento del rendimiento de TCP tanto en cargas altas como bajas cuando se emplea esta técnica.

2.10 Calidad de Servicio en OBS

Un aspecto muy importante en las redes es la posibilidad de ofrecer calidad de servicio (QoS) diferenciada. Cada tipo de aplicación y usuario tiene unos requisitos distintos, por lo que sus necesidades de transporte en la red son diferentes. Por ejemplo, mientras que descargando un correo no nos importa esperar unos segundos, unos cortes en un juego en tiempo real o una videoconferencia son molestos. Por otro lado, un usuario pagando una cuota mensual baja puede tolerar degradaciones puntuales, pero un banco que paga una cuota elevada no puede permitir problemas, incluso puntuales, por ejemplo en comunicaciones con los cajeros automáticos o sistemas de compra. Más aún, las llamadas de emergencia han de tener prioridad sobre el resto del tráfico.

En las redes IP, el IETF ha propuesto dos modelos principales para soportar QoS, *IntServ*, definido en la RFC 1633 [135] y *Diffserv*, definido en la RFC 2475 [136]. *IntServ* se basa en una reserva flujo a flujo. Mediante el protocolo RSVP [137] se puede reservar un camino con determinadas garantías de ancho de banda, retardo, perdida de paquetes, etc. Sin embargo esta aproximación tiene problemas de escalabilidad si se aplica el modelo a gran escala. Por otro lado, *DiffServ*, para poder conseguir una solución escalable, clasifica los paquetes de acuerdo a un código en la cabecera IP. Según este código, los paquetes tendrán diferentes tratamientos en las colas de los *routers*. Este modelo se emplea habitualmente en las redes IP.

En resumen, los mecanismos de QoS tratan de ofrecer a las aplicaciones y usuarios un comportamiento del tráfico acorde a sus necesidades y expectativas, en vez de tratar a todos los datos por igual. Por tanto, una red OBS también ha de ser capaz de implementar mecanismos de QoS, y poder ofrecer tratamientos diferentes a los paquetes que entran en la red.

En la Tabla 2-10 se muestra un resumen de los principales mecanismos propuestos en la literatura para ofrecer QoS en OBS, que se describen a continuación.

Mecanismo	Referencias
Incremento de *offset*	[79]
Ensamblado diferenciado	[138] [139]
Clonado de ráfagas	[140] [141]
Encaminamiento diferenciado	[142]
Asignación de longitud de onda diferenciado	[143]

Tabla 2-10 Técnicas de QoS en OBS

Incremento de *offset*

La primera técnica de QoS para redes con OBS JET sin *buffers* propuesta en la literatura consiste en un esquema de priorización basado en modificar el tiempo de *offset* [79]. En este esquema, a las ráfagas con mayor prioridad se les asigna un *offset* mayor, con lo que la posibilidad de reservar los recursos con éxito es similar. El concepto es similar a cuando se realiza una reserva de un vuelo o un hotel: cuanta mayor es la antelación con que se realiza la reserva, más fácil es que se tenga éxito, mientras que a última hora está todo agotado. Otra alternativa es indicar la prioridad en el paquete de control de las ráfagas, y proporcionar un tratamiento diferenciado en caso de resolución de contienda. Vokkarane *et al.* [144] proponen emplear conjuntamente con la priorización el empleo de segmentación de ráfagas.

Ensamblado de ráfaga diferenciado

Vokkarane *et al.* [138] proponen una nueva técnica de ensamblado de ráfagas para cumplir con los requisitos de retardo y pérdida de paquete de cada clase de servicio. La técnica consiste en crear ráfagas con paquetes de distintas clases. Los paquetes se ordenan en la ráfaga en orden descendente según su clase. De esta manera los paquetes con mayor prioridad están en la parte delantera de la ráfaga, mientras que los de menor prioridad están en la parte trasera. De esta forma, en caso de contienda, se descarta la parte trasera de la ráfaga, evitando la pérdida de los paquetes prioritarios.

Clonado de ráfagas diferenciado

Askar *et al.* [140] proponen una nueva técnica de clonado de ráfagas denominada *Classified Burst Cloning*. El principal problema del clonado de ráfagas es el aumento de la carga en la red. Aunque el esquema de clonado permite que las ráfagas normales tengan preferencia sobre las clonadas, la contienda entre las ráfagas clonadas es en igualdad de condiciones. Por tanto, en muchas ocasiones, los enlaces pueden ir muy cargados, sobre todo si se sigue un esquema de clonado que permita múltiples copias, con lo que probablemente las ráfagas clonadas sufran de una elevada pérdida.

En el esquema que proponen Askar *et al.* se tienen dos clases de servicios, y dos colas de ensamblado. En una de las colas, se introduce todo el tráfico, y se crean las ráfagas normales. En la otra cola se almacenan solo los paquetes de alta prioridad. Cuando se transmita la ráfaga original, también se transmite la clonada (pero solo con datos de alta prioridad). En caso de contienda, no hay ninguna prioridad. En un estudio mediante simulación Askar *et al.* muestran que el mecanismo obtiene un 50% menos de probabilidad de bloqueo para la clase superior [140]. Comparado con otros esquemas, el clonado, según dicho estudio, mantiene una baja latencia, sin introducir tiempos de *offset* adicionales.

Encaminamiento diferenciado

Anteriormente se ha visto que una buena estrategia de encaminamiento puede ayudar a reducir las situaciones de contienda evitando cuellos de botella. Si además la técnica de encaminamiento es multicamino, se puede realizar un mejor balanceo de carga en la red, al

aumentar el número de posibilidades por donde repartir el tráfico. Sin embargo, al emplear técnicas multicamino se han de escoger en ocasiones caminos más largos, con un retardo mayor del habitual, que puede ser intolerable para algunas aplicaciones. Thachayani y Nakkeeran [142] proponen una selección de camino diferenciada cuando se emplea una técnica multicamino. De esta forma, las ráfagas de mayor prioridad elegirían los caminos con menor retardo. En la propuesta de los autores mencionados, se calculan dos conjuntos de rutas, uno de ellos compuesto por las rutas más cortas, y un segundo conjunto compuesto por los caminos de menor coste, donde el coste se calcula mediante una combinación entre la ocupación de los enlaces si se emplease el camino más corto y el número de saltos. Las ráfagas, según su prioridad y su requisito de retardo máximo escogen entre los conjuntos de cola mencionados.

Como resultado, se consigue una reducción de la probabilidad de bloqueo por el mero hecho de emplear una técnica multicamino, y por otro lado se obtiene un tratamiento diferenciado al asignar las rutas más cortas al tráfico prioritario.

Asignación de longitud de onda diferenciada

Una de las formas de resolver la contienda de ráfagas es el empleo de convertidores de longitud de onda. Sin embargo, son elementos de coste elevado, por lo que no se puede asumir que un nodo OBS va a estar siempre equipado con convertidores. En caso de no disponer de convertidores de longitud de onda, se ha de emplear la misma longitud de onda en todo el camino de la ráfaga. Al tener que cumplir esta restricción, la contienda de ráfagas se agrava. Sabir *et al.* [143] proponen reservar un conjunto de longitudes de onda para las ráfagas prioritarias. Estas longitudes de onda serían asignadas en exclusiva durante unos *slots* de tiempo a cada nodo, para asegurar que, si escogen esa longitud de onda, no habrá otro nodo en la red que la escoja en ese mismo *slot*.

2.11 Demostradores de OBS

OBS ha tenido una gran acogida en el entorno de investigación en tecnologías de comunicaciones ópticas, con más de 1000 artículos en congresos y revistas en los últimos 10 años. No solamente se ha abordado OBS desde un punto de vista teórico, sino que se han realizado también prototipos y *testbeds* experimentales en centros de investigación de universidades y empresas de varios países que han confirmado la viabilidad tecnológica y han permitido realizar experimentos más allá de la simulación.

El concepto de OBS partió originalmente de propuestas de Qiao *et al.* [6] y Turner [24] alrededor de 1999. Sin embargo, el primer demostrador no llegaría hasta 2003, en el *testbed* de ATDnet (*Advanced Technology Demonstration Network)* situado a las afueras de Washington y dedicado a probar nuevas redes metropolitanas [43]. En este primer *testbed* se demostró experimentalmente el funcionamiento del protocolo de JIT, siendo una prueba de concepto de los protocolos de reserva de camino en un solo sentido sin confirmación, muy característicos de OBS. El protocolo se implementó en hardware en un prototipo de nodo OBS. En el prototipo, los mensajes del protocolo JIT se transmitieron fuera de banda, en una longitud de onda dedicada, llegándose a transmitir hasta 80000 mensajes de señalización por segundo. En este primer prototipo de nodo OBS, además de la señalización, se implementó el plano de datos, con un conmutador óptico de ráfagas basado en MEMS comerciales de *Firstwave*. Con los MEMS comerciales del nodo se obtuvieron, en 2003, unos tiempos de conmutación en el orden de la decena de ms. En este primer *testbed*, se desarrollaron 3 prototipos de nodo OBS, que se distribuyeron en el *testbed* ATDnet, con enlaces WDM de distancias relativamente cortas entre ellos, en el orden de varios kilómetros. La aplicación principal que se probó sobre este *testbed* fue la

transmisión de manera totalmente óptica, atravesando los 3 prototipos de nodo OBS, de un canal de alta definición de televisión de 1,5 Gbps.

En el transcurso de los dos años posteriores a la realización de la primera prueba de concepto experimental de OBS se realizaron nuevos demostradores en China y Japón en los que se exploraron nuevos aspectos de OBS. En 2004, en el *testbed* JGN II (*Japan Gigabit Network*), en el que participaron el fabricante Fujitsu, la operadora NTT y las universidades de Osaka y Tokio, se realizó una prueba de concepto de OBS con protocolo de reserva de canal en dos sentidos con confirmación, basándose para ello en una implementación de GMPLS [44] [45]. De esta forma, entre el *testbed* de ATDnet y el JGN II se verificaron las dos aproximaciones principales de OBS, señalización en uno y dos sentidos. Mientras que el *testbed* de ATDnet tenía 3 nodos OBS, el de JGN II constaba de 6 prototipos de nodo OBS. La mitad de los nodos, al igual que en el primer *testbed* de OBS, estaba equipada con MEMS para realizar la conmutación. Como innovación, en la mitad de los prototipos, en concreto en los realizados por Fujitsu, se empleó la tecnología PLC, que obtenía un tiempo de conmutación ligeramente inferior al obtenido mediante la tecnología MEMS (10 ms en los MEMS, frente a 3 ms en los PLC), es decir unos pocos milisegundos de diferencia. Como mejora frente al *testbed* anterior, que tenía el alcance de una pequeña red metropolitana, el *testbed* JGN II llegaba a recorrer 128 kilómetros, distancia suficiente para una red de alcance regional.

Los dos *testbeds* anteriores, ATDnet y JGN II, se centraron en dos pilares de OBS, la señalización y la tecnología de conmutación óptica, quedando aún un largo camino por realizar para poder demostrar OBS con todas las características que debe tener una tecnología de red. El *testbed* de la Universidad de Tokio, realizado en 2005, intentó ir un paso más allá de los demostradores anteriores, y centrarse en completar el resto de los aspectos de OBS, como son el proceso de ensamblado, los protocolos de encaminamiento, los protocolos de control (resolución de contienda y planificación) y la interacción con las redes de tecnologías existentes [46]. En este sentido, una de las principales aportaciones del *testbed* de Tokio fue el desarrollo de un prototipo de nodo de borde, con un ensamblador de ráfagas avanzado con múltiples colas y soporte a varias clases de servicio (CoS). Además del nodo de borde, se mejoraron los nodos de transporte, en los que se implementaron en hardware mecanismos de planificación de ráfagas, ya propuestos de manera teórica en la literatura y el conocido protocolo de señalización JET (anteriormente en el *testbed* ATDnet se implementó JIT). Sin embargo, la base de los tres nodos de transporte OBS implementados era la misma que en el *testbed* JGN II, con los conmutadores de ráfagas basados en PLC. En los experimentos llevados a cabo en el *testbed*, se conectó uno de los nodos con Internet, y se realizaron conexiones de vídeo en tiempo real, tanto basados en TCP como en UDP, entre los clientes conectados a los nodos OBS. Por tanto, el *testbed* de OBS de la universidad de Tokio se cumplió el hito de ser el primer demostrador de OBS completo e interconectado con Internet.

En el mismo periodo de tiempo que los *testbeds* japoneses mencionados, el gobierno chino lanzó el programa TBOBS (*Testbed for Optical Burst Switching Network*), con el objetivo de experimentar en distintos aspectos de OBS, y en el que participaron BUPT (*Beijing University of Posts and Telecommunications*) y la universidad de *Shaghai Jiaotong* [145] [146]. Una de las aportaciones del *testbed* chino fue en primer lugar la demostración en hardware de mecanismo de planificación de ráfagas LAUC-VF, del que hasta la fecha sólo se disponían evaluaciones mediante simulación. El *testbed* chino se ha utilizado desde entonces como base para estudiar de manera experimental el rendimiento de TCP sobre OBS, y se ha empleado en varios estudios [147] [148] [149] [150]. Finalmente, otra de las novedades del *testbed* chino con respecto a los demás demostradores de OBS fue el incluir un módulo de conmutación rápido basado en SOA, desarrollado por la propia universidad. Este *testbed* se

empleó recientemente para la primera demostración de interoperabilidad entre una red basada en GMPLS y una red OBS [151]

A partir de los resultados y experiencias del *testbed* JNG II citado anteriormente, ya en 2009, el consorcio japonés OITDA (*Optoelectronic Industry and Technology Development Association*), al que pertenecen la universidad de Tokio y KDDI, en su proyecto NEDO continuó avanzando en la tecnología OBS [152]. Un primer paso fue crear un prototipo de nodo OBS con tecnologías de conmutación más modernas. Mientras que en el antiguo *testbed* de JGN II se emplearon MEMS y PLC, en el nuevo *testbed* de OITDA, se empleó la tecnología de conmutación PZLT, que reducía en varios órdenes de magnitud los tiempos de conmutación ópticos. Otra de las novedades introducidas en el *testbed* fue el realizar una prueba de concepto de la reserva del camino con procesamiento salto a salto, sin un *offset* inicial como JET o reserva con confirmación, que se pudo realizar de manera satisfactoria gracias a la rapidez de los nuevos componentes introducidos. Al haber mejorado los tiempos de conmutación, el tamaño de las ráfagas era mucho menor que en *testbeds* anteriores, reportándose tiempos de ráfaga de 30 microsegundos, frente a los cientos de milisegundos de experimentos anteriores. Al igual que en el *testbed* JGN II, las distancias entre nodos en el demostrador son representativas de un alcance habitual en una red regional, habiéndose reportado transmisiones de hasta 92 km [47].

Mientras que los demostradores anteriores no estaban diseñados para una aplicación específica, la universidad de Essex, dentro del proyecto IST PHOSPHORUS [153], realizó en 2007 un *testbed* de OBS con una característica muy particular, el estar integrado con las aplicaciones y servicios de red (*application-aware*), en concreto con tecnologías de computación distribuida [154]. Dentro de los prototipos de nodo OBS, además de las funciones típicas de OBS, como el protocolo de reserva JIT, se implementaron mecanismos de planificación y control de trabajos. Al igual que el *testbed* chino, el *testbed* de Essex tomó la decisión de no emplear tecnología comercial, sino desarrollar sus propios conmutadores rápidos basados en SOA.

En esta sección se ha visto cómo los experimentos de OBS, recopilados con sus principales características en la Tabla 2-11, han ido evolucionando a lo largo de los años, y han ido demostrando los conceptos principales y la viabilidad tecnológica de OBS. Sin embargo, el concepto de OBS no se ha quedado únicamente en artículos o experimentos, sino que además se han realizado las primeras apuestas comerciales que emplean conceptos de OBS, que se verán en la próxima sección.

Testbed	Hitos	Tecnología Conmutación	Referencias
ATDnet	Primer *testbed* de OBS, prototipo de JIT.	MEMS	[43]
JGN II	Prototipo de reserva en 2 sentidos, 6 nodos	MEMS, PLC	[44] [45]
BUPT	Pruebas de TCP sobre OBS, interacción con GMPLS	SOA	[145] [147] [148] [149] [150] [151]
Universidad de Tokio	Primer *testbed* de OBS completo	PLC	[46]
OITDA	Reserva salto a salto, conmutador muy rápido	PLZT	[152], [47]
Universidad de Essex	OBS integrado con aplicación	SOA	[154]

Tabla 2-11 *Testbeds* de OBS

2.12 Primeros productos comerciales de OBS

La primera empresa que apostó por OBS fue la *start-up* Matisse Networks (2003-2009). Este primer producto comercial de OBS, que se denominó "*EtherBurst*" se centraba en entornos metropolitanos-regionales, con alcances de hasta 200 kilómetros y topologías en anillo [155]. El producto se basaba en ráfagas de tamaño constante, que se planificaban en el tiempo de una manera centralizada, en *slots* de tiempo fijos. La otra base del producto eran los láseres sintonizables de 10 Gbps para transmitir ráfagas ópticas, capaces de sintonizar en cualquier longitud de onda de la banda ITU-C en nanosegundos. Con este primer producto comercial, se podía transmitir en total de 10 a 640 Gbps en un entorno metropolitano. A pesar de las cualidades de la tecnología y de favorables informes económicos, como el llevado a cabo por Network Strategy Partners [156], Matisse Networks cerró sus operaciones en diciembre de 2009, al no poder conseguir más fondos para seguir adelante [157].

La siguiente empresa en apostar por OBS fue la compañía irlandesa Intune Networks [158]. La empresa irlandesa ha anunciado desde 2009 en varias notas de prensa su disponibilidad y continuo desarrollo de la tecnología OBS para el entorno metropolitano [159] [160]. Sin embargo, no ha sido hasta mayo de 2011 cuando oficialmente se ha lanzado el producto comercial, denominado *"Optical Packet Switch & Transport"* (OPST), nombre comercial "Verisma" [161]. Al igual que Matisse, la solución de Intune se centra en anillos metropolitanos y emplea láseres sintonizables rápido. La principal novedad de Intune es que se trata de una tecnología totalmente asíncrona, al contrario que Matisse, en la que los *slots* de tiempo eran fijos. Cada nodo de Intune dispone de un conjunto de colas, una para cada destino [162].

La última empresa en anunciar un producto de OBS ya no se trata de una *start-up* como las dos anteriores, sino de una empresa madura con productos comerciales en comunicaciones ópticas. Huawei anunció en la OFC (*Optical Fiber Conference*) de 2011 la demostración de una versión pre-comercial de su solución OBTN (*Optical Burst Transport Network*) [163]. En la siguiente edición de la OFC en 2012 Huawei continuó mostrando su interés por la tecnología OBS con un nuevo prototipo, denominado en esta ocasión como *Petabit (-per-second) Photonic Cross-Connect* (PPXC) [164]. Este prototipo de nodo OBS consta de una matriz de 80×80.

De esta forma, se aprecia que OBS va tomando poco a poco fuerza como alternativa para las redes de próxima generación y ha captado el interés de la industria. Sin embargo aún no hay ningún despliegue comercial en una operadora.

2.13 Resumen

En este capítulo de la tesis se ha realizado una revisión del estado del arte de la tecnología OBS. Se ha visto una clara evolución en esta tecnología, empezando por los estudios teóricos, primeros *testbeds* con sus pruebas de concepto, experimentos más complejos y finalmente las primeras implementaciones comerciales. La aparición de unos primeros productos, aún limitados a entornos metropolitanos, confirma el interés en OBS.

Por otro lado, se ha revisado cuáles son las tecnologías ópticas que han posibilitado la aparición de OBS. Entre ellas destacan los láseres sintonizables rápidos, cuya tecnología está bastante madura y las matrices de conmutación, que necesitan aún una evolución para llegar al rendimiento requerido en OBS. Otras tecnologías, como las relacionadas con el almacenamiento de ráfagas ópticas, clave para obtener un buen rendimiento, aún necesitan evolucionar. El avance que se ha visto en dichas tecnologías, sugiere que en cualquier

momento se pueden producir los saltos tecnológicos necesarios para que OBS tenga una viabilidad comercial.

Uno de los principales problemas de OBS es la contienda entre ráfagas, que puede provocar que se descarte el tráfico que viaja en las ráfagas. En el estudio del estado del arte se han revisado las técnicas propuestas en la literatura. Se ha visto que un aspecto con potencial para reducir la contienda y mejorar el rendimiento de OBS es una buena elección de las rutas de las ráfagas. En la literatura hay una gran cantidad de aportaciones en cuanto a la optimización de rutas dada una demanda conocida. Sin embargo, la demanda futura no es conocida con exactitud (es posible realizar medidas a posteriori y realizar previsiones). En cuanto empieza a funcionar la red, el tráfico evoluciona y cambia su distribución, muchas veces por eventos imprevistos, como por ejemplo el movimiento de tráfico provocado por el cierre de la página de descargas *Megaupload* [165], y hacen falta técnicas en que se adapten al tráfico. En este sentido, se ha detectado que hay una carencia de propuestas en la literatura. También se ha visto que el elemento de cálculo de rutas PCE propuesto por el IETF es muy adecuado para el mejorar el proceso de encaminamiento en OBS y puede ayudar en la mencionada labor de mejorar el rendimiento de OBS. Actualmente, el PCE está especificado para redes MPLS, empezando a especificarse para otra tipo de redes. Sin embargo, no hay ninguna propuesta en la literatura que lo proponga para OBS. Por tanto, estos serán algunos de los aspectos que se desarrollarán en los capítulos posteriores de esta tesis.

Capítulo 3

TCP sobre OBS

3.1 Introducción

TCP (*Transmission Control Protocol*) [8] se ha convertido en el estándar de facto de los protocolos de transporte y es empleado en la actualidad por la mayor parte de aplicaciones, como el correo electrónico, la navegación *web*, la transferencia de ficheros, como el caso de los servicios de descarga *Megaupload* [166] (recientemente clausurado [165]) o *Rapidshare* [167], aplicaciones P2P como *emule* [168] o *bittorrent* [169] [170], o incluso la transmisión de vídeo, como es el caso del popular servicio *youtube* [171] [172]. Para hacerse una idea de la importancia de TCP, se puede acudir a los estudios de tráfico, como el realizado por Wolfgang *et al.* sobre el tráfico transportado en un *backbone* sueco [173], en el que detectó que TCP acaparaba el 93% de los paquetes y el 97% de los bytes del enlace medido. En la página web de Caida [174], una asociación dedicada al análisis de datos de tráfico de Internet, se pueden observar datos similares, tanto en la actualidad, como en el histórico de datos. Por otro lado, la organización internacional IETF [175], la más importante e influyente en cuanto a arquitectura y protocolos empleados en Internet, mantiene su apoyo firme a TCP, por lo que se prevé que siga durante muchísimos años reinando en Internet como protocolo de transporte. A pesar de las múltiples propuestas académicas para cambiar TCP, este protocolo no ha cambiado sustancialmente durante años, principalmente debido a la política conservadora del IETF, que está al cargo de la tarea de mantenimiento del protocolo, y tiene como principio de actuación el incorporar únicamente cambios pequeños [176].

Dado que la mayor parte del tráfico en las redes emplea TCP, y las previsiones de que siga empleándose durante mucho tiempo, el diseño de nuevas tecnologías de redes de comunicaciones, como es el caso de OBS, lo ha de tener en cuenta de manera significativa. Hay que tenerlo en cuenta en dos sentidos. En primer lugar, hay que observar cómo se comporta la red cuando se carga con tráfico TCP, con sus características peculiares de emplear el ancho de banda disponible. Saber el tipo de tráfico que hay en la red, ayudará a dimensionar los *buffers* y los recursos necesarios, con lo que se podrá evaluar fácilmente la viabilidad de esta nueva tecnología. En segundo lugar, y en igualdad de importancia, hay que estudiar cómo impacta el funcionamiento de esta nueva tecnología de red en el tráfico TCP, y más en concreto, cómo impacta en el rendimiento de las aplicaciones de usuario que empleen TCP como medio de transporte.

En este capítulo se revisan en primer lugar los fundamentos de TCP para poder entender su comportamiento, realizando un recorrido por las distintas versiones y evoluciones del protocolo. Después se describe la problemática de TCP en redes ópticas de alta velocidad, independientemente de la tecnología específica, para pasar posteriormente a mostrar con detalle qué ocurre cuando se emplea TCP y OBS. En posteriores capítulos, se estudiarán aspectos específicos de la relación entre TCP y OBS, como el uso del asentimiento retardado, el comportamiento con tráfico agregado y la sincronización de flujos.

3.2 Fundamentos de TCP

TCP [177] es el principal protocolo de transporte en Internet. Es un protocolo orientado a conexión, encargado de proporcionar una comunicación fiable extremo a extremo en una red de paquetes, para lo cual emplea las facilidades que le proporciona la capa de red, habitualmente IP. TCP no asume ningún tipo de fiabilidad en la capa de red, por lo que incluye mecanismos para proporcionar una transmisión con éxito. El otro gran objetivo de TCP, además de la fiabilidad, es el de realizar un control de flujo, para evitar congestionar la red, y en caso de que haya congestión, ayudar a reducirla.

La principal referencia de TCP son las distintas RFCs (*Request For Comments*) que describen el protocolo TCP y los algoritmos que emplea, que se encuentran recopiladas en la RFC 4614, el "*roadmap* de especificaciones de TCP" [178], y la RFC 5783, que describe la historia del control de congestión en las RFCs [179]. Sin embargo, a pesar de que comportamiento de TCP está especificado en las RFCs, no hay un TCP "estándar", sino que existen versiones de TCP que emplean los distintos algoritmos para el control de la congestión descritos en las distintas RFCs. Incluso cada sistema operativo tiene su propia implementación de TCP, que escoge los mecanismos de control de flujo, congestión que cree más convenientes.

En esta sección se realiza un recorrido por los fundamentos de TCP necesarios para entender su dinámica, que será clave para entender su comportamiento posteriormente en redes OBS. Para profundizar en los detalles de TCP, además de las RFC, una buena referencia didáctica es el libro de Stevens [8].

3.2.1 Control de flujo y congestión

TCP envía la información en partes, denominadas segmentos, que son asentidos por el receptor mediante un mensaje ACK. Cada segmento está numerado, para posibilitar su ordenamiento en el destino y detectar posibles pérdidas.

Para realizar la tarea de control de flujo, TCP emplea un mecanismo de ventana deslizante. El tamaño de la ventana de transmisión determina cuántos datos pueden estar en tránsito, es decir, cuántos bytes pueden haber sido enviados por el transmisor sin que aún se haya recibido su asentimiento. Cuando se tienen en tránsito tanta cantidad de datos como indica la ventana de transmisión, el emisor deja de enviar nuevos segmentos. Cada vez que se recibe un asentimiento, y si el valor de la ventana de transmisión lo permite, TCP transmite nuevos datos. El valor de la ventana de transmisión se determina por el mínimo de dos límites, uno que está impuesto por el receptor, e indica el tamaño del *buffer* de recepción disponible, para evitar saturar el receptor, y otro impuesto por el propio emisor, denominado ventana de congestión, que tiene el objetivo de evitar el envío de más datos que los que la red soporta (Figura 3-1).

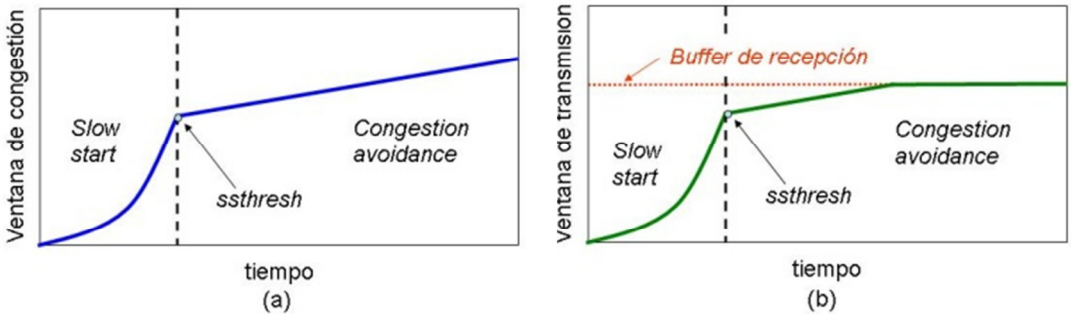


Figura 3-1 (a) Evolución de la ventana de congestión en TCP. (b) Evolución de la ventana de transmisión en TCP.

TCP tiene varias fases en la transmisión, en las que el valor de la ventana de congestión varía de manera diferente. TCP empieza la comunicación probando la red, enviando un solo segmento, y fija el tamaño de la ventana de congestión (*cwnd*) a un segmento. En cuanto recibe el asentimiento la ventana aumenta *cwnd* a dos, con lo que se envían dos segmentos, y así sucesivamente, resultando en un aumento exponencial de la ventana de congestión, como se puede observar en Figura 3-1. Este comportamiento se denomina *slow start*, y finaliza cuando la ventana de congestión alcanza un umbral denominado *ssthresh* (*show start threshold*). Realmente, la RFC 2581 [180] permite empezar la comunicación enviando más de un segmento, pero este comportamiento es aún poco habitual. Sin embargo, varios autores de Google han publicado recientemente un estudio en el que se muestran los beneficios en cuanto a latencia de conexiones HTTP (*Hypertext Transfer Protocol*) de poco tamaño cuando se incrementa la ventana de transmisión [181]. De hecho, Google está intentando influir en el IETF para el aumento de la ventana [182], y permitir un aumento de rapidez en las conexiones.

Después de la fase de *slow start*, la ventana de congestión aumenta de forma lineal en la fase denominada *congestion avoidance*, con el fin de no saturar la red. En la Figura 3-1b se observa la evolución de la ventana de transmisión, que, como se ha mencionado, es el mínimo entre la ventana de congestión y el tamaño del *buffer* de recepción.

Por lo tanto, el valor de la ventana de transmisión en cada momento da una indicación de la tasa de transmisión de TCP, ya que, en ausencia de errores se va a transmitir durante un RTT (*Round Trip Time*, tiempo de ida y vuelta) como máximo tantos segmentos como indique la ventana de transmisión. El RTT se define como el tiempo que transcurre desde que se envía un segmento hasta que llega el asentimiento que confirma su correcta llegada. Así pues, la relación entre la tasa de transmisión, $X(t)$, expresada en segmentos por segundo, y la ventana de transmisión, $W(t)$, expresada en segmentos se muestra en (3-1).

$$X(t) = \frac{W(t)}{RTT} \tag{3-1}$$

De esta forma, el rendimiento máximo de TCP viene dado por la ecuación (3-2), donde W_{max} es el tamaño máximo que puede alcanzar la ventana de transmisión [183].

$$X_{max} = \frac{W_{max}}{RTT} \tag{3-2}$$

3.2.2 Mecanismos de fiabilidad de TCP

TCP es un protocolo de transporte que garantiza la fiabilidad de la transmisión. Con tal fin dispone de una serie de mecanismos para actuar en caso de que se pierdan los segmentos. A lo largo de la historia de TCP, se han ido incorporando nuevos métodos de recuperación ante la pérdida de segmentos [178]. En los próximos párrafos, se describe en detalle cada uno de los distintos mecanismos.

Temporizador de retransmisión

TCP, como se ha comentado, numera los segmentos para poder reordenarlos si llegan en desorden y detectar pérdidas de segmentos. Un asentimiento (denominado ACK como abreviación) confirma la recepción no sólo de un segmento, sino de todos los bytes transmitidos hasta el número indicado en el asentimiento. Cuando el emisor envía un segmento, pone en marcha un temporizador denominado "temporizador de retransmisión". Si el asentimiento de dicho segmento llega antes del vencimiento del temporizador, se entiende que el segmento ha llegado correctamente. Sin embargo, si el temporizador vence antes de la llegada del asentimiento de dicho paquete, se entiende que el segmento se ha perdido. Tras el vencimiento del temporizador, el emisor ha de retransmitir el segmento y volver a iniciar el proceso de *slow start*.

Esta drástica reducción de la tasa de transmisión obedece a que TCP asume que si ha habido una pérdida de segmento es porque se ha producido una congestión en algún punto de la red, y para evitar saturarla, reduce la tasa de transmisión. Como TCP está pensado para las redes de paquetes con grandes *buffers*, una pérdida de segmentos es un indicativo de congestión en la red, y el objetivo de la recuperación es no sólo retransmitir los segmentos perdidos, sino evitar que vuelva a haber congestión en la red. En el caso de congestión (desbordamiento de colas), se perderán segmentos de muchos flujos, provocando una reducción masiva de la tasa de transferencia de las distintas fuentes TCP, solventándose la situación de congestión.

Fast Retransmit

Con el fin de reducir el tiempo de recuperación, se estandarizó la técnica denominada *fast retransmit* [180], la cual permite que un emisor conozca que se ha perdido un segmento incluso antes de que venza el temporizador de retransmisión. Cuando a un receptor TCP le llega un segmento cuyo número de secuencia no es el que espera, sino uno posterior y por lo tanto en desorden, debe enviar un ACK (asentimiento) duplicado inmediatamente, es decir, vuelve a asentir los mismos datos que ya asintió anteriormente. La técnica *fast retransmit* consiste en que en cuanto llegan al emisor tres asentimientos duplicados (cuatro ACKs idénticos), éste retransmite el segmento sin esperar a que venza el temporizador de retransmisión.

Fast Recovery

Empleando TCP con los mecanismos anteriores, las pérdidas provocan una reducción drástica del flujo de datos. Para evitar reducir de manera abrupta la tasa de transmisión cuando se detecta la pérdida de un segmento, se introdujo el algoritmo *fast recovery* [180] (recuperación rápida). Este algoritmo funciona de la siguiente forma:

1. Cuando llega el tercer ACK duplicado, se actualiza el valor de *ssthresh* a la mitad de *flightsize* (número de segmentos en tránsito, es decir que han sido enviados pero aún no han sido asentidos), se retransmite el segmento perdido, y se pone *cwnd* a *ssthresh* más 3 veces el tamaño máximo de segmento (el valor de este tamaño es un parámetro configurable).
2. Cada vez que llegue un nuevo ACK duplicado, se incrementa *cwnd* en el valor del tamaño máximo de segmento. En caso de que el número de segmentos en tránsito sea menor que el valor de la ventana de transmisión, se transmite un nuevo segmento.
3. Cuando llegue el ACK que asienta otro segmento, es decir, un ACK no duplicado, se actualiza *cwnd* con el valor de *ssthresh*. A continuación, se continua en la fase *congestion avoidance*.

Los mecanismos *fast retransmit* y *fast recovery* son eficaces cuando se pierde un único segmento. Sin embargo, cuando se pierden varios segmentos consecutivos, el algoritmo no se recupera con facilidad. Esto se debe a que después de la recuperación de la pérdida de un segmento, la ventana de congestión generalmente queda reducida a la mitad del valor que tenía la ventana de transmisión, con lo que en caso de que el asentimiento que llegue sea parcial, es decir, que no asienta a todos los segmentos que se habían enviado antes de entrar en *fast recovery*, la ventana es tan pequeña que no se pueden enviar nuevos segmentos, por lo que se ha de esperar al vencimiento del temporizador para continuar con la retransmisión. La versión de TCP que emplea *fast retransmit* y *fast recovery* tal y como se han descrito, se denomina TCP *Reno* [178].

Asentimiento selectivo

Con el objetivo de mejorar el comportamiento ante pérdidas múltiples, aparece el asentimiento selectivo (SACK, *Selective Acknowledgement*) [184] [185] [186]. Éste consiste en que en el asentimiento se incluye un campo adicional, en el que, cuando llegan segmentos en desorden, el receptor informa del número de segmento que ha llegado fuera de orden, así como cuántos segmentos contiguos a partir de él han sido recibidos. De esta manera, el emisor se hace una idea de cuántos segmentos se han perdido y cuántos han llegado al receptor. Con esta información, el emisor tiene información suficiente para retransmitir los segmentos que se han perdido sin esperar al temporizador de retransmisión.

TCP SACK, tal y como se especifica en la RFC 2018 [184], extendida en la RFC 2883 [185] para incluir casos especiales en los que se envía en el asentimiento selectivo información de segmentos duplicados, usa los mismos mecanismos para el control de la congestión que *Reno*. En la RFC 3517 [186] se define además un algoritmo para complementar *Reno* y poder aprovechar las ventajas del asentimiento selectivo. Como en el caso de *Reno*, cuando el emisor recibe el tercer ACK duplicado, entra en la fase *fast recovery*, y reduce a la mitad la ventana de congestión. En este caso, además, el emisor mantiene una nueva variable de estado, *pipe*, que indica el número de segmentos que se han enviado, pero no han recibido asentimiento (ni normal, ni selectivo). Se transmite (o retransmite) únicamente cuando *pipe* es menor que la ventana de congestión. La variable *pipe* se incrementa en uno cada vez que se envía un segmento, y se reduce cada vez que se reciben asentimientos con ciertas características. Si se recibe un asentimiento duplicado en el que su información de asentimiento selectivo indica que han llegado nuevos segmentos al receptor, entonces la variable *pipe* se reduce en una unidad. Si se recibe un asentimiento parcial, es decir, un asentimiento no duplicado pero que no confirma todos los segmentos enviados antes de entrar en *fast recovery*, se reduce en dos. La razón de reducirse en dos, es que se supone que se han enviado dos segmentos, el segmento original, y el segmento retransmitido. El objetivo de este mecanismo es no parar la transmisión, y continuar enviando nuevos segmentos además de retransmitir los segmentos perdidos. Una vez que ha llegado el asentimiento que confirma la recepción de todos los datos perdidos, incluyendo todos los datos que se habían enviado hasta el momento en el que se produjo la detección de la pérdida, se continúa con *congestion avoidance*.

Fast Recovery* con *New Reno

Los algoritmos de *fast retransmit* y *fast recovery,* empleados en la versión Reno, tienen problemas cuando se pierden múltiples segmentos de la ventana de transmisión. En el caso de estar disponible, SACK es la mejor opción para tomar una decisión inteligente en los casos de múltiples pérdidas. Sin embargo, en muchas ocasiones SACK no está habilitado. Para mejorar el comportamiento de *fast recovery,* IETF propuso una versión de TCP que modifica el comportamiento, denominada *New Reno* en la RFC 3782 [185], al ser una modificación de *Reno*. El comportamiento especial empieza tras la recepción de los tres ACKs duplicados que indican un situación anómala y termina cuando, o bien vence el temporizador de retransmisión, o bien llega un asentimiento que confirma la recepción de todos los paquetes que había en tránsito en el momento de la detección de la pérdida. En ese instante, se almacena en la variable *recover* el número de secuencia del último segmento transmitido, que servirá para identificar cuándo se ha superado la situación de pérdida múltiple. Después, el comportamiento es similar a Reno hasta el momento en que llega un asentimiento. Si el ACK es total, es decir, el número de secuencia almacenado en *recover* está confirmado en ese asentimiento, se finaliza el proceso de *fast retransmit*. Si por el contrario es un asentimiento parcial, eso significa que aún quedan segmentos perdidos, por lo que se retransmite un nuevo segmento, y se realiza una reducción parcial de la ventana. Para ello, en primer lugar se resta de la ventana el número de segmentos nuevos asentidos, y después

se suma uno, y si la ventana de recepción lo permite, se envían nuevos segmentos. De esta forma, cuando se haya recuperado del proceso, aproximadamente estarán en tránsito *ssthresh* bytes.

Resumen

En esta sección se han visto los principales mecanismos empleados por las distintas versiones de TCP. Hoy en día, las versiones de TCP más populares son TCP New Reno, o TCP Reno con SACK. Si se observan los sistemas operativos de usuario, tanto *Windows* como Mac OS X emplean versiones basadas en *Reno* y *New Reno* con SACK. Por otro lado *Linux* emplea también asentimiento selectivo, SACK, pero ha sustituido la forma de aumentar la venta de Reno por la de BIC (*Binary Increase Congestion*), que tiene un algoritmo de búsqueda binaria para incrementar la ventana de una manera más agresiva en redes de alta velocidad [187]. En la Tabla 3-1 se muestra un resumen de las versiones de TCP más conocidas, con las elecciones de cada versión.

	Tahoe	***Reno***	***NewReno***	***SACK***	**BIC*(Linux)***
Slow Start, congestion avoidance	Sí	Sí	Sí	Sí	Sí (incremento binario de la ventana)
Fast Retransmit	Sí	Sí	Sí	Sí	Sí
Fast Recovery	No	Sí (*Reno*)	Sí (*New Reno*)	Sí (*Reno* o *New Reno*)	Sí (similar a *Reno*)
ACK *selectivo*	No	No	No	Sí	Sí

Tabla 3-1 Principales versiones de TCP

Para hacernos una idea la variedad existente, se pueden consultar estudios como el de Medina *et al.* [188] en el que se analizan las versiones de TCP en servidores web a lo largo de varios años. Se puede ver cómo el casi el 70% de los servidores web tenían la posibilidad de emplear SACK (aunque sólo el 27% eran capaces de emplearla correctamente).

No sólo cada sistema operativo es diferente, sino que además cada equipo tiene una configuración personalizada, especialmente entre los servidores, que son mantenidos por administradores avanzados, que son capaces de manipular muchos parámetros. Entre estos parámetros, es posible encontrar la ventana inicial de transmisión, el uso de SACK, la ventana máxima de transmisión, etc. En los equipos de usuario, en la mayoría de los casos éstos emplean los valores por defecto del sistema operativo.

Por otro lado, la mayoría de los equipos, tanto de usuario como de servidores, tienen limitada la ventana de transmisión máxima. En el caso de los servidores, al margen de las malas prácticas de configuración, al tratarse de equipos accedidos por muchos clientes, se limita la ventana para poder compartir el ancho de banda y que este no sea absorbido por una única conexión. En el caso de los equipos de usuarios residenciales, en la gran mayoría de los casos, la ventana máxima es la que viene establecida por defecto en el sistema operativo.

3.2.3 Nuevas versiones de TCP

TCP ha sido un éxito al funcionar sobre todo tipo de redes, e interoperar con versiones heterogéneas del protocolo y además ser uno de los motores de Internet. Por conseguir esta flexibilidad, TCP ha tenido que renunciar a conseguir un rendimiento óptimo en redes de muy alta velocidad, en redes con gran producto ancho de banda retardo y en medios con muchas pérdidas, como los entornos inalámbricos. Desde hace más de una década, ha

habido continuas propuestas para mejorar TCP y adecuarlo a nuevos ambientes. Se pueden sintetizar las propuestas en 3 áreas, que se muestran a continuación.

El primer área de mejora es hacer TCP más adecuado para entornos de alta velocidad y aplicaciones de *grid computing* [189] (Fast TCP [190], HSTCP [191], STCP [192], HTCP [193], BIC [187] y CUBIC [194]). Las ideas principales se recopilan en la Tabla 3-2.

Versión TCP	Ideas principales
***Fast* TCP** [190]	La indicación de si hay congestión viene dada por el retardo. Funciona bien para redes de alta velocidad y grandes distancias.
***High Speed* TCP (HSTCP)** [191]	HSTCP modifica el esquema de cómo se abre la ventana en cada ronda, y cómo se cierra en función del tamaño absoluto de la ventana. Cuando la ventana es pequeña, HSTCP se comporta como TCP normal, pero cuando la ventana es grande, se incrementa más rápido, y decrece más lentamente.
***Scalable* TCP (STCP)** [192]	La principal idea de *Scalable* TCP es que el decremento de la ventana en caso de congestión es más suave (en concreto, 0,875 veces el valor de la ventana frente a la tradicional reducción a la mitad)
HTCP [193]	El factor de incremento de la ventana depende del tiempo transcurrido desde el último evento de congestión (en vez del valor de la ventana). Asimismo, para aumentar la equidad entre flujos competidores, el incremento de la ventana se ajusta según el RTT.
BIC [187]	BIC incluye un algoritmo de búsqueda binaria para incrementar la ventana de una manera más agresiva.
CUBIC [194]	CUBIC está inspirado en BIC y HTCP con el objetivo de aumentar la justicia (igualdad en el reparto del ancho de banda entre flujos con distintos RTT). La función de incremento de la ventana es cúbica y depende del tiempo desde el último evento de fallo y es menos agresiva y más sencilla que BIC.
Byte counting [195]	La ventana de TCP crece en función del número de bytes en vez del de segmentos como hace tradicionalmente. A pesar de estar especificada en una RFC, no se emplea prácticamente [188].

Tabla 3-2 Resumen de las principales ideas de TCP de alta velocidad

El siguiente área de mejora consiste en hacer TCP más robusto ante eventos que no impliquen congestión, pero que el TCP tradicional los interpreta como tales. Por ejemplo el reordenamiento de paquetes y retardos diferentes se pueden malinterpretar fácilmente. Las ideas más importantes se muestran en la Tabla 3-3.

Versión TCP	Ideas principales
TCP-*Peach* [196]	Emplea *sudden start* (enviar paquetes de prueba a la red para incrementar la ventana más rápido) y *rapid recovery* (emplea paquetes especiales de prueba en la recuperación para artificialmente conseguir más ventana). El problema es que se necesitan modificaciones en los *routers*.
TCP-NCR [197]	TCP - *Non Congestion Robustness* (NCR) son un conjunto de modificaciones a un emisor TCP para dar robustez. Son modificaciones sobre SACK
TCP-PR [198]	TCP-*Packet Reordering* (PR) detecta pérdidas usando temporizadores en vez de ACKs duplicados. Útil en redes en las que se desordenan los paquetes.
TCP-*Aix* [199]	TCP *Aix* separa el control de congestión de la recuperación por pérdida de segmento. Después de una pérdida, no reduce drásticamente la ventana, ni tampoco satura a la red, sino que espera un poco más de tiempo a recibir *feedback* del comportamiento para tomar una mejor decisión.

Tabla 3-3 Resumen de ideas de mejora de TCP ante eventos de no congestión

Por último, se han realizado propuestas para mejorar el rendimiento de TCP en entornos especiales, principalmente para optimizar el comportamiento en redes

inalámbricas [200]. Una recopilación de las modificaciones de TCP propuestas para entornos inalámbricos se muestra en la Tabla 3-4.

Versión TCP	Ideas principales
TCP *Vegas* [201]	Mide el retardo de los paquetes para calcular cual sería la tasa de transmisión óptima de la red a partir del retardo mínimo. Comparando con la tasa lograda, ajusta la ventana.
TCP *Veno* [202]	Intenta distinguir si la causa de una pérdida es aleatoria (un error común en un entorno inalámbrico), o por congestión. Si es por congestión, se comporta como TCP Reno, y en caso contrario, lo ignora.
TCP *Westwood* [203]	Estima el ancho de banda disponible mediante la tasa de recepción de ACKs. Es una modificación de *NewReno* en el transmisor.

Tabla 3-4 Resumen de las principales ideas de versiones de TCP útiles en entornos inalámbricos

Sin embargo, no hay que olvidar que durante la historia de TCP, aunque se han propuesto en la literatura muchas versiones de TCP dedicadas a resolver problemas con una determinada tecnología de red, o a obtener el máximo rendimiento posible en un enlace, estas no han tenido una gran aceptación debido a que TCP ha de funcionar en todo tipo de redes, ya que un ordenador o dispositivo puede conectarse a múltiples tipos de red. Por tanto, a no ser que se trate de un medio muy acotado, se asumirá que no habrá una versión de TCP específica para cada tipo de red.

Consideraciones sobre la versión de TCP a estudiar en la tesis

Ante la variedad de comportamientos de TCP, esta tesis se centra en estudiar TCP con los rasgos comunes a las versiones más importantes, teniendo en cuenta el marco temporal en el que se plantea este trabajo. Por tanto, se considera que una versión de TCP con asentimientos selectivos estará presente durante la próxima década. El número de segmentos iniciales o el ritmo de crecimiento de la ventana pueden variar, permitiéndose comportamientos más agresivos con el tiempo, pero no cambian la esencia del protocolo. Se tendrá en cuenta la diversidad de configuraciones con las que los clientes y servidores se pueden encontrar, y que limitarán el rendimiento. Por tanto, se evaluarán los parámetros sintonizables en los sistemas operativos, y se estudiará cómo pueden influenciar en el comportamiento en redes ópticas de conmutación de ráfagas, para unas mejores prácticas a la hora de escoger los parámetros.

3.3 Comportamiento de TCP en redes OBS

En esta sección se analiza el comportamiento básico de TCP en redes OBS desde un punto de vista cualitativo, y se realizan simulaciones de TCP en OBS mediante la herramienta OPNET Modeler [12] para ilustrar dicho comportamiento. El análisis se centrará en las versiones de TCP Reno, por ser la versión que ilustra el comportamiento general de TCP (el resto de versiones son modificaciones más o menos agresivas) y SACK (asentimiento selectivo, que permite obtener información adicional en caso de pérdidas múltiples), opción ampliamente utilizada en todos los sistemas operativos modernos. Finalmente, se realiza un recorrido por los estudios de la literatura con respecto a TCP sobre OBS y se analizan los aspectos abiertos que se estudian en esta tesis.

3.3.1 Problemática de TCP para entornos de alta velocidad

Las redes de alta velocidad como OBS sufren de un problema conocido como el producto ancho de banda retardo alto. Al ser TCP un protocolo de ventana deslizante, solo pueden estar en vuelo tantos datos como indique la ventana. Si el tamaño de ésta es menor que el producto ancho de banda por retardo, se está desperdiciando capacidad. La ventana

de TCP estándar es de 2^{16}–1 bytes (64 kbytes), muy baja para redes ópticas de alta velocidad. Las extensiones definidas en la RFC 1323 [204] permiten ampliar la ventana hasta 4 Gbytes. Sin embargo, el valor por defecto en los equipos de usuario es normalmente bajo, siendo de 512 kbytes en los casos más altos (ej. Mac OS X), y en los servidores se suele limitar para repartir el ancho de banda entre los usuarios.

Otro gran problema es el tiempo que tarda en crecer la ventana hasta llegar al tamaño ideal en ese medio [187]. Por ejemplo, con un RTT de 100 ms, para conseguir llegar a una tasa de transmisión de 10 Gbps se tiene llegar a una ventana de 83333 segmentos de 1500 bytes. En este caso, para llegar desde la mitad de la ventana a la ventana máxima, a un incremento de un segmento por RTT, se tardaría más una hora en conseguir llegar a la tasa máxima. Para reducir ese tiempo, las versiones de TCP incorporan cada vez incrementos más agresivos en el periodo de *congestion avoidance*, como el caso de BIC TCP [187], *High Speed* TCP [191], etc. Hay que recordar que en la Tabla 3-2 de la Sección 3.2 se mostró un resumen de estas versiones.

3.3.2 Influencia en el retardo

Los datagramas IP, los cuales contienen segmentos TCP, se agrupan en ráfagas en los nodos de ingreso de la red OBS. Este ensamblado de los paquetes introduce un retardo adicional, debido al tiempo de espera hasta que se completa la formación de la ráfaga, que reduce el rendimiento de TCP. El efecto es similar al de incrementar el RTT, que impacta en la tasa de transmisión como indica la ecuación (3-1) del apartado 3.2. En la literatura de TCP sobre OBS, este efecto se conoce como "penalización por retardo" [205]. El valor concreto del incremento del retardo dependerá del algoritmo de ensamblado utilizado y su configuración. Un problema adicional es que, al enviarse los segmentos TCP en ráfagas, el retardo experimentado por cada uno de ellos varía. TCP emplea la medida del valor de retardo (su media y varianza) para algunos cálculos, como por ejemplo el del temporizador de retransmisión. Por tanto, una gran variación en el retardo, puede provocar efectos adicionales inesperados en el cálculo de los temporizadores.

3.3.3 Influencia de los descartes de ráfagas

Sin embargo, el aspecto que más influye en el rendimiento de TCP es la pérdida de ráfagas debido a la contienda en los nodos intermedios. La pérdida de una ráfaga implica generalmente la pérdida de varios segmentos consecutivos pertenecientes a un mismo flujo. Tal y como se comentó en el apartado anterior, TCP tiene dos maneras de detectar la pérdida de un segmento, bien mediante la recepción de asentimientos duplicados, bien mediante el vencimiento de un temporizador. En OBS, las ráfagas contienen varios segmentos TCP, con lo que dependiendo de la cantidad de segmentos que se transmitan con relación al tamaño de la ventana de transmisión, TCP tiene una reacción u otra. Si en una ráfaga se transmite una ventana de TCP completa, vencerá el temporizador de retransmisión y comenzará la fase de *slow start*. En cambio, si no se transmite una ventana completa, llegarán al receptor segmentos fuera de orden, con lo que enviará asentimientos duplicados, que pondrán en marcha los mecanismos de recuperación de TCP explicados en la sección anterior. Por lo tanto, en el caso de que se pierda una ráfaga, la influencia en el rendimiento de TCP viene dada, no por el número total de segmentos que viajan en dicha ráfaga, sino por el número total de segmentos de un mismo flujo. Se pueden tener pérdidas en una ráfaga desde 1 hasta n segmentos de un mismo flujo, donde n es el número máximo de segmentos que caben en la ventana de transmisión. Principalmente son tres los casos interesantes, que se abordan en los apartados siguientes apoyados en simulaciones con la herramienta OPNET Modeler. Para ello, se ha implementado un ensamblador de ráfagas basado en temporizador y se ha simulado la transferencia de un fichero a través de una red OBS.

Pérdida de ráfagas conteniendo un solo segmento de un mismo flujo TCP

En el caso de que la ráfaga que se pierde contenga un único segmento de un mismo flujo TCP, y la ventana de transmisión sea mayor que un segmento, al receptor van a llegar segmentos fuera de orden, que provocarán el envío de asentimientos duplicados, uno por cada segmento fuera de orden. El valor de la ventana de transmisión cuando se recibe el primer duplicado determina cuántos segmentos pueden estar en tránsito. Si este número es mayor o igual que cuatro, entonces llegarán al receptor tres segmentos fuera de orden, generando tres asentimientos duplicados, que al llegar al emisor hacen que éste inicie sus mecanismos de protección ante pérdidas. En este caso, los asentimientos duplicados van a llegar siempre antes de que venza el temporizador, y TCP se va a poder recuperar en poco tiempo y sin reducir tan drásticamente la tasa de transmisión como lo haría si venciera éste. Es de interés destacar que en los primeros instantes de un flujo TCP, la ventana tiene menos de cuatro segmentos, con lo que una pérdida de una ráfaga al comienzo de la transmisión provocaría el vencimiento del temporizador con la consiguiente penalización en el rendimiento.

Para estudiar este escenario con TCP Reno, se utiliza la Figura 3-2, donde se muestra la evolución de la ventana de congestión en el tiempo para la transmisión de ráfagas conteniendo un único segmento de un flujo en cada ráfaga, y en la que se produce la pérdida de una ráfaga.

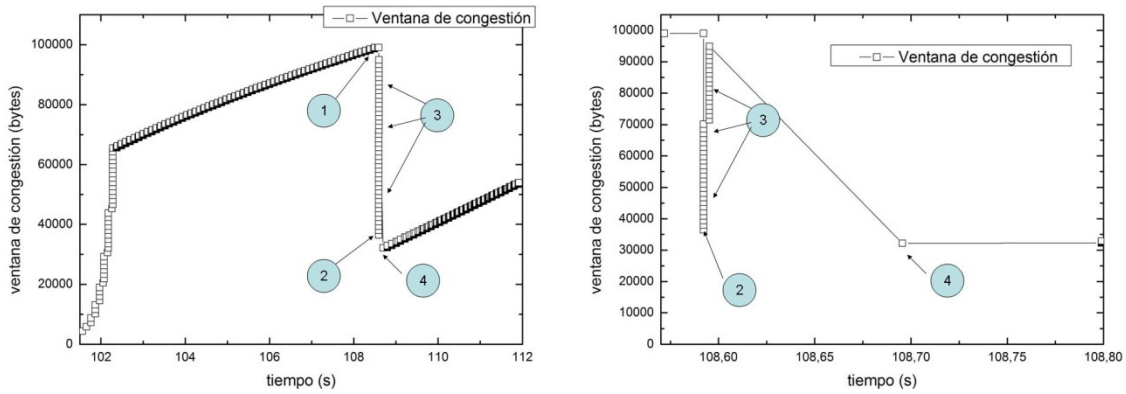


Figura 3-2 Pérdida de una ráfaga conteniendo un solo segmento de un mismo flujo con TCP Reno. (a) Transmisión completa (b) Detalle de la zona de recuperación.

En la Figura 3-2.a se muestra la transmisión completa, y la Figura 3-2.b el detalle de la recuperación. En ambas aparecen numerados los puntos de interés. En el punto ① se produce la pérdida de la ráfaga. Cuando llegan los tres asentimientos duplicados, entran en acción los mecanismos *fast retransmit* y *fast recovery* explicados en la sección anterior, con lo que primero se reduce la ventana a *flightsize*/2+3 MSS (*Maximum Segment Size*, tamaño máximo de segmento) (punto ② de la Figura 3-2), se retransmite el segmento, se incrementa la ventana cada vez que llega un asentimiento (punto ③), y finalmente se reduce la ventana a la mitad del número de segmentos en tránsito (punto ④).

En el caso de SACK (Figura 3-3), tras la pérdida de paquete (punto ①), cuando llegan los tres asentimientos duplicados (punto ②), se reduce la ventana a la mitad del número de segmentos en tránsito y se retransmite. A continuación van llegando más asentimientos duplicados (punto ③). Un RTT más tarde del envío del segmento retransmitido, llega el asentimiento que confirma su recepción, con lo que se continúa creciendo con *congestion avoidance* (punto ④).

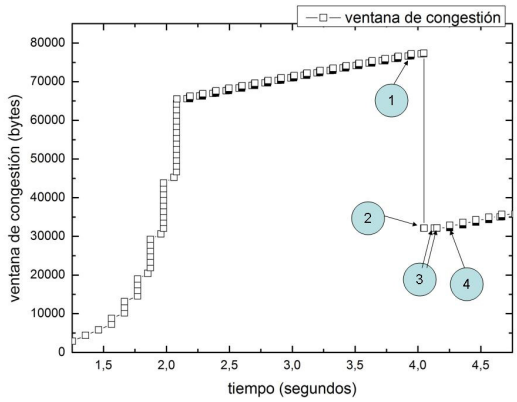


Figura 3-3 Pérdida de una ráfaga conteniendo un solo segmento de un mismo flujo con TCP SACK.

Tanto Reno como SACK consiguen recuperarse de la pérdida de un segmento en poco tiempo, un RTT desde la detección de los ACKs duplicados. Después de la recuperación en ambas versiones el resultado es que la ventana de congestión se reduce a la mitad del valor que tenía la ventana de transmisión antes de producirse las pérdidas. Por lo tanto, en este caso, el asentimiento selectivo no aporta ventajas significativas en el rendimiento.

Pérdida de ráfagas conteniendo dos o más segmentos de un mismo flujo TCP

Cuando se pierde una ráfaga que lleva dos o más segmentos de un mismo flujo, pueden darse dos escenarios posibles para la detección de la pérdida. Si la ventana de transmisión tiene un tamaño mayor o igual que el número de segmentos perdidos más tres, entonces el emisor acabará recibiendo tres asentimientos duplicados, detectando así la pérdida. Si la ventana de transmisión no alcanza dicho valor, entonces vencerá el temporizador de retransmisión. El caso de mayor interés, en el que Reno y SACK muestran diferente comportamiento, es el de la recepción de tres ACK duplicados, que se explica mediante un ejemplo. Se supone que se pierde una ráfaga que contiene dos segmentos de un flujo TCP y que la siguiente ráfaga transmitida (con nuevos segmentos) llega correctamente. La Figura 3-4 muestra el comportamiento de Reno y la Figura 3-5 el de SACK.

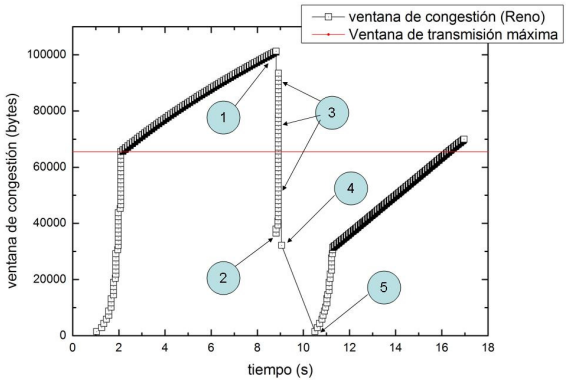


Figura 3-4 Pérdida de una ráfaga conteniendo dos segmentos de un mismo flujo con TCP Reno.

En el caso de TCP Reno, tras producirse la pérdida (punto ①), el emisor va a retransmitir el segmento, y va a reducir la ventana de congestión a la mitad de *flightsize* más tres segmentos (punto ②). A medida que van llegando nuevos asentimientos duplicados, la ventana de congestión se va incrementando en un segmento (punto ③). Sin embargo, no puede enviar nuevos segmentos, ya que la ventana de transmisión es superior a la ventana de congestión, ya que tiene almacenados todos los segmentos que se han enviado después de la pérdida. A continuación, llega el asentimiento del primer segmento perdido, que ha

sido generado tras la recepción del segmento retransmitido, y la ventana de congestión se reduce (punto ④). La ventana de transmisión en este punto, con el número de segmentos que hay en tránsito, no permite el envío de ningún segmento, por lo tanto el emisor permanece inactivo hasta que vence el temporizador de retransmisor asociado al segundo segmento perdido (punto ⑤), momento en el que se retransmite el segmento y se vuelve a *slow start*.

En el caso de SACK, como se muestra en la Figura 3-5, tras la pérdida de la ráfaga que lleva dos segmentos del flujo considerado (punto ①), cuando ocurre la detección de los tres asentimientos duplicados, se reduce la ventana a la mitad del número de segmentos en tránsito (punto ②). A medida que llegan los asentimientos duplicados con información de asentimientos selectivos, la variable *pipe* se va actualizando, como se ha explicado en el apartado 3.5, pero la ventana de congestión permanece constante (punto ③). Con la información de los asentimientos selectivos, se retransmiten los segmentos perdidos. Cuando llega el asentimiento que confirma la recepción de todos los seguidos, se continúa con la fase de *congestion avoidance* (punto ④).

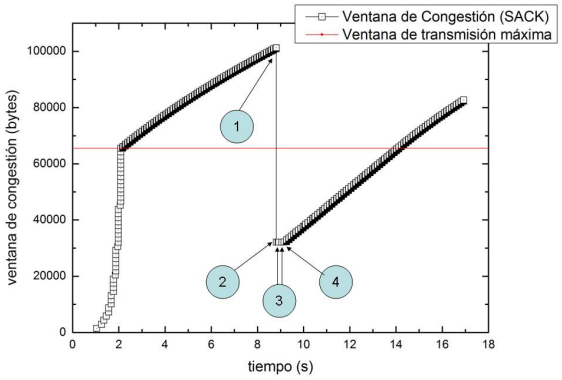


Figura 3-5 Pérdida de una ráfaga conteniendo dos segmentos de un mismo flujo con TCP SACK.

El caso que se ha explicado corresponde a la pérdida de una ráfaga que lleve dos segmentos pertenecientes a un mismo flujo TCP, pero si la ráfaga lleva más de dos segmentos (siempre que lleguen tres ACK duplicados) el comportamiento es similar.

En este escenario, la primera versión de TCP analizada, Reno, se recupera mediante el vencimiento del temporizador y continúa la transmisión con *slow start*, reduciendo drásticamente la tasa de transmisión. Sin embargo SACK se recupera en aproximadamente un RTT y continúa la transmisión con la ventana reducida a la mitad de lo que tenía antes en vez de reducirla a un segmento como en Reno. Por lo tanto, SACK ofrece una mejora significativa de rendimiento, por lo que su uso es altamente recomendable al transmitir en redes OBS.

Pérdida de ráfagas conteniendo una ventana completa de un mismo flujo TCP

En el caso de que en una ráfaga contenga todos los segmentos enviados en una ventana de transmisión TCP, la pérdida de la misma implica el vencimiento del temporizador. Esto se debe a que el emisor TCP, tanto Reno como SACK, no recibe asentimientos duplicados que le informen de dicha pérdida (Figura 3-6).

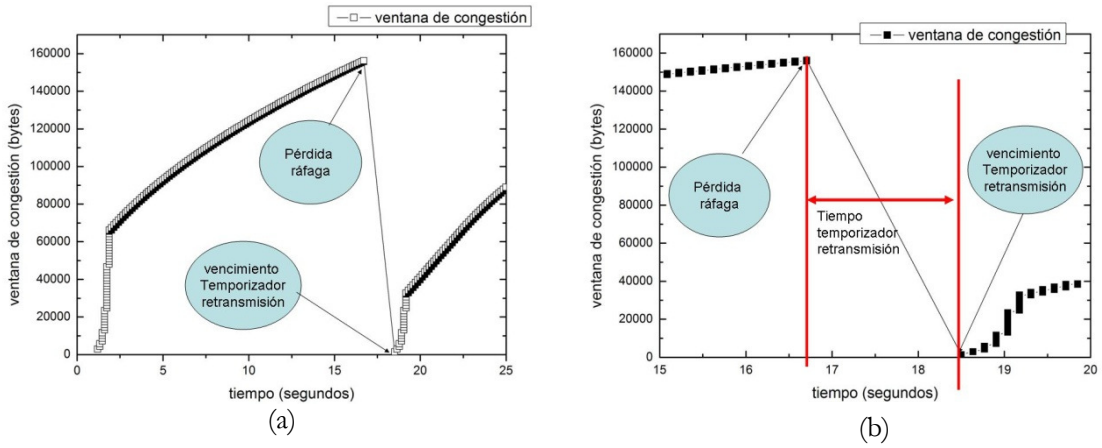


Figura 3-6 Pérdida de ráfaga conteniendo una ventana completa: (a) intervalo de tiempo completo, (b) detalle del instante de la pérdida

Resumen

Como se ha visto, una pérdida de ráfagas puede implicar la pérdida de varios segmentos consecutivos. Según el número de segmentos perdidos, y la versión de TCP, una pérdida de ráfaga puede tener una consecuencia más o menos importante en la tasa de transmisión TCP. Una vez se ha visto y entendido el comportamiento cualitativo, se presenta un resumen de los principales estudios de TCP sobre OBS.

3.3.4 Rendimiento de TCP en OBS

El primer estudio en la literatura sobre rendimiento de TCP en redes OBS en las que hay pérdidas de ráfagas fue llevado a cabo por Detti y Listanti [50], y en él se modela el comportamiento de un flujo TCP Reno que atraviesa un ensamblador OBS con una estrategia de ensamblado de temporizador fijo, basándose en un modelo anterior de TCP desarrollado por Padhye *et al.* [206]. En este estudio se observaron dos problemas, uno la reducción de *throughput* debido al incremento del retardo en el ensamblador, que se ha mencionado en el apartado 3.3.2, y otro que consiste en la reducción del rendimiento debido a las pérdidas de segmentos consecutivos, cuyo comportamiento detallado con ayuda de la simulación se ha mostrado en el apartado 3.3.3.

Beneficio de correlación / *Delayed First Loss* (DFL)

En este primer estudio se demuestra la existencia de un efecto positivo denominado "beneficio de correlación", por el que si se compara una red OBS (en la que se agregan varios segmentos en una ráfaga) en la que hay una determinada tasa de pérdida de segmento (siendo las pérdidas de segmento debidas únicamente a la pérdida de ráfagas, que se producen de manera aleatoria), y una red de paquetes con la misma tasa de pérdida de segmento (donde estas pérdidas de segmento se producen de manera aleatoria), el rendimiento que se obtiene en la red OBS es mayor. Esto se debe a que las pérdidas de segmento en OBS están correladas, ya que cuando se pierde un segmento se debe a que se ha perdido una ráfaga, y consecuentemente se habrán perdido más segmentos. TCP interpreta la pérdida de múltiples segmentos como un único evento de congestión. Por tanto, en la red OBS habría un menor número de eventos de congestión que en la red de paquetes (en la que cada pérdida de paquete equivale a un evento de congestión). Este comportamiento ha sido observado también por otros estudios, donde ha recibido diferentes nombres, como ganancia DFL (*Delayed First Loss*) [51] o efecto de amplificación [205].

Influencia de la velocidad de las fuentes de tráfico

El estudio de Detti y Listanti [50] y los de Yu *el al.* [66] [52] introducen la noción de fuentes TCP 'lentas', 'medias' y 'rápidas', según la relación entre el valor del temporizador de ensamblado y el ancho de banda de acceso y que determinará si en una ráfaga viaja un segmento de un flujo, varios, o la ventana completa y determinará la reacción de TCP ante la pérdida de la ráfaga. Si el número de segmentos de un mismo flujo en cada ráfaga es elevado, cuando hay una pérdida de ráfaga, el tiempo de recuperación es alto, lo que conlleva una bajada del rendimiento. Ahora bien, si el número de segmentos de un mismo flujo en cada ráfaga es uno, el comportamiento de TCP es similar al habitual en las redes actuales. Y si se está en un caso intermedio, siempre que se use SACK el comportamiento también se acercará al de las redes actuales, ya que no se ve afectado exceso por perdidas de varios segmentos. La casuística en el caso intermedio es muy amplia, y la reacción de TCP dependerá de cuántos segmentos no se hayan perdido y cual sea la ventana en ese momento.

Según los criterios establecidos por Yu *et al.* [52] y Detti y Listanti [205], suponiendo que el cuello de botella es el enlace de acceso, y este tiene una velocidad λ, medida en segmentos por segundo, y suponiendo que se emplea una estrategia de ensamblado de temporizador con periodo de ensamblado T_b, medido en segundos, se distinguen tres tipos de flujo:

- Flujo rápido: Si se verifica que $\lambda \cdot T_b \geq W_{max} - 1$, entonces todo el contenido de la ventana de transmisión cabe en una ráfaga.
- Flujo medio: Si se verifica que $1 \leq \lambda \cdot T_b < W_{max} - 1$, entonces el contenido de la ventana se reparte en varias ráfagas.
- Flujo lento: Si se verifica $\lambda \cdot T_b < 1$, entonces una ráfaga contiene un único segmento.

Sin embargo, el principal inconveniente de esta clasificación estricta en función del número de segmentos que cabrían teóricamente en una ráfaga con un ensamblado de un único flujo basado en temporizador es que se basa en un modelo con un único cliente y servidor, por lo que hay que tener cuidado en su interpretación. En esta tesis, se emplea la misma clasificación, pero en función del número de segmentos que caben cuando se ensamblan varios flujos (no un único flujo).

Número de clientes y servidores en los estudios de la literatura

El principal inconveniente del estudio de Detti y Listanti [50] al igual que el de Yu *et al.* [66] [52] es el empleo de un modelo de red muy simple, con un único cliente y servidor TCP conectados por un enlace de un solo sentido con pérdidas. Posteriormente, mejoran el modelo teórico considerando ambos sentidos en el enlace [205] y, adicionalmente consideran un caso de simulación con múltiples fuentes. Los resultados de simulación y analíticos divergen cuando se incluyen varias fuentes de tráfico en la simulación, sin proporcionar apenas explicación. Por tanto, parece evidente la necesidad de llevar a cabo estudios más extensivos, tanto en simulación como en estudio analítico cuando se consideran varias fuentes TCP.

En estos primeros estudios se hace mucho hincapié en la pérdida de ráfagas en OBS y reducción de rendimiento en TCP por las mismas, dando una imagen negativa de TCP y OBS. Sin embargo, en un escenario realista, Pleich *et al.* [207] llegan a la conclusión de que el tráfico TCP muestra un comportamiento muy robusto en OBS. En su estudio, se presentan resultados de simulación de TCP Reno sobre OBS incluyendo hasta 900 fuentes TCP. Consideraron que probablemente la sensibilidad de TCP mostrada en artículos anteriores, se deba al empleo de pocas fuentes y versiones viejas de TCP.

Comportamiento de TCP con distintas características de OBS

La mayoría de los estudios de TCP sobre OBS se centran en descubrir cuál es el tiempo de ensamblado que proporciona un rendimiento óptimo. Para un ensamblado basado en temporizador, en los estudios de Yu *et al.* [52] [51] los valores se obtienen en el rango de milisegundos, en los que el retardo introducido es bajo y a la vez se obtiene una buena ganancia de correlación.

Posteriormente, se empieza a estudiar otros aspectos de OBS. Por ejemplo, el modelo de Yu *et al.* es ampliado por Zhang *et al.* para tener en cuenta mecanismo de retransmisión de ráfagas [131]. La particularidad de las retransmisiones a nivel OBS es que pueden interferir con los mecanismos de TCP, ya que el receptor, cuando reciba los segmentos fuera de orden de la ráfaga retransmitida puede interpretar problemas en la red y reducir su tasa de transmisión, como si de una pérdida se tratase. En el modelo propuesto, Zhang *et al.* separan la probabilidad de pérdida de ráfaga en dos nuevos conceptos, la probabilidad de que suceda una contienda y probabilidad de descarte de ráfaga.

Estudios experimentales de TCP sobre OBS

El estudio de TCP sobre OBS no se ha centrado únicamente en simulación y modelado analítico. En el *testbed* de BUPT, que se describe en la sección 2.11, se han realizado un amplio conjunto de experimentos de TCP sobre OBS que corroboran los resultados de simulación [150] [208] [145]. El principal problema de los experimentos se asemeja al de los estudios teóricos de TCP sobre OBS. Debido a la limitación de nodos en los prototipos, las pruebas se reducen a unos pocos flujos. Por otro lado, como los prototipos se han planteado como una prueba de concepto, cuentan con componentes limitados, que hacen que el rendimiento sea menor que el esperado. De hecho, la principal conclusión que se obtiene es la robustez del tráfico TCP. A pesar de las pérdidas de ráfaga por el hecho de disponer un prototipo limitado, el rendimiento obtenido era adecuado para la aplicación empleada (ejemplo, transmisión de vídeo, etc.).

3.4 Resumen

En este capítulo se ha descrito el comportamiento fundamental de TCP y se han visto las diferencias entre las distintas versiones e implementaciones. Cabe destacar que aunque cada implementación particular de TCP realiza una elección entre los algoritmos disponibles, la tendencia es hacia el uso de asentimientos selectivos (SACK), y, muy progresivamente, hacia la aceptación de crecimientos de ventana cada vez más agresivos. Es importante mencionar que dentro de la misma implementación, TCP se puede configurar de manera particular, por lo que hay que tener en cuenta una gran variedad de configuraciones posibles.

TCP en redes OBS se enfrenta al problema clásico de las redes de alta velocidad, al que se le añade una penalización por el retardo introducido en el proceso de ensamblado y un problema debido al descarte de ráfagas, que repercute en la pérdida de segmentos TCP. Mediante la simulación con OPNET Modeler, en este capítulo se han analizado cualitativamente los distintos escenarios de pérdidas, en los que se ha descrito con detalle el comportamiento de TCP.

Los escenarios en los que la tasa de transmisión de TCP se ve más afectada son aquellos en los que TCP necesita esperar al vencimiento del temporizador de retransmisión. En el caso de pérdidas de múltiples segmentos, empleando asentimientos selectivos, TCP es capaz de realizar esa recuperación en caso de pérdidas múltiples como si de una pérdida simple se tratase.

Los estudios teóricos, mediante simulación y experimentales de la literatura han logrado estudiar con detalle la situación con un solo flujo. Estos estudios se han centrado en comparar las versiones de TCP, pero no han estudiado el aspecto de otros detalles de TCP, como por ejemplo el asentimiento retardado, que se estudia en el próximo capítulo. Asimismo, los estudios de la literatura mencionan el diferente comportamiento cuando se ensamblan múltiples flujos, pero no se llega a estudiar en profundidad para determinar el impacto en dicho caso. Por ello, uno de los aspectos fundamentales a estudiar en el próximo capítulo es el comportamiento de un flujo TCP cuando se encuentra con más tráfico.

Capítulo 4

Impacto del ACK retardado y el tráfico de fondo en TCP sobre OBS

4.1 Introducción

En el capítulo anterior se ha estudiado de manera cualitativa y mediante simulación cómo se combinan TCP y OBS, con énfasis en las pérdidas de ráfagas en las principales versiones de TCP. Sin embargo, hay muchos parámetros de TCP que se pueden modificar en las implementaciones de TCP, y pueden tener impacto en el rendimiento según la tecnología de red que se emplee. En este capítulo se estudia, en primer lugar, el impacto de un aspecto concreto de TCP, el asentimiento retardado.

Asimismo, se ha visto que distintos estudios de la literatura, como el de Pleich *et al.* [207], muestran que es necesario estudiar TCP con múltiples fuentes de tráfico, pero no llegan a una conclusión clara acerca de la influencia de ensamblar múltiples flujos. Por tanto, en este capítulo, mediante simulación, se estudia el comportamiento de un flujo TCP que se ensambla en un ensamblador OBS junto con tráfico perteneciente a otras conexiones, que se denomina tráfico de fondo, y se analiza qué impacto tiene en el rendimiento de TCP en redes OBS.

4.2 Asentimiento retardado en TCP

En las primeras versiones de TCP [177] se envía un asentimiento por cada segmento recibido. Este comportamiento se modificó posteriormente en la RFC 1122 [209], que detalla el algoritmo del asentimiento retardado (*delayed* ACK). Cuando llega un segmento TCP, no se envía el asentimiento de manera inmediata. Se espera a la llegada de un segundo segmento, que debe de recibirse en menos de 500 ms desde la llegada del segmento anterior. Aunque 500 ms es el valor indicado en la RFC, la mayoría de las implementaciones tienen como valor por defecto 200 ms (por ejemplo Windows). Este parámetro, aunque configurable en todos los sistemas operativos, suele dejarse siempre en su valor por defecto.

Para poder entender el comportamiento del algoritmo, se muestra un ejemplo en la Figura 4-1. En la parte izquierda de la figura se muestra un diagrama temporal con el flujo de segmentos TCP en una transferencia de datos en la que no está activado el asentimiento retardado. Cada flecha representa el envío de un segmento. Se envían datos de izquierda a derecha, mientras que de derecha a izquierda se envían únicamente los asentimientos de dichos datos. En el ejemplo representado en la figura, se envían 3 segmentos de datos, el primero con los bytes numerados del 1 al 1460, el segundo con los bytes numerados del 1461 al 2920 y el tercero con los bytes del 2921 al 4380. En la parte derecha se muestra el mismo intercambio de mensajes, pero en el caso de que TCP emplee asentimientos retardados. En la parte izquierda de la figura se observa que, al no estar activado el

asentimiento retardado, cada vez que llega un segmento con datos, se envía un asentimiento, en el que se indica el siguiente byte que se espera. Por tanto, hay una relación uno a uno entre segmento y asentimiento. En cambio, en el caso del asentimiento retardado en la parte derecha de la figura, cuando llega el primer segmento, el que contiene los bytes del 1 al 1460, no se envía inmediatamente ningún asentimiento, sino que se espera a que llegue otro segmento. Cuando llega el siguiente segmento, se envía el asentimiento, en el que se indica que se espera el byte 2921 (que será el primero del tercer segmento), con lo que se confirman los dos primeros segmentos. A continuación, cuando llega el segmento con bytes 2921 al 4380, no se envía ningún asentimiento de manera inmediata. En este caso, no llega después ningún otro segmento. La implementación habitual es basarse en los temporizadores incluidos en el sistema operativo. El ejemplo de la figura representa una implementación de Windows. En Windows, hay un temporizador que vence periódicamente cada 200 ms, que se ha denominado "*tick* de reloj" en la Figura 4-1. TCP, en vez de poner en marcha un temporizador de 200 ms cada vez que recibe un segmento, emplea este temporizador periódico del sistema operativo. Así, cuando vence ese temporizador y hay un segmento sin asentir, se envía el asentimiento de dicho segmento.

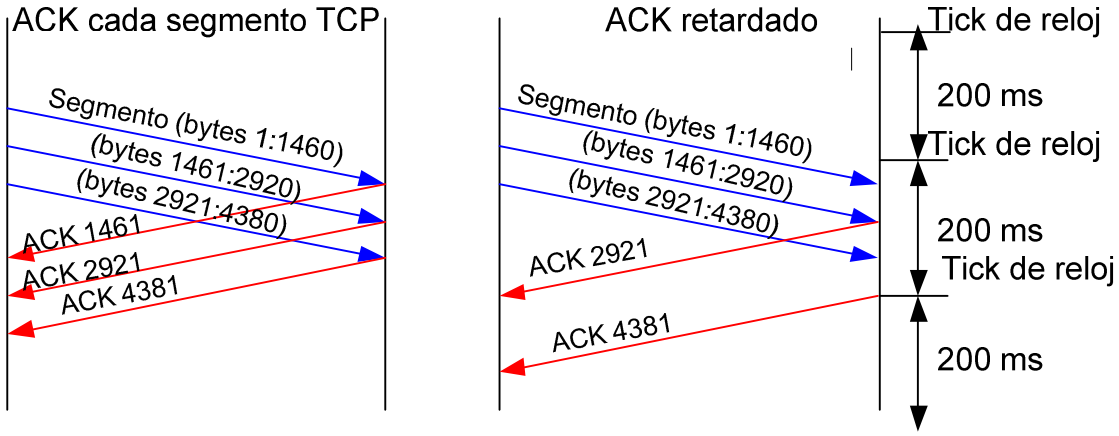


Figura 4-1 Funcionamiento de ACK retardado

El algoritmo del asentimiento retardado se introdujo para reducir la carga en la red en los equipos que envían y procesan los segmentos TCP. Como los asentimientos son acumulativos, usar ACK retardado no tiene ningún impacto en la fiabilidad de la transmisión. Además, el uso del asentimiento retardado mejora el rendimiento en redes asimétricas [210], como es el caso de las tecnologías xDSL [211], en las que la proporción de asimetría es muy alta y el canal de subida se satura fácilmente. Sin embargo, también se demostrado que el asentimiento retardado reduce el rendimiento en determinadas ocasiones. Tal y como describe la RFC 2581 [180], TCP incrementa la ventana según el número de asentimientos recibidos, por lo que cuantos menos asentimientos llegan, menos aumenta la ventana. De hecho, el tiempo que se pasa una conexión en la fase de *slow start* se puede llegar a doblar empleando *delayed* ACK [212]. El estudio de Dukkipati *et al.* [181] muestra la importancia de este tiempo, ya muchas de las transacciones TCP en Internet son conexiones cortas que pasan toda su vida en *slow start*. Asimismo, en la fase de *congestion avoidance*, el incremento es más lento. Hoy en día la mayoría de las tecnologías de acceso son el primer cuello de botella, y nos imponen una asimetría. Sin embargo, a medida que se van desplegando tecnologías de fibra, esta asimetría se irá reduciendo.

4.3 TCP sobre OBS con asentimiento retardado

En este apartado de la tesis, se procede a investigar el rendimiento de TCP, con y sin asentimiento retardado, sobre una red OBS mediante simulación, y dar unas explicaciones

cualitativas de los resultados. La mayoría de los estudios de TCP sobre OBS consideran que si el ancho de banda de acceso es lo suficientemente elevado como para que quepa una ventana entera de TCP en el tiempo de agregación de la ráfaga, toda la ventana de TCP se empaquetará y se enviará en una única ráfaga, y el emisor se quedará esperando a la llegada de asentimientos para enviar nuevos datos. Si no se emplea *delayed* ACK, cada segmento se asiente, y todos los ACKs viajarán también en una sola ráfaga, con lo que el emisor a su vez, volverá a transmitir todos sus datos de manera consecutiva. Este escenario de transmisión ideal se muestra en la Figura 4-2.

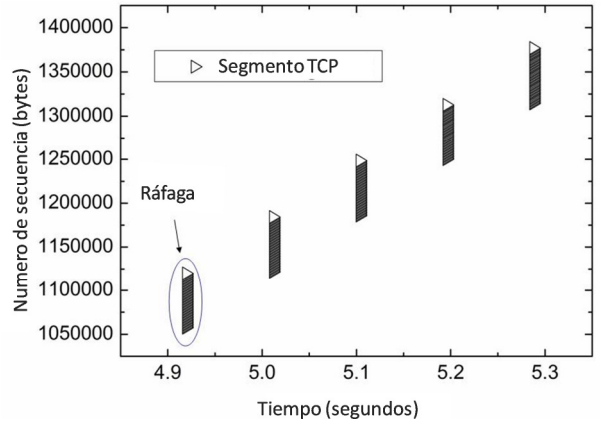


Figura 4-2 Transmisión ideal sin *delayed* ACK: todos los segmentos en una única ráfaga

Sin embargo, cuando se emplea *delayed* ACK, el último segmento de la ráfaga inicial, si esta contiene un número impar de segmentos, se transmitirá en un tiempo aleatorio (en el margen de 200 ms), por lo que ese ACK puede viajar separado en otra ráfaga. Este escenario es muy común en la fase de *slow start*, en la que la ventana crece con cada asentimiento recibido, y pueden aparecer ráfagas con números impartes. Debido a estos eventos, el emisor TCP no transmite todos los segmentos de manera consecutiva como en el caso ideal, sino que la ventana de transmisión se transmite en varias ráfagas en vez de una.

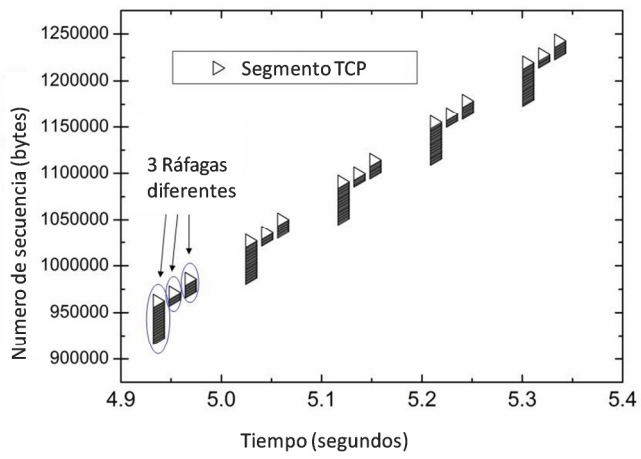


Figura 4-3 Transmisión con *delayed* ACK: segmentos repartidos en varias ráfagas

En la Figura 4-3 se muestra una transmisión TCP en OBS, simulada con OPNET Modeler, con *delayed* ACK activado, en la que se ha alcanzado el estado estacionario, con una ventana de 65 kbytes (unos 44 segmentos) y se muestra un triángulo por cada segmento que se envía en un instante de tiempo. Todos los segmentos que se transmiten de manera consecutiva, en muy poco espacio de tiempo, aparecen uno encima del otro. Hay que recordar que, en la fase de *slow start*, cada vez que llega un ACK, TCP se incrementa en uno la ventana. Al comienzo de la transmisión la ventana vale un segmento, el cual es asentido tras vencer el *tick* de reloj de *delayed* ACK. Después, se envían dos segmentos, que

se asienten en un único asentimiento. Cuando llega este asentimiento, se incrementa en uno la ventana, con lo que pasa a valer 3, y se envían 3 asentimientos. Como se ha visto en la Figura 4-1, cuando llegan 3 segmentos, se envían dos ACKs, uno tras la del segundo segmento y el otro tras el vencimiento del temporizador de asentimiento retardado, que, al ser de 200 ms ocurre después de que venza el temporizador de ensamblado de OBS (siempre y cuando éste sea menor de 200 ms). Así, un asentimiento viaja en una ráfaga, y otro en otra ráfaga. Este hecho vuelve a suceder más adelante, con lo que se genera una situación en la que se envían los segmentos en tres ráfagas diferentes.

Por tanto, en los ejemplos comparados, sin *delayed* ACK cada ráfaga tiene 44 segmentos, toda la ventana, mientras que con *delayed* ACK se reparten los segmentos, obteniéndose una media de 14 segmentos por ráfaga. Según los estudios de Detti y Listanti [50] el beneficio de correlación (denominado ganancia DFL en otros estudios) es proporcional a la raíz cuadrada del número medio de segmentos por ráfaga. Por tanto, uno de los efectos del *delayed* ACK es la reducción de la ganancia DFL. Sin embargo, se observa que en el caso con *delayed* ACK hay ráfagas que contienen solamente unos pocos segmentos. En el caso de pérdida de estas ráfagas, la recuperación de TCP será más fácil, aunque dependerá del tipo de TCP empleado, como se mostró en el capítulo anterior.

Por otro lado, ya que las pérdidas de ráfagas son comunes en OBS, otro de los impactos negativos del *delayed* ACK es el mayor tiempo que se tarda en volver a tener un valor de la ventana como el que se tenía antes de la pérdida. Esto se debe a que TCP incrementa su ventana según los asentimientos recibidos. Con *delayed* ACK activado, se reciben la mitad de asentimientos. Por tanto, ante una pérdida de ráfaga, puede tardarse hasta el doble de tiempo en alcanzar el mismo valor en la venta de transmisión que si no se tuviera ACK retardado. El comportamiento será apreciado en mayor medida cuanto mayor sea la ventana máxima del receptor.

4.3.1 Escenario de simulación

Se ha realizado un conjunto de simulaciones en OPNET Modeler [12] para analizar el impacto del asentimiento retardado. Estas simulaciones recrean un escenario en el que hay un usuario descargando archivos con un acceso de muy alta velocidad, y una red OBS hasta el servidor de archivos. El modelo que se ha programado y simulado está basado en el escenario ampliamente usado en la literatura [205] [52], y se muestra en la Figura 4-4. Los extremos "Cliente TCP" y "Servidor TCP" se conectan a la red OBS por un enlace sin pérdidas de 1 Gbit/s de ancho de banda y 10 ms de retardo, emulando un acceso futuro de alta velocidad. Estos nodos se conectan directamente a una fibra con un canal dedicado de 10 Gbit/s y 30 ms de retardo. Los nodos OBS tienen implementados un algoritmo de ensamblado de temporizador de 1 ms, que está en el rango de valores adecuados para el rendimiento de TCP obtenidos en estudios de la literatura [52] [205].

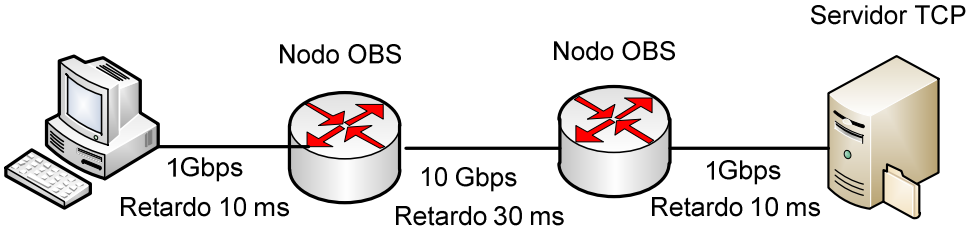


Figura 4-4 Escenario de simulación

Para cada configuración concreta simulada se realizan 500 transferencias de archivos de 10 Megabytes del Servidor TCP al Cliente mediante FTP. Como métrica para comparar el resultado de la simulación se va a emplear el *throughput,* que se obtiene dividiendo el tamaño del archivo entre el tiempo medio para transferir cada uno de los archivos. Esta métrica mide la tasa de transferencia de datos útiles, y excluye cabeceras, retransmisiones, etc. Por

ello, esta métrica a veces se denomina *goodput*. En la tesis se emplean ambos términos, si bien en cada apartado que se mencione *throughput* se aclarará qué incluye exactamente. A partir del *throughput* obtenido en las 500 transferencias se calcula el intervalo de confianza del 95%, que se muestra en las distintas gráficas con resultados. Se realizan simulaciones activando y desactivando el asentimiento retardado, y se emplean dos versiones de TCP, la clásica versión Reno y la versión con asentimiento retardado, SACK.

4.3.2 Efectos del *delayed* ACK en Reno y SACK sobre OBS

El objetivo de esta simulación es mostrar los efectos del asentimiento retardado en las versiones de TCP Reno y SACK. Los resultados se muestran en la Figura 4-5, en los que se detalla el *throughput* obtenido para distintas probabilidades de pérdida en cada versión, con y sin asentimiento retardado. A continuación, la Figura 4-6 muestra la ganancia obtenida en el *throughput* si se desactiva *delayed* ACK.

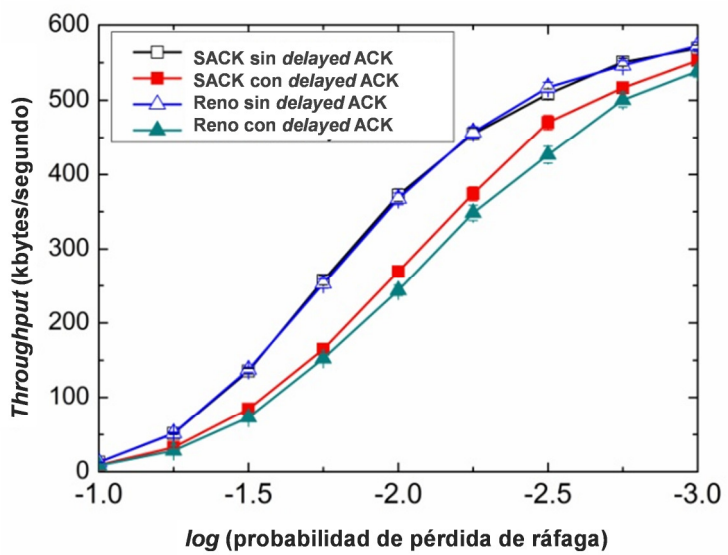


Figura 4-5 *Throughtput* en SACK y Reno con y sin *delayed* ACK

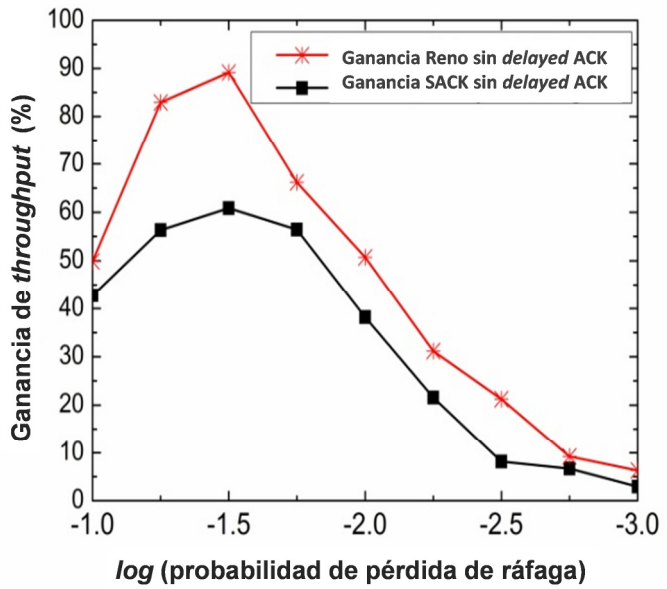


Figura 4-6 Ganancia de *throughput* cuando se desactiva *delayed* ACK

El asentimiento retardado, como se ha explicado anteriormente, rompe la secuencia de transmisión de los asentimientos, con lo que los segmentos acaban repartidos por varias ráfagas. Por tanto, para transmitir la misma cantidad de información se necesitan más ráfagas, con lo que el número de eventos de pérdida de ráfaga aumenta. Este hecho se

conoce como reducción del beneficio de correlación. Por otro lado, como TCP incrementa la ventana con los asentimientos, al llegar menos asentimientos si hay asentimiento retardado, el ritmo de crecimiento de la ventana es menor. Los resultados de simulación corroboran el análisis y se comprueba, como se observa en la Figura 4-5, que tanto empleando la versión de TCP Reno como la versión SACK, el *throughput* obtenido en las transferencias es mayor si no se activa *delayed* ACK.

Beneficios de SACK con ACK retardado

En caso de tener activado el asentimiento retardado, al romper la secuencia de transmisión en varias ráfagas, estas contienen menos segmentos, y en el caso de emplear el asentimiento selectivo (SACK), gracias a su información, permite recuperarse rápidamente de pérdidas de múltiples paquetes que TCP Reno (o cualquier versión de TCP que no use, o no tenga activado, asentimiento selectivo). Por ello, como este tipo de pérdidas ocurre cuando hay asentimiento retardado, se comprueba en la Figura 4-6, que si se emplea asentimiento retardado en SACK el rendimiento no decrece tanto como lo hace si se emplea en la versión Reno, aunque no es suficiente para compensar el resto de efectos negativos.

SACK y Reno sin ACK retardado

Por otro lado, cuando TCP se emplea sin activar *delayed* ACK, todas las ráfagas contienen toda la ventana de transmisión, por lo que la mayoría de las pérdidas se van a tener que recuperar mediante el temporizador de retransmisión. Con este tipo de pérdidas, el asentimiento selectivo no presenta ninguna ventaja, por lo que cuando no se activa *delayed* ACK, no se observan apenas diferencia entre TCP Reno y SACK en el escenario simulado con un único cliente y servidor.

Ganancia según la probabilidad de pérdida

Sin embargo, la diferencia entre usar o no *delayed* ACK varía según la probabilidad de pérdida. Cuando la probabilidad de pérdida de ráfaga es significativa, la ganancia de asentir cada uno de los segmentos es alta, llegando al 60% para SACK y al 90% para Reno en el caso de una probabilidad de $10^{-1,5}$. Para probabilidades de pérdida bajas, el impacto disminuye, ya que la transmisión ha estado en su mayor parte en una fase estable con la ventana de transmisión al máximo. Por tanto, el impacto de usar o no *delayed* ACK en transmisiones largas es significativo cuando hay pérdidas de ráfagas frecuentes y es importante aumentar la ventana con rapidez.

4.3.3 Impacto del uso del ACK retardado en SACK con escalado de ventana

A medida que se va disponiendo de un mayor ancho de banda, y los ordenadores disponen de más memoria, se recomienda activar la opción de escalado de ventana y emplear ventanas grandes. Es de esperar que en el momento que se despliegue una red OBS, la mayoría de las fuentes TCP cuenten con un buen acceso a la red, y tengan dicha opción activada. Por tanto, el impacto del asentimiento retardado en redes OBS ha de ser estudiado con *buffers* de recepción altos. En las simulaciones realizadas, el *buffer* de recepción, que limita la ventana de TCP, varía entre 2^{16}-1 bytes y 2^{22}-1 bytes, es decir entre 64 kbytes y 4 Mbytes). El estudio se centra en la versión SACK, pues una vez se han visto las diferencias en la simulación anterior, se ha comprobado que dicha versión obtiene un mejor rendimiento.

En primer lugar, se realiza un conjunto de simulaciones en las que se varía la probabilidad de pérdida de ráfaga para varios valores del *buffer* de recepción con el escalado de ventana activo. Los resultados de las simulaciones realizadas se pueden ver en la Figura 4-7, donde se muestra el *throughput* obtenido para cada *buffer*, con y sin *delayed* ACK para

distintas probabilidades de pérdida de ráfaga, y la Figura 4-8, en la que se muestra la ganancia de no utilizar *delayed* ACK frente a tenerlo activado para varios tamaños de *buffer* y distintas probabilidades de pérdida de ráfaga. Se puede comprobar cómo, lógicamente, el *troughput* se incrementa a medida que el *buffer* de TCP aumenta. Además, se observa que, sin escalado de ventana, por debajo de una cierta probabilidad de pérdida ya no se incrementa apenas el *throughput*. Esto se debe a que, después de una pérdida, TCP necesita un tiempo para recuperarse hasta llegar al valor máximo de la ventana. Este valor de la ventana es el mínimo de la ventana de congestión y el *buffer* de recepción. En la ventana estándar este valor de la ventana es tan bajo, que TCP tarda muy poco tiempo en llegar a dicho valor, independientemente de si hay o no asentimiento retardado.

Sin embargo, cuando se aumenta el *buffer* con el escalado de ventana, el tiempo que se tarda en alcanzar el máximo es mayor y, si se emplea *delayed* ACK, este tiempo será aún mayor. Por tanto, como muestra la Figura 4-8 el impacto del *delayed* ACK es mayor cuanto más grande sea el *buffer* de recepción. Como con escalado de ventana, el tiempo en recuperarse de una pérdida es significativo, no se llega a una situación estable tan rápidamente. A medida que el *buffer* va aumentando, el *throughput* se estabiliza a probabilidades de pérdida menores. De esta forma, cuanto menor sea la probabilidad de pérdida, más se nota el efecto de aumentar el *buffer* y el no tener asentimientos retardados.

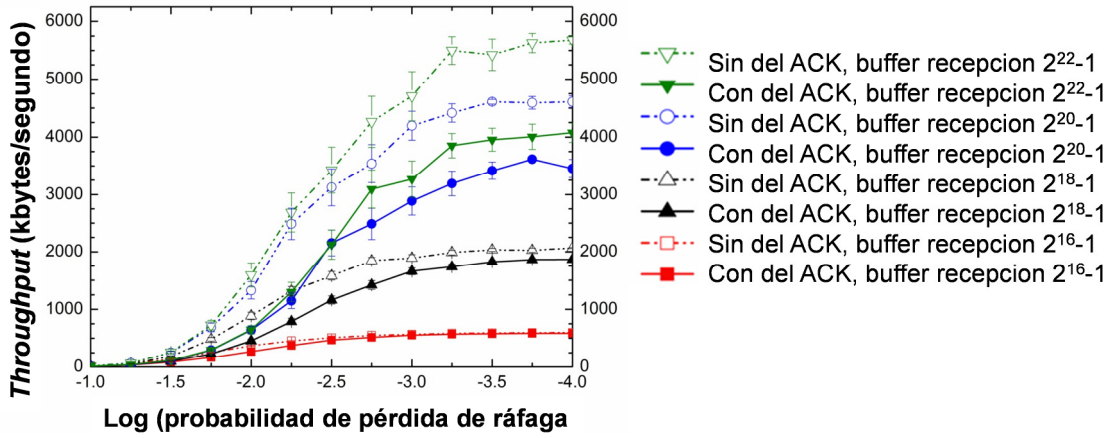


Figura 4-7 *Throughput* en SACK con escalado de ventana

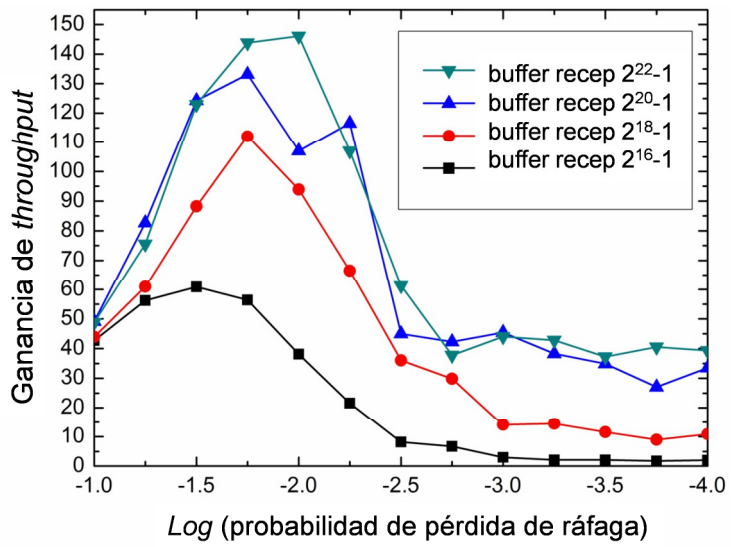


Figura 4-8 Ganancia en SACK sin *delayed* ACK

Por otro lado, sin asentimientos retardados, sin escalado de ventana (*buffer* de 64 Kbyes), la máxima ganancia de no activar *delayed* ACK frente a tenerlo activado se obtiene para una probabilidad de pérdida de $10^{-1,5}$, y es despreciable para pérdidas bajas. Sin embargo, con escalado de ventana, cuando el *buffer* se incrementa, la ganancia máxima por no tener asentimiento retardado se obtiene a probabilidades de pérdida de ráfagas baja (ej, 10^{-2} para un *buffer* de 2^{22}-1 bytes). Estos resultados sugieren que la máxima diferencia se obtiene cuando el tiempo medio entre ráfagas es aproximadamente el tiempo medio en que tarda la ventana de congestión en llegar al valor máximo del *buffer* de recepción después de una pérdida de ráfaga. Por tanto, cuanto mayor sea el *buffer*, el máximo de ganancia de no activar *delayed* ACK se obtiene a menor probabilidad de pérdida de ráfaga.

4.4 Estudio de TCP sobre OBS teniendo en cuenta tráfico de fondo

Como se mencionó en el capítulo anterior, los primeros estudios de TCP sobre OBS, llevados a cabo por Detti y Listanti [50], asumieron un modelo de red consistente únicamente en un cliente y un servidor ambos conectados por un enlace OBS con pérdidas y un enlace de retorno sin pérdidas. Posteriormente, el modelo se amplió teniendo en cuenta un enlace de vuelta con pérdidas [205]. Demostraron la existencia de un efecto, que nombraron "beneficio de correlación", que se debe al proceso de agregación, y que puede conducir, en algunos casos, al incremento significativo de la tasa de transmisión de TCP. Para poder comprobar la existencia del beneficio de correlación en escenarios pragmáticos, también estudiaron un escenario simple con fuentes de tráfico adicionales. Mientras que la simulación muestra el beneficio de correlación, los resultados cuantitativos en el escenario se desvían del modelo analítico, ya que el modelo se obtuvo del escenario cliente – servidor simple. Por tanto, parece clara la necesidad de llevar a cabo tanto estudios teóricos como simulación extensiva para evaluar el impacto de múltiples fuentes de tráfico en redes TCP sobre OBS. Yu *et al.* [66] [51] [52] compararon el impacto en el rendimiento de distintas versiones de TCP, Reno, SACK y New Reno, tanto de manera analítica como por simulación, mostrando que SACK es la versión con mejor *throughtput*. De todas formas, solo consideraron el escenario simple con un cliente TCP y un servidor. Gowda *et al.* [213] presentaron varios estudios simulación sobre el *throughput* y el retardo en función del tamaño máximo de las ráfagas y el temporizador de ensamblado cuando se alimenta un red OBS con entre 3 y 30 sesiones TCP, pero sin dar apenas resultados cualitativos. Pleich *et al.* [207] presentaron unos resultados de simulación de TCP Reno sobre OBS considerando 900 fuentes TCP, encontrando que el tráfico transportado por TCP en una red OBS es mucho más robusto que lo que se predecía en otros artículos. Consideraron que probablemente la sensibilidad del *throughput* a las pérdidas de ráfagas se deba a versiones simuladas antiguas de TCP o demasiadas pocas fuentes de tráfico. Sin embargo, el estudio se limita a proporcionar explicaciones cualitativas y presentar hipótesis sin mostrar resultados cuantitativos.

En este apartado, se realiza un estudio de simulación del rendimiento de TCP sobre OBS cuando múltiples fuentes alimentan a un generador de ráfagas. No solo se presentan resultados cuantitativos, sino explicaciones cualitativas del impacto de diferentes escenarios y parámetros, como la utilización de las distintas versiones de TCP, como Reno y SACK y el *delayed* ACK, complementando el estudio comenzado al principio del capítulo. El principal resultado de este apartado del capítulo es la comparación de escenarios con un único flujo y escenarios con tráfico de fondo adicional. Concretamente se obtiene una comparativa de la distribución de ráfagas en cada uno de los casos y se cuantifica en qué medida impacta el tener tráfico adicional.

4.5 Escenarios de simulación con tráfico de fondo

Se han diseñado dos modelos de simulación. El primero (Figura 3-5), se basa en el modelo clásico de la literatura [50] [52], y es similar al empleado en el Capítulo 4. En este modelo hay un cliente TCP y un servidor TCP conectados a la red OBS a través de un acceso sin pérdidas, con 10 ms de retardo y 100 Mbps bidireccional de ancho de banda, que se puede considerar un acceso típico en el futuro. Se modelan ambos sentidos de la transmisión, de tal forma que el ACK de TCP también se ensambla en ráfagas y se trasmite a través de un enlace OBS con pérdidas, con lo que es susceptible de perderse. El segundo modelo (Figura 4-10) es más pragmático y es la base de este estudio. El ensamblador de ráfagas OBS se alimenta no solo por un cliente TCP (o servidor TCP), sino también de tráfico adicional, que se crea artificialmente mediante generadores de tráfico fractal [214]. Los generadores de tráfico fractal pueden modelar el tráfico generado por miles de usuarios conectados a internet. El generador de tráfico fractal se ha configurado para introducir una media de 900 paquetes por segundo. La longitud de los paquetes sigue una distribución exponencial con una media de 1024 bytes, y un parámetro Hurst de 0,7.

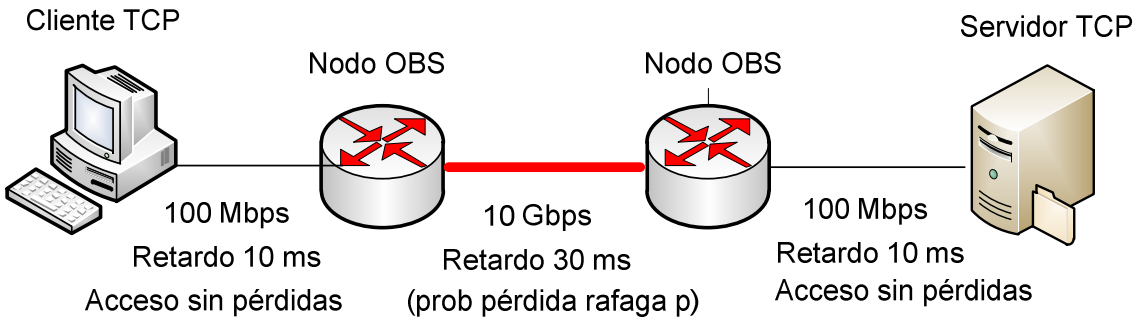


Figura 4-9 Escenario de simulación de TCP sobre OBS con un flujo TCP

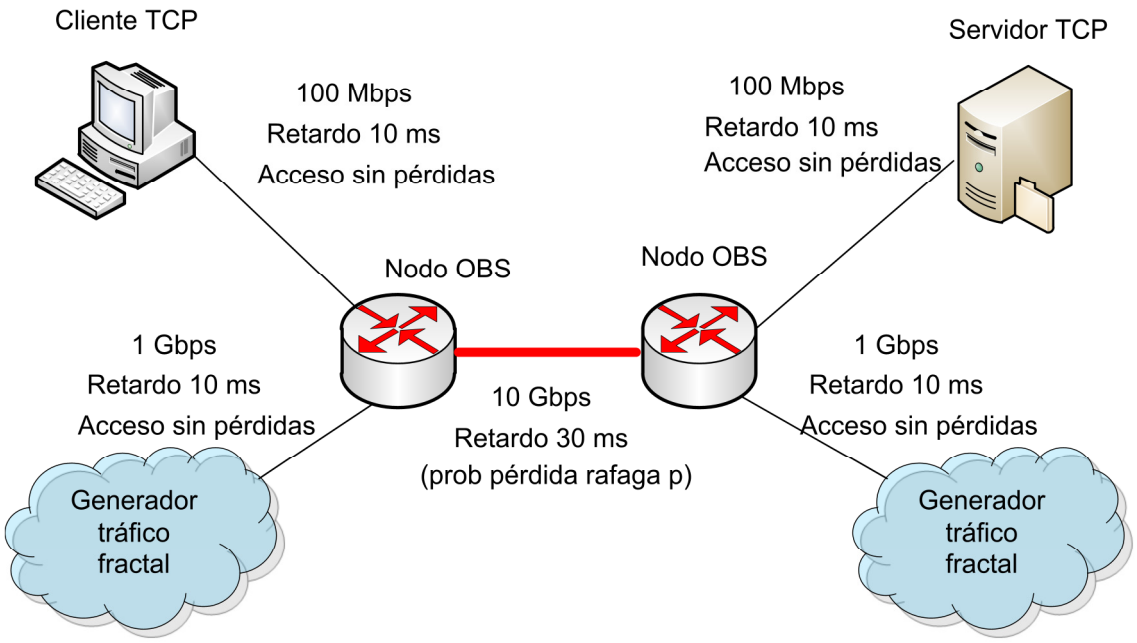


Figura 4-10 Escenario de simulación de TCP sobre OBS con tráfico fractal de fondo

En cuanto a los parámetros de TCP, la Tabla 4-1 recopila los principales valores empleados en las simulaciones.

Parámetro	Valor
Tamaño de segmento	1460 bytes
Tamaño máximo de ventana	65535 bytes (44 segmentos aprox.)
Versiones de TCP	Reno y SACK
Temporizador del *delayed* ACK	Cuando se usa, se fija en 200 ms
Temporizador de retransmisión	1 segundo

Tabla 4-1 Parámetros de TCP de las simulaciones de TCP con tráfico fractal

4.6 Estudio del número de segmentos TCP por ráfaga

En caso de pérdida de ráfaga, es importante cuantificar cuántos segmentos de un mismo flujo TCP viajan en la misma, ya que el impacto de la pérdida depende en gran medida de este valor. Para ello, se ha realizado un conjunto de simulaciones de transferencia de ficheros en una red OBS sin pérdidas, en concreto, ficheros de 20 Megabytes.

En cada escenario se ha simulado una situación ideal, con un único flujo, y a continuación se ha mezclado este flujo con tráfico de fondo adicional, emulando el tráfico que habría en una red real. Se realizaron 500 transferencias para cada caso estudiado, y se calculó el histograma del número de segmentos TCP por ráfaga. Se emplea un algoritmo de ensamblado de ráfagas basado en temporizador y se centra el estudio en flujos medios y rápidos, según la nomenclatura explicada en la sección 3.3.4. Dado que el ancho de banda del enlace de cuello de botella, que se corresponde con el enlace de acceso, es de 100 Mbit/s, y que los segmentos son como máximo de 1460 bytes, y que las cabeceras empleadas en la simulación ocupan 40 bytes, el valor del temporizador de creación de ráfagas se ha fijado en 10 ms en el caso de flujos rápidos, capaz de albergar los 44 segmentos de la ventana máxima, y en 1 ms en el caso de flujos medios, que permitirá ráfagas de hasta 9 segmentos.

Número de segmentos por ráfaga en el escenario de un flujo TCP

En primer lugar, se ha simulado el escenario simple de la Figura 4-9 sin *delayed* ACK. El resultado se muestra en la Figura 4-11 y la Figura 4-12 en las que se representa la distribución del número de segmentos por ráfaga para una fuente media y rápida respectivamente.

Como se esperaba, el resultado es muy determinista, y muestra que, en el caso de la fuente media, la mayoría de las ráfagas contienen 9 segmentos, el máximo que cabe en una ráfaga. Sin embargo, como la ventana es de 44 segmentos, cuando la ventana está al máximo, se transmiten 4 ráfagas de 9 segmentos, y una última de 8 segmentos. En cambio, en el caso de la fuente rápida, en la mayor parte de las ocasiones, las ráfagas tienen 44 segmentos. En este caso, todos los segmentos de la ventana de transmisión (44, como se muestra en la Tabla 4-1), se agregan en una única ráfaga y se transmiten juntos. Sus asentimientos también se agregan en una sola ráfaga y así sucesivamente.

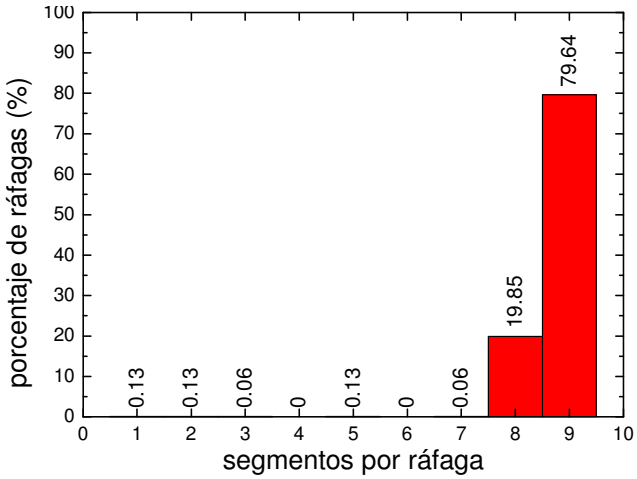


Figura 4-11 Histograma del número de segmentos de un flujo por ráfaga (flujo medio, sin *delayed* ACK, escenario de un solo flujo)

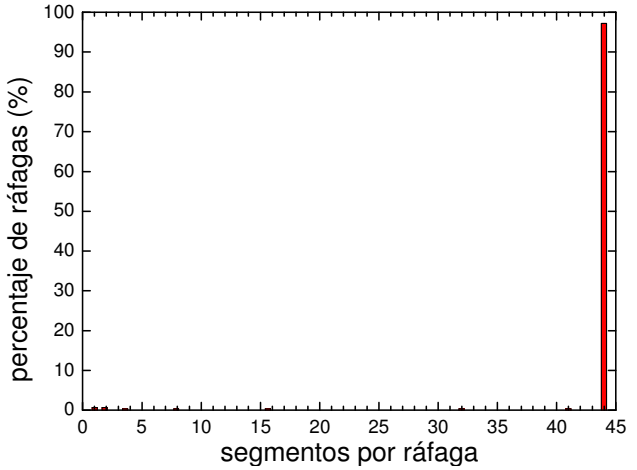


Figura 4-12 Histograma del número de segmentos de un flujo por ráfaga (flujo rápido, sin *delayed* ACK, escenario de un solo flujo)

Cuando hay *delayed* ACK, los histogramas del número de segmentos por ráfaga, mostrados en la Figura 4-13 para flujos medios y Figura 4-14 para flujos rápidos, se desvían de ese comportamiento, corroborando las hipótesis planteadas en la sección 4.3 de este capítulo. Debido a la fragmentación de las ráfagas, y al hecho de que un asentimiento confirma 2 segmentos, el histograma presenta picos para múltiplos de dos, además de un pico en 9, que es el valor máximo de segmentos que caben en una ráfaga con temporizador de 1 ms y ancho de banda de acceso de 100 Mbit/s. Es decir, como cada vez que llega un asentimiento, éste confirma 2 segmentos, se envían dos nuevos segmentos a continuación. De hecho, gran parte de las ráfagas transmite únicamente dos segmentos del flujo TCP.

Número de segmentos por ráfaga en el escenario con tráfico de fondo

Cuando se añade tráfico de fondo, el resultado es completamente diferente, como se aprecia en la Figura 4-15, la Figura 4-16, la Figura 4-17 y la Figura 4-18. Las ráfagas llevan, en todos los casos, un número de segmentos del flujo TCP menor que en su simulación equivalente del escenario simple. En el caso de que se añada tráfico de fondo, el instante de inicio de formación de las ráfagas puede venir no solamente de la llegada de segmentos del flujo TCP en cuestión, sino de cualquier paquete de cualquier otra fuente de tráfico. Por tanto, aparece una fragmentación de las secuencias de datos, que se enviarán en diversas ráfagas. Al igual que sucedía en el caso sin tráfico adicional, cuando se emplea el asentimiento retardado, cada vez que llega un ACK se asienten dos segmentos, con lo que el número de segmentos por ráfaga es mayoritariamente un número par.

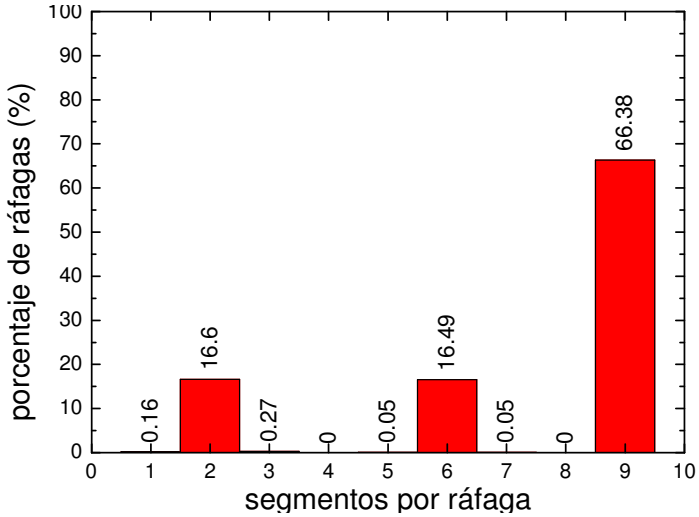


Figura 4-13 Histograma del número de segmentos de un flujo por ráfaga (flujo medio, con *delayed* ACK, escenario de un solo flujo)

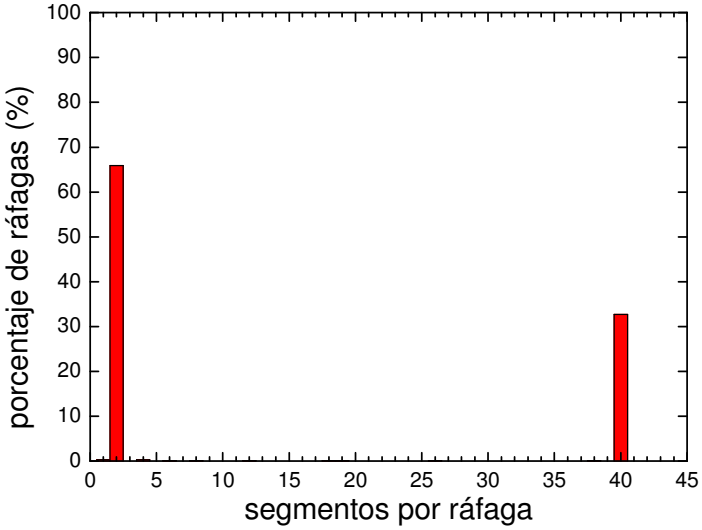


Figura 4-14 Histograma del número de segmentos de un flujo por ráfaga (flujo rápido, con *delayed* ACK, escenario de un solo flujo)

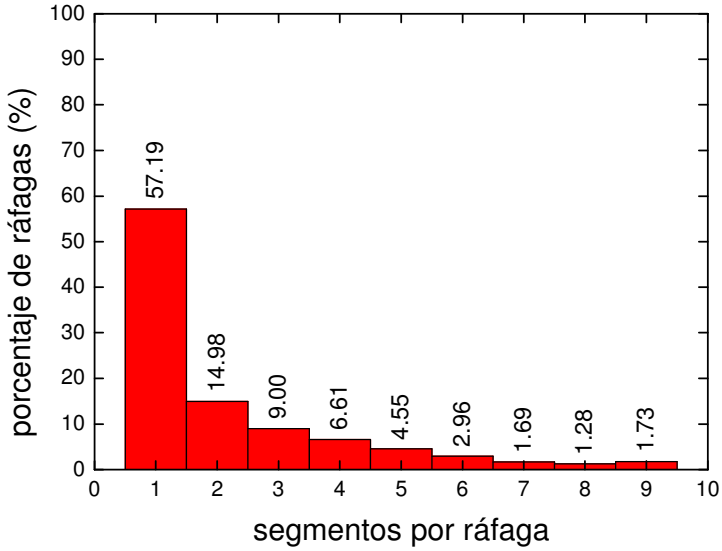


Figura 4-15 Histograma del número de segmentos de un flujo por ráfaga (flujo medio, sin *delayed* ACK, escenario de un flujo con tráfico de fondo)

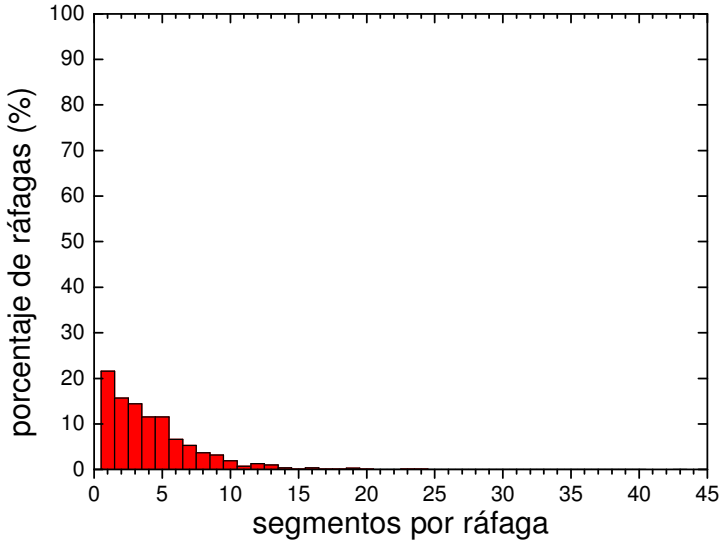


Figura 4-16 Histograma del número de segmentos de un flujo por ráfaga (flujo alto, sin *delayed* ACK, escenario de un flujo con tráfico de fondo)

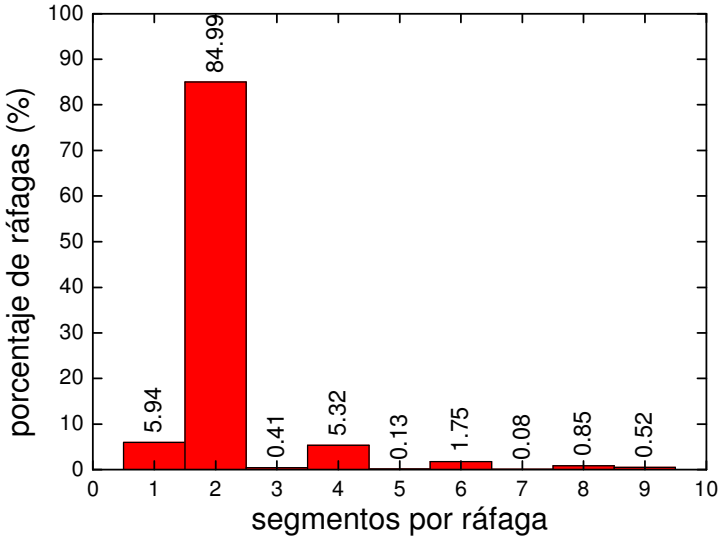


Figura 4-17 Histograma del número de segmentos de un flujo por ráfaga (flujo medio, con *delayed* ACK, escenario de un flujo con tráfico de fondo)

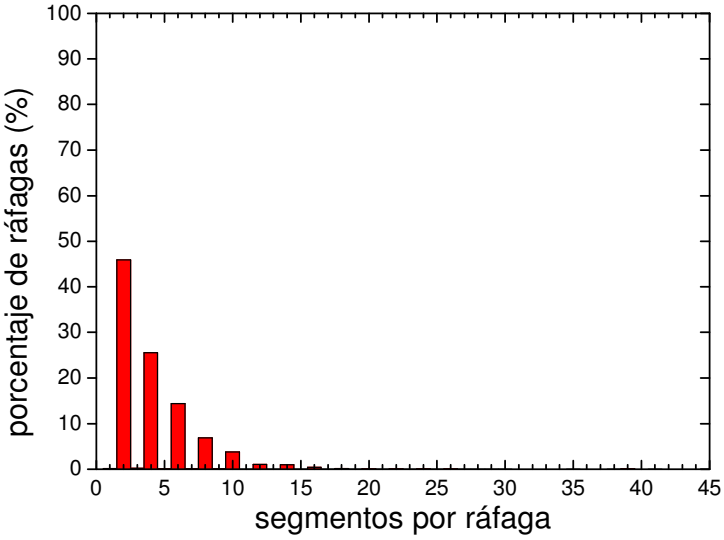


Figura 4-18 Histograma del número de segmentos de un flujo por ráfaga (flujo rápido, con *delayed* ACK, escenario de un flujo con tráfico de fondo)

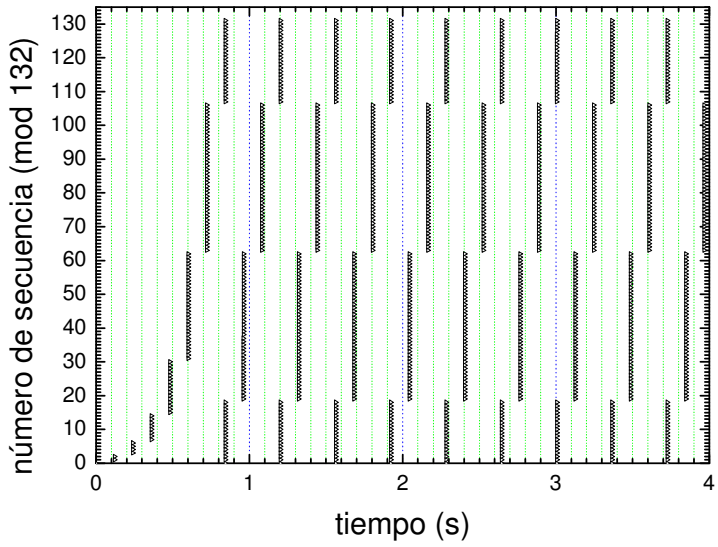


Figura 4-19 Traza de la transferencia de segmentos TCP (un solo flujo rápido)

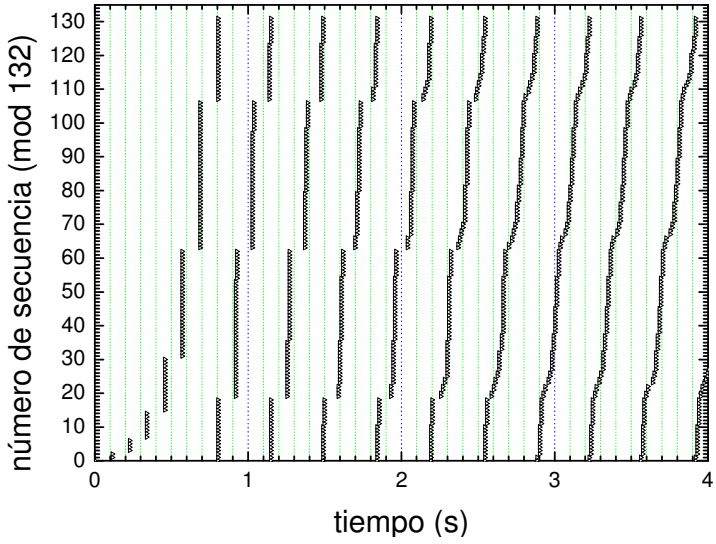


Figura 4-20 Traza de la transferencia de segmentos TCP de un flujo rápido con tráfico de fondo

Para explicar con más detalle el impacto de añadir tráfico de fondo, se emplean la Figura 4-19 y la Figura 4-20. La primera muestra el número de secuencia de los segmentos enviados por el emisor TCP sin tráfico de fondo para un flujo rápido sin *delayed* ACK. Se aprecia claramente que la transmisión es muy a ráfagas y sincronizada. Cada RTT, todos los segmentos de la ventana TCP se transmiten de manera consecutiva, con lo que se ensamblan en una única ráfaga. La Figura 4-20 muestra el caso en el que el ensamblador se alimenta con tráfico de fondo fractal. La transmisión comienza con el mismo comportamiento, todos los segmentos de la ventana TCP se ensamblan juntos en una ráfaga. Pero, a medida que pasa el tiempo, la transmisión de todos los segmentos de manera consecutiva se rompe. Por ejemplo, un paquete de otro flujo llega al ensamblador y lanza el proceso de generación de una nueva ráfaga, con lo que cuando la fila de segmentos del flujo TCP que se está observando llega al ensamblador, una parte entra en el *buffer* antes de que se finalice el tiempo de agregación de ráfaga, mientras que el resto llega después de que venza el temporizador, y por tanto se transmitirán en una ráfaga diferente. Por tanto, la transmisión de segmentos se extiende a lo largo del RTT, y cada ráfaga contiene unos pocos segmentos de cada flujo. De esta forma, después de varias rondas de transmisión, el número medio de segmentos por ráfaga (S) alcanza un estado estacionario (en la Figura 4-21 se ve la media móvil).

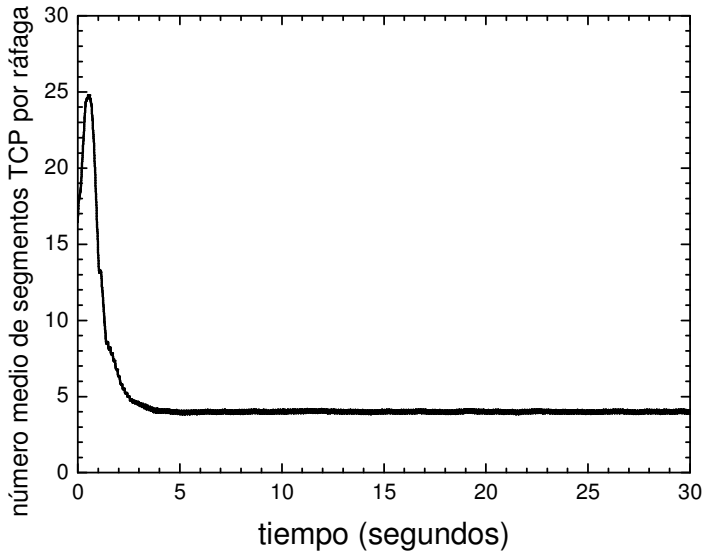


Figura 4-21 Evolución del número de segmentos por ráfaga a lo largo del tiempo en un flujo TCP

Para poder determinar el *throughput* de TCP en una red OBS, y saber cuál es la ganancia de correlación real, es necesario conocer el número medio de segmentos por ráfaga. Se ha medido este valor para los cuatros escenarios para distintos valores de temporizador de ensamblado. La Figura 4-22 muestra el número medio de segmentos de un flujo TCP por ráfaga para los distintos casos simulados, así como un resultado analítico para obtener el número medio de segmentos por ráfaga aplicando la ecuación (4-1), en la que T_b es el temporizador de ensamblado, B_a es el ancho de banda del enlace cuello de botella, y MSS_{con_cab} es el tamaño máximo de segmento, incluyendo las cabeceras.

$$\bar{S} = \left\lceil \frac{T_b B_a}{MSS_{con\_cab}} \right\rceil \tag{4-1}$$

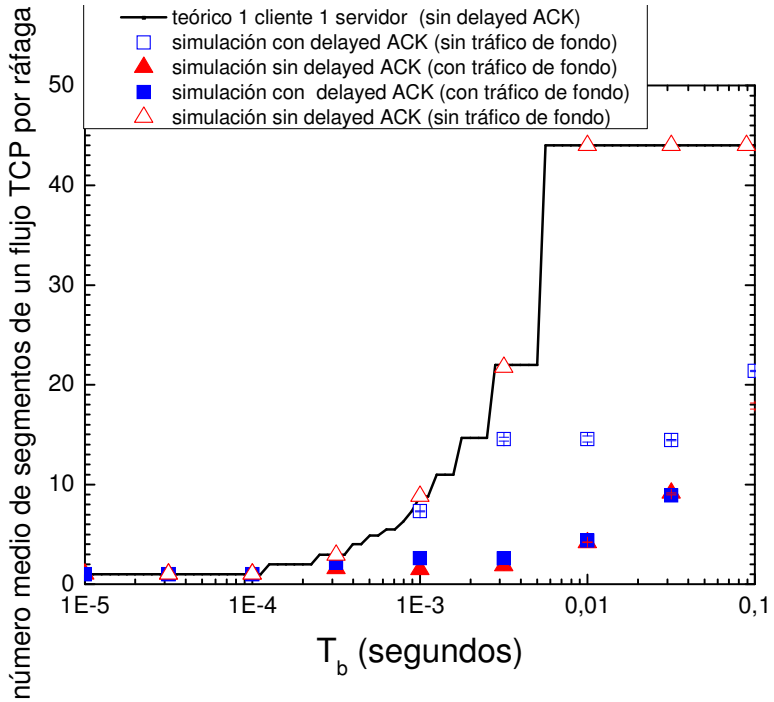


Figura 4-22 Número medio de segmentos de un flujo TCP por ráfaga

Los resultados de simulación muestran que cuando se considera trafico adicional, el número medio de segmentos de un flujo TCP por ráfaga es mucho menor que en el caso de un solo flujo, independientemente de si la fuente es lenta, media o rápida. Por otro lado, cuando se tiene en cuenta tráfico de fondo las diferencias entre tener o no activado el *delayed* ACK no son tan significativas como en el caso en que no había tráfico de fondo. Esto se debe a que, por el hecho de añadir tráfico de fondo se fragmentan las ráfagas, algo que ya ocurría al introducir el asentimiento retardado. En resumen, los resultados muestran que el número de segmentos por ráfaga se sobreestima en gran medida si se intenta extrapolar los resultados cuando el ensamblador se alimenta de un único flujo TCP con respecto al escenario más pragmático y realista en el que hay múltiples fuentes de tráfico.

4.6.1 Simulaciones para diferentes temporizadores de ensamblado de ráfaga

Ahora, nuestro objetivo es cuantificar el impacto en el *goodput*, que se define como la tasa de datos útiles recibidos por unidad de tiempo. Por tanto, esta medida descarta tanto los segmentos duplicados como la cabecera, como los asentimientos y se centra en los datos que transporta TCP. Las simulaciones han consistido en la transferencia de cien archivos de 20 Mbytes, fijando la probabilidad de pérdida de ráfaga en 10^{-3}. Probabilidades de pérdida mayores no serían realistas, ya que ningún operador desplegaría una red con semejante nivel de pérdida. Los valores de los temporizadores se han escogido en el rango de los estudios de la literatura. Los resultados se muestran en la Figura 4-23 y la Figura 4-24. Los valores medios se representan junto con el intervalo de confianza del 95%. A partir de estos datos se puede obtener un gran número de conclusiones.

TCP Reno vs. TCP SACK

El rendimiento de TCP Reno y SACK para temporizadores de ensamblado muy pequeños (10^{-5}-10^{-4}) y para temporizadores muy elevados (alrededor de 100 ms) es muy parecido. Para los temporizadores muy bajos, el número de segmentos del flujo TCP por ráfaga es uno, por lo que todas las pérdidas de ráfaga se solventan de la misma manera en Reno y en SACK, como se explicó en la sección 3.3.3. Para temporizadores muy altos, la pérdida de una ráfaga significa la pérdida de una ventana completa del flujo TCP, y por tanto, como se describió y la sección 3.3.3, ambas versiones se han de recuperar por el temporizador de retransmisión. Sin embargo, para el rango intermedio (el más realista) SACK obtiene un rendimiento mejor gracias a su capacidad de recuperarse de pérdidas múltiples de segmento.

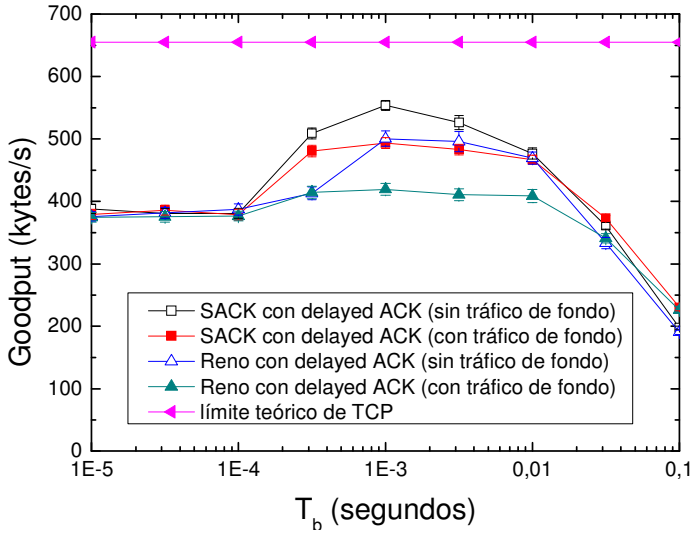


Figura 4-23 *Goodput* de TCP Reno y SACK, con y sin tráfico de fondo, con p =10^{-3}, usando *delayed* ACK

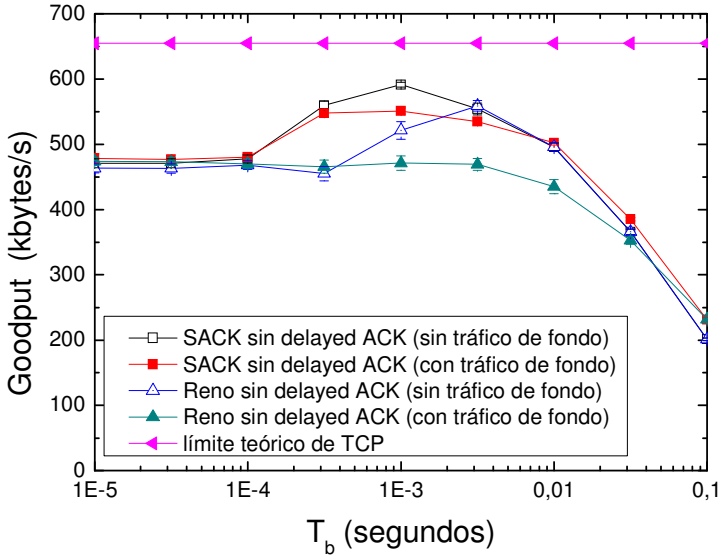


Figura 4-24 *Goodput* de TCP Reno y SACK, con y sin tráfico de fondo, con $p = 10^{-3}$, sin *delayed* ACK

Por tanto, los resultados confirman el análisis cualitativo realizado en capítulos anteriores. Si se compara el resultado con y sin *delayed* ACK, tal y como se vio en el apartado 4.3, el *goodput* (o *throughput*) cuando se emplea el algoritmo *delayed* ACK es menor que cuando no se activa.

Resultados con y sin tráfico adicional

Los resultados de las simulaciones muestran que para temporizadores bajos no hay ninguna diferencia al añadir el tráfico de fondo, ya que en ambos casos cada ráfaga transporta un único segmento. Sin embargo, para temporizadores entre 0,3 y 10 ms, el *goodput* es mayor en el escenario que no incluye tráfico extra que en el escenario realista con tráfico adicional. Esto se debe a que el número de segmentos de un flujo TCP por ráfaga es mayor cuando se estudia el flujo de manera aislada. Debido a la ganancia de correlación (o ganancia DFL, como se ha explicado en el apartado 3.3.4 [50] [205]), a mayor número de segmentos por ráfaga, mayor ganancia de correlación. Por otro lado, para temporizadores altos, el número de segmentos por ráfaga en el caso del flujo aislado es tan alto, que una pérdida de ráfaga siempre implica la pérdida de la ventana completa, mientras que en el caso del tráfico adicional, no siempre las pérdidas son de la ventana completa, por lo que con SACK pueden recuperarse en poco tiempo.

Valor óptimo del temporizador de ensamblado

El *goodput* depende del valor del temporizador de ensamblado. El mayor *goodput* se obtiene para valores del temporizador entre 0,3 y 10 ms. Esto se debe a la ganancia de correlación, que aumenta con el valor del temporizador. Sin embargo, si el valor del temporizador es demasiado alto, el *goodput* decrece, siendo incluso menor que el obtenido para temporizadores muy bajos donde no hay ganancia de correlación. La razón para dicho comportamiento es que no sólo las pérdidas afectan al rendimiento de TCP sobre OBS, sino también la penalización por el retardo de ensamblado, como se explicó en la sección 3.3.1.

Impacto del retardo de ensamblado de ráfaga

Tal y como se ha mencionado anteriormente, el rendimiento de TCP en OBS se ve afectado por un lado por las pérdidas de ráfagas y por otro por un incremento del RTT. En las secciones anteriores de este capítulo, el rendimiento obtenido (*goodput*) incluye todas las

contribuciones conjuntamente. A continuación se evalúa en qué medida contribuye el tiempo de ensamblado al decrecimiento del rendimiento. Para ello, se define una figura de mérito, denominada $F_{retardo}$ y mostrada en la ecuación (4-2), en la que se compara el *goodput* de una transmisión sin pérdidas que emplea un temporizador de ensamblado de valor T_b, con el *goodput* de la misma transmisión sin pérdidas, pero que emplea un temporizador de ensamblado de referencia, cuyo valor se denomina T_{b_ref}, de tal forma que dicho valor sea tan pequeño que todas las ráfagas siempre contengan un solo segmento y el retardo introducido por el ensamblador sea insignificante en comparación con el RTT. Como valor de T_{b_ref} se ha escogido 10^{-5} segundos, que se corresponde con el valor más pequeño empleado en las simulaciones y cumple con los criterios mencionados.

$$F_{retardo} = \frac{Goodput_{sin\_pérdidas}(T_b)}{Goodput_{sin\_pérdidas}(T_{b\_ref})} \qquad (4\text{-}2)$$

Por otro lado, se obtiene un valor analítico de la figura de mérito anterior. Para ello, se emplea el hecho de que la tasa de transmisión de TCP es inversamente proporcional al RTT (tiempo que tarda un segmento en transmitirse y que llegue su asentimiento de vuelta, cuando no hay ensamblado de ráfaga). Además, el RTT cuando se emplea un temporizador T_b es como máximo el valor del RTT sin temporizador (RTT_0) con un invremento de dos veces el valor del temporizador. El valor del temporizador de referencia es tan pequeño que se puede despreciar con respecto al valor del RTT sin temporizador. De esta forma, el valor de la figura de mérito $F_{retardo}$ obtenido de manera analítica se muestra en la ecuación (4-3).

$$F_{retardo} = \frac{RTT_0}{RTT_0 + 2T_b} \qquad (4\text{-}3)$$

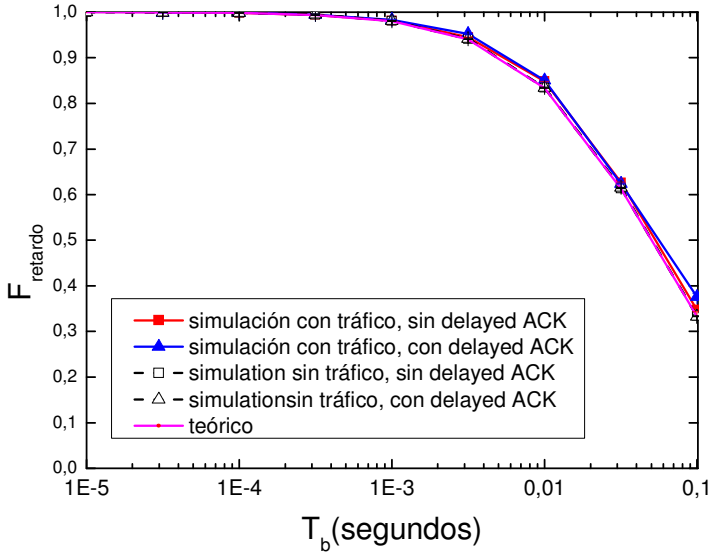


Figura 4-25 Figura de mérito de la penalización por retardo para RTT_0=100 ms

La Figura 4-25 muestra tanto el valor analítico de la figura de mérito, obtenido mediante la ecuación (4-3), como los resultados de las simulaciones. Corresponden al escenario de la Figura 4-9 y la Figura 4-10, en los que RTT_0 vale 100 ms, con y sin *delayed* ACK y con y sin tráfico adicional. En este caso hay que recalcar que, al no haber pérdidas, el comportamiento de Reno y SACK es similar. Como en los casos anteriores se ha simulado la transferencia de archivos de 20 Megabytes para obtener resultados experimentales. Los resultados de simulación encajan a la perfección con la figura de mérito de la penalización por retardo. En este caso, la diferencia entre considerar o no tráfico de fondo es despreciable. La conclusión más importante es que los valores de temporizador de ensamblado por debajo de 10 ms no dañan el *goodput* de TCP. Sin

embargo, para valores altos, hay una degradación del rendimiento muy alta. Por tanto, los temporizadores de ensamblado altos están muy afectados por la penalización de retardo, y es por esta razón por la cual el rendimiento decae significativamente para estos valores del temporizador en la Figura 4-23 y la Figura 4-24.

4.7 Resumen y conclusiones

En esta sección, se ha comparado de manera cualitativa y mediante simulación con OPNET Modeler el *throughput* de TCP sobre OBS con y sin la utilización del asentimiento retardado. El análisis ha señalado dos aspectos negativos y uno positivo cuando se utiliza dicho algoritmo: un aspecto negativo es un tiempo mayor para recuperarse de una pérdida de ráfaga. Otro aspecto negativo es una reducción de la ganancia de correlación, debido a una transmisión no ideal de los segmentos de TCP, que hacen que una ventana de TCP se distribuya en varias ráfagas. El aspecto positivo es la mayor facilidad para TCP de recuperarse de una pérdida de ráfaga empleando asentimientos selectivos, al no concentrarse en cada ráfaga toda la ventana.

Por tanto, en unas condiciones de acceso simétrico como las simuladas, conviene que los clientes TCP no activen el algoritmo de asentimiento retardado, ya que se aumenta el *throughput* de las conexiones, como se observa en las simulaciones realizadas. Sin embargo, en condiciones de asimetría, si no se emplea asentimiento retardado se puede saturar el sentido de la transmisión limitante (típicamente el canal de subida al usuario). Para saber cuánto es la relación de asimetría a partir de la cual se puede no emplear el asentimiento retardado, hay que comparar los tamaños de los segmentos completos y solo de asentimiento. Un segmento, como máximo, tiene habitualmente 1460 bytes, que con los 20 bytes de la cabecera TCP, y 20 bytes IP, son unos 1500 bytes. Un asentimiento no tiene datos, por lo que tiene 40 bytes (20 de la cabecera TCP más otros 20 de la cabecera IP). A estos valores habría que sumarles el valor de la cabecera del nivel de enlace (será dependiente de la tecnología). Por tanto, sin contar el nivel de enlace, aproximadamente un asentimiento es 30 veces más pequeño que un segmento típico TCP. Cuando la relación ancho de banda de subida entre ancho de banda de bajada sea menor que 1/30 (es decir, se acerque más hacia una relación simétrica), se podría enviar un asentimiento por segmento sin que se resienta la conexión. Teniendo en cuenta que el enlace puede ser compartido con más flujos, para no llenar por completo el enlace de subida, se puede emplear como límite una relación aproximada de 1 a 10. Por debajo de esta relación de asimetría (es decir, a medida que se tiene una conexión menos asimétrica), se considera que el *delayed* ACK deja de tener su beneficio, y por tanto conviene desactivarlo. Otra opción es usar la técnica de *byte counting*, por la que se incrementa la ventana no por el número de asentimientos recibidos, sino por el número de bytes [195]. Sin embargo, esta opción prácticamente no se ha implementado, a pesar de recomendarse en el IETF [178].

Además de estudiar el impacto de un aspecto concreto de TCP, se ha añadido tráfico a un ensamblador OBS, y se ha comparado el comportamiento de TCP sobre OBS con y sin tráfico de fondo. Se ha comprobado que, si un flujo TCP se ensambla con flujos adicionales, se produce una fragmentación de la transmisión continua de la ráfaga, lo que lleva a una disminución del número de segmentos por ráfaga. Esto lleva a que cuando se estudia un flujo TCP junto con tráfico adicional, la distinción entre fuentes medias y rápidas se desvanece, y se observa un comportamiento parecido en todos los casos. Esto se debe a que al introducir tráfico adicional, la transmisión de segmentos TCP se fragmenta en distintas ráfagas, y el número medio de segmentos por ráfaga de un flujo concreto disminuye.

Cuando se estudia TCP considerando múltiples fuentes de tráfico en una red OBS, se observa que una pérdida de ráfaga no es siempre tan negativa como se pensaba en estudios previos, al contener la mayoría de ráfagas pocos segmentos. De esta forma, empleando asentimientos selectivos, TCP puede recuperarse rápidamente de la mayor parte de pérdidas de ráfagas en OBS. Para las probabilidades de pérdida de ráfaga empleadas, TCP obtiene un buen rendimiento, sin ser penalizado en exceso por dichas pérdidas.

Capítulo 5

Modelo periódico de TCP sobre OBS

En los capítulos anteriores se ha estudiado TCP sobre OBS empleando la técnica de la simulación. En este capítulo, con el objetivo por un lado de profundizar en el entendimiento de la dinámica de TCP sobre OBS, y por otro lado de obtener una expresión sencilla que pueda relacionar los principales parámetros del diseño de OBS con el rendimiento de una conexión TCP, se ha diseñado un modelo teórico para el rendimiento de TCP en redes OBS. Con este modelo se puede entender de una manera clara qué aspectos de OBS afectan a TCP.

5.1 Modelado de TCP

La dinámica del protocolo TCP es conocida y se puede replicar de una manera muy cercana a la realidad mediante simulación. Sin embargo, en las redes de comunicaciones, por ejemplo Internet, circulan de manera simultánea millones de flujos TCP, interactuando unos con otros en distintos puntos de la red. Por tanto, cuando se extiende el estudio de TCP a un entorno de esta magnitud, la simulación de la dinámica de TCP de miles de flujos se complica y requiere elevadísimos tiempos de simulación. Los modelos analíticos de TCP pueden complementar el estudio de los casos en los que la simulación no escala.

Por otro lado, cuando las aplicaciones emplean TCP como protocolo de transporte sobre una determinada tecnología de red, que por sus características concretas impacta sobre la dinámica del protocolo TCP, se puede realizar un modelado del comportamiento concreto de la red en aquellos aspectos que influyan a TCP. Por ejemplo, en una red inalámbrica se modelarán las pérdidas del canal [215], en una red de satélites unos retardos muy amplios [216], y en una red OBS, objetivo de esta tesis, se modela el proceso de pérdida de ráfagas. Así, el modelado matemático de TCP permite entender con mayor detalle el comportamiento e impacto de las distintas tecnologías de red en TCP complementando los estudios de simulación. De esta forma, el modelado permite estudiar el comportamiento de TCP en distintos tipos de redes.

5.1.1 Fundamentos del modelado de TCP

En la literatura se puede encontrar una amplia variedad de modelos teóricos de TCP [183]. En todo modelo matemático de TCP, son dos principalmente los aspectos que se representan, la dinámica la ventana de transmisión, y el proceso de pérdidas de segmentos TCP.

Dinámica de la ventana de transmisión

La dinámica de la ventana de la transmisión de TCP decide cuántos paquetes pueden transmitirse a la red, como se ha visto en detalle en el Capítulo 3, donde se estudian los aspectos concretos que determinan el valor de la ventana de transmisión. Según la dinámica de la ventana, la tasa de transmisión de una fuente TCP en un instante de tiempo *t*, denominada $X(t)$, independientemente de la versión específica de TCP, está relacionada

con el valor de la ventana en un instante ese instante de tiempo *t*, denominado $W(t)$, y el RTT (*Round Trip Time*, tiempo transcurrido desde que se envía un segmento TCP hasta que llega su asentimiento) mediante la ecuación (5-1) [183]:

$$X(t) = \frac{W(t)}{RTT} \tag{5-1}$$

El incremento y la disminución de la ventana de transmisión de TCP, tal y como se ha explicado en la sección 3.2 viene marcado por la versión específica y opciones concretas de TCP, y en su versión más genérica, coincide con un incremento lineal de la ventana con el tiempo, y un decremento a la mitad tras una pérdida. En otras versiones de TCP experimentales el incremento puede ser más agresivo, y el decremento no reducir la ventana en una proporción menor. No hay que olvidar además que la ventana de transmisión está limitada por el valor de la ventana de transmisión máxima, para evitar desbordar al cliente TCP.

Modelado del proceso de pérdidas

Las pérdidas de segmentos TCP se interpretan como un indicativo de carga o congestión en la red y provocan una reducción del tamaño de la ventana de transmisión. El proceso de pérdidas en una red de paquetes se suele modelar como un proceso estocástico, con una probabilidad *p* de perder un paquete en la red. La clave en el modelo es cómo TCP detecta esa pérdida y cómo reacciona ante ella. Habitualmente TCP detectará la pérdida o bien por el vencimiento de un temporizador tras la ausencia de asentimiento del segmento perdido, o mediante la llegada de asentimientos duplicados. Sin embargo, a pesar de ser las más comunes, éstas no son las únicas formas mediante las que TCP puede detectar pérdidas y situaciones de congestión, ya que es posible que TCP pueda recibir una señalización explícita de la red informando de la congestión [217], o que, como ocurre en la versión de TCP Vegas [201], se detecte la congestión por las variaciones percibidas en las medidas del RTT.

5.1.2 Modelos de TCP

El modelo más intuitivo y que captura la esencia de TCP es el denominado "Modelo periódico de TCP", propuesto por Mathis *et al.* [218], que considera un patrón periódico en la dinámica de la evolución de la ventana y el proceso de pérdidas. El modelo solo considera la evolución de la ventana en la fase de *congestion avoidance* y no tiene en cuenta el temporizador de retransmisión. A pesar de que el comportamiento de TCP se simplifica mucho, obtiene una expresión ampliamente utilizada y válida en la mayoría de las situaciones. De hecho, aplicando simplificaciones al resto de los modelos, se obtiene la expresión del modelo periódico. Además, del modelo periódico se obtiene una propiedad fundamental del rendimiento de TCP, la ley del inverso de la raíz cuadrada de *p*, siendo *p* la probabilidad de perder un paquete en la red, por la cual la tasa de transmisión de TCP es inversamente proporcional a la raíz cuadrada de la probabilidad de pérdida de paquete y es inversamente proporcional al RTT. La expresión básica de la tasa media de transmisión de TCP, en segmentos por segundo, viene dada por la ecuación (5-2):

$$\bar{X}(p) = \frac{1}{RTT}\sqrt{\frac{3}{2p}} \tag{5-2}$$

En la sección 5.3 se detalla el modelo propuesto por Mathis *et al.*, que se emplea en esta tesis como base del modelo periódico de TCP sobre OBS.

El siguiente tipo de modelo que se ha planteado en la literatura, también bastante intuitivo, pero más complejo, es el denominado modelo detallado de pérdida de Padhye *et al.* [206], corregido posteriormente en algunos detalles por Chen *et al.* [219]. En este modelo

se captura el detalle concreto y específico de la reacción de TCP ante una pérdida. En concreto, el modelo de Padhye *et al.* estudia en detalle el comportamiento de TCP Reno. Este modelo, y todos sus derivados se basan en calcular el número medio de segmentos entre pérdidas y la duración media del intervalo entre pérdidas. En estado estacionario, la tasa media de transmisión de TCP para una probabilidad de pérdida de segmento *p*, denominada $\bar{X}(p)$, es igual a la división de la media de segmentos enviados entre dos pérdidas de segmento ($E[Y]$) y el tiempo medio entre dos pérdidas de segmento ($E[A]$), como se muestra en la ecuación (5-3).

$$\bar{X}(p) = \frac{E[Y]}{E[A]} \tag{5-3}$$

Para calcular la media de los segmentos transmitidos entre pérdidas, se emplea por un lado el hecho de que en cada ronda se envían tantos segmentos como dice el valor de la ventana de transmisión, y por otro lado una estimación de los segmentos enviados tras la pérdida del segmento. El número de paquetes que se envían después de la pérdida depende del tipo de fallo en concreto (uno o varios segmentos) así como de la versión de TCP empleada. Este número puede variar desde cero en el caso de perder la ventana completa hasta varios, en los casos en los que, o se ha perdido un segmento, o se han perdido varios segmentos y se esté empleando asentimientos selectivos.

Por otro lado, para calcular el tiempo entre pérdidas, el modelo propone en primer lugar dividir el tiempo en rondas de transmisión, siendo el tiempo de una ronda equivalente al RTT. Luego, se emplea la suposición de que la probabilidad de que en una ronda haya una pérdida depende de cuantos segmentos se hayan transmitido. De esta forma, sabiendo el número de segmentos por ronda, se puede saber con qué probabilidad ocurrirá una pérdida en cada ronda, y a partir de ahí, el número de rondas medio entre fallos. Además del tiempo del número de rondas hasta que ocurre la pérdida, hay que añadir el tiempo desde que se produce la pérdida, hasta que TCP se entera de la misma y reacciona. Por ejemplo, en el caso de que se detecte la pérdida mediante el temporizador de retransmisión, hay que añadir dicho tiempo. El modelo de Padhye *et al.* incluye un cálculo muy detallado de dicho temporizador. De esta forma, el modelo detallado de pérdidas modela tanto el caso en el que la recuperación de las pérdidas se lleve a cabo tanto por asentimientos duplicados como cuando se lleva a cabo por temporizador de retransmisión.

El modelo de Phadye obtiene una expresión para calcular la tasa de transmisión media $(\bar{X})$ en función de la probabilidad de pérdida de segmento *p*, que se muestra la ecuación (5-4). El modelo es válido para el caso en el que las pérdidas se detectan siempre con triple asentimiento duplicado y se envía un asentimiento cada *b* segmentos. Hay que recordar que, como se vio en el Capítulo 4, si se emplea la técnica de asentimiento retardado, en la mayoría de las ocasiones se envía un asentimiento cada 2 dos segmentos TCP.

$$\bar{X}(p) = \frac{\frac{1-p}{p} + \frac{2+b}{b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2}}{RTT\left(\frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} + 1\right)} \tag{5-4}$$

Para hacerse una idea de la diferencia de precisión entre el modelo de Padhye *et al.* y el modelo periódico, para una conexión con un *round trip time* de 200 ms y p=0,05, el modelo periódico obtiene una tasa de 27,4 paquetes por segundo, mientras que el modelo detallado de pérdidas, obtiene 26,69 [183]. Es decir, para una probabilidad de pérdida de segmento bastante alta como es 0,05 la diferencia es menor del 3%. Por lo tanto, a pesar de ser un modelo más complejo, los resultados son apenas diferentes. Por otro lado, en el modelo de

Phadye *et al.*, la relación entre los parámetros (la probabilidad de pérdida de segmento y el número de segmentos asentidos por ACK) y la tasa de transmisión no es evidente, al contrario que en el modelo periódico donde se identifica rápidamente que la tasa de transmisión de TCP crece con el inverso de la raíz de la probabilidad de pérdida de segmento.

De hecho, para valores pequeños de la probabilidad de pérdida p (los habituales en una red en estado normal), y para b igual a 1 (un asentimiento por segmento), la expresión del modelo de Phadye *et al.* se puede simplificar y se obtiene la ecuación (5-2) del modelo periódico de Mathis *et al.*

El modelo detallado de pérdidas de Padhye *et al.* tiene en cuenta un caso muy interesante, en el que la ventana de transmisión de TCP está limitada, al contrario que el modelo periódico propuesto por Mathis *et al.* que solamente considera ventana ilimitada.

Posteriormente, el modelo de Padhye *et al.* ha sido ampliado y detallado para situaciones concretas y versiones específicas de TCP. Por ejemplo, Sikdar *et al.* [220] realizaron un estudio detallado de la latencia (retardo) de la transmisión de los segmentos TCP, especialmente en la fase de *slow start*. Con este análisis, se puede estimar con mayor detalle el *throughput* de TCP para conexiones de muy corta duración. Para conexiones de larga duración, la mejora de este nuevo modelo en el cálculo del *throughput* es insignificante.

Por otro lado, con respecto a otras versiones de TCP, Zhou *et al.* [221] completaron el modelo de Padhye *et al.* [206] y de Sikdar *et al.* [220] obteniendo un nivel de precisión muy alto, al modelar con detalle TCP SACK [Zhou05]. El modelado de SACK fue completado con aún más detalle por Xu *et al.* [222], que incluyeron las pérdidas de segmentos retransmitidos, con los temporizadores que entran en juego. La versión de TCP *New Reno* fue modelada por Zhou *et al.* [223] siguiendo la misma aproximación, y ampliado por Parvez *et al.* [224].

Aunque el modelado de TCP basado en la aproximación de Padhye *et al.* ha sido muy popular en la literatura, se han realizado otras aproximaciones alternativas. Por ejemplo, se ha realizado un modelado de TCP basado en una cadena de Markov [225]. En este caso, cada estado es un valor de la ventana, y se modelan las transiciones de una ventana a otra. El principal inconveniente es el elevado número de estados que se tiene, lo que hace que solo sea factible la obtención de resultados numéricos.

Finalmente, hay que destacar que el modelado de TCP se ha empleado para estudiar el comportamiento de TCP y nuevas versiones de TCP sobre distintos tipos de redes. Uno de los entornos más estudiados ha sido el inalámbrico con numerosos modelos propuestos [215]. Otro entorno muy estudiado ha sido el de las redes de satélites [216]. Esta tesis, con respecto al modelado de TCP, se centra en el comportamiento sobre redes OBS.

5.2 Modelado de TCP sobre OBS

El primer estudio del comportamiento detallado de TCP sobre OBS fue llevado a cabo por Detti y Listanti [50], quienes modelan el comportamiento de un flujo TCP Reno que atraviesa un ensamblador OBS con una estrategia de ensamblado de temporizador fijo. Este primer modelo se basa en el modelo anterior de TCP desarrollado por Padhye *et al.* [206] en el que, como se ha visto en la sección previa, se modela el detalle de las pérdidas de segmentos TCP de una conexión TCP Reno de larga duración, con y sin limitación de ventana, y sin *delayed* ACK.

En el modelo de Detti y Listanti, para calcular la tasa de transmisión de TCP se obtiene el número medio de segmentos entre pérdidas de ráfagas y se divide entre el tiempo medio entre pérdidas. A la hora de realizar estos cálculos, Detti y Listanti son los primeros en introducir la noción de fuentes TCP 'lentas', 'medias' y 'rápidas', según la relación entre el

valor del temporizador de ensamblado y el ancho de banda de acceso, y que determinará si en una ráfaga viaja o bien un único segmento del flujo TCP (fuentes lentas), varios segmentos del flujo TCP (fuentes medias), o la ventana completa del flujo (fuentes rápidas), que condicionará la reacción de TCP ante la pérdida de la ráfaga.

Detti y Listanti [50] en su estudio modelan la tasa de transmisión media de fuentes TCP lentas y fuentes TCP rápidas. En el caso de las fuentes lentas, la tasa de transmisión se obtiene aplicando directamente el modelo de Padhye *et al.* con el valor de RTT teniendo cuenta el tiempo de ensamblado (se suma al RTT sin ensamblador el valor del temporizador de ensamblado) y el valor del temporizador de retransmisión, que Detti y Listanti calculan de manera exacta, ya que éste depende de la varianza de las medidas del RTT (que están afectadas por el ensamblador).

Para el modelo de fuentes rápidas, al modelar TCP Reno, que como se ha visto en las secciones anteriores, necesita recuperarse de las pérdidas múltiples con temporizador de retransmisión, el modelo de Detti y Listanti centra su desarrollo en aplicar el modelo de Padhye *et al.* cuando hay únicamente pérdidas detectadas con dicho temporizador.

La principal aportación del modelo de Detti y Listanti con respecto a TCP y OBS es la formalización de dos de sus efectos, uno, el más obvio, denominado 'penalización por retardo', por el que se reduce la tasa de transmisión TCP y otro denominado 'beneficio de correlación', por el que se incrementa la tasa de transmisión TCP con respecto a un hipotético caso en el que no hubiera ensamblado, pero se mantuviera la misma probabilidad de pérdida de segmento que de ráfaga. De este 'beneficio de correlación', Detti y Listanti apuntan, de una manera cualitativa, que es mayor cuantos más segmentos haya por ráfaga, y que tiene su máximo para valores de la probabilidad de pérdida alrededor del inverso de la ventana máxima de transmisión. El principal inconveniente de este modelo es que solo cubre los casos de flujos extremos de temporizador de ensamblado (fuentes 'lentas' y fuentes 'rápidas'), en los que o solo va un flujo, o viaja toda la ventana en una ráfaga.

El siguiente modelo fue desarrollado por Yu *et al.* [51], que completó el modelo de Reno de Detti y Listanti incluyendo las fuentes TCP de velocidad 'media'. Además del modelo de Reno, Yu *et al.* proponen un modelo de TCP New Reno y SACK. Para todos ellos, al igual que Detti y Listanti, Yu *et al.* se basan en el modelo de Padhye *el al.* [206] en el que se analiza en detalle la pérdida de paquetes.

El modelo de Yu *et al.* llega a unas conclusiones parecidas a las de Detti y Listanti, pero las formaliza y cuantifica. De esta forma definen dos efectos, uno negativo, la penalización por retardo (ya mencionada por Detti y Listanti), y la ganancia denominada DFL (*delayed first loss*), que Detti y Listanti denominan beneficio de correlación. Esta ganancia es proporcional a la raíz cuadrada del número medio de segmentos por ráfaga, y se debe a que la probabilidad de que haya una pérdida en una ronda de transmisión es menor cuanto mayor sea el número de segmentos por ráfaga, ya que la probabilidad de pérdida de ráfaga se asume constante e independiente de las características de la ráfaga.

El modelo de Yu *et al.* se fundamenta en que en cada ronda de transmisión el número de ráfagas que se van a transmitir es conocido y la probabilidad de pérdida para ese instante es directamente proporcional al número de ráfagas necesarias para transmitir esa ventana. Para ello, primero estiman el número medio de segmentos por ráfaga de una manera trivial, calculando cuantos segmentos cabrían en una ráfaga dada una velocidad de acceso, y luego dividen la ventana de transmisión entre ese número. Sin embargo, tal y como se comprueba mediante simulación en el capítulo anterior, al obtener el número medio de segmentos de un flujo por ráfaga de esta forma se comete un error muy grande, lo que hace que el modelo se comporte mal con pérdidas medias.

El modelo de Yu *et al.* fue ampliado posteriormente por Zhang *et al.* [131] para incluir retransmisiones. La particularidad de las retransmisiones a nivel OBS es que pueden interferir con los mecanismos de TCP. Por un lado, para el caso de las fuentes 'rápidas', si se pierde una ráfaga con todos los segmentos de la ventana del flujo, y la ráfaga se retransmite, TCP no detectará ninguna pérdida. El único impacto será un mayor retardo, disminuyendo ligeramente el *throughput*. En el modelo propuesto, Zhang separa la probabilidad de pérdida de ráfaga en dos nuevos conceptos, la probabilidad de que suceda una contienda y probabilidad de descarte de ráfaga. Cuando hay una contienda, la ráfaga se pierde y se retransmite, pero con cierta probabilidad la retransmisión no tiene éxito, y en el resto de los casos, se retransmite satisfactoriamente. Por tanto, para tener en cuenta el impacto de las retransmisiones, se retoca el modelo de Yu *et al.* incluyendo la probabilidad de descarte de ráfaga en el lugar de la probabilidad de pérdida, y para el RTT se añade la probabilidad de que haya habido descarte y se haya recuperado con retransmisión multiplicada por el tiempo extra de la retransmisión.

5.3 Propuesta de modelo

Los modelos propuestos en la literatura, por un lado, obtienen expresiones en las que es difícil observar cuál es la relación principal de la tasa de transmisión de TCP y OBS. Por otro lado, realizan una aproximación demasiado simple para el número de segmentos por ráfaga que limita la validez de los modelos. Este hecho hace que los modelos teóricos de la literatura sean únicamente aplicables con precisión al caso de un único flujo. El error cuando se ensamblan múltiples flujos puede ser muy elevado.

Así pues, se propone un modelo que tenga en cuenta cómo se ensambla un flujo TCP cuando comparte ensamblador con más flujos TCP. Además, debido a la dificultad de identificar la influencia de los parámetros principales de OBS en los modelos de la literatura, se buscará que se obtenga una expresión analítica de la que fácilmente se pueda derivar cómo afectan los parámetros de OBS a la tasa de transmisión.

Por tanto, con los dos objetivos mencionados, esta tesis propone un nuevo modelo para la estimación del rendimiento de TCP sobre OBS partiendo del modelo periódico de TCP propuesto por Mathis *et al.* [218]. Para entender el modelo, se comienza revisando en detalle dicho modelo periódico. A continuación, se amplía el modelo para considerar la limitación de ventana de transmisión, ya que en la literatura se ha comprobado que esta es la carencia principal del modelo de Mathis *et al.* Después, se particulariza el modelo periódico de Mathis *et al.* para el transporte sobre una red OBS (primero sin limitación de ventana y después incluyendo la limitación). Uno de los aspectos clave en los que los modelo de la literatura no han conseguido dar respuesta es en estimar correctamente el número de segmentos por ráfaga (Yu *et al.* realizan una aproximación únicamente válida para el caso de un único flujo TCP). En la tesis se obtiene de manera analítica una cota superior y otra inferior para dicho número. Después se obtiene mediante simulación, en varios escenarios, la distribución del número de ráfagas necesarias para transmitir una determinada ventana de TCP. A partir de estos datos empíricos de las simulaciones, se obtienen indicaciones de cómo estimar mejor el número de segmentos por ráfaga y así completar el modelo periódico de TCP. Finalmente, se procede a validar el modelo con un escenario de simulación en el que se transfieren un conjunto de ficheros de gran tamaño y se mide su rendimiento (velocidad a la que se enviaron los ficheros).

Detalle del modelo periódico de TCP

El modelo periódico de TCP [218] se basa en reproducir un patrón periódico en el comportamiento de la dinámica de la ventana de TCP. Este modelo no asume ninguna versión específica de TCP, y se centra en el comportamiento de una conexión en estado estacionario en la que ocurren pérdidas de paquetes (leves o moderadas). En esta sección se procede a revisar y desarrollar dicho modelo, ya que es la base del modelo propuesto para TCP sobre OBS.

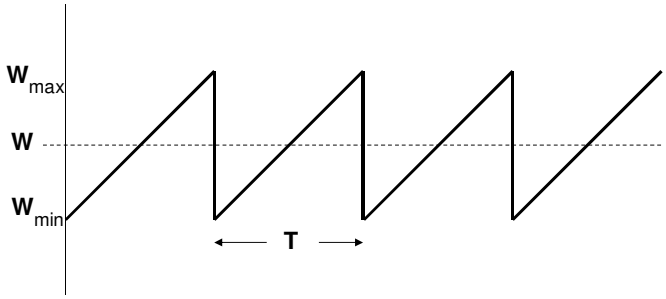


Figura 5-1 Evolución de la ventana de TCP en el modelo periódico

El principio de este modelo es que, partiendo del supuesto en el que hay una probabilidad de pérdida de segmento *p*, se realiza la simplificación de que se pierde un segmento exactamente cada *1/p* segmentos. Este comportamiento 'perfecto', no se corresponde fielmente con la realidad, pero como aproximación para representar el estado estacionario y estudiar la tasa de transferencia media, es adecuado (no así si se quisiéra estudiar más allá de la media, por ejemplo la varianza). Por otro lado, por la regla del crecimiento aditivo (fase de *congestion avoidance*), la ventana de transmisión crece en un segmento cada RTT (sin *delayed* ACK), mientras que por la regla del decrecimiento multiplicativo, la ventana se reduce a la mitad en el caso de pérdida (este es el comportamiento más habitual). Hay que recordar que el *Round Trip Time* (RTT) es el tiempo que transcurre desde que se envía un segmento TCP hasta que se recibe un asentimiento que confirma su correcta recepción. La evolución periódica de la ventana en forma de sierra perfecta se puede observar gráficamente en la Figura 5-1 . En esta figura, se define *T* como el periodo entre pérdidas, W_{max} como el valor máximo de la ventana de transmisión, W_{min} como el valor mínimo de la ventana de transmisión y W como el valor medio de la ventana de transmisión.

La tasa de transferencia media de TCP se obtendrá dividiendo el número de segmentos transmitidos en el intervalo entre fallos entre la duración de dicho intervalo entre fallos. El número de segmentos transmitidos se puede obtener fácilmente calculando el área de uno de los periodos del diente de sierra, ya que la ventana indica el número de segmentos que se pueden transmitir en cada RTT. Este área se obtiene fácilmente sumando el área de un rectángulo de cuya base es el número de RTTs en el periodo de tiempo *T*, y altura W_{min}, y el área del triángulo superior, de igual base y lado $W_{max}/2$ (la altura se obtiene restando de W_{max} el valor de W_{min}, y como la ventana se reduce a la mitad en caso de perdida $W_{min} = W_{max}/2$). Teniendo en cuenta que la ventana de transmisión se incrementa con cada asentimiento recibido, es decir, en $1/b$ segmentos cada RTT, el número de RTTs necesarios para incrementar la ventana desde W_{min} hasta W_{max} es $bW_{max}/2$. Así, el número de segmentos transmitidos en el intervalo de tiempo *T* viene dado por la ecuación (5-5).

$$n_{seg_{tx}} = b\left(\frac{W_{max}}{2}\right)^2 + \frac{1}{2}b\left(\frac{W_{max}}{2}\right)^2 = \frac{3}{8}b(W_{max})^2 \qquad (5\text{-}5)$$

Por otro lado, como el número de segmentos transmitidos ($n_{seg_{tx}}$) es, por definición en el modelo, igual a *1/p*, sustituyendo en (5-5), la ventana de transmisión máxima W_{max} se obtiene con la ecuación (5-6).

$$W_{max} = \sqrt{\frac{8}{3bp}} \tag{5-6}$$

De las ecuaciones (5-5) y (5-6), sabiendo que la duración del periodo de pérdidas *T* es $RTT \cdot b \cdot W_{max}/2$, y teniendo cada segmento TCP un tamaño de MSS (*Maximum Segment Size*) bytes, se puede deducir la tasa de transferencia media ($\bar{X}$), que se muestra en la ecuación (5-7).

$$\bar{X} = \frac{n_{seg_{tx}} \cdot MSS}{T} = \frac{\frac{3}{8} b (W_{max})^2 \cdot MSS}{RTT \cdot b \cdot \frac{W_{max}}{2}} = \frac{3}{4} W_{max} \frac{MSS}{RTT} = \frac{MSS}{RTT} \frac{\sqrt{3/2b}}{\sqrt{p}} \tag{5-7}$$

El resultado es una aproximación válida para muchos casos, y universalmente reconocida. Las principales carencias de este modelo son:

- No modela la limitación de ventana, por lo que si una conexión se encuentra en su mayor parte del tiempo con la ventana al máximo este método sobreestima el rendimiento.
- Asume que el emisor siempre tiene listos datos para transmitir (esto se aplica a todos los modelos de TCP).
- No modela las pérdidas por detección de temporizador de retransmisión y en casos de pérdidas elevadas, o en casos en los que SACK no está activo, puede haber este tipo de pérdidas.
- En ciertas ocasiones, se tarda en llegar al estado estacionario, por ejemplo en el caso de conexiones cortas. En estos casos el modelo no es válido.

Inclusión del asentimiento retardado

Si se tiene en cuenta el algoritmo *delayed* ACK (asentimiento retardado), la ventana de transmisión se incrementa en un ritmo de 1 segmento cada 2 RTT, en vez de cada RTT. Esto se debe a que, aproximadamente, se envía un asentimiento cada 2 segmentos, y TCP incrementa la ventana con la llegada de asentimientos, independientemente de la cantidad de datos que sean asentidos. Puede haber casos en los que se envíe un asentimiento cada 1 en vez de 2, pero se desprecian estos casos para esta aproximación. Por tanto, al poner *b*=2, se obtiene la tasa de transmisión con la ecuación (5-8).

$$\bar{X}_{del\_ack} = \frac{MSS}{RTT} \frac{\sqrt{3}/2}{\sqrt{p}} \tag{5-8}$$

Por lo tanto, la relación entre la tasa de transmisión de TCP con y sin *delayed* ACK es la mostrada en la ecuación (5-9), suponiendo que no hay limitación en el sentido de envío de los ACKs:

$$\bar{X}_{del\_ack} = \frac{\bar{X}_{\text{sin}\_del\_ack}}{\sqrt{2}} \tag{5-9}$$

5.4 Limitación de ventana en el modelo periódico

Como se ha visto en el apartado anterior, el modelo periódico de TCP propuesto por Mathis *et al.* no tiene en cuenta la limitación de ventana máxima de TCP. Esta limitación ha de tenerse en cuenta, debido a que los clientes y servidores imponen unos tamaños

máximos de ventana. Esto hace que la conexión se encuentre durante un tiempo significativo en el estado de ventana máxima, sin aumentar ni reducir. La limitación de ventana ha sido incluida en otros modelos, como por ejemplo en el modelo detallado de pérdida [206].

En esta sección se propone una extensión al modelo periódico para calcular el rendimiento aproximado de TCP partiendo del modelo periódico añadiéndole la limitación de ventana. De este modo, se obtiene una aproximación mejor que la anterior para los casos en los que las transmisiones lleguen a alcanzar la ventana máxima.

Si se considera la limitación de ventana, la evolución de la ventana de transmisión deja de ser un diente de sierra, para incorporar unas zonas 'planas', en las que la ventana no crece más. Como en el modelo periódico básico, se considera que se produce una pérdida cada $1/p$ segmentos. La evolución de la ventana se puede observar en la Figura 5-2, en la que se representan varios tiempos, T_1 es el tiempo durante el cual la ventana crece hasta llegar a su máximo valor, T_2 es el tiempo que permanece el flujo TCP con la ventana a su máximo valor hasta que se produce una pérdida de segmento, y T es el tiempo total entre dos pérdidas de segmentos.

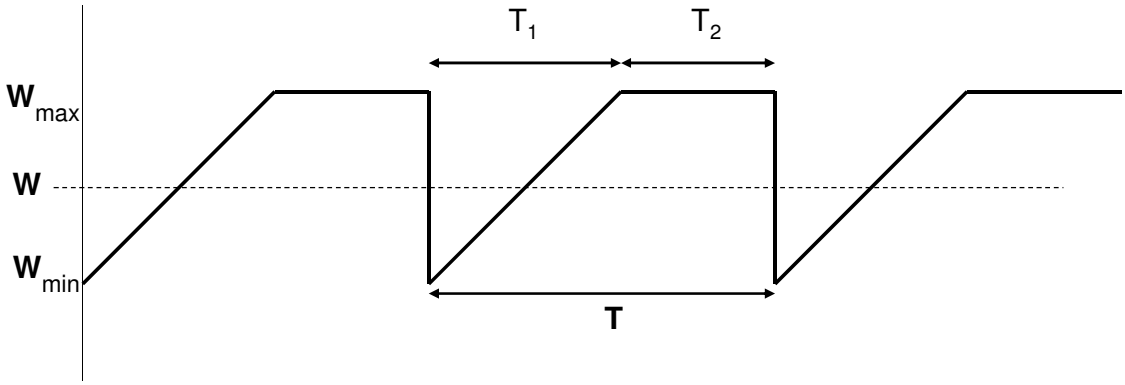


Figura 5-2 Evolución de la ventana de TCP en el modelo periódico con limitacion de ventana

Para el caso en que en la evolución de la ventana se tengan zonas "planas", el método a emplear será el siguiente. El *throughput* se btendrá de dividir el número de segmentos transmitidos en T, dividido entre T. El número de segmentos transmitidos es, igual que en el caso anterior, $1/p$, ya que el modelo propuesto asume que hay una pérdida cada $1/p$ segmentos. Por otro lado se calcula T a partir de los dos intervalos, denominados T_1 y T_2 en la Figura 5-2. La clave es que en este caso W_{max} es conocida e igual al valor máximo del *buffer* de recepción. Por tanto, es fácil deducir el valor del primer intervalo T_1. Por la regla del crecimiento aditivo (la ventana crece en $1/b$ segmentos cada RTT) se obtiene la ecuación (5-10).

$$W_{max} = W_{min} + T_1 \frac{1}{bRTT} \tag{5-10}$$

Aplicando la regla del decrecimiento multiplicativo, por la que, en la mayoría de las ocasiones, la ventana se divide a la mitad tras una pérdida, con lo que $W_{min} = W_{max}/2$, se obtiene la ecuación (5-11).

$$T_1 = \frac{1}{2} W_{max} \cdot b \cdot RTT \tag{5-11}$$

Una vez conocido el valor del tiempo en el que la ventana crece (T_1), se procede a calcular T_2. Esta segunda componente del tiempo entre fallos se obtiene calculando primero el número de segmentos transmitidos en dicho tiempo de ventana al máximo, y partir de esta cifra, como durante T_2 la ventana permanece constante, se puede calcular de manera trivial el número de rondas necesarias para transmitir esos segmentos. El número de segmentos en T_2 se calcula restando del número de segmentos transmitidos en total (que

es conocido e igual a *1/p*) el número de segmentos transmitidos en T_1, que se puede calcular fácilmente a partir de la ecuación (5-5). Por tanto, el número de segmentos transmitidos en T_2, que se denomina $n_{seg_{tx}enT_2}$, viene dado por la ecuación (5-12).

$$n_{seg_{tx}enT_2} = \frac{1}{p} - \frac{3}{8} b (W_{max})^2 \tag{5-12}$$

Como en cada ronda de duración RTT se transmiten W_{max} segmentos, se obtiene el tiempo de permanencia en la ventana máxima de transmisión con la ecuación (5-13):

$$T_2 = \left(\frac{1}{p} - \frac{3}{8} b (W_{max})^2\right) \frac{RTT}{W_{max}} \tag{5-13}$$

Dividiendo el número de segmentos transmitidos en T ($n_{seg_{tx}enT}$, que por definición es *1/p*) entre el tiempo entre pérdidas T, se obtiene la tasa de transmisión media, considerando que en algún momento se alcanza la ventana máxima ($\bar{X}_{win\_max}$), mediante la ecuación (5-14).

$$\begin{aligned} \bar{X}_{win\_max} &= \frac{n_{seg_{tx}enT}}{T} MSS = \frac{1/p}{T_1 + T_2} MSS \\ &= \frac{1/p}{\frac{1}{2} W_{max} \cdot b \cdot RTT + \frac{RTT}{p \cdot W_{max}} - \frac{3}{8} bRTT \cdot W_{max}} MSS \end{aligned} \tag{5-14}$$

Simplificando (5-14) se obtiene la expresión de la ecuación (5-15) para dicha tasa de transmisión media $\bar{X}_{win\_max}$.

$$\bar{X}_{win\_max} = \frac{MSS}{RTT \cdot \left(\frac{b}{8} p \cdot W_{max} + \frac{1}{W_{max}}\right)} \tag{5-15}$$

Con lo cual, el *throughput* medio de TCP con limitación de ventana, se puede obtener de una manera aproximada mediante la ecuación (5-16), donde se agrupan las ecuaciones (5-7) y (5-15) para considerar los dos casos posibles: aquel en el que no se alcanza la ventana de transmisión máxima y aquel en el que sí se alcanza.

$$\bar{X}_{win\_\max} = \begin{cases} \dfrac{MSS}{RTT \cdot \left(\frac{b}{8} p \cdot W_{max} + \frac{1}{W_{max}}\right)} & \text{si } \dfrac{1}{p} > \dfrac{3}{8} b W_{max}^2 \\ \dfrac{MSS}{RTT} \dfrac{\sqrt{3/2b}}{\sqrt{p}} & \text{si } \dfrac{1}{p} \leq \dfrac{3}{8} b W_{max}^2 \end{cases} \tag{5-16}$$

5.5 Modelo periódico de TCP en OBS sin limitación de ventana

Por un lado, por el hecho de ensamblar los paquetes en ráfagas, OBS introduce un retardo adicional a los paquetes. En una estrategia de ensamblado basada en temporizador, se envía una nueva ráfaga periódicamente, y el tiempo entre ráfagas es constante e igual al valor del temporizador. El tiempo que permanece un segmento antes de que se envíe la ráfaga está dentro del intervalo [0, T_b] donde T_b es el periodo de generación de ráfagas. El asentimiento generado, tendrá que entrar de igual manera en un generador de ráfagas, esperando un tiempo acotador por el intervalo [0, T_b]. Por lo tanto, el tiempo entre la transmisión de un segmento y la llegada de su correspondiente asentimiento está dentro del intervalo [RTT_0, RTT_0+2T_b], donde RTT_0 es el *RTT* sin tener en cuenta el ensamblado. Así, en el caso peor, el RTT máximo en OBS (RTT_{OBS_MAX}) se obtiene con la ecuación (5-17).

$$RTT_{OBS\_MAX} = RTT_0 + 2T_b \quad (5\text{-}17)$$

El RTT que observa un segmento TCP en media (RTT_{OBS}) se obtiene con la ecuación (5-18).

$$RTT_{OBS} = RTT_0 + T_b \quad (5\text{-}18)$$

Por lo tanto, tiempos de agregación grandes en comparación con el RTT pueden provocar notables descensos del *throughput*. Para otro tipo de algoritmos de ensamblado el valor del temporizador puede variar, por ejemplo en el caso de un algoritmo de tamaño máximo de ráfaga, si el tráfico es bajo, el tiempo hasta completar la ráfaga puede ser bajo. En todo caso, en una red OBS es conveniente que se limite el tiempo máximo de formación de ráfaga, para evitar retardos excesivos.

Pérdidas de ráfagas

Por otro lado, en OBS no se dispone de probabilidad de pérdida de segmento *p*, sino de probabilidad de pérdida de ráfaga, p_b. Se realiza la suposición de que la probabilidad de pérdida de ráfaga es independiente de T_b. Esta aproximación es válida siempre que todos los ensambladores de la red usen el mismo temporizador, ya que en caso contrario, los temporizadores más pequeños producirán ráfagas de menor tamaño que tienen mayor probabilidad de encontrar hueco que las grandes. Como primera aproximación, se asume que en cada ráfaga viajan en media *S* segmentos de un mismo flujo TCP, donde *S* tiene un valor entre 1 y S_{max}, siendo S_{max} el máximo número de segmentos que se pueden transmitir en una ráfaga dada una determinada velocidad de acceso. En secciones posteriores, se profundiza en el número de segmentos TCP por ráfaga. Por lo tanto, realizando la mencionada suposición, se perderá una ráfaga cada $1/p_b$ ráfagas, que con la aproximación mencionada equivale a una pérdida de ráfaga cada S/p_b segmentos.

Se realiza el supuesto, al igual que en el modelo periódico básico, de que la ventana de transmisión crece de acuerdo al crecimiento aditivo, al ritmo de un segmento cada RTT (o cada 2 si hay asentimientos retardados, o generalizando *b* segmentos por ronda). Sin embargo, cuando se pierde una ráfaga, se pierden *S* segmentos. La regla de reducir a la mitad la ventana de transmisión en caso de pérdida de ráfaga con *S* mayor que 2 segmentos, sólo puede aplicarse en la versión SACK, siempre y cuando *S* sea menor que W_{max}. Por lo tanto, en un principio se centra el estudio en el *throughput* de TCP con asentimientos selectivos en OBS. Hay que destacar que en la actualidad SACK está muy extendido, con lo que se puede asumir que se empleen los asentimientos selectivos.

Para TCP con SACK, con la suposición de que *S* es tal que TCP pueda recuperarse sin recurrir al vencimiento del temporizador, se pueden emplear los supuestos del modelo periódico básico y emplear la ecuación (5-5) que por la que se calcula el valor del número de segmentos transmitidos entre dos eventos de pérdida de segmento. Por otro lado, como el número de ráfagas entre pérdidas $1/p_b$ y el número medio de segmentos por ráfaga es S, el número de segmentos entre eventos de pérdida ($n_{seg_{tx}}$) es S/p_b, que sustituyendo en la ecuación (5-5), se obtiene la igualdad de la ecuación (5-19):

$$\frac{3}{8} b (W_{max})^2 = \frac{S}{p_b} \quad (5\text{-}19)$$

Despejando de la ecuación (5-19), se obtiene el valor de W_{max} en la ecuación (5-20).

$$W_{max} = \sqrt{\frac{8S}{3bp_b}} \quad (5\text{-}20)$$

De la ecuación (5-20), sabiendo que el *throughput* medio es el número de segmentos entre pérdidas ($n_{seg_{tx}}$) multiplicado por el tamaño de cada segmento (se asume el tamaño máximo de segmento *MSS*) dividido entre el la duración del periodo de pérdidas *T*, y que

este es $RTT_{OBS} \cdot b \cdot W_{max}/2$, se puede deducir la tasa de transferencia media, que se denomina $\bar{X}_{OBS}$, y que se muestra en la ecuación (5-21).

$$\bar{X}_{OBS} = \frac{n_{seg_{tx}} \cdot MSS}{T} = \frac{\frac{3}{8} b (W_{max})^2 \cdot MSS}{RTT_{OBS} \cdot \frac{W_{max}}{2} b} = \frac{3}{4} W_{max} \frac{MSS}{RTT} = \tag{5-21}$$

$$\bar{X}_{OBS} = \frac{MSS}{RTT_{OBS}} \frac{\sqrt{3S/2b}}{\sqrt{p_b}} \tag{5-22}$$

Por lo tanto, el *throughput* de TCP se incrementa en proporción a $\sqrt{S}$. Sin embargo, hay que considerar el incremento del RTT por el proceso de ensamblado, con lo que combinando (5-22) y (5-18) se obtiene que la tasa de transferencia media en OBS ($\bar{X}_{OBS}$) viene determinada por la ecuación (5-23).

$$\bar{X}_{OBS} = \frac{MSS}{(RTT_0 + T_b)} \frac{\sqrt{3S/2b}}{\sqrt{p_b}} \tag{5-23}$$

5.6 Modelo periódico con limitación de ventana

Tal y como se ha comentado anteriormente, los extremos de una conexión TCP imponen siempre límites a la ventana de transmisión, de forma que solamente a nivel teórico la ventana puede crecer hasta el infinito. Por tanto, al igual que se ha realizado con el modelo periódico básico, se añade al modelo periódico la limitación de ventana.

Añadiendo la restricción de que la ventana de TCP no pueda crecer por encima de un valor, se tiene un comportamiento como el de la Figura 5-3.

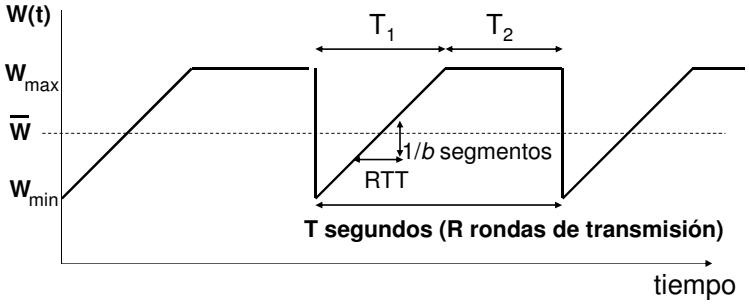


Figura 5-3 Comportamiento ideal de la evolución de la ventana de TCP con limitación de ventana máxima

Se sigue el mismo planteamiento que cuando se añadió la limitación de ventana al modelo periódico básico, por lo que se obtiene T_1 (periodo de tiempo en el que la ventana de transmisión está creciendo) y T_2 (periodo de tiempo en el que la ventana de transmisión está al máximo) de la misma manera. El número de segmentos en T_2 se calcula restando del número de segmentos transmitidos en total (que es conocido e igual a S/p) el número de segmentos transmitidos en T_1, que se puede calcular fácilmente a partir de la ecuación (5-5). Así, el número de segmentos en T_2 ($n_{seg_{tx}enT_2}$), se muestra en la ecuación (5-24).

$$n_{seg_{tx}enT_2} = \frac{S}{p_b} - \frac{3}{8} b (W_{max})^2 \tag{5-24}$$

Como durante el intervalo de tiempo T_2 la ventana de transmisión está a su valor máximo W_{max}, en cada ronda de duración RTT_{OBS} se transmiten W_{max} segmentos. Por tanto, el valor del intervalo de tiempo T_2 se obtiene con la ecuación (5-25).

$$T_2 = \left(\frac{S}{p_b} - \frac{3}{8} b (W_{max})^2\right) \frac{RTT_{OBS}}{W_{max}} \tag{5-25}$$

La tasa de transmisión de TCP en OBS, cuando en algún momento se alcanza la ventana de transmisión máxima, denominado $\bar{X}_{win\_\max\_OBS}$, se obtiene dividiendo el número de segmentos transmitidos en el intervalo de tiempo *T* ($n_{seg_{tx}enT}$) multiplicado por su tamaño (MSS), entre el valor del intervalo de tiempo *T* (que es la suma de T_1 y T_2). El valor de $\bar{X}_{win\_\max\_OBS}$ se muestra en la ecuación (5-26).

$$\begin{aligned} \bar{X}_{win\_\max\_OBS} &= \frac{n_{seg_{tx}enT_b}}{T} MSS = \frac{S/p_b}{T_1 + T_2} MSS \\ &= \frac{S/p_b}{\frac{1}{2} W_{max} \cdot b \cdot RTT_{OBS} + \frac{S \cdot RTT_{OBS}}{p_b \cdot W_{max}} - \frac{3}{8} \cdot b \cdot RTT_{OBS} \cdot W_{max}} MSS \end{aligned} \tag{5-26}$$

Simplificando la ecuación (5-26), se obtiene en la ecuación (5-27) una expresión para la tasa de transmisión de TCP en OBS considerando que se alcanza la ventana máxima de transmisión.

$$\bar{X}_{win\_\max\_OBS} = \frac{MSS}{RTT_{OBS} \cdot \left(\frac{b}{8} \frac{p_b}{S} \cdot W_{max} + \frac{1}{W_{max}}\right)} \tag{5-27}$$

A partir de las ecuaciones (5-27) y (5-23), y buscando el valor de *p* para encontrar el punto exacto en el que la pérdida de ráfaga en el modelo periódico ocurre en el instante en que la ventana llega a su máximo valor, el *throughput* de TCP ($\bar{X}_{OBS}$), teniendo en cuenta la limitación de ventana, se puede obtener mediante la ecuación (5-28).

$$\bar{X}_{OBS} = \begin{cases} \dfrac{MSS}{(RTT_0 + T_b) \cdot \left(\frac{b}{8} \frac{p_b}{S} \cdot W_{max} + \frac{1}{W_{max}}\right)} & \text{si } \dfrac{S}{p_b} > \dfrac{3b}{8} W_{max}^2 \\ \dfrac{MSS}{(RTT_0 + T_b)} \dfrac{\sqrt{3S/2b}}{\sqrt{p}} & \text{si } \dfrac{S}{p_b} \leq \dfrac{3b}{8} W_{max}^2 \end{cases} \tag{5-28}$$

5.7 Estudio del número de segmentos de un flujo por ráfaga

Para poder modelar un flujo TCP sobre OBS correctamente, es necesario saber cuántas ráfagas se necesitan para transmitir cada ronda de transmisión. Cuantas más ráfagas se necesiten, más probabilidades habrá de que se pierda algún segmento en una ronda de transmisión. Por tanto, la probabilidad de que en una ronda de transmisión ocurra una pérdida de segmento es directamente proporcional al número de ráfagas necesarias para transmitir la ventana de transmisión. Los estudios de la literatura, así como el modelo periódico, realizan una simplificación y asumen que en cualquier instante de la transmisión el número medio de segmentos por ráfaga es el mismo. Así, el número de ráfagas para transmitir una ventana de transmisión solo depende del valor de la ventana y del número medio de segmentos por ráfaga. Los estudios de la literatura además han realizado una aproximación simplista para calcular este número, suponiendo que un solo flujo alimenta al ensamblador y obteniendo un valor alejado del real.

En esta sección, con el fin de obtener una mayor precisión en el modelo periódico se va a estudiar con detalle el número de ráfagas que se transmiten en cada ronda. Con este fin, se va a calcular primero dos cotas, una inferior, y una superior del número de ráfagas por ronda según el valor de la ventana. A continuación, mediante simulación, se estudia la relación entre el número de segmentos por ráfaga y el resto de parámetros en OBS,

aproximando en la medida de lo posible, cuál es la relación entre este número y el valor de la ventana. Finalmente se determina si es válido emplear una suposición de un número medio de segmentos por ráfaga independientemente del valor de la ventana, uno de los fundamentos del modelo periódico para OBS.

5.7.1 Cota inferior del número de ráfagas por ronda

La cota inferior del número de segmentos por ráfaga se obtiene asumiendo que toda la ventana se transmite de manera consecutiva, de manera que los segmentos llegan seguidos al ensamblador al ritmo permitido por el ancho de banda del enlace del cuello de botella, típicamente el enlace de acceso. En el caso de una estrategia de temporizador, el caso de mayor número de segmentos comienza cuando un segmento perteneciente al flujo en cuestión inicia el temporizador. En concreto, el temporizador se inicia cuando finalice la recepción completa del segmento. A continuación, irán llegando segmentos al ritmo marcado por la velocidad del enlace cuello de botella (se supone que es el de acceso).

Por tanto, el número de segmentos que llegan como máximo en el tiempo de formación de la ráfaga, que se denomina S_{max}, para una estrategia basada únicamente en temporizador, depende del tamaño de los segmentos con las cabeceras de red incluidas (MSS_{con_cab}), la velocidad de acceso (suponiendo que el acceso es el cuello de botella) (B_a) y el temporizador (T_b), se puede obtener tal y como muestra la ecuación (5-29). El redondeo hacia abajo se debe a que, al contabilizar únicamente segmentos completos, en la fracción de tiempo sobrante podría estar recibiéndose un segmento en el ensamblador, pero nunca daría tiempo a que se ensamblara. Por otro lado, el segmento adicional que se suma se debe a que el temporizador se pone en marcha al recibir el primer segmento completo.

$$S_{max} = \left\lfloor \frac{T_b B_a}{MSS_{con\_cab}} + 1 \right\rfloor \tag{5-29}$$

Se va a denominar ronda de transmisión al intervalo de tiempo en el que se envía una ventana de transmisión completa. Entonces, a partir de esta expresión de la ecuación (5-29), se obtiene la cota inferior del número de ráfagas que se transmiten en una ronda de transmisión de TCP, que se denomina $\bar{R}_{min}$, para un determinado tamaño de ventana de transmisión. Dicha cota se obtiene dividiendo el tamaño de la ventana en segmentos W_{seg} (es decir, el tamaño de la ventana en bytes, *W*, entre *MSS*) entre la cota superior del número medio de segmentos por ráfaga, como muestra la ecuación (5-30). Se redondea hacia arriba ya que como mínimo hacen falta varias ráfagas con el máximo número de segmentos, y una con menos segmentos para completar la ventana.

$$\bar{R}_{min}(W) = \left\lceil \frac{W}{MSS \cdot S_{max}} \right\rceil = \left\lceil \frac{W}{MSS \cdot \left\lfloor \frac{T_b B_a}{MSS} + 1 \right\rfloor} \right\rceil \tag{5-30}$$

5.7.2 Cota superior del número de ráfagas por ronda

La cota superior del número de ráfagas por ronda se obtiene cuando la ventana, en vez de transmitirse de manera consecutiva, se transmite de manera equiespaciada en el tiempo durante la duración de la ronda (RTT). Es decir, la cota superior se obtiene suponiendo que se transmite una ráfaga tras otra de manera consecutiva, independientemente del valor de la ventana, y los segmentos se reparten en ellas. Por tanto, se calcula durante un RTT cuántas ráfagas como máximo pueden transmitirse. Este número máximo de ráfagas se obtiene dividiendo el RTT entre el valor del temporizador, como muestra la ecuación (5-31).

$$\bar{R}_{max}(W) = \left\lceil \frac{RTT}{T_b} \right\rceil \tag{5-31}$$

5.7.3 Análisis mediante simulación del número de ráfagas por ronda

En los apartados anteriores se han obtenido dos cotas, una inferior y una superior, que nos indican en qué zona se mueve el número de ráfagas necesarias para transmitir una ventana, que da una indicación del número segmentos por ráfaga en cada ronda. Sin embargo, para obtener cual es el valor medio del número de ráfagas por ronda más cercano a una situación real, se procede a observar el comportamiento mediante simulación. Para ello, se plantea un modelo de simulación con varios escenarios, en los que se modifican los distintos parámetros encontrados en las cotas inferior y superior, como son el RTT, el temporizador de ensamblado. En las simulaciones se observa la evolución del número de ráfagas para transmitir una determinada ventana, y se comprobará si se puede aplicar una simplificación de emplear un número medio de segmentos por ráfaga independientemente del valor de la ventana. Para la simulación, se ha empleado un modelo, representado en la Figura 5-4 con 25 fuentes TCP conectadas a un ensamblador junto con un generador de tráfico exponencial para que se generen continuamente ráfagas. Las simulaciones se han realizado con OMNeT++, debido a la gran flexibilidad para manejar el contenido de las ráfagas y segmentos TCP.

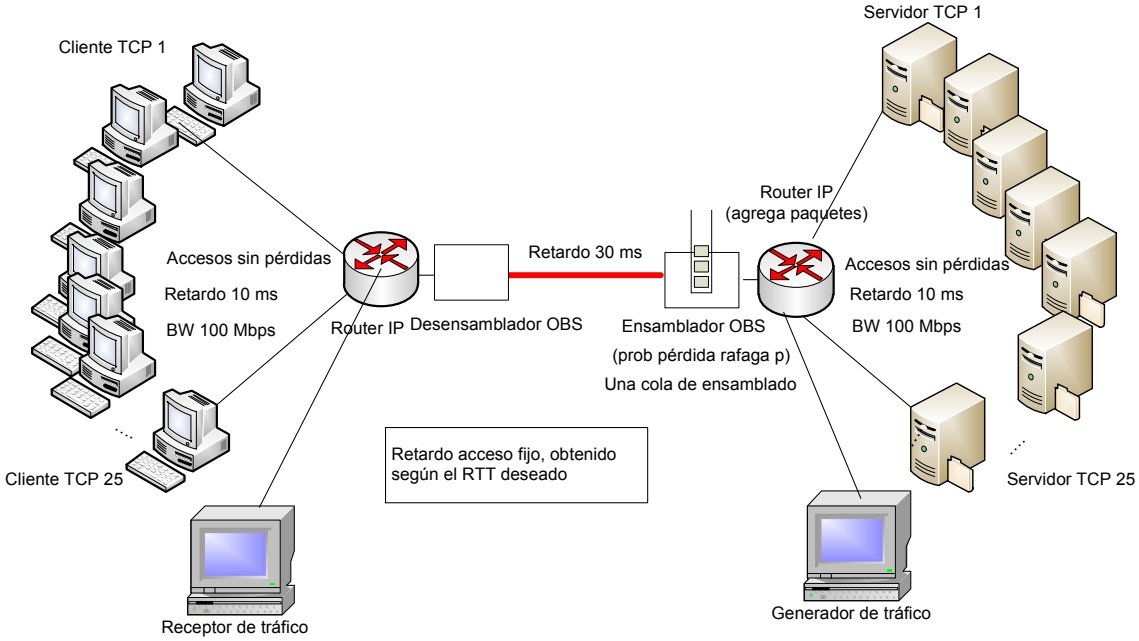


Figura 5-4 Escenario de simulación para medir número de ráfagas por ronda y ventana

En primer lugar, se realiza una simulación en la que se fija el RTT en 100 ms, mediante 30 ms en el enlace OBS y 10 ms en cada uno de los tramos de acceso, así como una probabilidad de pérdida de 0,001 y un temporizador de 1 ms. En esta simulación se realizan medidas de tráfico (bytes transmitidos) en intervalos de 100 ms (el valor del RTT) para cada una de las fuentes TCP. Dichas medidas se toman a la salida del ensamblador OBS justo al entrar en el enlace OBS. En el dibujo de la Figura 5-4 los datos se envían desde los servidores TCP, situados a la derecha de dicha figura, hacia los clientes, situados a la izquierda de la imagen. Mediante la medida de los bytes transmitidos por cada flujo en intervalos de 100 ms, al ser este intervalo de tiempo de igual valor que el RTT, se obtiene una estimación del valor de la ventana en ese intervalo. Por otro lado, en cada uno de los intervalos de medida se toma una muestra del número de ráfagas que se han empleado para transmitir cada uno de los flujos de tráfico, así como el valor del número de segmentos de cada flujo por ráfaga. Con las medidas mencionadas, se puede observar, para cada flujo, la

relación entre número de ráfagas y valor de la ventana. En la Figura 5-5 y la Figura 5-6 se representa, para un par de los flujos de la simulación, en concreto para los flujos TCP nº1 y nº3, en el eje *x* el valor de la ventana de transmisión (en segmentos) y en el eje *y*, el número de ráfagas necesarias para transmitir la ventana en cuestión. En dichas gráficas se representa mediante un círculo de color negro cada una de las muestras tomadas, es decir, para cada intervalo de medida se toma un par valor de ventana de transmisión (estimado a partir del tráfico) – número de ráfagas para transmitir dicha ventana. Además de estos puntos, en la Figura 5-5 y la Figura 5-6 se muestra en color rojo la media del número de ráfagas necesarias para cada valor de la ventana de transmisión.

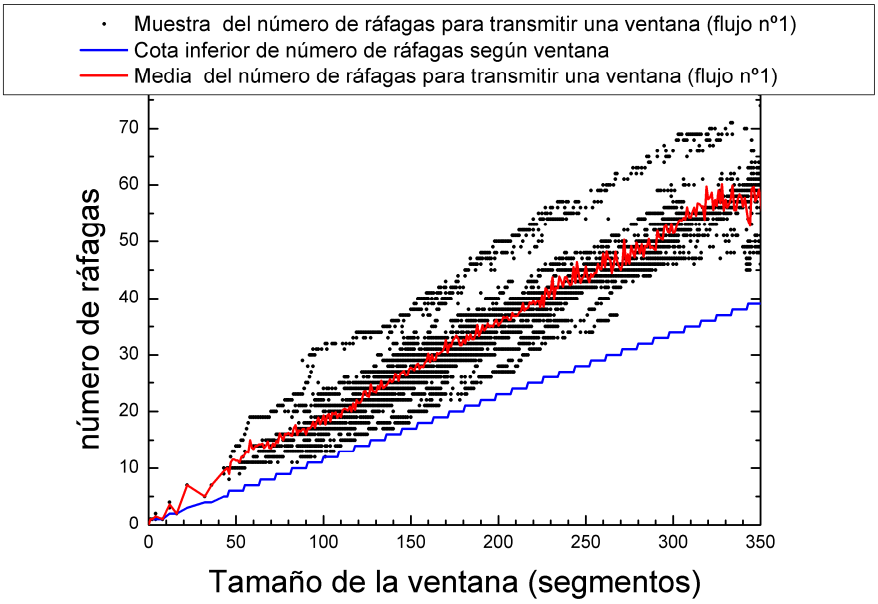


Figura 5-5 Número de ráfagas para transmitir una ventana (flujo TCP nº1)

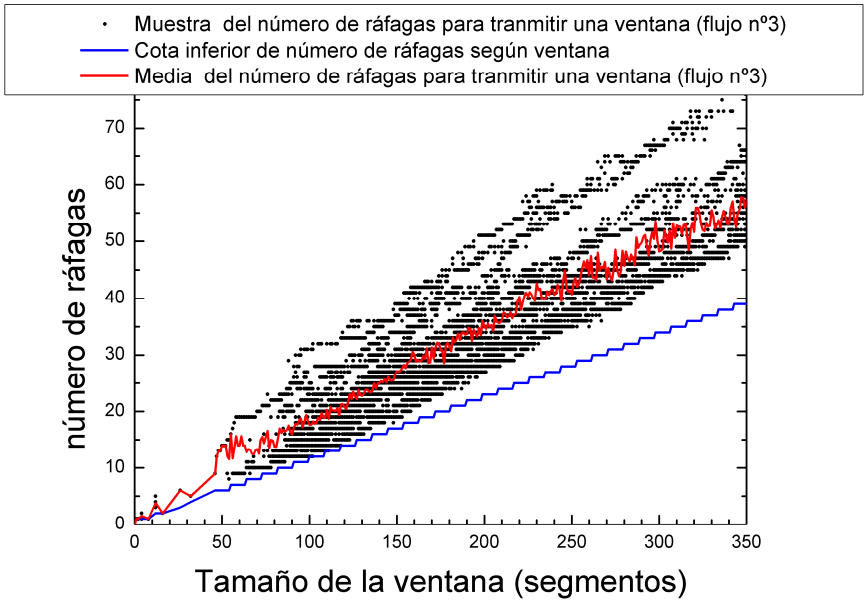


Figura 5-6 Número de ráfagas para transmitir una ventana (flujo TCP nº3)

Además, en las mencionadas figuras se incluye, en color azul la cota inferior del número de segmentos por ráfaga, obtenido mediante la ecuación (5-30) del apartado anterior. La cota superior, obtenida con la ecuación (5-31), tiene un valor fijo e independiente de la ventana de 120 ráfagas, que, en este caso, queda muy por encima de los valores obtenidos y no se muestra en las figuras.

En primer lugar se puede observar el buen funcionamiento de la cota inferior, rozada en muchas ocasiones. Además, se observa que la media del número de ráfagas según el valor de la ventana sigue una relación prácticamente lineal. Esto quiere decir que el número medio de segmentos por ráfaga es prácticamente independiente del valor de la ventana. Solamente para valores altos de la ventana de transmisión, se observa en ambas gráficas un comportamiento más irregular. Esta irregularidad se debe a que el número de muestras de ventanas altas es bajo, ya que en la simulación se llega a estos valores en pocas ocasiones. Para eliminar este error y aumentar el número de muestras, se procesan los datos de los 25 flujos. La Figura 5-7 representa los valores analizados con las medidas tomadas en todos los flujos TCP de la simulación. En este caso se observa el perfecto funcionamiento de la cota, y un comportamiento más regular para valores altos de la ventana.

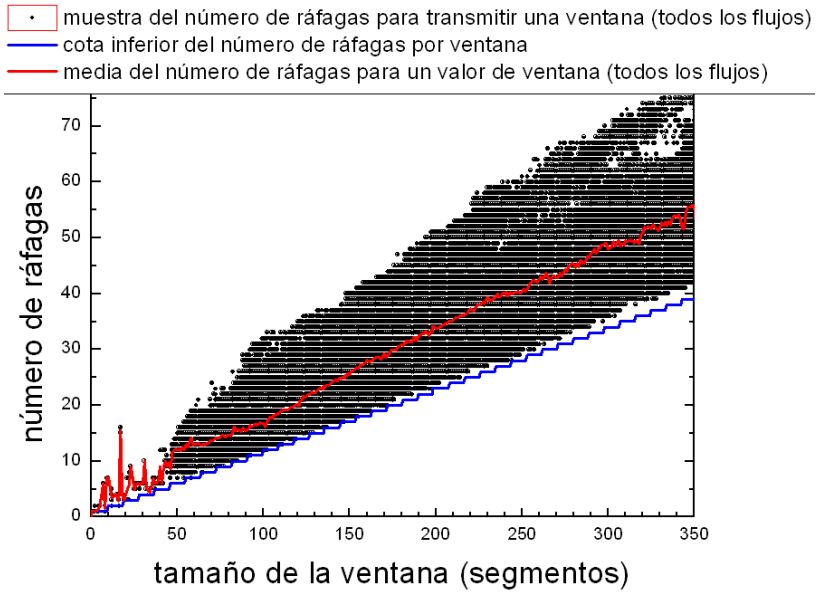


Figura 5-7 Número de ráfagas para transmitir una ventana de flujos medios (muestras de todos los flujos)

Al observar la media del número de ráfagas según el tamaño de la ventana se aprecian dos zonas, una primera, hasta 50 segmentos aproximadamente en la que la media fluctúa bastante, otra a continuación, en la que se observa un comportamiento lineal. Sin embargo, la mayor parte del tiempo, de hecho la gran mayoría de los intervalos muestreados, están en valores de la ventana por encima de 50 segmentos (que corresponde aproximadamente por encima de los 65 kbytes de la ventana TCP estándar).

A continuación, mediante la herramienta matemática Matlab [226], se busca un polinomio de primer grado que aproxime la media del número de ráfagas según la ventana. El resultado de la aproximación lineal se muestra en la ecuación (5-32) para el caso de la simulación con RTT = 100 ms, en la que R es el número medio de ráfagas por ronda estimado, y W_s es el valor de la ventana en segmentos.

$$R_{sim\_RTT\_100}(W) = 0{,}1586\, W_s + 0{,}703 \qquad (5\text{-}32)$$

En la Figura 5-8 se muestra gráficamente la aproximación lineal obtenida frente a la media de número de ráfagas necesario para transmitir una determinada ventana de transmisión. Si se desprecia el coeficiente de grado cero, que vale 0,703, a medida que la ventana crece su contribución es cada vez más pequeña y se tiene que el número medio de ráfagas por ventana es directamente proporcional al valor de la ventana. De esta forma, el número medio de segmentos por ráfaga es independiente del valor de la ventana, y la aproximación del modelo periódico es válida.

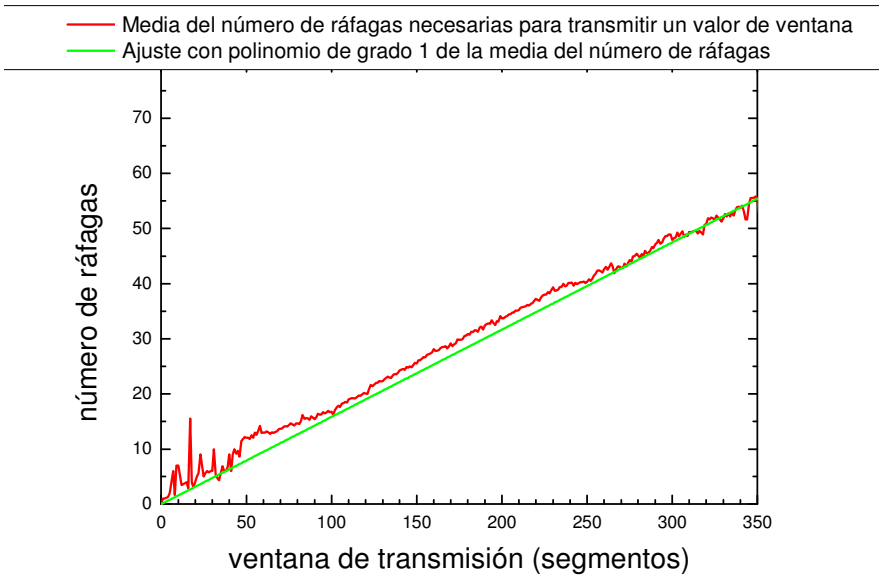


Figura 5-8 Media y ajuste lineal del número de ráfagas por ventana (RTT=100 ms)

Para complementar el modelo periódico, se propone incluir un factor correctivo al valor empleado típicamente en la literatura, en el que se obtiene el número medio de segmentos por ráfaga mediante la ecuación (5-33) [50]. Este valor es parecido, pero no igual, a la cota inferior que se ha obtenido en la ecuación (5-30) y cuyo buen comportamiento se ha visto en la simulación.

$$\bar{S}_{tipico} = \frac{T_b B_a}{MSS_{con\_cab}} \cdot W_s \tag{5-33}$$

Se define un factor, α, denominado factor de partición de ráfagas, de tal forma que la estimación del número de ráfagas por ventana $\tilde{R}$ viene dada por la ecuación (5-34).

$$\tilde{R} = \alpha \frac{W_s}{\bar{S}_{tipico}} = \alpha \frac{MSS_{con\_cab}}{T_b B_a} \cdot W_s \tag{5-34}$$

En el caso de la simulación realizada, el factor α vale 1,32, es decir, en el caso simulado, para realizar la transmisión TCP se necesita aproximadamente un 32% más de ráfagas que en el escenario más optimista en el que todas las ráfagas llevan el máximo número de segmentos TCP posibles.

Resultados variando el RTT

Los resultados anteriores se han obtenido para unas condiciones concretas, con un RTT fijado, un ancho de banda fijo y un determinado valor de temporizador. A continuación se procede a determinar si la aproximación puramente lineal, despreciando el factor de grado cero, es válida para otros valores RTT, y, en caso de ser válida, el factor α permanece constante o varía de RTT.

En primer lugar, se muestra en la Figura 5-9 el número de ráfagas por ventana (un punto por cada intervalo de medida y la media para el caso de RTT=80 ms. Dicho valor de RTT se ha obtenido reduciendo en 10 ms el retardo del enlace OBS. Se realiza el ajuste lineal y se obtiene el factor en la ecuación (5-35).

$$\bar{R}_{sim\_RTT\_80}(W) = 0{,}1384 \cdot W_s + 1{,}8947 \tag{5-35}$$

En el caso del RTT de 80 ms, el coeficiente de grado cero tiene un valor pequeño, pero apreciable, ya que añade una constante de casi dos ráfagas a cada valor de la ventana. Por tanto, si se emplea únicamente el término de grado uno, se obtiene un error de 2 ráfagas, que, para una ventana de 150 segmentos aproximadamente, representa el 10% del total. Si se calcula el factor de partición α, se obtiene un valor de 1,15. Este valor es menor que en

el caso del RTT de 100ms, pero no es directamente comparable ya que el término de grado cero tiene en este caso bastante peso.

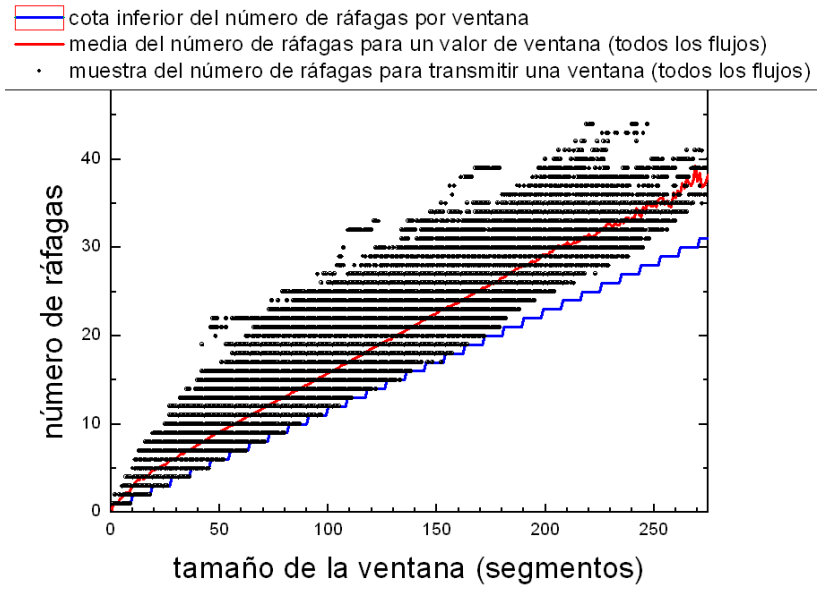


Figura 5-9 Número de ráfagas para transmitir una ventana RTT = 80 ms

En las simulaciones anteriores se ha empleado un RTT relativamente alto. Se realiza en esta ocasión una simulación con un valor de RTT de 50 ms, para lo cual se modifica tanto el retardo del enlace OBS como el retardo de acceso. El resultado, que se observa en la Figura 5-10, muestra como el límite superior se aproxima bastante bien.

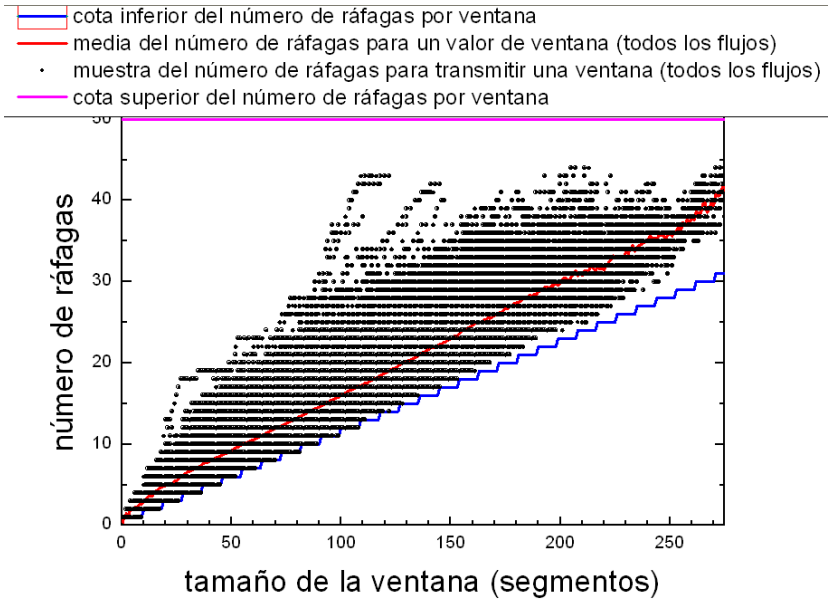


Figura 5-10 Número de ráfagas para transmitir una ventana RTT = 50 ms

En la Tabla 5-1 se muestran los coeficientes de grado cero y uno de los polinomios que se han encontrado mediante Matlab para ajustar los resultados de cada una de las simulaciones. Como se observa en la tabla, el polinomio que aproxima la función lineal varía en su factor de grado uno entre 0,138 y 0,158, mientras que el de grado cero entre 0,7 y 2,05. Para estos valores, el factor α estaría entre 1,22 y 1,32. Sin embargo, no es posible establecer una relación directa entre RTT y el factor de partición de ráfagas. En algunos casos, si se descarta el grado cero, se puede cometer un error no despreciable.

R= WX(1)+ X(0)	X(1)	X(0)
RTT=50ms	0,1521	0,6909
RTT=60 ms	0,1466	1,2098
RTT=70ms	0,1388	1,8817
RTT=80ms	0,1384	1,8947
RTT=90ms	0,1375	2,056
RTT=100ms	0,1586	0,7073

Tabla 5-1 Ajustes lineales para distintos RTTs

5.7.4 Comportamiento en fuentes rápidas

Las simulaciones anteriores se han realizado para flujos de tipo 'medio', en las que el número de segmentos por ráfagas es relativamente bajo, y para transmitir una ventana se emplea una gran cantidad de ráfagas. A continuación se simula el caso de flujos rápidos. En primer lugar se ha escogido un escenario con RTT 100 ms. En la Figura 5-11 se representa con un punto cada muestra del número de ráfagas para transmitir una ventana en un determinado intervalo, en verde la cota superior del número de ráfagas por ventana, en azul la cota inferior y en rojo la media.

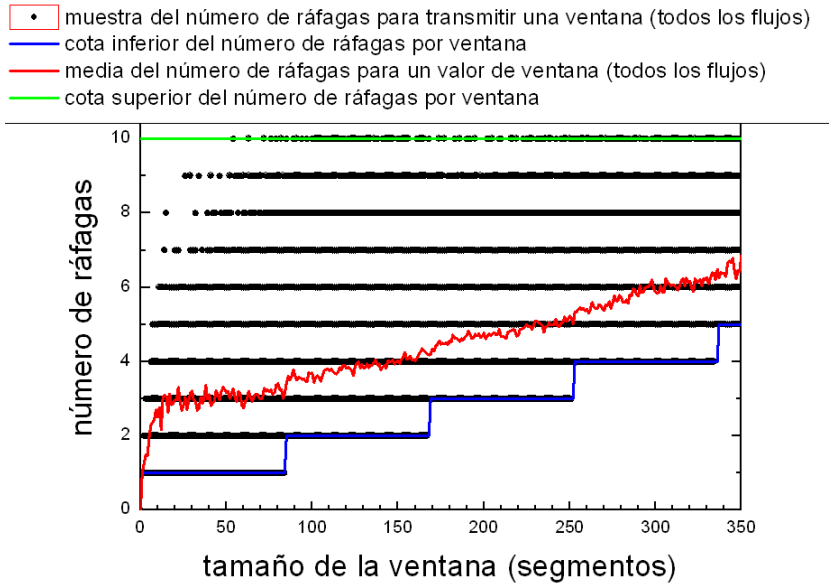


Figura 5-11 Número de ráfagas para transmitir una ventana de flujos rápidos (RTT=100 ms, Tb =10ms)

Se puede observar que para el caso de las fuentes rápidas, la cota inferior y la cota superior funcionan perfectamente, habiendo muestras que corresponden con ambas cotas. En el caso de las fuentes medias, se observó que si bien la cota inferior se cumple perfectamente, la cota superior queda muy alejada.

Al igual que para las fuentes medias, se realiza la aproximación lineal de la media del número de segmentos por ráfaga según la ventana. El resultado se muestra en la ecuación (5-35) y se representa gráficamente en la Figura 5-12.

$$R_{sim\_RTT\_100\_Tb\_100ms}(W)=0{,}0188\,W_s+0{,}7191 \qquad (5\text{-}36)$$

En esta ocasión, el ajuste lineal obtenido Matlab se aleja de la media para valores de la ventana inferiores a 150 segmentos aproximadamente. Para valores mayores, el ajuste se comporta mejor y comete menos error. Se puede comprobar como en este caso la simplificación de un número medio de segmentos por ráfaga constante e independiente del valor de la ventana no se cumple. Por tanto, cuando se emplee esta aproximación en el modelo teórico se cometerá un pequeño error, sobreestimando el número de segmentos

por ráfaga para valores pequeños de la ventana, y subestimando para valores altos. Si la ventana sigue creciendo, se puede apreciar que el número de ráfagas para transmitir una ventana se aproximaría a la cota superior, con lo que el número de segmentos por ráfaga llegaría a su valor máximo.

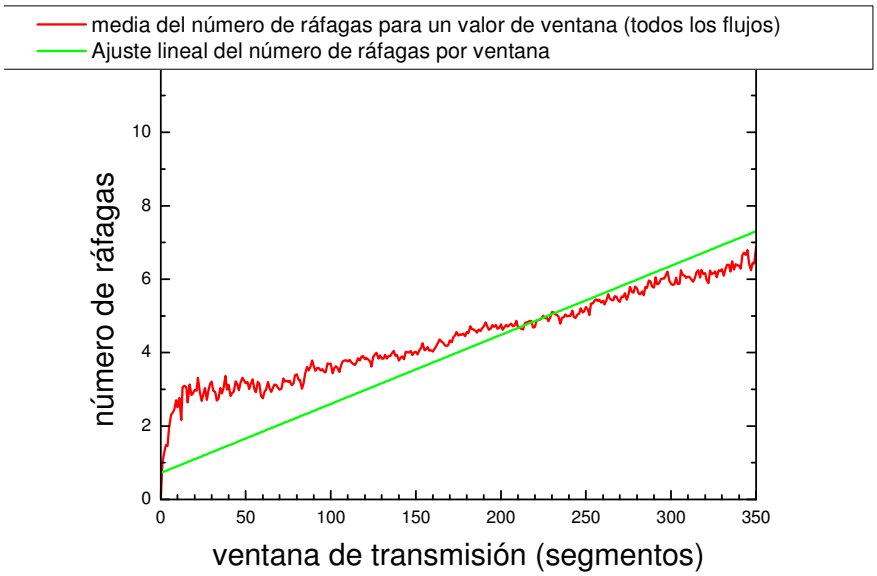


Figura 5-12 Media y ajuste lineal del número de segmentos por ráfaga (RTT=100 ms, Tb=10 ms)

Asimismo, se calcula el factor α, que vale 1,56, es decir, en el caso simulado, para realizar la transmisión TCP se necesita aproximadamente un 56% más de ráfagas que en el escenario más optimista en el que todas las ráfagas llevan el máximo número de segmentos TCP posibles. El factor es ligeramente superior al obtenido en el caso de los flujos lentos, cuyo máximo fue de 1,32. Sin embargo, como se observa en la Figura 5-12, la aproximación lineal no se ajusta 100% al caso simulado.

Se ha replicado la simulación con otros valores de temporizador. La Tabla 5-2 muestra una recopilación de los datos obtenidos. Se puede comprobar que, a medida que aumenta el valor del temporizador, el número de segmentos por ráfaga obtenido se aproxima a la mitad del máximo teórico. Sin embargo, no hay una relación directa entre el número medio de segmentos por ráfaga obtenido, el máximo teórico y el factor α, que se obtiene a partir del ajuste lineal. El principal problema del ajuste lineal es que, como se ha visto en la Tabla 5-1 y la Tabla 5-2, tiene un coeficiente de grado cero no despreciable en muchos casos. En concreto, en aquellos caso en los que el coeficiente de grado cero es alto, por ejemplo si T_b = 0,3 ms, el factor α es bajo, de 1,13 en el ejemplo, y en los que el coeficiente es despreciable, el factor α sube, por ejemplo en el caso de T_b =0,7 ms sube a 1,7.

	T_b =1 ms	T_b =0.3 ms	T_b =0.7ms	T_b = 10 ms
Número medio de segmentos por ráfaga	6,1049	16,0474	29,9051	41,4809
Número máximo de segmentos por ráfaga	9	26	59	84
Ajuste lineal	0,1586 W_s+ 0,7073	0,0455 W_s+ 2,3135	0,0306 W_s+ 0.3954	0,0188 W_s+ 0.7191
Factor α	1,32	1,1364	1,7833	1,5672

Tabla 5-2 Resumen datos simulaciones para RTT= 100 ms variando temporizador

En la Tabla 5-2 se ha calculado además el número medio de segmentos por ráfaga obtenido en el global de la simulación. Se puede comprobar la relación entre la media del número de segmentos por ráfaga y el máximo teórico depende del valor del temporizador, y es un valor más estable que factor α. Se comprueba que la aproximación lineal pura, sin

coeficiente de grado cero comete pequeños errores, que dependerán del valor de la ventana.

5.8 Modelo periódico completo

Se ha comprobado en las simulaciones que en cada ronda de transmisión el número de ráfagas necesarias para transmitir la ventana varía en gran medida durante el tiempo de vida del flujo. Sin embargo, se puede hacer una simplificación y suponer que dicho número de ráfagas es directamente proporcional al valor de la ventana. El valor del coeficiente (α) depende del escenario concreto (valor del temporizador de ensamblado, RTT, ancho de banda de acceso, etc). Si W_s es el valor de la ventana en segmentos, MSS_{con_cab} es el tamaño máximo del segmento TCP en bytes con las cabeceras de los protocolos TCP, B_a es el ancho de banda del enlace cuello de botella y T_b el valor del temporizador de ensamblado, el número medio de ráfagas por ronda ($\bar{R}(W)$) viene dado por la ecuación (5-37).

$$\bar{R}(W) = \alpha \cdot \frac{W_s}{\bar{S}_{tipico}} = \alpha \cdot \frac{W_s MSS_{con\_cab}}{T_b B_a} \tag{5-37}$$

Bajo el supuesto de que el número de ráfagas por ronda es proporcional a la ventana, se puede realizar la suposición de que el número de ráfagas para transmitir un determinado valor de la ventana se puede obtener aproximadamente dividiendo el valor de la ventana entre el número medio de segmentos por ráfaga *S*, como se muestra en (5-38).

$$\bar{R}(W) \approx \frac{W_s}{S} \tag{5-38}$$

Así, realizando la suposición de independencia de la ronda en la que se esté del número de segmentos por ráfaga, el modelo periódico que se desarrolla en el apartado 5.6 es válido y se puede sustituir en la ecuación (5-28) el valor del número de segmentos por ráfaga S (que se obtiene a partir de (5-37) y (5-38)). Por tanto, el *throughput* de TCP en OBS se puede obtener medianta la ecuación (5-39), en la que B_a es el ancho de banda del enlace cuello de botella, MSS_{con_cab} es el tamaño máximo del segmento TCP en bytes con las cabeceras de los protocolos TCP y de red añadidos, T_b el valor del temporizador de ensamblado, α el factor de partición de ráfagas, W_{max} el tamaño máximo de la ventana y p_b la probabilidad de pérdida de ráfaga.

$$\bar{X}_{OBS} = \begin{cases} \dfrac{MSS}{RTT_{OBS}\left(\dfrac{bMSS_{con\_cab}\alpha p_b}{8T_bB_a}W_{max} + \dfrac{1}{W_{max}}\right)} & \text{si } p_b < \dfrac{8T_bB_a}{3bW_{max}^2\alpha MSS_{con\_cab}} \\ \dfrac{MSS}{RTT_{OBS}}\dfrac{\sqrt{3T_bB_a/\left(2MSS_{con\_cab}\alpha b\right)}}{\sqrt{p_b}} & \text{si } p_b \geq \dfrac{8T_bB_a}{3bW_{max}^2\alpha MSS_{con\_cab}} \end{cases} \tag{5-39}$$

5.9 Validación mediante simulación

El modelo periódico desarrollado se va a validar mediante simulación en un escenario similar al del apartado 4.5. En este escenario, mostrado en la Figura 5-13, hay un servidor TCP enviando ficheros de tráfico periódicamente a un cliente TCP. Por otro lado, los generadores de tráfico alimentan los nodos OBS. Nótese que, a diferencia del apartado anterior, en este escenario hay dos ensambladores, uno en cada extremo. En las simulaciones se mide el *goodput*, es decir la cantidad de datos útiles sin tener en cuenta cabeceras, por unidad de tiempo. Aunque muchas veces se emplea indistintamente *goodput y*

throughput, el primero se emplea para indicar el tráfico a nivel de aplicación, y el segundo a nivel de red, por lo que incluye la sobrecarga de las cabeceras. Para cada simulación, se transmiten aproximadamente 100 ficheros de 200 Mbytes, y se calcula el *goodput* dividiendo 200 Mbytes entre el tiempo que se tarda en realizar la transferencia.

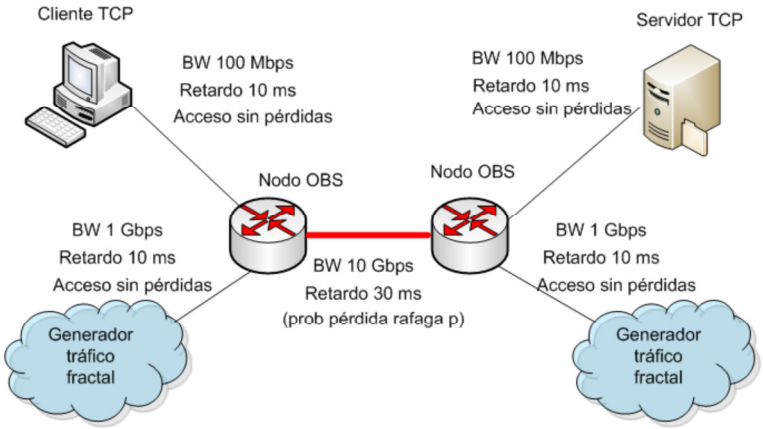


Figura 5-13 Escenario de simulación de TCP sobre OBS con trafico de fondo

Verificación del modelo para flujos lentos

En primer lugar se realiza la verificación del modelo con el caso de los flujos lentos, en los que cada ráfaga lleva un segmento de un flujo TCP y el comportamiento es similar a una red de paquetes. Por tanto, en este caso la media, mínimo y máximo del número de segmentos por ráfaga coincide y es igual a uno. La principal aportación del modelo propuesto en este caso que coincide con el de una red de paquetes es el tener en cuenta la limitación de la ventana. Por ello, se han realizado tres conjuntos de simulaciones, el primero con la ventana estándar, el segundo con una ventana de 130 kbytes y el último con una ventana máxima de 260 kbytes. Para cada una de los escenarios, se realizan varias simulaciones con distintas probabilidades de pérdida.

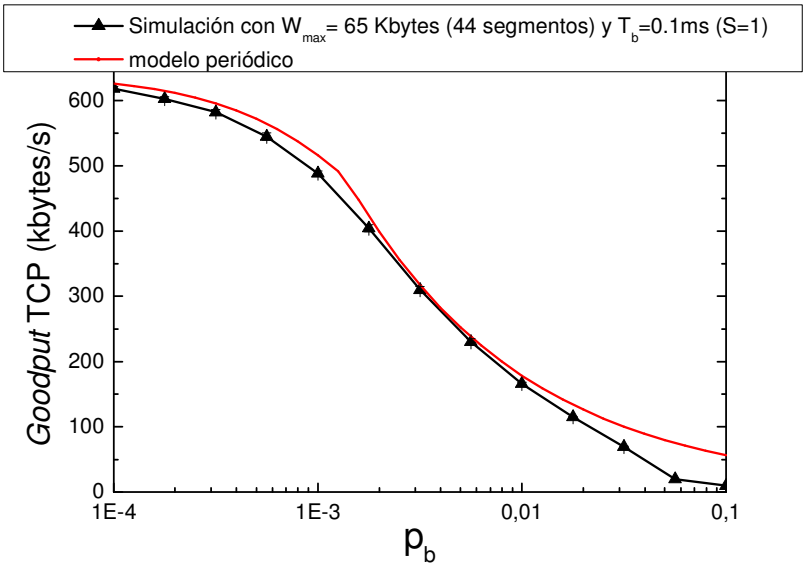


Figura 5-14 *Goodput* de TCP para flujo lento (T_b = 0,1 ms) y ventana de 65 kbytes

En la Figura 5-14 se puede ver la tasa de transferencia obtenida en la simulación en negro, y la obtenida con el modelo periódico en rojo. Se puede apreciar el buen comportamiento del modelo periódico. Solamente con probabilidades de pérdidas muy altas hay un ligero desvío debido a que el modelo periódico no contempla el vencimiento por temporizador, muy habitual en casos con múltiples pérdidas.

En la Figura 5-15 se muestran las tasas de transferencia para una ventana de 130 kbytes, y en la Figura 5-16 para una de 260 kbytes. En el caso de la ventana de 130 kbytes, el ajuste es muy bueno, comportándose de manera similar al caso de la ventana estándar.

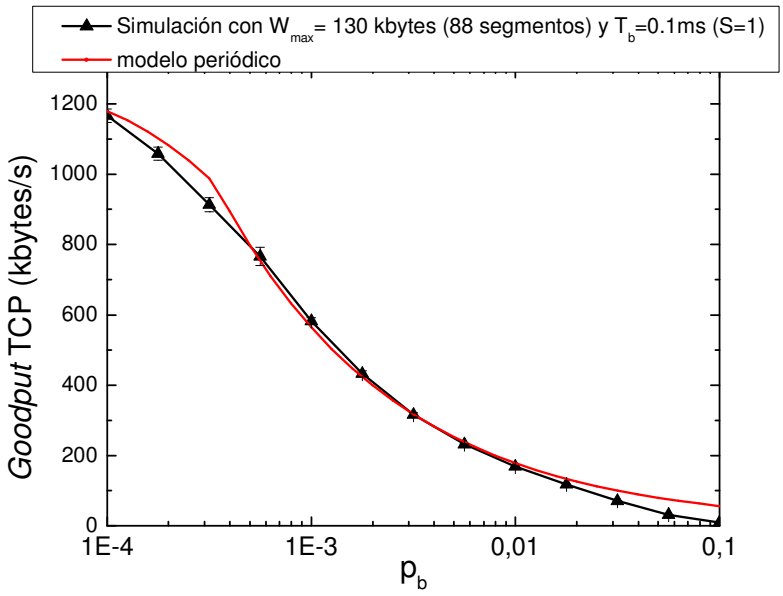


Figura 5-15 *Goodput* de TCP para flujo lento (T_b = 0,1 ms) y ventana de 130 kbytes

En el caso de la ventana grande de 260 kbytes, el modelo se ajusta casi perfectamente a la simulación. Por otro lado, cabe destacar que a medida que la simulación emplea una ventana mayor, hay una mayor varianza en los resultados, por lo que el intervalo de confianza es relativamente mayor que en los casos anteriores.

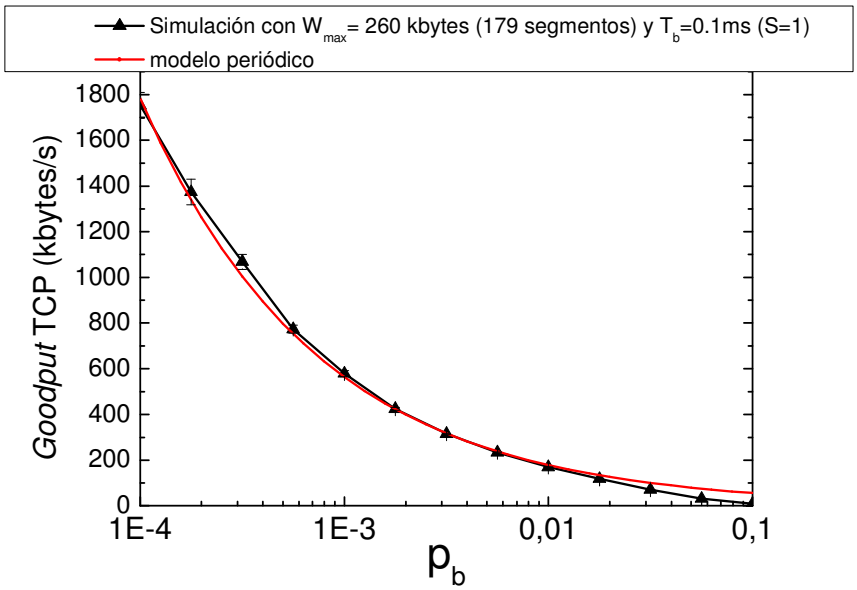


Figura 5-16 *Goodput* de TCP para flujo lento (T_b = 0,1 ms) y ventana de 260 kbytes

Las simulaciones realizadas muestran que, para flujos lentos en los que el comportamiento es como una red de paquetes, el modelo periódico tiene un excelente comportamiento tanto en los rangos de pérdidas de ráfaga que un operador de red asumiría, entre 10^{-3} y 10^{-4} como en pérdidas elevadas del 10%. Una vez comprobado el caso de los flujos lentos, se analiza el caso de los flujos medios y rápidos, en los que las ráfagas llevan más segmentos.

Validación del modelo con flujos medios

Una vez que se ha comprobado el buen funcionamiento del modelo para el caso más parecido a una red de paquetes, se continúa con el caso de los flujos medios, en concreto, se simula el temporizador de 1 ms. Para cada una de las ventanas de transmisión, se muestra el modelo periódico con el valor típico del número de segmentos por ráfaga, y también se muestra el modelo periódico con el factor de partición de ráfagas α obtenido en la simulación y otros posibles valores del factor.

Para el caso de la ventana estándar, el principal problema es que la ventana no supera los 44 segmentos. Como se comprobó en la sección anterior, en esta zona de ventanas, la partición de las ráfagas es mayor que para ventanas altas.

El modelo periódico con el valor máximo del número de segmentos por ráfaga se comporta bien a pérdidas bajas, en la que el comportamiento está dominado por el límite de la ventana, y para probabilidades de pérdida altas, mantiene una forma parecida, pero se aleja de la simulación. Esto se debe a que el número medio de segmentos por ráfaga en estas simulaciones es mucho menor que el máximo.

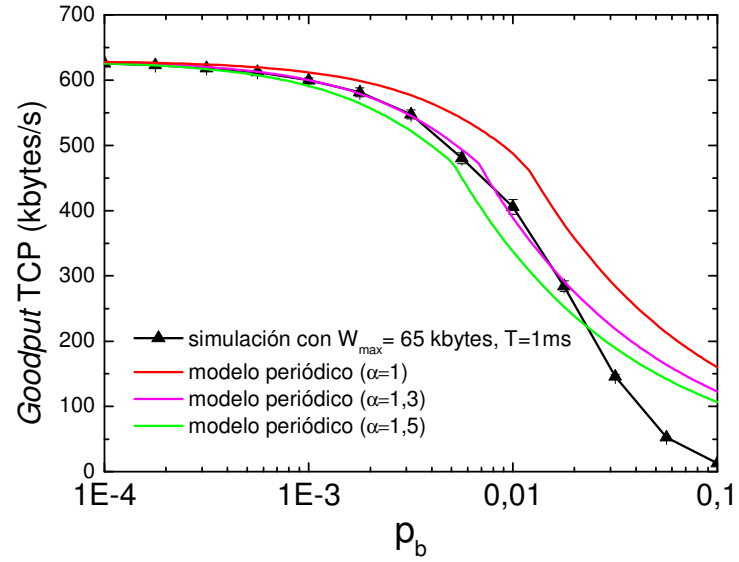


Figura 5-17 *Goodput* de TCP para flujo medio (T_b = 1ms) y ventana de 65 kbytes

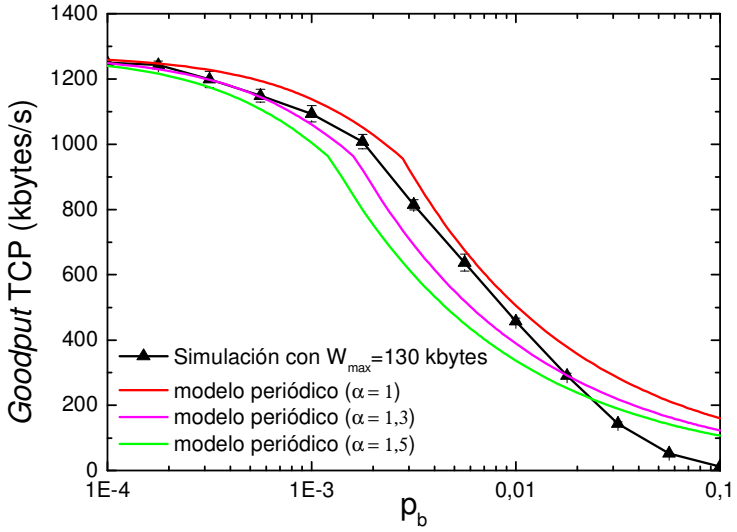


Figura 5-18 *Goodput* de TCP para flujo medio (T_b = 1ms) y ventana de 130 kbytes

A continuación, se realiza una simulación con una ventana grande, de 130 kbytes (88 segmentos como máximo en un ráfaga), cuyo resultado se muestra en la Figura 5-18. En este caso, se observa que el factor de partición está entre 1 y 1,3. Esto quiere decir que, para este valor de ventana máximo, la transmisión de la ventana tiende a concentrarse y apenas hay partición de ráfagas. Aun así, la aproximación con el valor de 1,3 obtenido en la simulación tiene un error menor del 10%.

Finalmente, se vuelve a doblar la ventana y se emplea una ventana de transmisión de 260 kbytes (176 segmentos). En este caso, en la simulación, cuyo resultado se muestra en la Figura 5-19, el comportamiento más cercano al resultado de la simulación se ha obtenido con el modelo periódico sin ningún tipo de corrección. Este hecho puede deberse a que en la simulación, en la que hay dos ensambladores, cuando la ventana es muy alta, se esté produciendo una concentración de los segmentos en las ráfagas, por lo que el factor de partición es menor al que se obtuvo en la simulación con un único ensamblador.

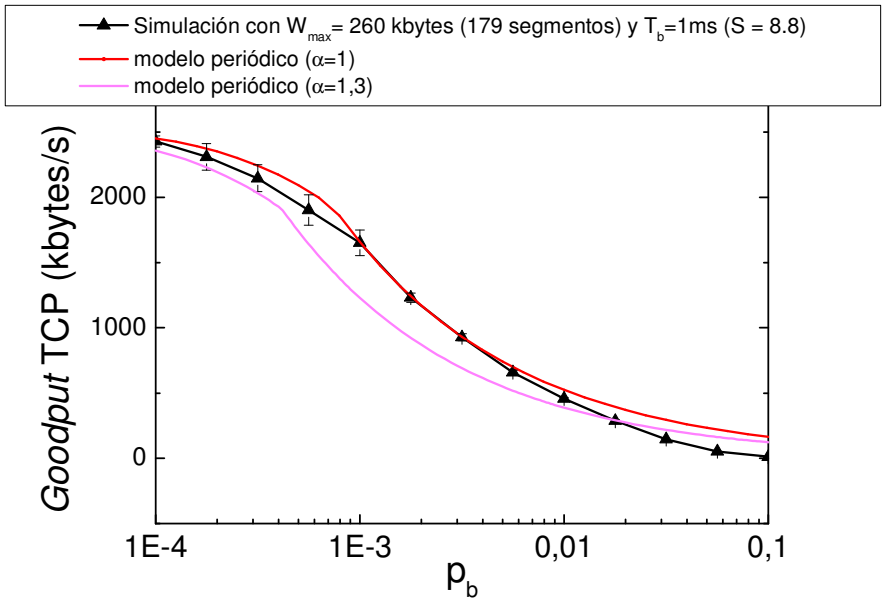


Figura 5-19 *Goodput* de TCP para flujo medio (T_b = 1 ms) y ventana de 260 kbytes

Comportamiento de fuentes rápidas

Por último, se comprueba el comportamiento del modelo periódico en el caso de las fuentes rápidas, donde el número de segmentos por ráfaga es mucho mayor. A priori, en este caso el modelo periódico cuenta con el hándicap de no tener en cuenta las pérdidas resueltas mediante temporizador de retransmisión, que se producen cuando se pierde un número elevado de segmentos. Por otro lado, se ha comprobado que la ventana, aun en el caso de una fuente rápida, se divide en bastantes ráfagas, con lo que la recuperación se produce en la mayoría de los casos simplemente reduciendo la ventana a la mitad, caso contemplado por el modelo periódico.

Por tanto, el mayor inconveniente es la correcta estimación del número medio de segmentos por ráfaga, y en segundo lugar que la premisa de que el número medio de segmentos por ráfaga sea independiente del valor de la ventana.

El primer caso analizado en las fuentes rápidas es el de la ventana máxima estándar. En la Figura 5-19 se aprecia que el *goodput* obtenido con el modelo periódico está sobreestimado en gran medida cuando se aplica un factor de partición de ráfagas de valor 1. Sin embargo, si se emplea el factor de partición de 2, que se obtuvo en el apartado anterior, se obtiene un mejor resultado, pero aún alejado de la tasa de transferencia obtenida en la simulación. Estos factores están obtenidos empleando un valor de ventana máxima muy por encima del valor de la ventana estándar de TCP. En las simulaciones realizadas en el

apartado anterior se comprobó que el número de ráfagas para transmitir el valor de la ventana (relacionado directamente con el número de segmentos por ráfaga) seguía un patrón lineal para valores de la ventana de transmisión mayores que la ventana estándar (cuyo valor es aproximadamente 44 segmentos). Por debajo de ese valor el comportamiento era diferente y la ventana de TCP se fragmentaba en más ráfagas. Hay que recordar que en el Capítulo 4 de la tesis se estudió el número medio de segmentos por ráfaga empleando la ventana de TCP estándar. En dichas simulaciones se obtuvo que el número medio de segmentos por ráfaga era una cuarta parte del valor máximo del número de segmentos en una ráfaga. Esto quiere decir que hay un factor de partición 4. Así, empleando el factor de partición obtenido a partir de los resultados de las simulaciones del Capítulo 4, el resultado de la tasa de tranferencia con el modelo periódico, como se aprecia en la Figura 5-19, se asemeja en gran medida al resultado de simulación desde la probabilidad de pérdida más pequeña simulada (10^{-4}) hasta una probabilidad de pérdida alta, pero poco realista, de 10^{-2}. De hecho, en este rango, el error obtenido en la tasa media es mínimo, menor del 1%.

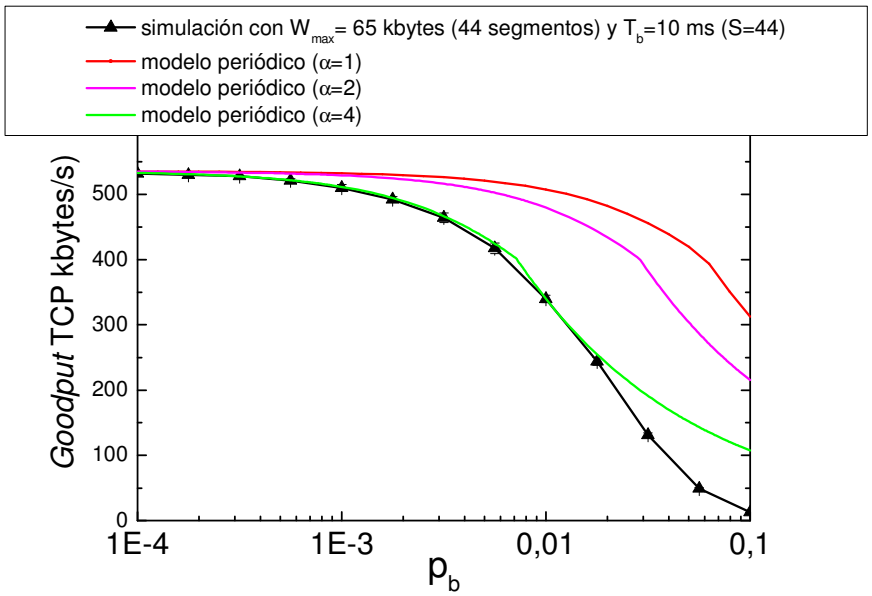


Figura 5-20 *Goodput* de TCP para flujo rápido (T_b = 10 ms) y ventana de 65 kbytes

A continuación, se examina el caso de una ventana grande, en concreto de 130 kbytes, el doble que el estándar. El resultado se muestra en la Figura 5-21, donde se compara el resultado experimental con el modelo teórico empleando dos factores de partición, 1 y 2. En este caso, al llegarse a ventanas mayores, empleando el factor de partición 2 se consigue un resultado que se acerca más a la tasa de transferencia obtenida mediante simulación. Aun así, no se consigue el resultado exacto, principalmente por la sobreestimación del número de segmentos por ráfaga. Sin embargo, con $\alpha = 2$, el error cometido con p igual a 10^{-3} es menor del 3%. Incluso, con una probabilidad de pérdida alta como es 10^{-2} el error es menor del 20%.

Finalmente, se examina el caso de una ventana muy grande, en concreto de 2600 kbytes, cuatro veces la ventana estándar. En este caso, se muestran en la Figura 5-22 el resultado del modelo periódico con 3 valores diferentes de α. Se observa que para probabilidades de bloqueo por debajo del milisegundo, un factor de α de 1,5 consigue replicar exactamente el comportamiento de la simulación. Para valores altos, se sobreestima el rendimiento de la conexión en todos los casos. El modelo teórico no tiene en cuenta los casos de pérdidas de ráfaga consecutivas que se producen cuando las pérdidas son muy elevadas.

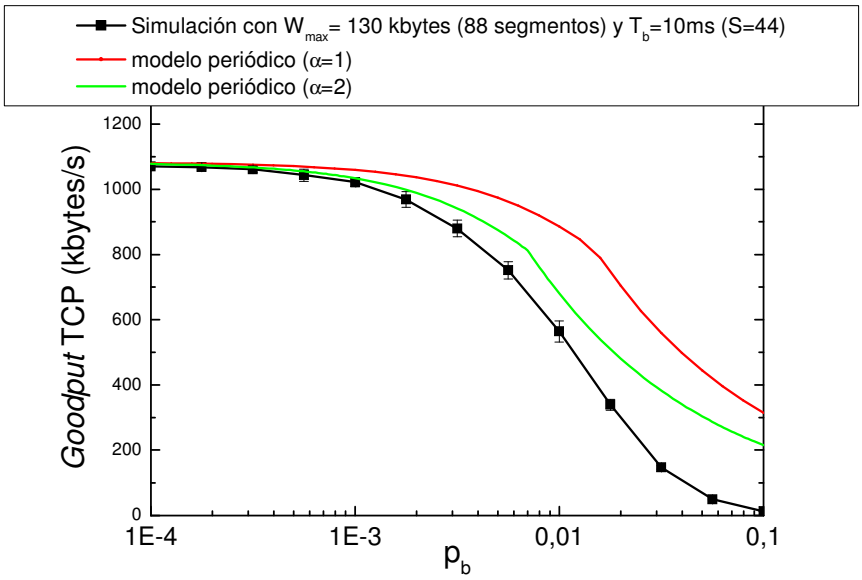


Figura 5-21 *Goodput* de TCP para flujo rápido (T_b = 10 ms) y ventana de 130 kbytes

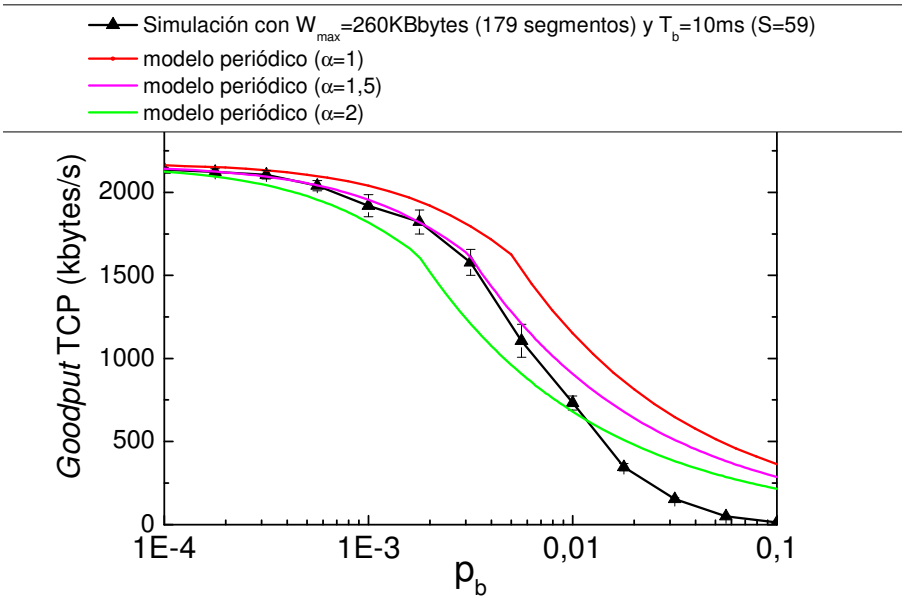


Figura 5-22 *Goodput* de TCP para flujo rápido (T_b = 10 ms) y ventana de 260 kbytes

Por tanto, se puede apreciar como en el caso de las fuentes rápidas, cuanto mayor es la ventana empleada, menor factor de partición hay que emplear. Es decir, a medida que la ventana es mayor, los segmentos tienden a reagruparse en mayor medida.

Los resultados de la simulación en un escenario realista muestran que, tanto para flujos medios como rápidos, a medida que la ventana máxima aumenta, el número de segmentos por ráfaga aumenta. Es decir, cuando la ventana máxima es pequeña se produce una mayor fragmentación de las ráfagas. En el caso de emplear la ventana TCP estándar, se han obtenido buenos resultados empleando el modelo periódico con un factor de partición de 1,3 para flujos medios, y de 4 para flujos rápidos. A medida que ha aumentado la ventana, los resultados han sido buenos con el modelo periódico sin factor de corrección para los flujos medios, y para los flujos rápidos, un factor de 2 doblando la ventana, y bajando a 1,5 volviendo a doblar la ventana.

Para todos los casos el modelo periódico ha obtenido un resultado exacto para una probabilidad de pérdida de 10^{-4}. Esto demuestra que el modelo periódico puede emplearse para conocer el rendimiento de OBS en condiciones normales. Para probabilidades de

pérdida mayores, puede emplearse el modelo empleando el factor de partición adecuado para obtener mejores resultados. Encontrar un factor de partición exacto es complicado, más aun teniendo en cuenta que el número de segmentos por ráfaga se ha demostrado no ser exactamente constante. Sin embargo, la aproximación en la que se emplea un valor constante, corregido en un factor de partición α con respecto al máximo teórico, obtiene buenos resultados.

5.10 Conclusión

En este capítulo se ha presentado un modelo periódico de TCP sobre OBS con el que obtener el rendimiento aproximado de un flujo que atraviesa una red OBS a partir de una serie de parámetros de la red, como son el ancho de banda del enlace de cuello de botella, el temporizador de ensamblado, el *Round Trip Time* (RTT) y el tamaño máximo de segmento (MSS).

Un parámetro crítico del modelo es el número de segmentos TCP por ráfaga, que determina cuántas ráfagas son necesarias para transmitir todos los datos de la ventana de TCP. Se ha realizado un análisis de dicho número de segmentos por ráfaga empleando el simulador OMNeT++, que ofrece la flexibilidad necesaria para poder monitorizar en detalle el contenido de las ráfagas. Mediante simulación se ha comprobado que, asumiendo un error relativamente pequeño para las probabilidades de pérdida de interés y valores de ventana de tranmisión de interés, se puede aplicar la suposición de que el número medio de segmentos por ráfaga sea constante. Para ello hay que aplicar un factor de corrección, que se ha denominado factor partición de ráfagas α al valor del número de segmentos por ráfaga. En los casos específicos simulados, el factor calculado oscila entre 1.3 y 1.5 para flujos medios y entre 1.5 y 4 en el caso de flujos rápidos. Dicho factor de partición se aplica directamente a la formulación obtenida en el modelo periódico.

Aunque el modelo parte del supuesto de conexiones TCP de duración infinita y un comportamiento exactamente periódico, se ha evaluado su bondad en un escenario pragmático, en el que las transferencias de fichero tienen un tamaño finito. Para realizar esta evaluación mediante simulación se ha empleado en esta ocasión OPNET Modeler, con los modelos de simulación desarrollados en el capítulo anterior, en los que se realizaban transferencias de ficheros. Como no se puede obtener el factor de partición α a priori de manera exacta, se han obtenido los resultados analíticos del modelo con los valores del factor obtenidos anteriormente mediante simulación (con un simulador diferente y un escenario de transferencias infinitas, pero mismos parámetros de TCP y OBS). El modelo, en este escenario pragmático, con los valores del factor obtenidos anteriormente, ha mostrado un buen comportamiento, especialmente en las zonas de probabilidad de pérdida de interés, entre 10^{-4} y 10^{-3}, que coincide con el rango en el que la suposición del número de segmentos TCP por ráfaga constante era válida. Únicamente en las zonas de probabilidades de pérdida baja, el modelo sobreestima el rendimiento, ya que asume una recuperación sencilla, y no mediante temporizador.

Capítulo 6

Sincronización de flujos TCP en OBS

En este capítulo se estudia el efecto de la sincronización de flujos TCP en una red OBS. La sincronización de flujos TCP aparece cuando una ráfaga con segmentos de distintos flujos TCP se pierde, haciendo que todos esos flujos TCP reduzcan su ventana simultáneamente. Este hecho lleva a un uso ineficiente del ancho de banda disponible, por lo que si la sincronización es excesiva, el ancho de banda necesario para ofrecer el mismo rendimiento a las conexiones TCP es mayor que en un caso sin sincronización. Para cuantificar el efecto de sincronización en los mecanismos clásicos de ensamblado se van a aplicar distintas métricas. Por otro lado, se propone un nuevo esquema de ensamblado que de manera estática o dinámica asigna flujos a un conjunto de colas para controlar el proceso de agregación de flujos. Se evalúa la bondad de este esquema, mostrando una mejora significativa en la utilización del ancho de banda del enlace óptico.

6.1 Efecto de sincronización TCP

Sincronización TCP

La sincronización de TCP es un efecto muy conocido en las redes de paquetes [227]. El efecto aparece cuando varias conexiones TCP comparten un enlace y los mecanismos de control de congestión de los clientes y servidores TCP involucrados en la transmisión reaccionan al mismo tiempo. Para comprender este fenómeno hay que recordar las explicaciones de los fundamentos de TCP de la sección 3.2. Una conexión TCP sigue el principio AIMD (*Additive Increase, Multiplicative Decrease*): incremento aditivo, disminución multiplicativa. En una situación ideal, cada conexión TCP disminuye su ventana en instantes de tiempo diferentes, con lo que se logra un uso eficiente del ancho de banda del enlace. Para ilustrar este efecto, se ha realizado una simulación en OMNeT++ de 10 fuentes TCP conectadas a un *router* IP en el que se descartan paquetes de manera aleatoria, y el *router* IP tiene un único enlace de salida que comparten todos los flujos. Se ha medido en dicho enlace compartido el ancho de banda ocupado por cada uno de los flujos TCP (midiendo el tráfico en intervalos de 500 ms). El resultado se muestra en la Figura 6-1, en la que aparece en un color diferente el ancho de banda ocupado por cada fuente, y el ancho de banda agregado es la suma de todos los anchos de banda individuales. Sin embargo, si los instantes en los que se producen las pérdidas coinciden, el tráfico se parecerá a un diente de sierra, y se desperdiciará una gran cantidad de ancho de banda, llevando a un uso ineficiente del enlace, como se puede ver en la Figura 6-2. En esta figura se ha modificado a propósito el *router* IP del escenario anterior de modo que cada vez que se produce un evento de fallo, se descarta un paquete de cada uno de los 10 flujos, pero manteniendo la misma tasa de pérdida de paquetes global. El efecto por el cual múltiples conexiones incrementan y disminuyen su ventana de transmisión de manera simultánea, lo cual se aprecia en la Figura 6-2, se llama sincronización de TCP.

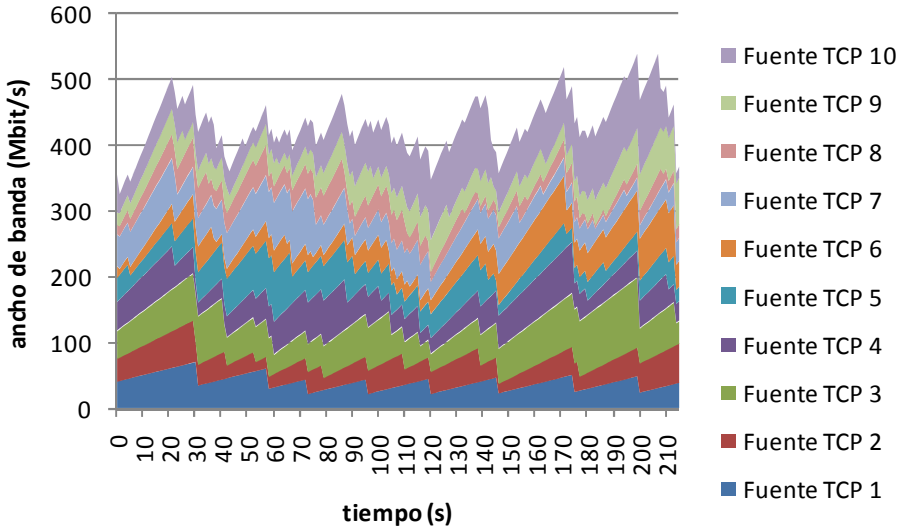


Figura 6-1 Consumo del ancho de banda por distintas fuentes TCP (no sincronizadas)

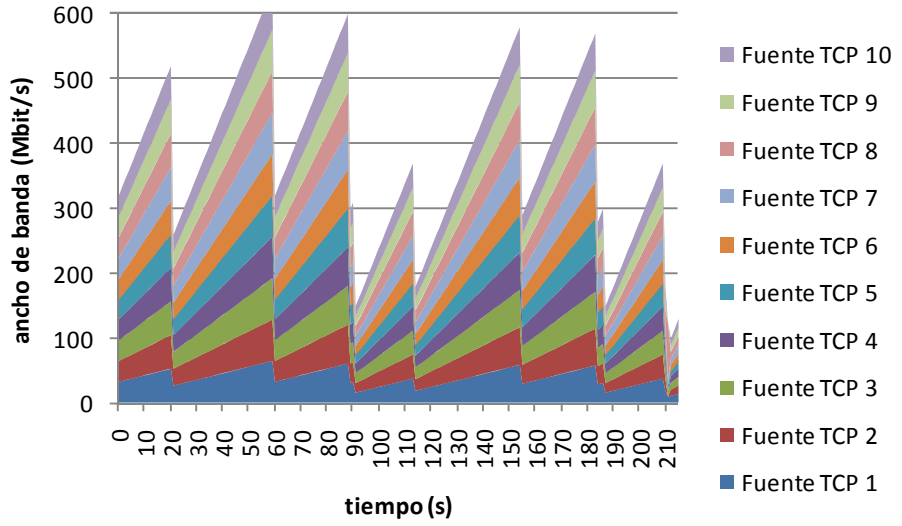


Figura 6-2 Consumo del ancho de banda por distintas fuentes TCP (sincronizadas)

Sincronización TCP en Internet

Hoy en día el fenómeno de la sincronización está presente en Internet, y en todas las redes de paquetes alimentadas con tráfico TCP. La principal causa de este fenómeno de sincronización TCP es el desbordamiento de los *buffers* de los *routers* de paquetes [228]. Los *routers* IP/MPLS están provistos de memorias (*buffers*) para poder almacenar, de manera temporal, paquetes cuando el enlace de salida está ocupado. De esta forma un *router* puede hacer frente a picos puntuales de tráfico, siempre y cuando estos sean de corta duración. La clave para la sincronización está en cómo se gestionan las colas (los *buffers* son un conjunto de colas) de los *routers*. Existen varias alternativas en cuanto a estrategias de gestión de colas. El esquema más popular, y que está desplegado en mayor número de equipos, es el *DropTail* [229]. Con este esquema, cuando se llenan los *buffers*, todo el tráfico entrante se descarta. Por tanto, hay un riesgo muy alto de descartar paquetes de múltiples flujos de manera consecutiva, sincronizando sus mecanismos de control de flujo y congestión. Hay otros esquemas de gestión de cola más inteligentes, como los de tipo AQM (*Active Queue Monitoring*) [230], en los que se toman decisiones antes de que el *buffer* se llene, ayudando así a prevenir la sincronización. Por ejemplo, un tipo de esquema de gestión de cola AQM es el esquema RED (*Random Early Detection*) [231], que empieza a descartar paquetes al azar antes de que el *buffer* se llene. Este esquema tiene la gran ventaja de que, al descartar paquetes aleatoriamente, regula las tasas de transmisión de los flujos TCP, y ayuda a evitar

que se produzcan situaciones de congestión y que llenen los *buffers*, reduciendo así las pérdidas consecutivas de múltiples flujos, demostrando además que pérdidas periódicas no sólo no son malas, sino que benefician a la estabilidad de la red [232]. Sin embargo, por un lado, el uso de RED no es generalizado, y por otro, no garantiza que no se produzcan congestiones, por lo que hoy en día sigue apareciendo la sincronización TCP en Internet.

Sincronización TCP en redes OBS

Las redes OBS son diferentes a las redes de paquetes IP interconectadas por *routers* con *buffers*. En OBS, típicamente no hay *buffers* (salvo que se empleen FDLs), y los paquetes se agregan en el borde de la red para luego viajar en ráfagas, como se ha explicado en la sección 2.5. Por tanto, cada ráfaga contiene segmentos de varios flujos TCP. El problema ocurre cuando, al no haber *buffers*, hay que descartar una ráfaga. En ese caso se perderían paquetes de múltiples flujos, que sincronizarían sus sistemas de control de flujo y congestión. Cuanto mayor sea el número de flujos diferentes en una ráfaga, mayor será la sincronización.

Impacto de la sincronización

La sincronización de las fuentes TCP, como se observa en el ejemplo de la Figura 6-2, hace que el ancho de banda no se emplee de una manera eficiente. Con el mismo número de transmisiones activas, durante ciertos periodos de tiempo el uso del ancho de banda es bajo, mientras que en otros es muy alto. Este hecho afectará a la capacidad, en términos de ancho de banda, que se tiene que reservar en un enlace para estos flujos. La capacidad de un enlace que transporta flujos agregados ha de ser igual a la media del tráfico agregado más un cierto margen para acomodar los picos de tráfico [233]. Este margen de seguridad depende de la varianza del tráfico, de tal forma que a mayor varianza, mayor margen de seguridad, y por lo tanto, mayor capacidad hay que reservar [234]. Dando la vuelta al razonamiento, se tiene que, para un mismo ancho de banda (por ejemplo, la capacidad de un enlace OBS), el número de flujos que se pueden transportar es menor cuanto mayor sea la varianza. La sincronización provoca que la varianza sea mayor. Por tanto, es importante controlar la sincronización para que la red admita un número mayor de usuarios.

Medida de la sincronización

Una vez que se ha visto en qué consiste el efecto y cómo impacta, se necesitan métricas que ayuden a medir el nivel de sincronización y saber cómo es de peligrosa. La sincronización se puede medir observando la evolución de la ventana de congestión de los distintos flujos TCP. Los puntos clave en dicha evolución son los instantes en los que se producen las pérdidas. A partir de la evolución de la ventana se propone un índice de sincronización. Para ello, se mide la ventana de cada flujo TCP en intervalos fijos y, para cada intervalo *i* en el que al menos un flujo haya reducido su ventana, se cuenta el número de flujos que han reducido su ventana con respecto al intervalo anterior (N_{di}), y se resta uno, pues si un solo flujo ha reducido su ventana no hay sincronización. Después se suman estos valores en todos los intervalos considerados ($\sum_i(N_{di}-1)$) y se divide por el resultado de sumar, para cada intervalo en el que al menos un flujo ha reducido su ventana, el número total de flujos que han estado activos restando uno ($\sum_i(N_{fi}-1)N_{sdi}$, donde N_{fi} es el número de flujos activos en el intervalo *i* y N_{sdi} es una variable binaria que vale 1 si al menos un flujo ha reducido su ventana con respecto al intervalo *i*-1 y vale 0 en caso contrario). Por tanto, se define el índice de sincronización *I* como:

$$I = \frac{\sum_i(N_{di}-1)}{\sum_i(N_{fi}-1)N_{sdi}} \tag{6-1}$$

La ecuación (6-1) es válida siempre y cuando $N_{fi} \geq 2$. Si en cada intervalo en el que sucede una pérdida de ráfaga únicamente hay un flujo que reduce su ventana, el índice de

sincronización vale 0. En cambio, si en esos intervalos todos los flujos reducen su ventana, el índice de sincronización vale 1. En la sección 4.6 se realizaron medidas del número de paquetes de un flujo TCP por ráfaga, con el objetivo de estudiar el tipo de reacción de TCP ante la pérdida de ráfaga. En cuanto a la sincronización de distintos flujos TCP, hay que acercarse al estudio de la composición de las ráfagas desde otro ángulo. En esta ocasión se va a examinar, cuando se alimenta un ensamblador con un conjunto de flujos, cómo se distribuyen estos en las ráfagas. El objetivo que se persigue es determinar si las ráfagas contienen segmentos de todos los flujos o concentran segmentos de unos pocos flujos. Por tanto, la distribución del número de flujos diferentes por ráfaga será un factor clave.

Para medir el impacto de la sincronización hay que fijarse en el tráfico agregado y su comportamiento. En primer lugar se consideran dos métricas con respecto al tráfico, la media, que refleja cuál es tasa de transferencia agregada de las conexiones TCP, y la varianza, que muestra cómo varía la ocupación del ancho de banda. Una forma de observar este comportamiento agregado es a partir de las ventanas de congestión [235]. Tal y como se ha visto en capítulos anteriores, a partir de la ventana de congestión se puede averiguar la tasa de transmisión de una fuente TCP. Por tanto, observando la ventana de todos los flujos TCP, es posible obtener su tasa de transmisión agregada. Sin embargo, este método sólo es eficaz si todos los flujos tienen el mismo RTT, ya que la tasa de transmisión es igual al tamaño de la ventana dividido entre el RTT. La otra forma de obtener el tráfico es medir los bytes que circulan por un enlace en un intervalo de tiempo determinado. Este es el mejor método si los flujos tienen distintos retardos de extremo a extremo. Esta medida es muy dependiente del intervalo de tiempo que se emplee a la hora de realizar la medida y condicionará el tipo de resultado que se puede obtener. Por ejemplo, un intervalo de medida largo, como pueden ser varios minutos, no captura las variaciones puntuales del tráfico. Para poder observar las fluctuaciones del tráfico hay que emplear intervalos de medida pequeños, por ejemplo debajo del segundo. Por tanto, cuanto mayor sea el intervalo en el que se miden los datos, menores serán las fluctuaciones del tráfico. A partir de las medidas de tráfico, se puede inferir cuál sería la capacidad mínima necesaria para poder transportar los flujos obteniendo el mismo rendimiento. Es decir, si se ofreciera una capacidad menor, los flujos experimentaría retardos adicionales y su tasa de transmisión descendería. Para calcular cuál es esta capacidad mínima que hay que provisionar, se calcula la distribución del tráfico agregado (a partir de las muestras de tráfico en los intervalos de medida), y se obtiene el valor para el cual con una probabilidad 1- ε el tráfico agregado está por debajo de esa cifra, típicamente el 95% [234].

6.2 Estrategias de ensamblado consideradas

En el estudio se emplea una estrategia de ensamblado de ráfaga basada en temporizador, sin límite en el tamaño máximo de ráfaga. En el nodo OBS se emplean de una a varias colas por par origen-destino dentro de la red OBS. Según el número de colas, se definen tres estrategias:

PF (*Per Flow*): Hay una cola para cada flujo. Es una estrategia con interés teórico, y será la estrategia con menor sincronización. A pesar de ser una estrategia más teórica, es ampliamente usada en la literatura en los estudios de TCP sobre OBS [50] [52] [236].

MF (*Mixed Flow*): Hay una cola para todos los flujos a un mismo par origen-destino y clase de servicio. Es la estrategia que, a priori, más sincronización va a tener, y la estrategia que se suele asumir en los estudios de OBS relacionados con el ensamblado de flujos de tráfico [64] [65] [139].

MQ (*Multi Queue*): En este caso, se propone el empleo de un esquema intermedio en el que hay varias colas que se comparten entre todos los flujos de un mismo par origen-

destino (origen y destino en la red OBS) y clase de servicio. De esta forma, cada cola agrega varios flujos. Hay varias posibilidades para repartir los paquetes con datos de los flujos de tráfico a las colas. Una primera alternativa sería un método puramente aleatorio, en el que se tome la decisión de qué ensamblador utilizar paquete a paquete. Sin embargo, una técnica de estas características desordena los datos en exceso, que TCP puede interpretar como congestión, por lo que no es aconsejable para tráfico TCP. Otra alternativa que se propone consiste en asignar a cada cola un conjunto de flujos determinados, que se pueden distinguir por alguna característica (por ejemplo protocolo, dirección IP y puerto origen, dirección IP y puerto destino). Una forma de realizar esta asignación estática es asignar unos rangos de direcciones a cada ensamblador. Esta es la estrategia que se ha implementado y cuyo rendimiento de cara a reducir la sincronización ha sido evaluado.

6.3 Escenarios de simulación

Con el objetivo de analizar en detalle el comportamiento de TCP cuando se agregan múltiples fuentes, se han implementado en OMNeT++ 4.1 [13] varios escenarios de simulación. Estos escenarios simulados van a ayudar a entender y caracterizar los efectos de la sincronización TCP.

En el primer escenario, mostrado en la Figura 6-3, se implementa la estrategia PF y consiste en un conjunto de clientes TCP cada uno de los cuales establece una conexión TCP con uno de los servidores TCP situados al extremo opuesto de la red OBS. Los clientes TCP reciben datos de los servidores, que realizan una transferencia FTP de un archivo de gran tamaño, suficiente para que durante toda la simulación siempre haya datos listos para transmitirse. Por tanto, en este escenario los segmentos de datos se envían de servidor a cliente, y los asentimientos de cliente a servidor.

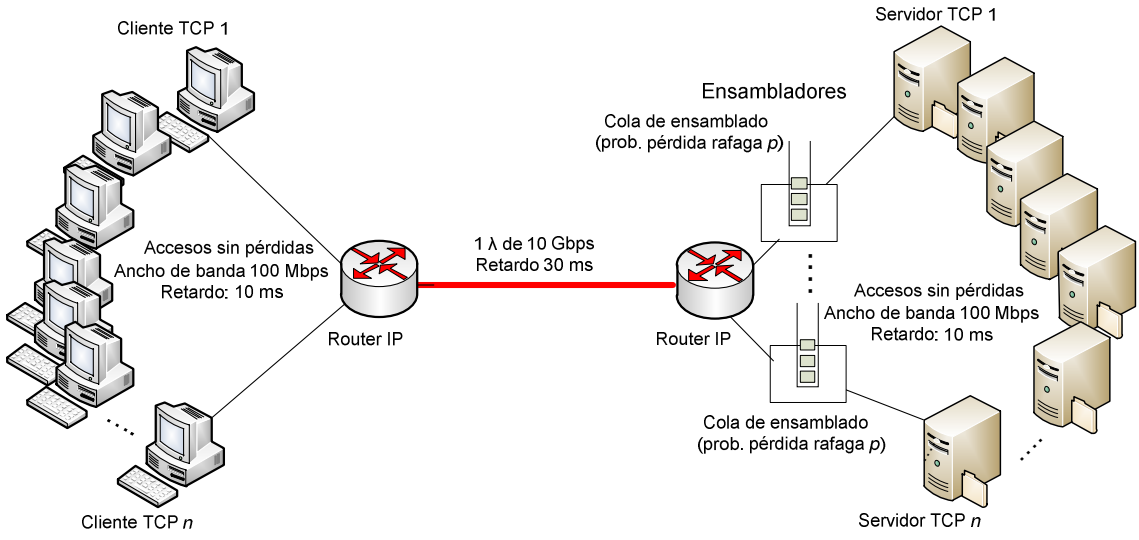


Figura 6-3 Escenario PF: *n* flujos TCP y una cola de ensamblado de ráfaga por flujo

Cada servidor TCP se conecta a través de un enlace de acceso de 100 Mbps y 10 ms de retardo a un ensamblador de ráfagas diferente. Es decir, hay un ensamblador de ráfagas por cada servidor TCP. Los ensambladores de ráfagas implementan una estrategia de ensamblado de temporizador fijo sin máximo de tamaño de ráfaga. Los ensambladores se encargan de agregar los segmentos TCP provenientes de los servidores TCP y, una vez formada una ráfaga, emulan el envío de una ráfaga óptica transmitiendo los segmentos a 10 Gbps al *router* IP. Se trata de un escenario simplificado en el que se pretende evaluar el impacto de la sincronización provocada por la agregación de segmentos en ensambladores.

Para simplificar, en vez de modelar una red OBS completa, se introduce entre los *router* IP un enlace con un solo canal óptico de 10 Gbps y se modela un proceso de pérdida de ráfagas en los ensambladores. Todos los ensambladores se configuran con el mismo valor de temporizador y la misma probabilidad de pérdida de ráfaga. Los *router* IP se han introducido para facilitar el encaminamiento IP de los segmentos TCP. De hecho, se ha asignado automáticamente una IP a cada cliente, servidor y *router* IP.

En el siguiente escenario, cuyo esquema se muestra en la Figura 6-4, se implementa la estrategia MF y consiste, al igual que el escenario anterior, en un conjunto de clientes TCP cada uno de los cuales establece una conexión TCP con uno de los servidores TCP situados al extremo opuesto de la red OBS. A diferencia del escenario anterior, los servidores TCP están conectados a un *router* IP que agrega los datagramas de todos servidores (y se encarga también de enviar los datagramas con los asentimientos provenientes de los clientes a cada uno de los servidores), y a cuya salida se conecta un único ensamblador de ráfagas basado en temporizador fijo sin máximo de tamaño de ráfaga. Solo se ha incluido el ensamblador que recoge los segmentos en sentido servidor-cliente, mientras que los asentimientos generados por el cliente no son ensamblados y se pasan directamente sin retardo alguno. Se ha tomado esta decisión para poder estudiar con detalle cómo afecta un ensamblador al tráfico. Con dos ensambladores, los efectos son más difíciles de apreciar, ya que sería complicado saber si el efecto proviene de uno u otro.

Al igual que en el escenario anterior, en vez de modelar una red OBS completa con múltiples nodos, se ha introducido un enlace punto a punto de una única longitud de onda de 10 Gbps entre el ensamblador y el desensamblador de ráfagas. En el mismo ensamblador se modela un proceso de pérdida de ráfagas, representando lo que ocurriría en una red OBS y se envía la ráfaga (formada por los segmentos TCP) sin añadir a la misma cabeceras adicionales.

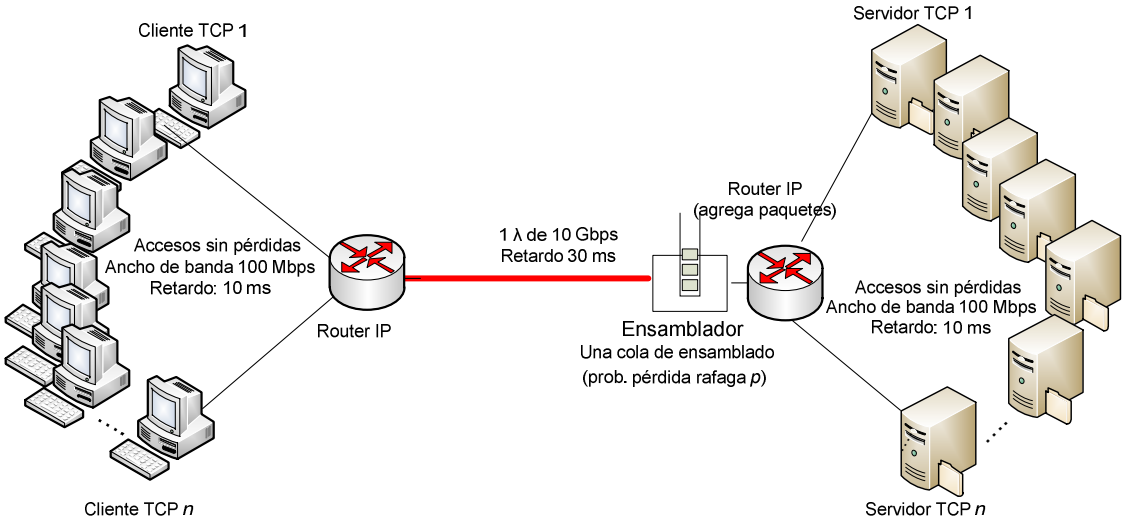


Figura 6-4 Escenario MF: *n* flujos TCP y una cola de ensamblado de ráfaga para todos los flujos

Tráfico TCP estudiado en los escenarios

En ambos escenarios, las fuentes de tráfico se han configurado para enviar datos de manera continua, simulando los flujos denominados popularmente "elefantes" o transferencias masivas (*elephants* o *bulk transfers*) [237], ya que son los que consiguen un mayor *throughput* y realizan un mayor consumo de ancho de banda y marcan el comportamiento del tráfico agregado. En los reportes semanales de Internet 2 (disponibles en [238]) se observa que los flujos TCP de transferencias masivas generan el 40% del total del tráfico de los enlaces. Estos flujos tienen una amplia duración en el tiempo y son los más susceptibles de ser afectados por la sincronización de TCP.

Según la relación entre el ancho de banda de acceso y el temporizador, se puede distinguir entre flujos lentos, medios y rápidos, según el número medio de segmentos de un flujo TCP por ráfaga. En anteriores capítulos se ha analizado el comportamiento de dichos flujos desde el punto de vista del rendimiento, mientras que en el capítulo actual se estudian desde el punto de vista de la sincronización. Considerando el uso de SACK, la reacción de TCP ante pérdidas está gobernada por el temporizador de retransmisión en el caso de las fuentes rápidas, y por la detección de asentimientos duplicados y reducción de la ventana a la mitad en el caso de las fuentes medias y lentas. Desde el punto de vista de la sincronización, cada uno de los comportamientos tiene unas repercusiones diferentes, ya que cuanto más drástica sea la reducción de velocidad de un flujo, cuando se considera un comportamiento agregado, mayores serán las fluctuaciones del tráfico agregado. Por tanto, se estudiarán ambos comportamientos.

En primer lugar se comienza con el estudio del comportamiento de la sincronización en los escenarios PF y MF empleando flujos TCP de velocidad media, en los que cada ráfaga contiene unos pocos segmentos de cada flujo, y en el que todos los flujos tienen el mismo retardo de transmisión en los distintos tramos. A continuación, se estudia la sincronización cuando en vez de flujos de velocidad media, hay flujos rápidos, con capacidad para transmitir un mayor número segmentos. En ambos caso se observará el comportamiento del *throughput* agregado, a partir del cual se obtendrá el dimensionado para conocer el ancho de banda que es necesario garantizar para obtener dicho *throughput*.

Después, se propone y estudia el impacto de la estrategia de ensamblado MQ, por la que los flujos se reparten en varias colas. Mediante esta técnica se pretende reducir los efectos nocivos de la sincronización y a la vez ofrecer una solución escalable y viable. El escenario de simulación de MQ se describe posteriormente.

Finalmente se estudia un escenario más pragmático en el que cada flujo TCP, en vez de tener el mismo retardo de transmisión, tiene un retardo de transmisión diferente, haciendo que cada flujo tenga un RTT diferente. Para ello, se varía aleatoriamente en un margen de 5 ms el retardo de transmisión en el segmento acceso que se ha descrito en el escenario. En este escenario con RTTs diferentes se comparan las estrategias MF, PF y la propuesta MQ.

6.4 Flujos medios con pérdidas de ráfaga y mismo RTT

En primer lugar, se procede a estudiar el comportamiento de flujos TCP de velocidad 'media' de larga duración cuando se agregan cada uno por separado (escenario PF) y ese mismo conjunto de flujos cuando se agregan en un mismo ensamblador (escenario MF), en ambos casos con pérdidas de ráfaga en los ensambladores. Por flujo 'medio' se entiende que la relación entre el ancho de banda de acceso y el valor del temporizador de ensamblado es tal que en una ráfaga pueden viajar varios segmentos de un mismo flujo, de tal manera que si se perdiera esta ráfaga, TCP, aprovechando las ventajas de los asentimientos selectivos (SACK) para recuperarse de múltiples pérdidas, se puede recuperar, en la mayoría de las ocasiones, únicamente reduciendo la ventana a la mitad. Por tanto, para conseguir que los flujos analizados se consideren en la categoría de 'medios', se ha escogido un valor de temporizador de ensamblado de un milisegundo, con lo que, para un tamaño máximo de segmento de 1460 bytes, una ráfaga ensambla como máximo 9 segmentos.

En este primer análisis, se agregan flujos con el mismo RTT, y las fuentes TCP comienzan a transmitir de manera simultánea. El RTT, sin tener en cuenta el tiempo de ensamblado de las ráfagas, es de 100 ms, ya que, como se puede observar en la Figura 6-3 y la Figura 6-4, en retardo de cada uno de los accesos es de 10 ms (tanto en el lado cliente, como en el lado servidor) y en el tramo del enlace OBS el retardo es de 30 ms. Se ha

implementado un escenario adicional en el que las fuentes empiezan a transmitir de manera aleatoria. En las simulaciones realizadas se ha observado que el comportamiento es solamente ligeramente distinto al comienzo de la simulación, pero a los pocos segundos de simulación el comportamiento en el caso de que las fuentes hayan comenzado simultáneamente y en el caso en el que han empezado en instantes aleatorios es similar, obteniéndose resultados finales muy parecidos. Por tanto, todas las simulaciones se realizarán con las fuentes comenzando a transmitir en el mismo instante de tiempo y los datos se analizan a partir del segundo 50 de la simulación, para evitar el comienzo de las transmisiones y centrar el estudio en el estado estacionario. Cada una de las simulaciones realizadas en todo este capítulo tiene una duración de 1000 segundos, y para cada caso se han realizado 4 simulaciones con semillas diferentes. Con una duración de 1000 s y un RTT de 100 ms se simulan 10000 rondas de transmisión en cada simulación.

La probabilidad de pérdida de ráfaga en un ensamblador se ha fijado a 10^{-3}, un valor típico considerado en estudios previos. Con respecto a TCP, se ha empleado un tamaño máximo de ventana de 512 kbytes, un valor típico en los sistemas operativos, como se ha mencionado en la sección 3.2.

Por tanto, en primer lugar se analiza qué sucede cuando hay un ensamblador para cada flujo, estrategia PF, y se compara con el caso en que se agregan varios flujos en un mismo ensamblador, estrategia MF. En la Figura 6-5 y la Figura 6-7 se muestra el tráfico en el enlace OBS, medido en intervalos de 100 ms, para una muestra de 250 segundos de simulación en la que hay 25 flujos TCP para ambas estrategias de ensamblado (PF y MF respectivamente).

Las medidas de tráfico se han realizado contabilizando los bytes de la ráfaga a la salida del ensamblador, de forma que cuando una ráfaga empieza a ser transmitida se contabilizan todos sus segmentos en el intervalo de medida. Así, según cuando se termine de formar una ráfaga, según en qué instante de tiempo finalice, puede ocurrir que parte de la ráfaga se transmita durante el intervalo de medida y parte en el que siguiente, pero se contabiliza únicamente en un intervalo. En cualquier caso, las fluctuaciones introducidas por estas estas medidas serán pequeñas, ya que la duración de la transmisión de una ráfaga es pequeña en comparación con el intervalo de medida (100 ms).

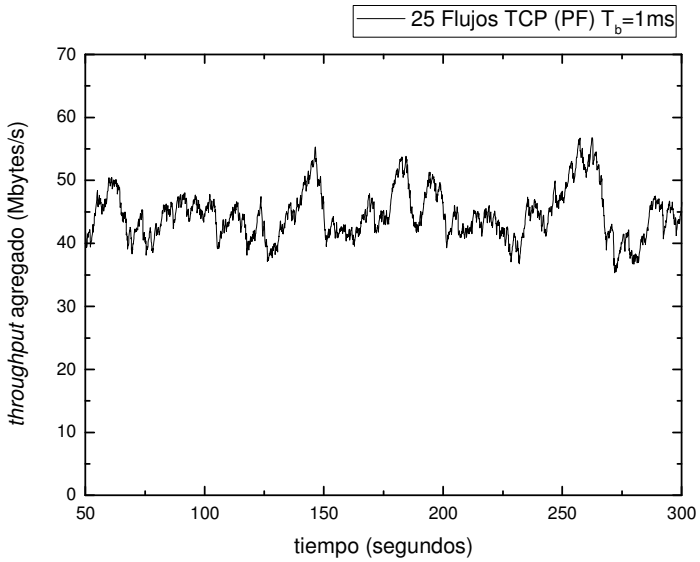


Figura 6-5 *Throughput* agregado de 25 flujos medios con una cola por flujo (PF), T_b= 1 ms

En la Figura 6-5 se muestra el tráfico agregado en el enlace OBS para el caso PF (una cola por flujo), y se observa que dicho tráfico agregado es bastante estable, oscilando entre 35 y 55 Mbytes/s. Esto se debe a que cada flujo reduce su ventana a la mitad (en la mayoría

de las ocasiones) de manera independiente a los demás, es decir, los procesos de pérdida son completamente independientes para cada flujo.

En la Figura 6-6 se muestra el detalle de la evolución del *throughput* individual de varios de los flujos involucrados (se ha medido el *throughput* en intervalos de 100 ms), en el que se comprueba que cada flujo aumenta y disminuye su consumo de ancho de banda en instantes diferentes, obteniéndose en media un tráfico con relativamente pocas fluctuaciones. Los pequeños descensos puntuales que se observan se deben a tres factores, el retardo introducido por los *routers*, el retardo en el ensamblador, y la forma de realizar las medidas. En todo caso, las variaciones puntuales son muy pequeñas. Si se aumenta ligeramente el intervalo de medida, se eliminan las fluctuaciones puntuales.

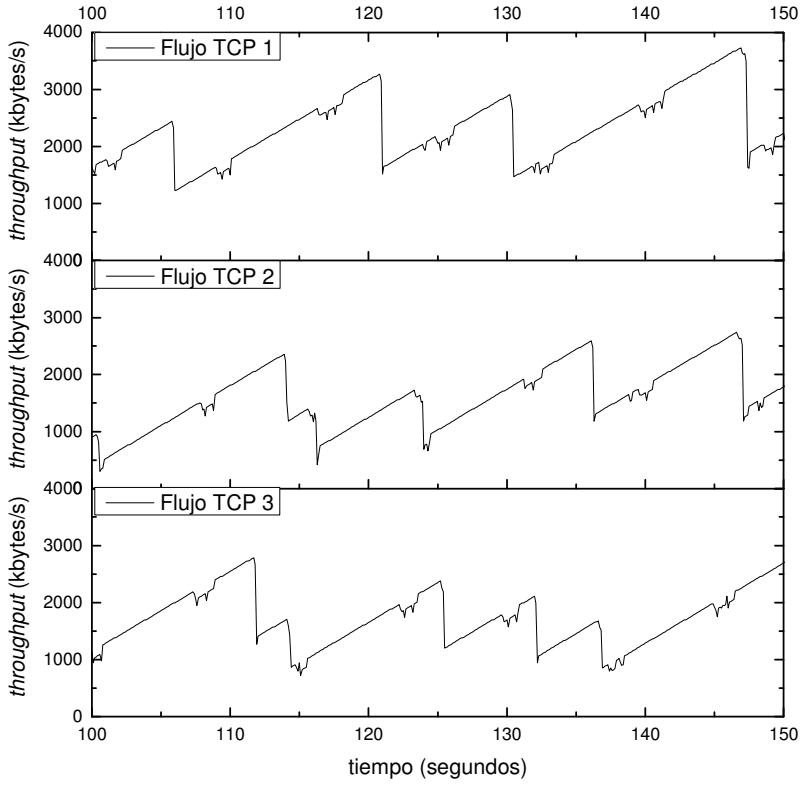


Figura 6-6 Detalle del *throughput* individual de 3 de los flujos (PF)

Sin embargo, en la Figura 6-7, en la que se muestra el *throughput* agregado medido en intervalos de 100 ms cuando se comparte una única cola de ensamblado, hay una gran fluctuación en el consumo de ancho de banda agregado. Tal y como se observa en la Figura 6-7, hay algunos intervalos en los que el *throughput* agregado decae hasta 15 Mbytes/s y hay picos de hasta 65 Mbytes/s, altos en comparación con el caso de una cola por flujo, en el que el *throughput* agregado oscila entre 35 y 55 Mbytes/segundo. Esto se debe a que cada vez que se produce una pérdida, se pierden segmentos de varios flujos TCP. Para ayudar a comprender la magnitud de las pérdidas y su impacto en las variaciones de tráfico se ha medido el número de flujos diferentes por ráfaga en el caso MF. El resultado es que todas las ráfagas contienen segmentos de todos los flujos. Es decir, es un escenario de sincronización total.

La razón por las que cada ráfaga contiene segmentos de absolutamente todos los flujos TCP es que, como en el escenario todos los flujos experimentan exactamente el mismo retardo de transmisión en el segmento de acceso y el de núcleo, cada vez que hay una pérdida de ráfaga, siguen comportamientos similares. Los asentimientos se generan asimismo en un instante de tiempo similar, provocando que los nuevos segmentos se transmitan a su vez en ese mismo intervalo de tiempo, llegando al ensamblador en instantes similares, ensamblándose en la misma ráfaga. Este es el peor escenario posible en cuanto a

sincronización de flujos TCP. En apartados posteriores del capítulo, se estudia un escenario más pragmático, en el que cada flujo experimenta un retardo de acceso diferente, por lo que los RTTs son variados y se elimina la sincronización estricta.

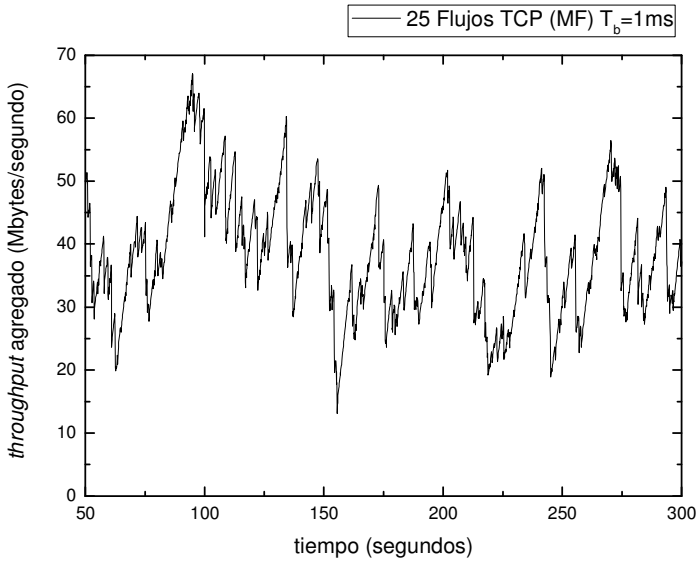


Figura 6-7 *Throughput* agregado de 25 flujos medios con una cola para todos los flujos (MF), T_b= 1ms

En la Tabla 6-1 se ha recopilado un conjunto de medidas de interés para entender las diferencias del comportamiento de TCP en las estrategias de ensamblado consideradas, incluyendo la media y la varianza del *throughput* agregado. Para obtener los valores se han realizado 4 simulaciones largas de 1000 segundos de transmisión TCP y se han descartado los primeros 50 segundos de cada una de las simulaciones.

	Ensamblado PF de 25 flujos (T_b=1 ms)	Ensamblado MF de 25 flujos (T_b=1 ms)
Media del *throughput* agregado	42,81 Mbytes/s	42,97 Mbytes/s
Varianza del *throughput* agregado	13,03 $(\text{Mbytes/s})^2$	357 $(\text{Mbytes/s})^2$
Número medio de segmentos de un flujo TCP por ráfaga	8,35	8,39
Índice de sincronización	0,035	1
Número medio de flujos TCP diferentes por ráfaga	1	25

Tabla 6-1 Recopilación de resultados de los casos MF y PF para flujos medios con mismo RTT

Se observa que en el caso de flujos medios no hay diferencias significativas en el *throughput* medio entre la estrategia MF y PF. Esto se debe a varios motivos. En primer lugar, como hay ancho de banda suficiente en el enlace OBS, el hecho de que haya picos de tráfico no impacta en el tráfico medio. Si el ancho de banda en el enlace OBS tuviera un ancho de banda menor que los picos de tráfico, se limitaría el *throughput*. Por otro lado, se ha obtenido el número medio de segmentos de un flujo TCP por ráfaga en un caso y otro y se observa que apenas hay variación entre ambas estrategias. Esto se debe a que, a pesar de que los flujos se ensamblan conjuntamente en el caso MF, como los accesos son independientes, y la capacidad del enlace OBS excede la de la suma de todos los accesos, los paquetes de unos accesos apenas afectan a los demás, y como el RTT es el mismo, todos los flujos siguen el mismo comportamiento, prácticamente como si se ensamblaran solos.

Aunque en el *throughput* medio no hay apenas diferencia, en la varianza sí hay una diferencia muy significativa. Se pasa de 13 $(\text{Mbytes/s})^2$ en el caso PF a 357 $(\text{Mbytes/s})^2$ en el

caso MF. Es decir, unas 27 veces más en el caso MF comparado con el caso PF. Esto se debe a que en MF cada vez que hay una pérdida se pierden segmentos de varias ráfagas, descendiendo el *throughput* drásticamente. Como se ha indicado anteriormente, en el caso de ensamblado MF, el índice de sincronización es de 1, sincronización total. En el caso PF el índice de sincronización es casi cero, con un pequeño valor debido a intervalos en los que han coincidido pérdidas de ráfagas de forma fortuita.

A continuación se va a analizar en detalle el impacto del *throughput* agregado en el dimensionado del enlace de una red OBS. Para ello, a partir de las medidas de *throughput* agregado obtenidas en las simulaciones anteriores se calcula primero la función de distribución acumulada de dicho *throughput* agregado, que se representa en la Figura 6-8.

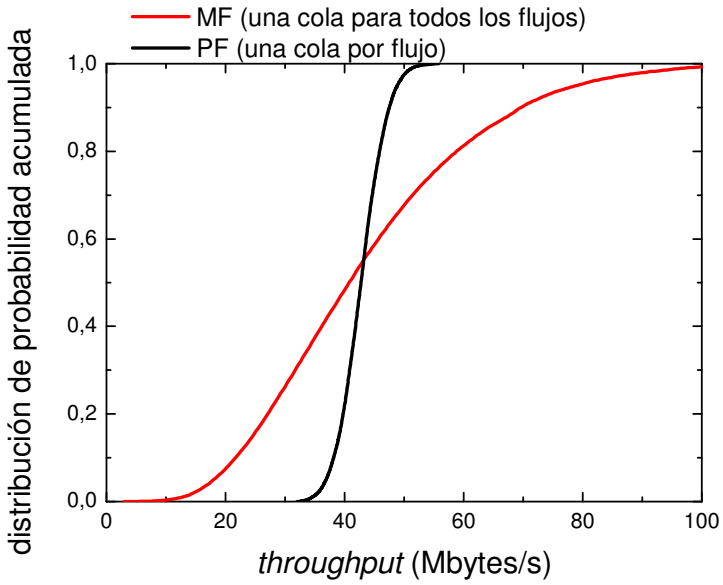


Figura 6-8 Distribución de probabilidad acumulada del *throughput* para fuentes medias con mismo RTT base

A partir de ella, se calcula el valor para el cual el X% del tiempo el *throughput* agregado sea menor a igual a dicho valor, y se denomina $C_{x\%}$. Este valor será el ancho de banda mínimo que hay que provisionar en dicho enlace. El valor de X dependerá de cómo se quiera dimensionar la red, pero típicamente se fija a 95% [239] (cuanto mayor sea el porcentaje fijado, se es más estrictos, con lo que el ancho de banda necesario será mayor). Los valores obtenidos se han recopilado en la Tabla 6-2.

	Ensamblado PF de 25 flujos (T_b=1 ms)	Ensamblado MF de 25 flujos (T_b=1 ms)
$C_{95\%}$	48,83 Mbytes/s	78,69 Mbytes/s
$C_{99\%}$	51,32 Mbytes/s	97,06 Mbytes/s
$C_{95\%}$/*throughput* medio	1,14	1,83
$C_{99\%}$/*throughput* medio	1,2	2,26

Tabla 6-2 Capacidad necesaria para flujos medios con mismo RTT

Para el caso de una cola por flujo, $C_{95\%}$ vale 48,83 Mbytes/s. Para el caso de una cola para todos los flujos, $C_{95\%}$ vale 78,69 Mbytes/s, es decir, la capacidad necesaria es un 61% mayor. Si se observa la relación capacidad/*throughput* medio, para el caso de una cola por flujo, este valor es de 1,14, lo que quiere decir que el sobredimensionamiento necesario es pequeño, un 14% por encima de la media, mientras que para el caso de una cola para todos los flujos, este valor es de 1,83, por lo que el sobredimensionamiento necesario es bastante mayor, un 83% por encima de la media. En este estudio se denomina

sobredimensionamiento al ancho de banda por encima de la media que se ha de proveer en un enlace para soportar los picos de tráfico. Si en vez del 95%, se escoge un objetivo del 99% [233], $C_{99\%}$ vale 97,06 Mbytes/s para el caso MF, y $C_{99\%}$ vale 51,32 Mbytes/s para el caso PF, aumentando en este caso la diferencia entre ambos a un 89%. La relación capacidad/*throughput* medio con el objetivo del 99% es de 1,2 para el caso PF, y de 2,25 para el caso MF. Es decir, con el objetivo más estricto, en el caso MF hace falta más del doble del ancho de banda que el valor de la media. Cuanto más alto se fije el porcentaje objetivo, mayor es el sobredimensionamiento necesario comparando el caso MF con el caso PF.

Resumiendo, a tenor de los resultados de simulación y el análisis realizado, con la estrategia de ensamblado PF, es decir, una cola de ensamblado independiente para cada flujo, se consigue para los flujos medios el mismo *throughput* agregado pero requiriendo provisionarse mucha menor capacidad que con la estrategia de ensamblado MF en la que todos los flujos comparten la misma cola de ensamblado. Sin embargo, la estrategia de ensamblado es muy costosa de realizar, ya que implica una cola por cada flujo, por lo que es poco pragmática.

6.5 Flujos rápidos con pérdidas de ráfaga y mismo RTT

Tras ver el comportamiento de los flujos TCP medios, en esta sección se va a analizar el comportamiento de un conjunto de flujos TCP rápidos de larga duración en los mismos escenarios, es decir, cuando se agregan cada uno en una cola diferente (escenario PF) y ese mismo conjunto de flujos cuando se agregan en un mismo ensamblador (escenario MF), en ambos casos con pérdidas de ráfaga. Por flujo rápido se entiende que la relación entre el ancho de banda de acceso y el valor del temporizador de ensamblado es tal que el número de segmentos TCP que potencialmente caben en una ráfaga es lo suficientemente alto como para que TCP en la mayoría de las pérdidas de ráfaga deba recuperarse con el temporizador de retransmisión, dejando de transmitir durante un tiempo y reduciendo su ventana a uno, independientemente de la versión de TCP.

Al igual que en el estudio de los flujos medios, en el escenario de simulación se ha fijado un ancho de banda de acceso de 100 Mbps. Por tanto, para conseguir que los flujos analizados se consideren en la categoría de 'rápidos', se ha escogido un valor de temporizador de ensamblado de 10^{-2} s, con lo que, con el tamaño máximo de segmento de 1452 bytes empleado en las simulaciones, una ráfaga ensambla como máximo 85 segmentos. Igualmente, para comparar de una manera equitativa el comportamiento en cuanto a sincronización de flujos medios y rápidos, se realizan los mismos supuestos: se agregan flujos con el mismo RTT base, de 100 ms, las fuentes TCP comienzan a transmitir de manera simultánea, la probabilidad de pérdida de ráfaga en un ensamblador se ha fijado a 10^{-3}, y, con respecto a TCP, se ha empleado un tamaño máximo de ventana de 512 kbytes (aproximadamente 350 segmentos).

Por tanto, en primer lugar se analiza qué sucede cuando se agregan varios flujos en un mismo ensamblador (estrategia MF) y se compara con el caso en el que hay un ensamblador para cada flujo. En la Figura 6-9 y la Figura 6-11 se puede ver el tráfico en bytes por segundo en el enlace OBS, medido en intervalos de 100 ms, para una muestra de 250 segundos de simulación en la que hay 25 flujos TCP rápidos para las dos estrategias de ensamblado a estudiar.

En la Figura 6-9 se muestra el tráfico en el enlace OBS para el caso PF (una cola por flujo), y se observa que el tráfico es relativamente estable, al igual que en el caso de los flujos medios. Esto se debe a que cada flujo reduce su ventana de manera independiente a los demás, es decir, los procesos de pérdida son completamente independientes para cada

flujo y cada flujo aumenta y disminuye su consumo de ancho de banda en instantes diferentes. Se obtiene en media un tráfico con relativamente pocas fluctuaciones, teniendo en cuenta que, al tratarse de flujos rápidos, las pérdidas de ráfaga no se pueden recuperar incluso con TCP SACK de una manera fácil. Una pérdida de ráfaga implica la pérdida de muchos segmentos, que no permiten a TCP recuperarse simplemente reduciendo la ventana a la mitad en muchos casos, sino que llegaría a decaer el *throughput* a cero.

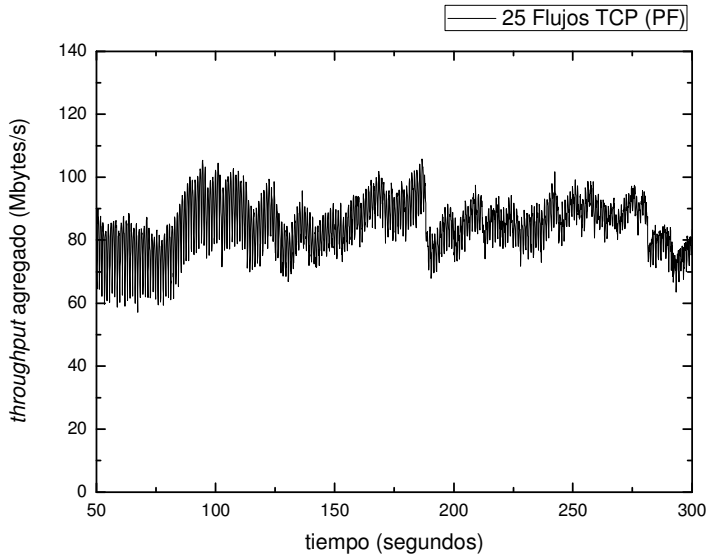


Figura 6-9 *Throughput* agregado de 25 flujos rápidos con una cola por flujo (PF)

Como ejemplo, se muestra en la Figura 6-10 el detalle del comportamiento de uno de los flujos ensamblados, en el que se puede observar cómo decae drásticamente la ventana en algunas ocasiones y en otras cae a la mitad. Por tanto, la principal diferencia entre flujos medios y rápidos es una ligera mayor fluctuación, debido a la mayor severidad de las pérdidas, que se cuantifica más adelante.

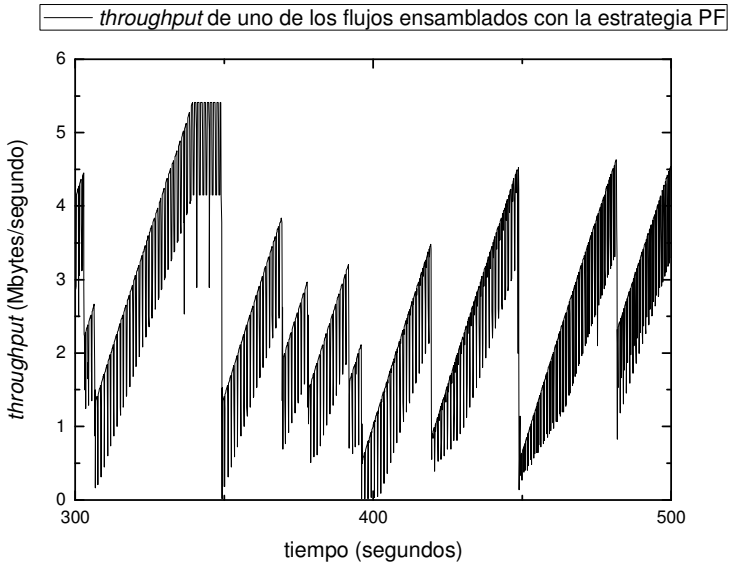


Figura 6-10 Detalle del *throughput* de uno de los flujos en el caso de la estrategia PF

Al igual que en el caso de los flujos medios, se observan unos *glitches* en la evolución del *throughput*. En este caso, como las ráfagas son de más tamaño, el impacto del retardo en las colas y, en general, la influencia del retardo de transmisión de segmentos de otros flujos, es significativa, y a simple vista se pueden observar dichas fluctuaciones.

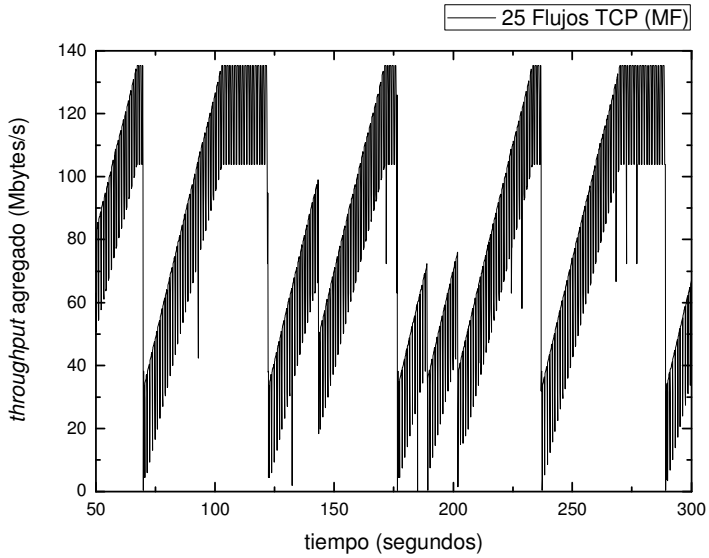


Figura 6-11 *Throughput* agregado de 25 flujos rápidos con una cola común para todos los flujos (MF)

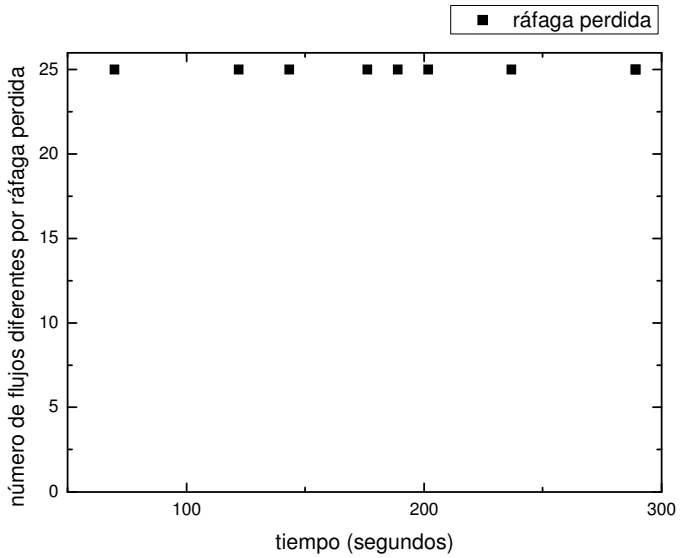


Figura 6-12 Número de flujos diferentes por ráfaga perdida en el caso de la estrategia MF

En cuanto al escenario MF, en la Figura 6-11 se puede comprobar que cuando se comparte el ensamblador, hay una gran fluctuación en el consumo de ancho de banda agregado, mayor que en el caso de flujos medios. Esto se debe a que cada vez que se produce una pérdida, se pierden segmentos de varios flujos TCP, y además la recuperación de estas pérdidas es más difícil que en el caso de flujos medios. Para ayudar a comprender la magnitud de la pérdidas y su impacto en las variaciones de tráfico, se ha medido el número de flujos diferentes que han perdido al menos un segmento, y se ha representado de manera gráfica en la Figura 6-12. Como se puede comprobar, cada ráfaga perdida contiene segmentos de todos los flujos TCP, obteniéndose la peor situación posible en cuanto a sincronización.

Tal y como se observa en la Figura 6-11, hay algunos intervalos en los que el *throughput* agregado decae prácticamente a cero y hay picos de hasta 135 Mbytes/s, que coincide con todas las fuentes transmitiendo en la ventana máxima de 524288 bytes que se ha fijado en la simulación. Se puede observar que en el tiempo en el que las ventanas están al máximo hay una fluctuación de la tasa de transmisión. Esto se debe a que el RTT no es realmente 100 ms. Como se ha explicado en capítulos anteriores, el temporizador de ensamblado introduce un retardo adicional en los paquetes, que varía entre 0 y el valor del temporizador. Por tanto, el RTT que realmente observa una conexión TCP tiene en cuenta

todos los retardos de la red, no solo los de propagación, e incluye, entre otros, el introducido en el ensamblador. Además, como se trata de un caso de fuentes rápidas, el número de segmentos transmitidos en las ráfagas es tal que el tiempo en que tardan en transmitirse los segmentos de otros flujos a 10 Gbps no es despreciable en comparación con lo que tarda en transmitirse un segmento en el tramo de acceso a 100 Mbit/s. Por ejemplo, en el peor de los casos, delante de un segmento de un flujo pueden ir todos los segmentos del resto de flujos (24 flujos), que como puede haber un máximo de 85 segmentos por flujo, se tendrían 2040 segmentos, que siendo de 1492 bytes, tardarían, a 10 Gbps, unos 2,4 ms, que podrían añadirse al RTT. Incluso, los segmentos del propio flujo, que saldrán del desensamblador de ráfagas hacia el cliente al ritmo de la velocidad de acceso introducen un retardo adicional, de hasta 10 ms (por lo que en media, serán 5 ms en caso de que viaje la ráfaga con el máximo de segmentos de un flujo). De esta forma, aunque la ventana de transmisión de TCP de todos los flujos esté al máximo, su *throughput* puede oscilar entre 135 Mbytes/s y 1100 Mbytes/s. Sumando los tamaños máximos de las ventanas de las 25 fuentes, multiplicando por 1,027, que es el *overhead* para introducir las cabeceras TCP e IP (1452 bytes del tamaño máximo de segmento empleado en las simulaciones, más 20 bytes de la cabecera TCP más 20 bytes de la cabecera IP, divididos entre 1452, el tamaño del segmento sin cabeceras) y dividiendo entre el RTT mínimo (100 ms), que incluye únicamente los retardos de propagación, se obtiene el valor de 135 Mbytes/s. Si en vez de emplear el RTT mínimo se considera el caso peor de 122,5 ms, obtenido sumando a los 100 ms del RTT mínimo, 10 ms del retardo de ensamblado máximo, 2,5 ms de espera a la transmisión del resto de segmentos de otros flujos y 10 ms de retardo máximo de espera a los segmentos del mismo flujo en el tramo de acceso tras llegar del enlace de 10 Gbps, se obtiene aproximadamente el valor mínimo de 1100 Mbytes/s.

Con respecto al análisis del caso MF, se observa que hay instantes en los que el *throughput* está al máximo posible por la ventana de transmisión máxima, y otros en los que decae a cero, en los que se pierden todos los flujos, como se ve en la Figura 6-12. En cambio, en el caso de una cola por flujo, el *throughput* agregado oscila entre 60 y 105 Mbytes/s. Comparando con el caso de los flujos medios, las fluctuaciones son mayores, debido a la mayor reducción de la ventana en el caso de una pérdida de ráfaga con flujos rápidos. Gracias a que se alcanza el tamaño máximo de ventana y la transmisión transcurre durante bastante tiempo con la ventana al máximo, las fluctuaciones se limitan con respecto a un caso en el que no hubiera limitación de ventana, en cuyo caso la ventana seguiría creciendo, por lo que el *throughput* podría seguir aumentando. En la Tabla 6-3 se ha recopilado la media y la varianza del *throughput* agregado para los casos estudiados. Para obtener estos valores, al igual que en el caso de flujos medios, se han realizado cuatro simulaciones largas de 1000 segundos de transmisión TCP y se han descartado los primeros 50 segundos de simulación.

	Ensamblado PF de 25 flujos (T_b=10 ms)	**Ensamblado MF de 25 flujos (T_b=10 ms)**
Media del *throughput* agregado	84,83 Mbytes/s	87,82 Mbytes/s
Varianza del *throughput* agregado	46,17 $(\text{Mbytes/s})^2$	1277 $(\text{Mbytes/s})^2$
Índice de sincronización	0,001	1

Tabla 6-3 Recopilación de resultados en los casos PF y MF para 25 flujos rápidos con mismo RTT

A diferencia del caso de los flujos medios, como se ve en la Tabla 6-3, el *throughput* medio es ligeramente superior en el caso de la estrategia MF que con la estrategia PF. En cambio, la varianza sí desciende drásticamente del caso PF al caso MF, al igual que sucedía en el caso de los flujos medios, y lo hace variando en este caso de 1277 $(\text{Mbytes/s})^2$ en el caso MF a 46 $(\text{Mbytes/s})^2$ en el caso PF. Comparando los flujos rápidos con los medios, en

los flujos rápidos la varianza es mucho mayor, tanto en la estrategia PF como en la MF. Esto se debe a la mayor severidad de las pérdidas, y las mayores fluctuaciones puntuales de tráfico. El índice de sincronización, igualmente, en el caso PF es muy pequeño, y se debe únicamente a las pocas pérdidas que coinciden (en concreto en 10 de los 416 intervalos en los que hubo al menos una pérdida de ráfaga). En el caso MF, como era de esperar, hay una sincronización total.

Al igual que en el apartado anterior, se calcula la función de distribución acumulada del *throughput* agregado y, a partir de ella se obtiene el valor para el cual el X % del tiempo el *throughput* agregado sea menor a igual a dicho valor, denominado $C_{X\%}$. Los resultados se recopilan en la Tabla 6-4. Para el caso de una cola por flujo (PF), $C_{95\%}$ vale 95,86 Mbytes/s, mientras que para el caso de una cola para todos los flujos (MF), $C_{95\%}$ asciende a 135,28 Mbytes/s. En las simulaciones realizadas se ha observado que durante más de un 5% del tiempo todos los flujos están transmitiendo al máximo posible ya que han alcanzado la ventana de transmisión máxima configurada. Por esta razón, el valor de $C_{95\%}$ coincide con el valor del *throughput* agregado máximo. A partir de la relación entre $C_{95\%}$ y el *throughput* medio, se obtiene el porcentaje de tráfico extra con respecto a la media que se ha de sobredimensionar. En el caso PF, planificando para el 95%, se ha de proporcionar únicamente un 13% adicional de ancho de banda con respecto al *throughput* medio, mientras que en el caso MF se ha de proporcionar un 54% adicional con respecto al *throughput* medio. Si en vez del objetivo del 95%, se escoge el criterio del 99%, se observa que para el caso MF el valor de $C_{99\%}$ ya no sube más que el valor de $C_{95\%}$, ya que, como se ha dicho, el valor coincide con el *throughput* agregado máximo y la transmisión se encuentra en más del 5% del tiempo en el máximo de ventana. Por tanto, para caso MF, el ancho de banda que es necesario garantizar es el valor del *throughput* agregado máximo. Si se compara con el resultado obtenido en el análisis de los flujos medios, la capacidad adicional a proveer es menor. Sin embargo, este resultado está condicionado por la influencia del límite del *throughput* máximo por el hecho comentado de estar la ventana de transmisión a su máximo valor durante gran parte de la transmisión.

	Ensamblado PF de 25 flujos (T_b=10 ms)	**Ensamblado MF de 25 flujos (T_b=10 ms)**
$C_{95\%}$	95,86 Mbytes/s	135,28 Mbytes/s
$C_{99\%}$	100,2 Mbytes/s	135,28 Mbytes/s
$C_{95\%}$/ *throughput* medio	1,13	1,54
$C_{99\%}$/ *throughput* medio	1,18	1,54

Tabla 6-4 Capacidad necesaria para 25 flujos rápidos en PF y MF con el mismo RTT base

Resumiendo, a tenor de los resultados de simulación y el análisis realizado, para flujos rápidos, con la estrategia de ensamblado PF, es decir, una cola de ensamblado independiente se necesitan 41 puntos porcentuales menos de sobredimensionamiento que en el caso MF cuando se escoge un objetivo del 95%. El análisis muestra un gran impacto de la ventana máxima de transmisión en las fuentes rápidas, hecho que no tenía apenas impacto en las fuentes medias, en las que el flujo apenas estaba tiempo con la ventana de transmisión al máximo. Aunque el análisis realizado para fuentes medias y rápidas con la mima probabilidad de pérdida de ráfaga ha mostrado comportamientos ligeramente diferentes, en ambos casos (tanto para fuentes medias como rápidas) el ancho de banda adicional con respecto al *throughput* medio necesario para soportar tráfico TCP es mucho mayor en el caso MF que en el caso PF.

6.6 Reducción de sincronización con múltiples colas (MQ)

En base a los análisis realizados, se ha corroborado que las pérdidas de ráfagas con segmentos de múltiples flujos son las causantes de la sincronización de los flujos TCP. Asimismo, se ha visto que un mayor tiempo de ensamblado provoca una sincronización mayor. También se ha comprobado que es el número de flujos por ráfaga el factor que importa en la sincronización, mientras que la reacción de TCP ante las pérdidas influye en el impacto de la sincronización, aumentando la varianza del *throughput* agregado cuanto más drástica sea la reducción de ventana. Por tanto, para reducir la sincronización en la red OBS, se debe tratar de reducir el número de flujos diferentes por ráfaga.

Para ello, se propone un esquema de ensamblado con múltiples colas (*Multiple Queues*, MQ), en el que cada ensamblador consta de varias colas en las que se reparten los flujos con un mismo origen, destino y clase de servicio. El reparto de los flujos entre las colas podría hacerse de varias maneras, por ejemplo, segmento a segmento, o flujo a flujo. Si se reparte segmento a segmento, puede producirse llegadas en desorden de los segmentos TCP, que son interpretadas por TCP como señales de congestión y las trata como si de pérdidas se tratasen [240]. Por tanto, es aconsejable definir un criterio para un flujo, por ejemplo protocolo, dirección IP origen, puerto origen, dirección IP destino y puerto destino y, una vez asignado un flujo a una cola, este se mantenga asociado a ella, evitando así llegadas en desorden. Si, por motivos de rendimiento, es necesario realizar menos procesamiento para identificar a qué flujo pertenece un segmento, se puede reducir la caracterización a direcciones IP origen y destino. En la propuesta, se plantea una asignación estática de los flujos a las colas, es decir, fijar un criterio para la asignación y mantenerlo siempre. El principal inconveniente de este método es que no tiene por qué garantizar un balanceo de carga entre las colas. El estudio actual se centra en comprobar si este tipo de mecanismos con varias colas reporta beneficios suficientes como para plantearse su implantación.

6.6.1 Escenario de simulación

Mediante simulación se va a estudiar el comportamiento de un esquema de ensamblado MQ2, es decir 2 colas a compartir entre todos los flujos, y un esquema MQ4, con 4 colas a compartir, en el que la asignación de cola se realiza de manera estática por dirección IP origen. La asignación se ha simplificado para el escenario concreto simulado, que se describe a continuación. El escenario MQ2 se muestra en la Figura 6-13. El escenario MQ4 es similar, pero en vez de 2 colas, hay 4 colas a compartir entre todos los flujos.

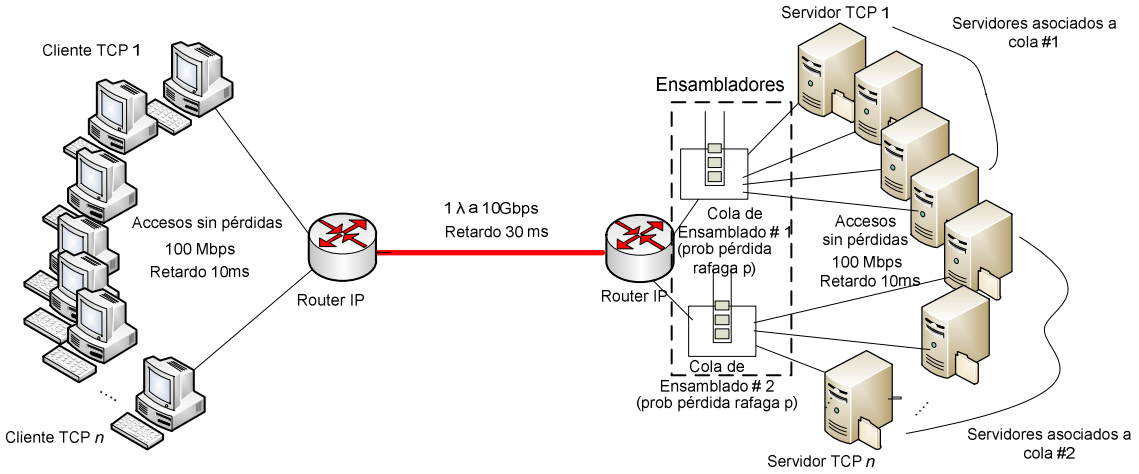


Figura 6-13 Escenario MQ2: *n* flujos TCP y dos colas de ensamblado a compartir entre todos los flujos

La asignación de flujos a colas, como se ha mencionado se ha realizado de manera estática en función de la dirección IP, con lo que, en el caso de 25 flujos simulados y MQ2, 13 flujos se han asignado a un ensamblador y 12 al otro. En el caso del escenario de simulación MQ4, a cada cola se le han asignado 6 flujos, excepto a la última a la que se han asignado 7 flujos. Para emular este comportamiento, se ha creado un enlace entre cada servidor TCP con la cola de ensamblado asignada. La salida de las colas de ensamblado, como en el caso PF, se conecta a un *router* IP para agregar las ráfagas que comparten el enlace de salida común. Hay que recordar que el hecho de que sea IP es simplemente para facilitar el direccionamiento, ya que su única labor es controlar el acceso al enlace OBS. Al igual que en los escenarios MF y PF, los segmentos TCP con datos se envían de servidor a cliente (derecha a izquierda en la Figura 6-13), y los segmentos que actúan de asentimiento de cliente a servidor (izquierda a derecha en la figura). Igualmente, el comportamiento de OBS se ha simplificado y emulado mediante un proceso de pérdidas independiente en cada uno de los ensambladores y la transmisión de la ráfaga se ha emulado transmitiendo los segmentos de manera consecutiva a 10 Gbps.

A continuación se analiza el comportamiento de flujos medios con las estrategias propuestas MQ2 y MQ4, para después realizar el análisis en el caso de los flujos rápidos, también con ambas estrategias. En ambos casos, además del comportamiento del *throughput* agregado se estudia el sobredimensionamiento necesario.

6.6.2 Análisis de MQ para flujos medios

En el escenario con la estrategia MQ de 2 colas, cuyo *throughput* medio medido en intervalos de 100 ms para 25 flujos medios con el mismo RTT base de 100 ms se muestra en la Figura 6-14, se observa que el tráfico agregado es bastante más estable que cuando se emplea una única cola para todos los flujos. En el caso MF visto en el apartado anterior, el tráfico en el ejemplo del intervalo mostrado en la Figura 6-7 oscilaba entre 15 y 65 Mbytes/s, mientras que en el escenario MQ de 2 colas, apenas baja el tráfico de 30 Mbytes/s, y tiene un único pico de 70 Mbytes/s, manteniéndose en general por debajo de 50 Mbytes/s el resto de la simulación.

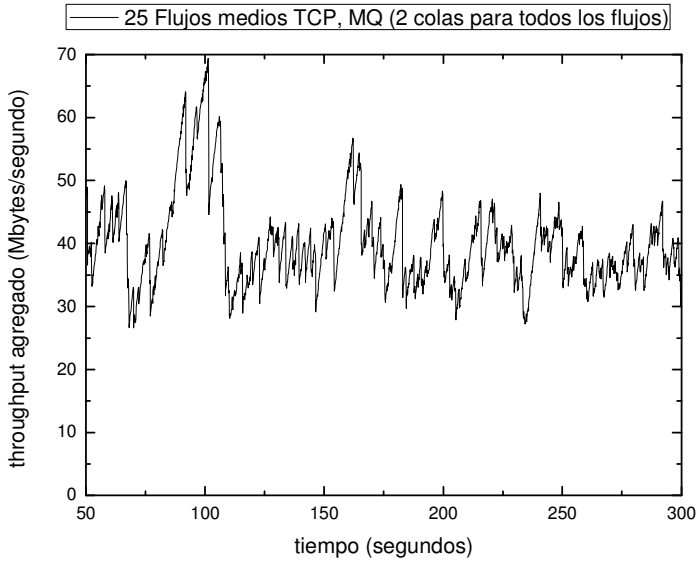


Figura 6-14 *Throughput* agregado de 25 flujos medios con MQ2 de 2 colas y mismo RTT base

La Tabla 6-5 recopila los resultados de las simulaciones, tanto con la técnica MQ2 con 2 colas, como con la estrategia MQ4 con 4 colas. El *throughput* agregado medio obtenido con la estrategia MQ2 apenas varía con el obtenido en las estrategias MF y PF. Sin embargo, la varianza empleando la técnica MQ2 desciende en gran medida, pasando de 357 $(\text{Mbytes/s})^2$ del caso MF (ver Tabla 6-1) a 136,57 $(\text{Mbytes/s})^2$, es decir desciende más del

50%. De esta forma, simplemente con añadir una cola se consigue una reducción importante de la varianza, estabilizando el tráfico y apenas se varía el *throughput* medio. De hecho, se observa que el índice de sincronización desciende desde 1, sincronización total, a 0,3913. El número de flujos diferentes por ráfaga desciende por debajo de la mitad de los 25 flujos, pasando a 12,14 flujos (cuando la mitad serían 12,5 flujos). El número medio de segmentos por ráfaga no varía, lo que se refleja en el hecho de que la media del *throughput* agregado no cambie apenas. Por tanto, la estrategia MQ2 es efectiva para reducir la sincronización y no impactar en el resto de parámetros.

	Ensamblado MQ2 con 2 colas y 25 flujos (T_b=1 ms)	**Ensamblado MQ4 con 4 colas y 25 flujos (T_b=1 ms)**
Media del *throughput* agregado	41,88 Mbytes/s	42,80 Mbytes/s
Varianza del *throughput* agregado	136,57 $(\text{Mbytes/s})^2$	76,91 $(\text{Mbytes/s})^2$
Número medio de segmentos de un flujo TCP por ráfaga	8,37	8,37
Índice de sincronización	0,3913	0,2144
Número medio de flujos TCP diferentes por ráfaga	12,14	5,4

Tabla 6-5 Resumen resultados MQ2 y MQ4 para 25 flujos medios y mismo RTT base

Añadiendo una cola se han visto mejoras significativas en la varianza, por lo que si se dobla el número de colas a 4, es de esperar que continúe reduciéndose la varianza y se siga manteniendo el hecho de no afectar al *throughput* agregado medio. Para el caso de MQ con 4 colas se muestra en la Figura 6-15 un ejemplo de un intervalo de la transmisión TCP, representando el *throughput* agregado medido en intervalos de 100 ms.

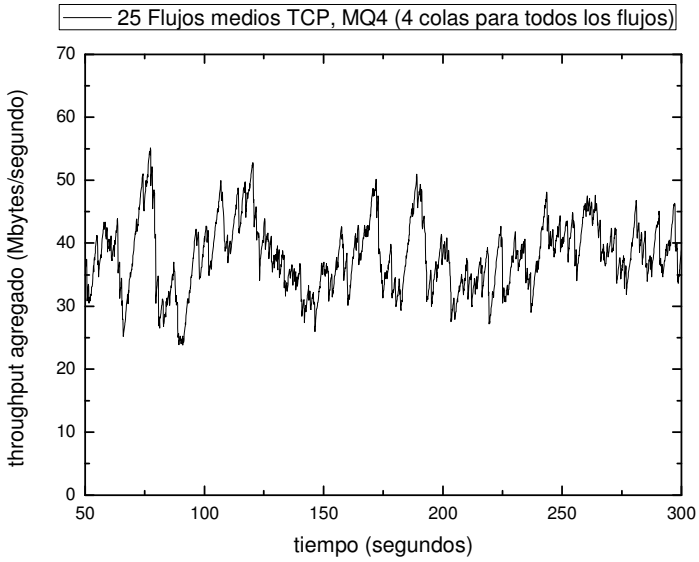


Figura 6-15 *Throughput* agregado de 25 flujos con MQ4 de 4 colas

En la Figura 6-15 se observa que el tráfico agregado es un poco más estable que el caso de MQ con 2 colas. Visualmente se ve que el tráfico apenas supera los 50 Mbytes/s de pico. Sin embargo, el análisis sobre la figura no permite determinar con exactitud la magnitud de variaciones en el *throughput*. Para analizar los datos con una mayor precisión, y averiguar si merece la pena elevar el número de colas, se puede recurrir a la recopilación de resultados de la Tabla 6-5 comparándolos con los de la Tabla 6-1 con los datos de las estrategias MF y PF. En la tabla mencionada se observa que la varianza con la estrategia MQ4 desciende a 76,91 $(\text{Mbytes/s})^2$, por debajo de la mitad del caso MQ2 y una cuarta

parte que en el caso de la estrategia MF, aunque aún lejos de 13 $(\text{Mbytes/s})^2$ que se obtiene en la estrategia PF. De esta forma, aunque no se llega a la bajísima varianza del PF, sí se consigue una reducción significativa y se sigue manteniendo el *throughput* medio. En cuanto al índice de sincronización se aprecia que se reduce en la estrategia MQ4 a la cuarta parte del caso MF, a costa de cuadruplicar el número de colas. Por tanto, la ventaja principal de MQ4 es la reducción de la varianza manteniendo el *throughput*.

A continuación, para cuantificar el impacto de la sincronización, se va a analizar el dimensionado del enlace OBS necesario para flujos medios con las estrategias MQ, para los casos de 2 y 4 colas, y se compara con los casos MF y PF, todos ellos en un escenario con el mismo RTT base de 100 ms.

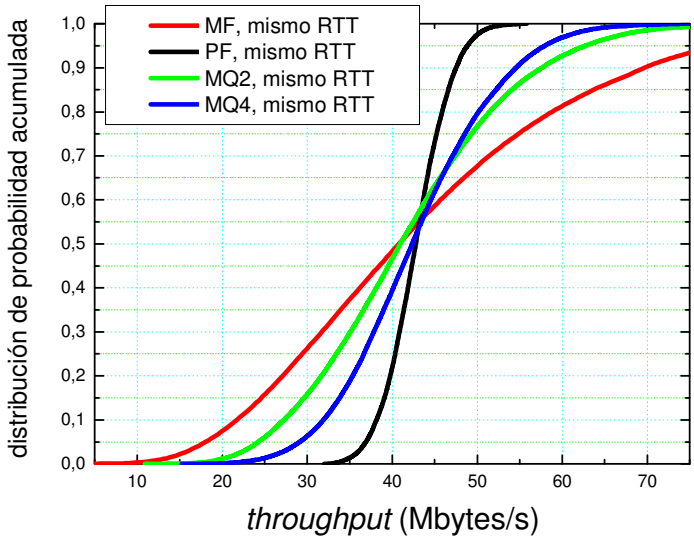


Figura 6-16 Distribución de probabilidad acumulada del *throughput* agregado para flujos medios (PF, MF, MQ2, MQ4) con el mismo RTT base

La Figura 6-16 muestra las distribuciones de probabilidad acumulada para los distintos casos, MF, PF, MQ2 y MQ4, obtenidas a partir de los resultados de simulación descartando los primeros 50 segundos. Para observar con más detalle la zona de interés para el dimensionado, se muestra en la Figura 6-17 el detalle de la zona de las distribuciones alrededor del 95% y 99%, mostrando el resumen de los resultados en la Tabla 6-6

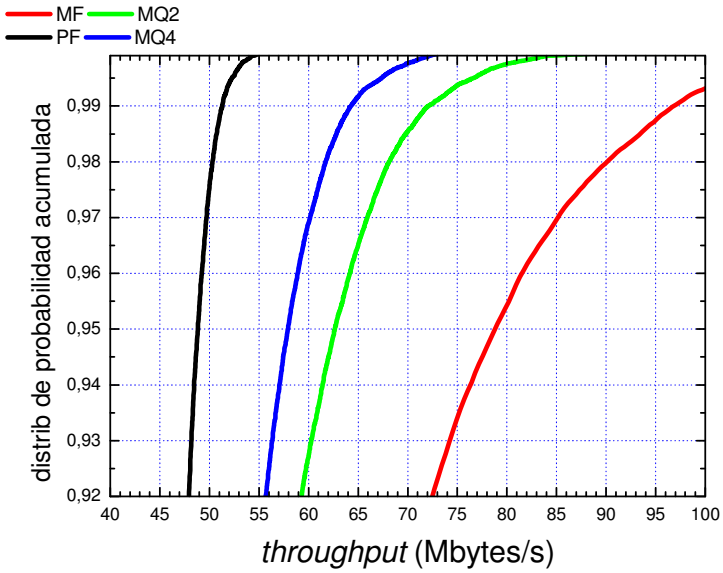


Figura 6-17 Detalle de las distribuciones de probabilidad acumulada del *throughput* agregado para 25 flujos medios (PF, MF, MQ2, MQ4) y mismo RTT base

En el caso MQ2, con dos colas para compartir entre los flujos, $C_{95\%}$ vale 62,64 Mbytes/s, es decir, se necesita en total un 14% menos de ancho de banda que en el caso MF, en el que $C_{95\%}$ vale 72,86, como se indica en la Tabla 6-2. Si se duplica de nuevo el número de colas, pasando a MQ4, se observan diferencias significativas, obteniéndose un dimensionado de enlace OBS de 57,94 Mbytes/s, un 20% de reducción en el ancho de banda total necesario.

	Ensamblado MQ2 de 25 flujos (T_b=1 ms)	**Ensamblado MQ4 de 25 flujos (T_b=1 ms)**
$C_{95\%}$	62,64 Mbytes/s	57,94 Mbytes/s
$C_{99\%}$	72,09 Mbytes/s	64,29 Mbytes/s
$C_{95\%}$/*throughput* medio	1,49	1,35
$C_{99\%}$/*throughput* medio	1,72	1,50

Tabla 6-6 Capacidad necesaria para flujos medios, estrategias MQ2 y MQ4

Si en vez del 95%, se escoge un objetivo del 99%, $C_{99\%}$ vale 72,09 Mbytes/s para el caso MQ2, un 26% de reducción de necesidad de ancho de banda total con respecto al caso MF, que necesitaba 97,06 Mbytes/s (Tabla 6-2). Doblando el número de colas, en el caso MQ4, el ancho de banda necesario con el objetivo del 99% baja hasta 64,29 Mbytes/segundo, un 34% menos que en el caso MF. Es decir, se obtiene una reducción mayor que la que se obtiene considerando un objetivo del 95%.

Para representar de una manera clara el sobredimensionamiento necesario en cada caso, se ha representado en la Figura 6-18 el porcentaje de capacidad adicional con respecto a la media del tráfico que hay que proveer, cuando se emplea un dimensionado al 95%. Este sobredimensionado se obtiene de la relación entre la capacidad necesaria y la media. Se aprecia como, simplemente añadiendo una cola, se pasa de necesitar un 83% de capacidad extra con respecto a la media con la estrategia MF a un 49% con respecto a la media con la estrategia MQ2. En el caso de duplicar el número de colas, pasando a las 4 de la estrategia MQ4, se necesita únicamente un 35% de capacidad adicional con respecto a la media en el caso simulado de 25 flujos.

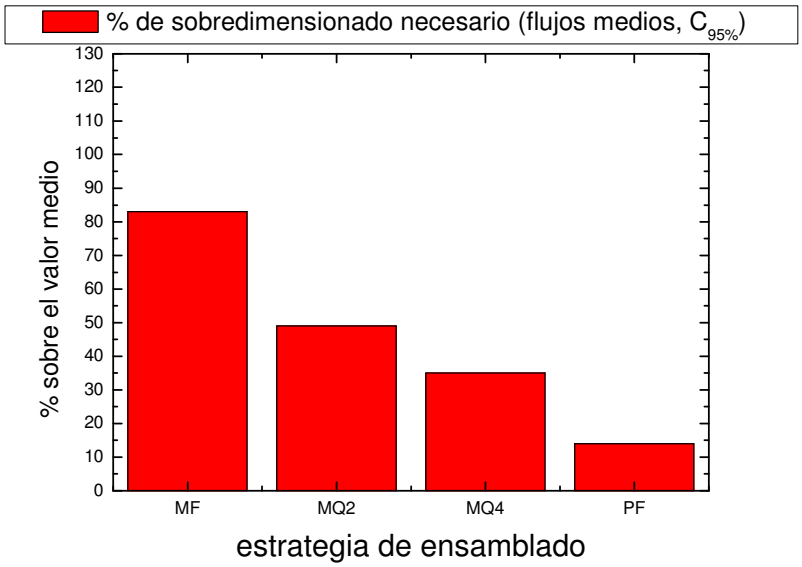


Figura 6-18 Sobredimensionado en % sobre la media para 25 flujos medios con mismo RTT base, con $C_{95\%}$

Si se fija el objetivo de capacidad más estricto del 99%, el sobredimensionado necesario es mayor. Sin embargo, como se observa en la Figura 6-19 la reducción de ancho de banda adicional necesario es mayor empleando las estrategias MQ2 y MQ4. Es decir, estas

estrategias con múltiples colas tienen un mayor impacto cuanto más estricto sea el criterio de dimensionado. Por ejemplo, se pasa de tener que proporcionar un 123% de capacidad adicional con respecto a la media en el caso MF a un 72% en el caso MQ2, y al 50% en el caso MQ4. Aun así, el sobredimensionado queda lejos del caso ideal de la estrategia PF, que necesita únicamente un 20% con respecto a la media.

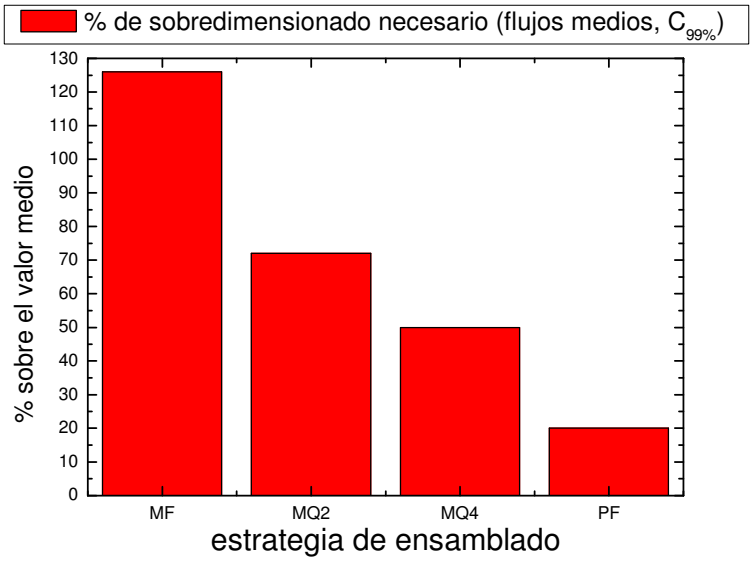


Figura 6-19 Sobredimensionado en % sobre la media para 25 flujos medios con mismo RTT base, con $C_{99\%}$

Por tanto, para los flujos medios, las estrategias MQ2 y MQ4 propuestas mantienen el *throughput* medio, reducen la varianza significativamente y consiguen ahorros sustanciales en cuanto a capacidad adicional con respecto a la media para dimensionar enlaces. Por ejemplo, con el dimensionado más estricto, se pasa de un sobredimensionado del 126% con respecto a la media en el caso MF a un sobredimensionado del 50% con respecto a la media con MQ4.

6.6.3 Análisis de MQ para flujos rápidos

Una vez analizado el caso de los flujos medios, se continúa con los flujos rápidos, en los que la sincronización es mayor, al contener más flujos por ráfaga y el impacto de la sincronización es también mayor, al tener mayor número de segmentos por ráfaga, provocando mayores picos de tráfico.

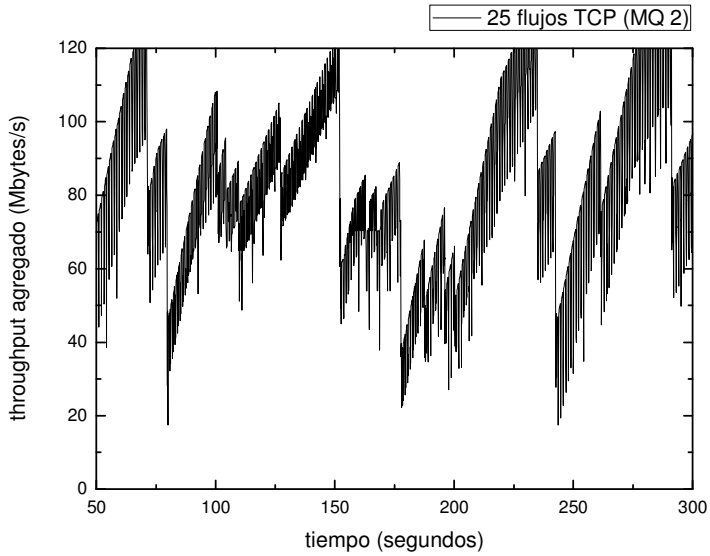


Figura 6-20 *Throughput* agregado de 25 flujos rápidos con mismo RTT base con estrategia MQ2

En el análisis hay que recordar el contexto del caso estudiado de fuentes rápidas. En primer lugar, la ventana máxima de TCP ha sido el factor limitante en el *throughput*, y ha provocado una concentración elevada de periodos a tasa máxima. El hecho de alcanzar la ventana de TCP máxima ha ayudado a mitigar en parte los efectos de la sincronización, ya que impide que la ventana (y por tanto la tasa de transmisión) crezca indefinidamente, con lo que se limita el tamaño y número de picos. Por otro lado, el hecho de que todos los flujos tengan el mismo RTT base hace que los flujos tengan un comportamiento similar al atravesar el mismo ensamblador.

Al analizar las gráficas de la evolución del *throughput* agregado, en el caso MQ2, mostrado en la Figura 6-20, se aprecia que las oscilaciones de tráfico son menores que el caso MF. En caso de MF en distintas ocasiones se llega casi a cero (Figura 6-11), mientras que en el caso MQ2 apenas se baja de 20 Mbytes/s. Por otro lado, los instantes de tiempo en los que todas las fuentes transmiten a su máxima tasa son muy breves, de menor duración que en el ejemplo mostrado de la estrategia MF en la Figura 6-11. Por tanto, con una sola cola más, se aprecia visualmente un incremento de la estabilidad, que se corrobora con los datos de la Tabla 6-7. La varianza desciende a la mitad, desde 1277 $(\text{Mbytes/s})^2$ en la estrategia PF a 650 $(\text{Mbytes/s})^2$ con la estrategia MQ2.

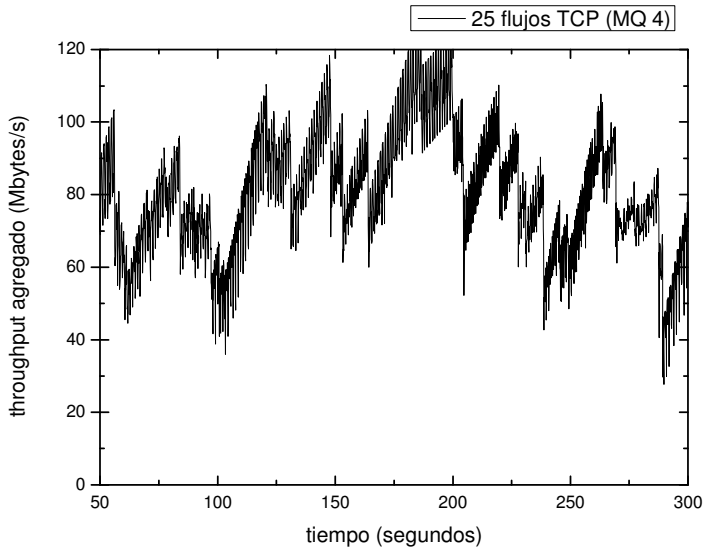


Figura 6-21 *Throughput* agregado de 25 flujos rápidos con mismo RTT base con estrategia MQ4

Si se aumenta el número de colas, pasando a la estrategia MQ4, se aprecia visualmente en el fragmento de la simulación mostrado en la Figura 6-21 que el *throughput* agregado es aún más estable, con menos fluctuaciones. Se observa que en este caso el número de veces en que el *throughput* agregado supera los 120 Mbytes/s es bajo, y una única vez desciende a 35 Mbytes/s y un par de instantes de tiempo cerca de los 40 Mbytes/s.

	Ensamblado MQ2 y 25 flujos (T_b=10 ms)	Ensamblado MQ4 y 25 flujos (T_b=10 ms)
Media del *throughput* agregado	80,33 Mbytes/s	83,69 Mbytes/s
Varianza del *throughput* agregado	650 $(\text{Mbytes/s})^2$	296,25 $(\text{Mbytes/s})^2$
Número medio de segmentos de un flujo TCP por ráfaga	66,4	67,3
Índice de sincronización	0,4349	0,2224
Número de medio de flujos TCP diferentes por ráfaga	10,77	6,2

Tabla 6-7 Resumen resultados MQ2 y MQ4 para flujos rápidos

Tanto con la estrategia MQ2 como con la MQ4 el *throughput* agregado medio es muy parecido al obtenido con las estrategias MF y PF. En cambio, en la varianza se confirman las apreciaciones visuales y se desciende hasta 296,25 $(\text{Mbytes/s})^2$ en el caso de MQ4. El índice de sincronización obtenido es ligeramente superior al de los flujos medios.

Al igual que en el caso de los flujos medios, se analiza el dimensionamiento necesario. Fijando el objetivo del 95%, en vez de necesitarse como en la estrategia MF tanto ancho de banda como el valor del *throughput* agregado máximo (135 Mbytes/s, como se indica en la Tabla 6-2), es decir dimensionar al pico máximo, se reduce la capacidad necesaria a 123,02 Mbytes/s en el caso de MQ2, llegando a reducirse a 111,57 Mbytes/s en el caso MQ4 como se muestra en la Tabla 6-8. En cambio, si se fija el objetivo más estricto del 99%, en el caso MQ2 se vuelve a obtener el valor del *throughput* agregado máximo, debido a que aún los flujos se encuentran a la capacidad máxima durante al menos el 1% del tiempo.

	Ensamblado MQ2 de 25 flujos (T_b=10 ms)	Ensamblado MQ4 de 25 flujos (T_b=10 ms)
$C_{95\%}$	123,02 Mbytes/s	111,57 Mbytes/s
$C_{99\%}$	135,28 Mbytes/s	120,17 Mbytes/s
$C_{95\%}$/*throughput* medio	1,53	1,33
$C_{99\%}$/*throughput* medio	1,68	1,43

Tabla 6-8 Capacidad necesaria para flujos rápidos, estrategias MQ2 y MQ4

Se aprecia como el sobredimensionado cuando se emplea un dimensionado al 95% es menor que en el caso de los flujos medios, debido al efecto de la ventana máxima de transmisión. En el caso MQ2 apenas hay variación, ya que aunque la capacidad necesaria obtenida es menor, también lo es el *throughput* medio obtenido, y se necesita un 53% con respecto a la media. Pasando a 4 colas, la capacidad extra necesaria desciende a un 33% sobre la media.

En cambio, si se fija el objetivo de capacidad más estricto del 99%, el efecto de la ventana máxima de transmisión condiciona los resultados. Por ejemplo, como en MQ2 la capacidad necesaria con el criterio del 99% es igual a la suma de la tasa máxima de transmisión de cada flujo, al igual que en el caso MF, pero el *throughput* agregado que se obtiene es ligeramente menor que en el caso MF, se obtiene un sobredimensionado peor que en MF. En cambio, en el caso de MQ4, el efecto de la ventana máxima de transmisión se diluye, y se consigue descender del 53% de sobredimensionado necesario con la estrategia MF para pasar al 43% con la estrategia de ensamblado MQ4.

Por tanto, para los flujos rápidos, las estrategias MQ2 y MQ4 apenas reducen el *throughput* medio, pero reducen la varianza significativamente. En el caso simulado, las conexiones TCP pasan una buena parte del tiempo transmitiendo a tasa máxima, limitados por la ventana de transmisión máxima, lo que reduce las variaciones de tráfico y limita la bondad y efectos positivos de las estrategias MQ2 y MQ4. Sin embargo, aún con el efecto de la ventana máxima gobernando las transferencias TCP, se consiguen ahorros en cuanto a ancho de banda en las fuentes rápidas estudiadas cuando se aplica un criterio de dimensionado del 95%.

6.7 Sincronización de TCP con OBS con distintos RTTs

Se ha visto en los apartados anteriores cómo, por el hecho de que todas las fuentes tuvieran el mismo retardo de transmisión en el acceso y el enlace OBS, todas las transmisiones tenían prácticamente el mismo RTT, y se producía una sincronización total a la hora de ensamblar todas las fuentes en una única cola. Sin embargo, es razonable

suponer que los flujos que compartan un enlace OBS provengan de accesos diferentes, e incluso tengan un destino final, fuera de la red OBS, diferente, con lo que los RTT serán distintos para cada flujo. En este caso, el comportamiento es ligeramente diferente, ya que, tras una pérdida múltiple, aunque todas las fuentes sincronicen su transmisión y comiencen a transmitir a la vez, no llegarán todos los segmentos a la vez al ensamblador, sino que se repartirán en instantes de tiempo diferentes.

Para comprobar el impacto de tener flujos con RTT base distinto con respecto a tener exactamente el mismo RTT base, se compara el conjunto de simulaciones realizadas en la sección 6.4 con 25 flujos medios con el mismo RTT base de 100 ms, y un nuevo conjunto de simulaciones en el mismo escenario, pero en el que el retardo de transmisión del segmento de acceso es distinto para cada flujo. En concreto, para el valor del retardo de los distintos accesos se han obtenido 25 valores con una variable aleatoria uniforme entre 7.5 y 12,5 ms, que se muestran recopilados en la Tabla 6-9. En dicha tabla cada número de acceso corresponde con el número de servidor y cliente TCP de la Figura 6-3, la Figura 6-4 y la Figura 6-13. El valor del retardo de transmisión se aplica tanto al tramo de acceso del servidor TCP como al tramo de acceso del cliente TCP, por lo que el RTT varía entre 90 ms y 110 ms. Todas las simulaciones cuyo resultado se muestra en este estudio se han realizado con los valores de la Tabla 6-9 para poder llevar a cabo una comparación justa entre los casos, y que el resultado no venga marcado por la aleatoriedad en la elección del retardo de transmisión concreto. Cabe mencionar que se han realizado las mismas simulaciones con otro conjunto de valores de RTT y se ha comprobado que el resultado es similar, por lo que se ha decido mostrar los resultados con un único conjunto de valores de retardos de transmisión.

Acceso de flujo	Retardo	Acceso de flujo	retardo
Acceso nº1	10,878223 ms	Acceso nº14	11,524359 ms
Acceso nº2	10,996067 ms	Acceso nº15	12,041988 ms
Acceso nº3	11,137546 ms	Acceso nº16	8,6594716 ms
Acceso nº4	9,8919219 ms	Acceso nº16	8,6965628 ms
Acceso nº5	10,274210 ms	Acceso nº18	7,7487724 ms
Acceso nº6	8,1052356 ms	Acceso nº19	7,8919204 ms
Acceso nº7	9,7537697 ms	Acceso nº20	10,704077 ms
Acceso nº8	11,079415 ms	Acceso nº21	8,4544329 ms
Acceso nº9	11,964208 ms	Acceso nº22	11,719347 ms
Acceso nº10	8,8655124 ms	Acceso nº23	8,3695012 ms
Acceso nº11	8,7738465 ms	Acceso nº 24	8,3539641 ms
Acceso nº12	11,828017 ms	Acceso nº 25	12,471477 ms
Acceso nº13	8,6617519 ms		

Tabla 6-9 Retardos del tramo de acceso de cada flujo

Con respecto a los parámetros de las simulaciones, al igual que en las secciones anteriores, la probabilidad de pérdida de ráfaga en un ensamblador se ha fijado a 10^{-3} y se ha empleado un tamaño máximo de ventana de 524288 bytes, con segmentos TCP de tamaño máximo 1452 bytes, y 40 bytes de cabeceras, que corresponden con 20 bytes cabecera IP y 20 bytes de cabecera TCP, sin cabeceras adicionales para el nivel de enlace.

Se han simulado los casos PF, MF, MQ2 y MQ4 para fuentes de tipo medio. Hay que recordar de los apartados anteriores que se ha escogido un temporizador de 1 ms para las fuentes de tipo medio. Los resultados en cuanto a *throughput* agregado medio, varianza del *throughput* agregado, número medio de segmentos TCP por ráfaga, índice de sincronización

y número se flujos diferentes por ráfaga, tanto con RTTs base fijos como con RTTs base variados, se han recopilado en la Tabla 6-10.

	Ensamblado PF de 25 flujos (T_b=1 ms)		Ensamblado MF de 25 flujos (T_b=1 ms)		Ensamblado MQ2 de 25 flujos(T_b=1 ms)		Ensamblado MQ4 de 25 flujos(T_b=1 ms)	
	RTTs fijos	RTTs varios	RTTs fijos	RTTs varios	RTTs fijos	RTTs varios	RTTs fijos	RTTs varios
Media del *throughput* agregado (Mbytes/s)	42,81	43,08	42,97	23,92	41,88	23,49	42,80	25,17
Varianza del *throughput* agregado $(Mbytes/s)^2$	13,03	13,82	357	29,12	136,57	17,12	76,91	10,52
Número medio de segmentos de un flujo TCP por ráfaga	8,35	8,42	8,39	2,58	8,37	2,51	8,37	2,88
Índice de sincronización	0,035	0,0346	1	0,2852	0,4349	0,1633	0,2224	0,1008
Número medio de flujos TCP diferentes por ráfaga	1	1	25	6,42	12,14	3,49	5,4	1,99

Tabla 6-10 Resumen de resultados con RTTs base fijos vs distintos, para PF, MF, MQ2 y MQ4

Análisis de la estrategia PF con flujos con RTTs base variados

En primer lugar se observa que en el caso de ensamblado PF, la media y la varianza obtenida con flujos con RTT base fijo (en este estudio se denomina RTT base al retardo debido a las contribuciones del retardo de acceso y el del enlace OBS, excluyendo el retardo variable introducido por el ensamblador OBS) es similar a la obtenida en el del caso de los RTT base diferentes para cada flujo, que se muestra en la Tabla 6-10.

Al ensamblarse cada flujo de manera independiente, el hecho de que los accesos varíen no tiene influencia en el comportamiento del *throughput* agregado. Las pérdidas de ráfaga no afectan a otros flujos. Además, como los RTT en media son de 100 ms, el resultado numérico en cuanto a *throughput* agregado medio es similar. El resto de parámetros analizados es similar. Por tanto, en la estrategia PF la variedad de RTTs no tiene influencia.

Análisis de la estrategia MF con flujos con RTTs base variados

La diferencia principal de comportamiento se tiene en el caso de la estrategia MF. En este caso, la media del *throughput* decae drásticamente desde 42,97 Mbytes/s del caso de RTT base fijo para todos los flujos hasta 23,92 Mbytes/s con RTT base variados. Al tener accesos con retardos de transmisión diferentes en vez de exactamente el mismo valor, las segmentos de las distintos flujos llegan al ensamblador en instantes de tiempo diferentes. Se observa además que el comportamiento es muy parecido al que ocurre al ensamblar un flujo con tráfico adicional, como se ha visto en la sección 4.4, y que daba como resultado que la transmisión de segmentos TCP se fragmentara en más ráfagas en vez de concentrarse en unas pocas ráfagas. Para comprobarlo hay que observar el número medio de segmentos por ráfaga. En la Tabla 6-10 se muestra que cómo en el caso de los retardos del tramo de acceso variados el número medio de segmentos de un flujo por ráfaga desciende a 2,58 segmentos, mucho menor que en el caso de los accesos con el mismo valor de retardo en los que dicho número medio de segmentos por ráfaga es 8,35. Esto se debe a que, al llegar los segmentos de los distintos flujos más espaciados al ensamblador, se generan ráfagas (más pequeñas) constantemente, mientras que en caso de los accesos fijos, se crean ráfagas más grandes que acumulan segmentos de todas las ráfagas. Para ayudar a visualizar las diferencias de comportamiento, en la Figura 6-22 se muestra el histograma del

número de segmentos de un flujo por ráfaga para el caso MF con RTTs base distintos, en el que se aprecia que las ráfagas contienen desde 1 hasta 9 segmentos de un mismo flujo, mientras que en el histograma del número de segmentos de un flujo por ráfaga para el caso MF con RTTs base iguales, mostrado en la Figura 6-23, la mayor parte de las ráfagas contiene 9 segmentos de un flujo TCP, el máximo posible, siendo muy pocas las ráfagas que contienen otro valor. De esta forma se aprecia la altísima concentración de segmentos en ráfagas en el caso de los accesos de igual retardo.

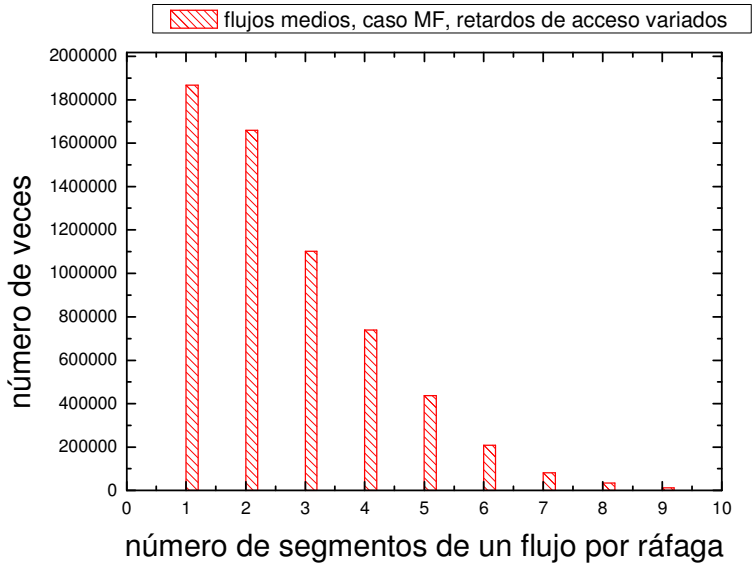


Figura 6-22 Histograma del número de segmentos de un flujo por ráfaga con retardos de acceso variados

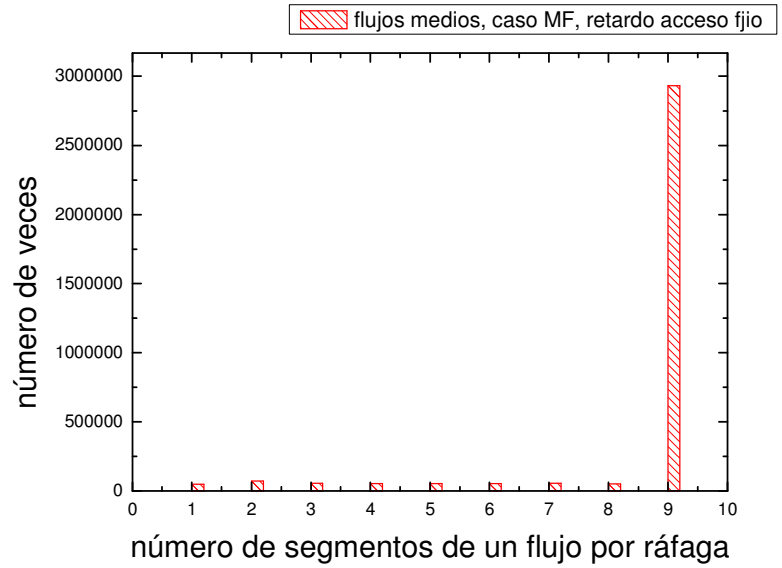


Figura 6-23 Histograma del número de segmentos de un flujo por ráfaga con retardos de acceso iguales

Como en el caso MF el número de segmentos de un flujo por ráfaga pasa de 8,35 en el caso del mismo RTT a 2,58 en el caso de RTTs variados, el *throughput* de los flujos TCP, como se había observado los capítulos anteriores al estudiar los flujos TCP con tráfico de fondo, desciende. Aplicando las averiguaciones del modelo periódico del Capítulo 5, en las que se calculaba el *throughput* medio de TCP en función la probabilidad de pérdida y del número de segmentos por ráfaga, se veía que el *throughput* era proporcional a la raíz cuadrada del número de segmentos del flujo TCP por ráfaga. Dividiendo la raíz de 8,35, entre la de 2,58, se obtiene 1,8, que es exactamente la división entre el *throughput* con mismo

RTT 42,97 y el *throughput* con RTTs variados, 23,92, que da 1.8. De esta forma, por un lado, se corrobora la aplicabilidad del modelo periódico y por otro lado se corrobora el impacto del ensamblado con flujos TCP adicionales, que se había comprobado principalmente con tráfico de fondo sintético.

Sin embargo, la diferencia principal en cuanto a sincronización se obtiene al observar el índice de sincronización, que desciende de 1, sincronización total, a 0,259, que nos indica que aproximadamente la cuarta parte de los flujos desciende su *throughput* en el intervalo en el que se produce una pérdida. Este índice está muy relacionado con el número de flujos diferentes por ráfaga. En la Figura 6-24 se muestra, para el caso de los RTTs variados el histograma normalizado del número de flujos diferentes por ráfaga. Se aprecia que hay una gran variedad de valores, por lo que los flujos se reparten entre las distintas ráfagas, y no todas las ráfagas llevan el mismo número de flujos. Esto contrasta con el caso de los RTTs fijos, en los que todas las ráfagas llevan 25 flujos, es decir, todas las ráfagas llevan segmentos de todos los flujos.

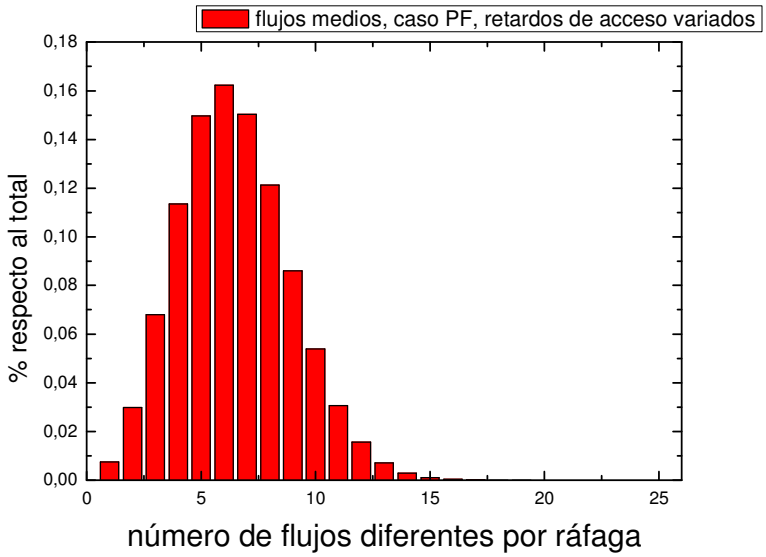


Figura 6-24 Histograma normalizado del número de flujos diferentes por ráfaga con RTTs variados

Análisis de la estrategia MQ con flujos con RTTs base variados

En el caso de la estrategia MQ, tanto con 2 colas como con 4 colas, se observa un comportamiento diferente en el caso de los RTT base variados. Por ejemplo, en la estrategia MQ2, la media del *throughput* decae drásticamente desde 41,88 Mbytes/s del caso de RTT base fijo para todos los flujos hasta 23,49 Mbytes/s con RTT base variados. Sin embargo, al igual que en el caso de los RTTs fijos, con los RTT base variados el *throughput* agregado medio es similar en las estrategias MF, MQ2 y MQ4, por lo que el comportamiento entre las distintas estrategias se mantiene. En cuanto a la varianza, esta desciende en los casos MQ2 y MQ4 en el caso de RTTs variados con respecto al caso de los RTTs fijos.

En concreto, según los datos de la Tabla 6-10 en la estrategia MQ2 se pasa de 136,57 $(\text{Mbytes/s})^2$ con RTTs fijos a 17,12 $(\text{Mbytes/s})^2$, descendiendo drásticamente. El mismo descenso ocurre en el caso de la estrategia MQ4 en la que se pasa de 76,91 $(\text{Mbytes/s})^2$ a 10,52 $(\text{Mbytes/s})^2$. Al igual que con el *throughput* agregado medio, la variación de varianza relativa del *throughput* agregado entre los casos MF, MQ2 y MQ4 se mantiene. En el caso de los RTTs base fijos, el descenso de varianza entre el caso MQ2 y el MF es de un 62% (pasa de 357 $(\text{Mbytes/s})^2$ a 136,57 $(\text{Mbytes/s})^2$), mientras que en caso de los RTT base variados, el descenso de varianza entre el caso MQ2 y el MF es de un 41% (pasa de 29,12 $(\text{Mbytes/s})^2$ a 17,12 $(\text{Mbytes/s})^2$). Para el caso MQ4, el descenso con respecto al caso MF,

con RTTs base fijos, es del 78% (pasa de 357 (Mbytes/s)^2 a 76,91 (Mbytes/s)^2), mientras que en caso de los RTT base variados, el descenso de varianza entre el caso MQ4 y el MF es de un 64% (pasa de 29,12 (Mbytes/s)^2 a 10,52 (Mbytes/s)^2). Por tanto la diferencia de varianza entre la estrategia MF y las estrategias MQ2 y MQ4 se mantiene tanto con RTTs fijos como variados, pero con un porcentaje de descenso menor en el caso de RTTs variados.

El índice de sincronización que se obtiene en el caso de MQ2 con RTTs variados es aproximadamente la mitad que el que se obtiene en el caso de los RTTs fijos, pasando de 0,4157 a 0,1967. En el caso de MQ4, el índice de sincronización también pasa a aproximadamente a la mitad, de 0,2459 a 0,1368.

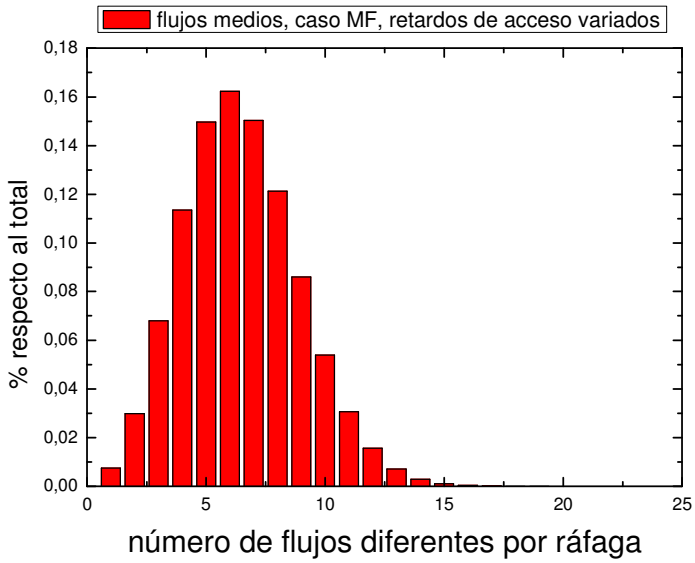


Figura 6-25 Histograma normalizado del número de flujos diferentes por ráfaga con RTTs variados para MQ2

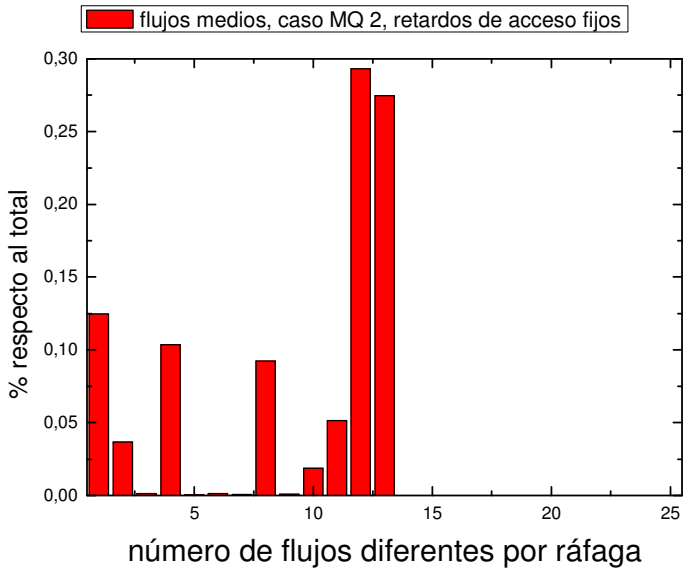


Figura 6-26 Histograma normalizado del número de flujos diferentes por ráfaga con RTTs fijos para MQ2

Como se ha mencionado anteriormente, es importante observar el número de flujos diferentes por ráfaga. En la Figura 6-25 se muestra, para el caso de los RTTs variados el histograma normalizado del número de flujos diferentes por ráfaga obtenido con la estrategia MQ2. Se aprecia que hay una gran variedad de valores, que contrasta con el histograma normalizado del número de flujos diferentes por ráfaga con RTTs fijos para el mismo caso MQ2, mostrado en la Figura 6-26, en el que hay dos picos claros para 12 y 13 segmentos por ráfaga (el máximo posible en cada una de las colas de ensamblado). Por

tanto, al igual que ocurría en el caso MF, los flujos se reparten entre las ráfagas en el caso de los RTTs variados. En el caso de MQ4 se aprecia un comportamiento similar, como se puede observar en los histogramas normalizados del número de flujos diferentes por ráfaga, mostrados en la Figura 6-27 para el caso de RTTs variados, y en la Figura 6-28 para RTTs fijos.

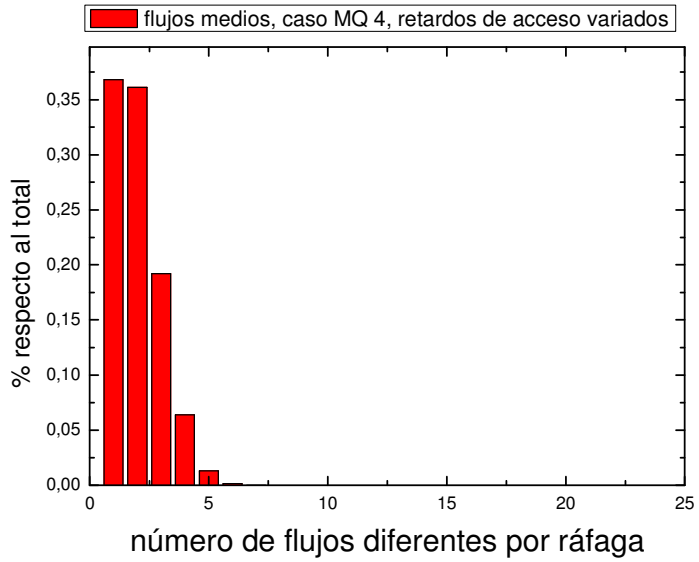


Figura 6-27 Histograma normalizado del número de flujos diferentes por ráfaga con RTTs variados para MQ4

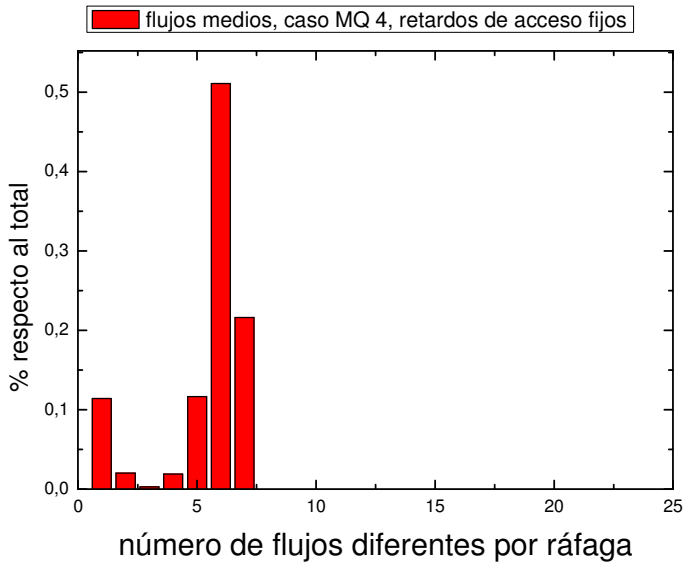


Figura 6-28 Histograma normalizado del número de flujos diferentes por ráfaga con RTTs fijos para MQ4

Análisis del sobredimensionamiento con RTTs base variados

A continuación se va a analizar en primer lugar la capacidad necesaria a proveer en el caso de los flujos medios con RTTs variados, para después obtener el sobredimensionamiento necesario. En la Figura 6-29 se muestra la distribución de probabilidad acumulada del *throughput* agregado en intervalos de 100 ms para cada una de las estrategias, PF, MF, MQ2 y MQ4 con los RTTs base variados. Esta figura se puede comparar con la Figura 6-16 que muestra dichas distribuciones para el caso de RTTs fijos. Como ya se indicó anteriormente, la estrategia PF se comporta de la misma manera con RTTs fijos y distintos. En cambio, el resto de casos está desplazado, principalmente debido al mayor número de ráfagas que hace falta para transmitir los mismos datos, lo que se traduce en una reducción del *throughput*.

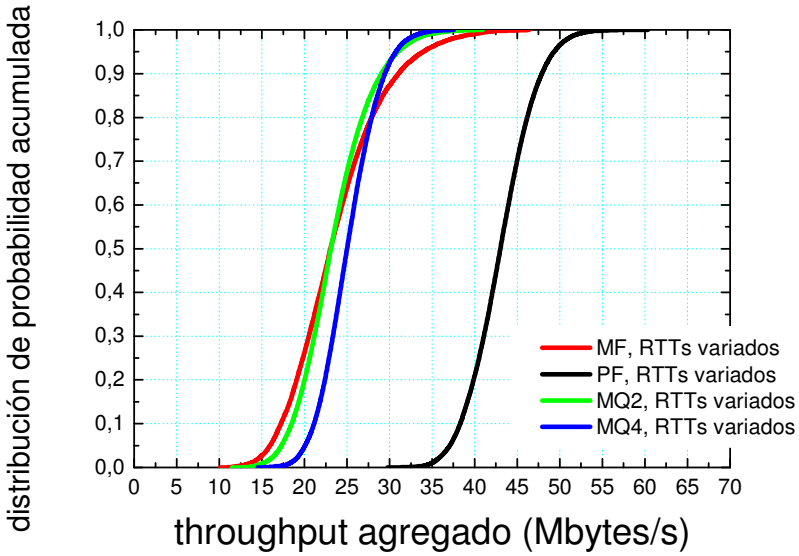


Figura 6-29 Distribución de probabilidad acumulada del *throughput* agregado con RTTs varios

En la Tabla 6-11 se muestra la capacidad necesaria para los flujos medios con un objetivo del 95% y del 99%, tanto con RTTs fijos como variados para todas las estrategias. En este caso, no se pueden comparar directamente los valores absolutos entre el caso de RTT fijo y variados para las estrategias MF, MQ2 y MQ4, ya que el *throughput* medio obtenido es diferente. Por ejemplo, en el caso MF, se obtiene 78,69 para $C_{95\%}$ con RTTs fijos y 33,79 para RTTs variados.

	Ensamblado PF de 25 flujos (T_b=1 ms)		Ensamblado MF de 25 flujos (T_b =1 ms)		Ensamblado MQ2 de 25 flujos (T_b=1 ms)		Ensamblado MQ4 de 25 flujos (T_b =1 ms)	
	RTTs fijo	RTTs varios	RTTs fijo	RTTs varios	RTTs fijo	RTTs varios	RTTs fijo	RTTs varios
$C_{95\%}$ (Mbytes/s)	48,83	49,31	78,69	33,79	62,64	30,99	57,94	30,74
$C_{99\%}$ (Mbytes/s)	51,32	52,02	97,06	39,72	72,06	34,58	64,29	33,35
$C_{95\%}$/ *throughput* medio	1,14	1,14	1,83	1,41	1,49	1,32	1,35	1,22
$C_{99\%}$/ *throughput* medio	1,2	1,2	2,26	1,66	1,72	1,47	1,50	1,32

Tabla 6-11 Capacidad necesaria para flujos medios con RTTs diferentes para PF, MF, MQ2 y MQ4

Por tanto, para poder comparar la capacidad necesaria cuando se consideran RTTs variados frente al caso en que son fijos, hay que normalizar por el *throughput* medio, para así obtener el porcentaje con respecto a la media de capacidad adicional que hay que provisionar en el enlace. El sobredimensionamiento se ha representado de manera gráfica en la Figura 6-30 para el objetivo del 95% y en la Figura 6-31 para el objetivo del 99%.

Por tanto, cuando se considera un escenario pragmático con RTTs variados en vez de RTTs fijos, la sincronización disminuye, pero no se elimina. En el caso de la estrategia MF, se necesita, en el caso más estricto del 99%, hasta un 66% de capacidad adicional con respecto a la media, que se baja al 47% con MQ2 y al 32% en el caso de MQ4.

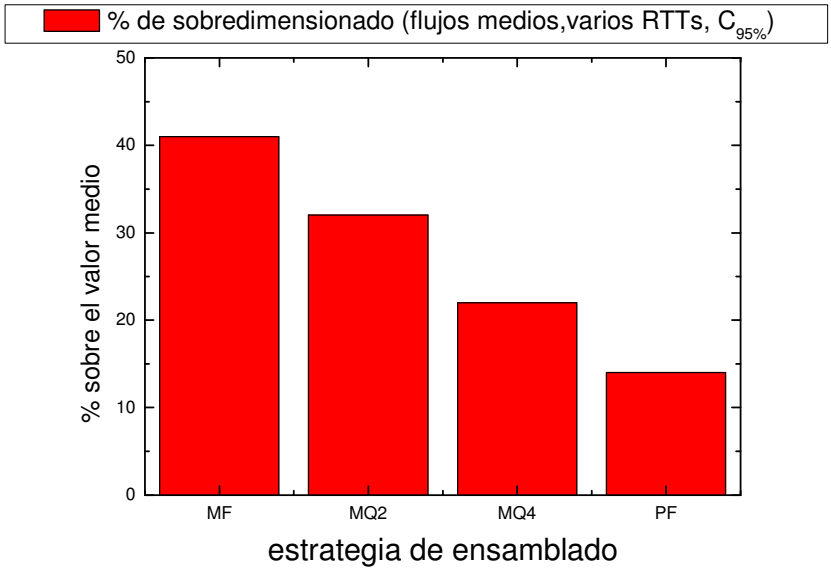


Figura 6-30 Sobredimensionado en % sobre la media para 25 flujos medios con RTT variados, con $C_{95\%}$

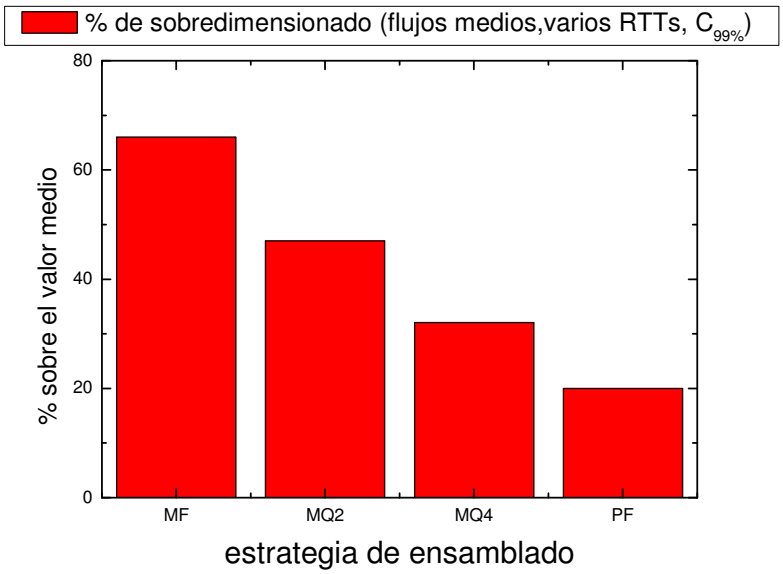


Figura 6-31 Sobredimensionado en % sobre la media para 25 flujos medios con RTT variados, con $C_{99\%}$

6.8 Resumen y conclusiones

En este capítulo de la tesis se ha estudiado el efecto de sincronización TCP, que se da en situaciones en las que varias conexiones TCP comparten un enlace y los mecanismos de control de congestión de los clientes y servidores TCP involucrados en la transmisión reaccionan al mismo tiempo. Este efecto hace que el perfil del tráfico presente una mayor varianza y que para obtener un mismo *throughput* medio, el ancho de banda que hay que provisionar en los enlaces sea mayor. Por tanto, cuanto mayor sea la sincronización, menos eficiente es el uso de los recursos disponibles.

En el caso de OBS se ha visto que el mencionado efecto de sincronización TCP aparece, ya que las ráfagas contienen segmentos de varios flujos, con lo que una pérdida de ráfaga implica la reducción de la tasa de transmisión de varias fuentes simultáneamente.

Cuando se tiene un único ensamblador de ráfagas compartido por varios flujos TCP las necesidades de ancho de banda son muy superiores al caso ideal, pero no realista, de que haya un ensamblador para cada flujo TCP. Para reducir las necesidades de

sobredimensionado en OBS se ha propuesto el empleo de varias colas de ensamblado. Mediante el estudio realizado, apoyado con simulaciones en OMNeT++, se ha constatado que con sólo pasar a 2 colas de ensamblado se reduce la varianza del tráfico y se consiguen ahorros de capacidad en el dimensionado. Si se aumenta a 4 colas, el sobredimensionado necesario se reduce aún más y se acerca al mejor comportamiento obtenido en el caso ideal de una cola por flujo.

A lo largo del capítulo se ha cuantificado el impacto de la sincronización en OBS tanto para un caso peor de sincronización total (en el que todos los flujos tienen el mismo RTT), como para un escenario pragmático, en el que los RTTs varían. Por ejemplo, en el escenario pragmático estudiado, como se ha visto en la sección 6.7, para un dimensionado en el que se garantice que el 99% del tiempo el *throughput* agregado es menor que el valor del ancho de banda a provisionar, con un temporizador de ensamblado de un milisegundo, pasa de un 60% de sobredimensionado en el caso de una cola de ensamblado a un 20% de sobredimensionado en el caso ideal de una cola por flujo. Con la estrategia MQ2 propuesta, que consiste en emplear dos colas, el sobredimensionado baja a un 47%. Si se utiliza la estrategia MQ4, en la que hay 4 colas de ensamblado, el sobredimensionado baja aún más, situándose en el 32% con respecto al tráfico medio, muy cerca del caso ideal.

Capítulo 7

Encaminamiento multicamino en OBS sin confirmación

El encaminamiento en OBS es el proceso por el cual se decide el conjunto de enlaces y nodos que empleará una ráfaga de datos para ir desde el nodo origen al nodo destino. En la alternativa de OBS con reserva de camino sin confirmación, la contienda de ráfagas que desemboca en pérdida de ráfaga es algo común. Un buen encaminamiento puede ayudar a repartir el tráfico en la red de tal forma que las contiendas se minimicen, reduciéndose así la probabilidad global de pérdida de ráfaga. En esta sección se realiza una propuesta de algoritmo multicamino que ayude a reducir la probabilidad de bloqueo en redes OBS con reserva de camino sin confirmación.

7.1 Introducción

La aproximación principal en OBS es realizar un encaminamiento basado en el camino más corto para minimizar el número de recursos empleados. El principal problema de las técnicas de encaminamiento que escogen un único camino fijo para un par origen destino es que saturan en exceso determinados enlaces, que pueden convertirse en cuellos de botella, mientras que otros enlaces de la red pueden estar desocupados. En estos enlaces saturados, la probabilidad de bloqueo de ráfagas es muy elevada, con lo que en global, se obtiene una probabilidad de bloqueo alta. Tal y como se mencionó en la sección 2.7, mediante formulaciones ILP, a partir de una demanda de tráfico conocida, Teng y Rouskas lograban obtener un conjunto de rutas óptimo [95]. De hecho, Teng y Rouskas demuestran que para conseguir la solución óptima hay que repartir las ráfagas que van de un mismo origen a un mismo destino entre varias rutas, es decir, hay que realizar encaminamiento multicamino.

Las técnicas de optimización proporcionan un camino óptimo a partir de una determinada distribución de tráfico conocida. Sin embargo, este dato no siempre es posible conocerlo, y, aún en caso de conocerlo, el tráfico real puede diferenciarse del previsto. Por ello, se necesitan técnicas de encaminamiento adaptativas que reaccionen ante las distintas situaciones en la red.

En la literatura se han propuesto varias técnicas adaptativas para OBS, que se han resumido en la sección 2.7, y que se mencionan brevemente a continuación. Hay algoritmos reactivos, que reaccionan ante situaciones en la red. Por ejemplo, Thodime *et al.* [99] proponen la técnica denominada *Congestion-Based Static-Route Calculation*. Esta técnica realiza encaminamiento de fuente, por lo que el nodo origen de las ráfagas es el único encargado de tomar las decisiones en cuanto a la ruta de la ráfaga. Se calcula de forma estática un conjunto de caminos disjuntos alternativos y se selecciona uno de ellos de forma dinámica en base a la información de congestión recolectada de los nodos del núcleo. En los nodos se toman medidas de carga de manera periódica en sus enlaces de salida. Si se supera un umbral de carga, se indica que ese nodo está congestionado y se disemina esta información.

A la hora de escoger el camino, si el camino más corto atraviesa algún enlace congestionado, se escoge el camino alternativo con menor número de tramos congestionados.

Klinkowski *et al.* [98] [92] proponen la técnica *Bypass Path* (BP), en la que se selecciona un camino en el nodo de origen del paquete como función del estado de sus colas de salida, por lo que emplea únicamente información local. La ruta sólo puede modificarse cuando el paquete se encuentra con un enlace saturado. En ese caso, el nodo intenta hacer un "bypass" de ese enlace saturado, utilizando un nodo intermedio para alcanzar al próximo salto.

Klinkowski *et al.* [98] también proponen la técnica *Distributed Path* (DP), con variantes con 1 o varias alternativas (DP-1, DP-*k*). Es una estrategia de encaminamiento adaptativa distribuida que emplea información global para la toma de las decisiones. En esta técnica, cada nodo pre-calcula los *k* LSPs más cortos hacia cualquier destino. Posteriormente, el algoritmo elige un camino entre todos ellos en función de un coste compuesto a base de tres parámetros, la media de ráfagas perdidas de cada enlace (ABL, *Average Burst Loss*), la utilización media del enlace (ALU, *Average Link Utilization*) y la ocupación media de *buffer* (ABO, *Average Buffer Occupancy*). Cada LSP de la red tiene diferente coste, calculado como la suma de las contribuciones de cada enlace que forma el camino. Para que los nodos tengan conocimiento de esta información de los demás, cada nodo mide el estado de sus parámetros (ABL, ALU y ABO) en un intervalo de tiempo determinado, y a continuación inunda la red con la información recolectada para que llegue a los demás nodos y estos puedan recalcular el coste de sus caminos.

Para evitar estos cuellos de botella y ayudar a reducir la probabilidad de bloqueo, en este capítulo se diseña una estrategia de encaminamiento heurística, distribuida, adaptativa, multicamino que se pueda realizar empleando únicamente información local del nodo. De esta forma, al utilizar información disponible en el propio equipo, se reducen los requisitos de intercambio de información con el resto de la red OBS. Para ello, se toma como punto de partida la estrategia multicamino para redes IP denominada MRDV (*Multi-path Routing with Dynamic Variance*) [241].

En este capítulo se realiza en primer lugar un recorrido por la técnica MRDV mencionada. A continuación se describe cómo se han extraído los principales conceptos y se ha diseñado el algoritmo adaptado a una red OBS con señalización en un único sentido, por ejemplo mediante JET. Finalmente se realiza una evaluación experimental del mismo en un escenario de red de tamaño nacional.

7.1.1 Algoritmo MRDV

El algoritmo MRDV [241] es una técnica de encaminamiento para redes IP que combina el encaminamiento multicamino con varianza y encaminamiento distribuido dinámico. El concepto clave del algoritmo MRDV es que cuando los caminos de menor coste están congestionados, se escogen caminos alternativos, de mayor coste, pero con menor tráfico. Mediante un parámetro, denominado varianza (no hay que confundir con la varianza matemática), que depende del tráfico en el enlace de salida, se decide en cada nodo si un camino es o no utilizable. Para el enlace de salida *j*, el parámetro varianza V_j, se calcula mediante la ecuación (7-1), node V_{max} es un parámetro fijado por el usuario que representa el valor máximo que puede alcanzar la varianza (cuando la carga está al 100%), ρ_j es la carga del enlace *j* y s un número real positivo que define la forma de la curva del valor de la varianza.

$$V_j = 1 + (V_{max} - 1) \cdot {\rho_j}^s \tag{7-1}$$

La carga ρ_j se define mediante la ecuación (7-2), donde T_m es la duración en segundos del intervalo de medida de la carga (fijado por el usuario), $x(T_m)$ es el número de bytes transmitidos en el enlace en el intervalo de medida T_m y B_{link} es la tasa binaria del enlace en bytes/s.

$$\rho_j = \frac{x(T_m)}{T_m B_{link}} \tag{7-2}$$

En cada nodo, se calculan los *k* caminos más cortos hacia un destino. Si M_i es la métrica del camino número *i* hacia el destino, V_{sp} el valor del parámetro varianza en el enlace de salida que utilice el camino más corto, y M_{min} la métrica del camino más corto, si se cumple que $M_i \leq M_{min} \cdot V_{sp}$, el camino es escogido como utilizable. Por tanto, para cada destino, el nodo tendrá una o varias posibilidades.

La técnica ECMP (*equal cost multipath,* múltiples caminos del mismo coste), por la que se reparte el tráfico entre todos los caminos del mismo coste, es un caso particular que se obtiene cuando el parámetro varianza *V* es igual a 1. En este caso, se pueden emplear todos los caminos que tengan el mismo coste.

El algoritmo MRDV ajusta la varianza de manera dinámica de acuerdo a la carga que detecta el *router* en el enlace al siguiente salto del camino considerado. Por tanto, hay una varianza diferente por cada interfaz de salida. De esta forma, el *router* ha de monitorizar continuamente sus interfaces, para actualizar el valor de la varianza. Para un determinado camino, se emplea la varianza de la interfaz correspondiente al enlace de salida del camino más corto.

Como se ha mencionado, según la varianza y el coste del camino, nuevos caminos se consideran adecuados. Una vez que se ha escogido qué caminos son adecuados para ir a un determinado destino, el algoritmo reparte la carga entre todos los caminos elegidos. La forma de repartir la carga es inversamente proporcional al coste del camino, por lo que a menor coste, mayor tráfico recibe ese camino.

De esta forma, el *router* reacciona con su propia visión de la red, que obtiene de sus interfaces de salida. Así, las decisiones de encaminamiento se basan en información local y no global. Para evitar inestabilidades, se añade un ciclo de histéresis en el cálculo de la varianza, de tal forma que cuando la carga sube, se emplea una curva, y cuando baja otra. El algoritmo MRDV modifica periódicamente la tabla de rutas del *router* con los criterios mencionados, por lo que, cada vez que llega un datagrama IP, se envía acorde a dicha tabla de rutas. La forma en la que el *router* distribuye los paquetes entre los distintos caminos depende de cómo lo maneje el *router* en sí mimo, y queda fuera del alcance del algoritmo.

Resumiendo, el algoritmo MRDV consigue balancear la carga de la red y reducir la congestión, mejorándose el *jitter* y el retardo de los datagramas IP en la red debido a un mejor uso de las colas.

7.2 Diseño del algoritmo de encaminamiento distribuido adaptativo multicamino

Tal y como se ha mencionado, los estudios de optimización muestran que, en una red OBS, si se consigue balancear adecuadamente la carga en la red, se puede obtener un mejor comportamiento en la planificación de las ráfagas y conseguir reducir la probabilidad de pérdida de ráfaga global. De esta forma, se emplean los conceptos básicos del encaminamiento multicamino con varianza MRDV que se acaba de explicar para diseñar una estrategia de encaminamiento multicamino para OBS.

7.2.1 Hipótesis de partida

En las hipótesis de partida se asume una red OBS con reserva sin confirmación tipo JET o con la reserva salto a salto, tipo *offset* emulado. Se realiza además la suposición de que los nodos tendrán implementado un protocolo de estado de enlace como OSPF o IS-IS para el intercambio de información de encaminamiento, por lo que son capaces de construir de manera autónoma un grafo de la red que se puede emplear en el algoritmo de encaminamiento. Con respecto a las características del nodo OBS, no se asume ninguna arquitectura específica, pudiendo haber o no FDLs, así como tampoco se asume ningún algoritmo de planificación de ráfagas concreto.

7.2.2 MRDV directamente sobre OBS

Una primera aproximación propuesta por Aracil *et al.* [242] consiste en aplicar directamente MRDV en cada nodo, y emplear la información de la tabla de rutas para escoger el enlace de salida a la hora de encaminar las ráfagas. En el estudio de Aracil *et al.* se evalúa la estabilidad del reparto de carga entre los enlaces alternativos. Se proponen dos políticas de reparto, una en la que se reparte de manera inversamente proporcional al tráfico en cada enlace, y otra en la que se reparte de manera inversamente proporcional a la probabilidad de bloqueo. En dicho estudio se ha descubierto que si se aplica directamente MRDV para OBS, el resultado puede ser muy inestable. Esto se debe a que no se puede emplear directamente la carga o la probabilidad de bloqueo en la función de MRDV para escoger los caminos, ya que no está optimizado para las pérdidas de ráfagas OBS, sino para el retardo en una cola. El comportamiento que se observa es que el tráfico en los nodos fluctúa de unos enlaces a otros, provocando inestabilidades. Entre las métricas de reparto estudiadas, el empleo de la carga genera menos inestabilidades. Por tanto, es necesario realizar un rediseño del algoritmo para tener en cuenta las particularidades de OBS y obtener un algoritmo estable.

7.2.3 Diseño del algoritmo para una red OBS

Las redes de conmutación óptica de ráfagas tienen muchas particularidades, que se han detallado en el primer capítulo de esta tesis. Por ello, para diseñar un algoritmo de encaminamiento especialmente adaptado para una red OBS, se va a realizar un recorrido por cada uno de los principios de diseño y se va a justificar las distintas elecciones. Una vez presentadas y justificadas las decisiones de diseño, se procede a describir el detalle del algoritmo concreto.

Algoritmo adaptativo al tráfico

La distribución del tráfico en una red evoluciona con el tiempo. Hay varios marcos temporales diferentes, las evoluciones a medio-largo plazo, motivadas por cambios en el patrón de uso de los usuarios, las evoluciones diarias, motivadas por la movilidad de los usuarios (tráfico en la oficina de día, tráfico de vídeo por la noche, etc.) y las evoluciones a muy corto plazo. Una red se diseña con unas predicciones de patrón de distribución de tráfico determinadas. Las evoluciones de medio plazo pueden provocar que el tráfico en la red no se ajuste a las previsiones, con lo que hay dos alternativas, o redistribuir físicamente los recursos (complicado y costoso), añadir nuevos recursos donde hagan falta (gasto importante) o bien el encaminamiento se adapta de manera automática y es capaz de distribuir el tráfico de una manera más eficiente. Una técnica adaptativa es capaz de ajustarse a cambios en las distribuciones de tráfico. La técnica propuesta se centra en detectar estos cambios de tráfico prolongados (en intervalos grandes, en el orden de decenas de minutos) y ajustarse a ellos. La técnica no pretende ajustarse a cambios en escalas de tiempo muy pequeñas (por debajo de minutos) ya que crearía inestabilidades en

la red difíciles de controlar. Por tanto, el primer criterio de diseño es que la técnica se adapte a modificaciones del tráfico en escalas de tiempo de decenas de minutos.

Algoritmo multicamino

Si para cada par origen-destino se tiene que enviar todo el tráfico por el mismo camino, el reparto global de la carga es sub-óptimo. Para poder adaptarse a un cambio en el patrón de tráfico de la mejor manera, es necesario poder distribuir flujos en varias rutas. En los estudios de optimización de rutas en OBS de Teng y Rouskas se demuestra que se obtiene un mejor resultado empleado múltiples caminos por par origen destino y repartiendo la carga entre ellos frente a emplear un único camino por par [95].

A pesar de las ventajas de emplear múltiples caminos, hay que tener cuidado con la reacción de los protocolos de capas superiores con el multicamino [240]. Por ejemplo, aunque TCP se encarga de ordenar los segmentos de un flujo, reacciona ante los segmentos en desorden como si se tratara de pérdidas, ajustando su ventana. Para evitarlo, o bien se ordenan las ráfagas a la salida de la red OBS, o bien se incluyen mecanismos para que haya un reparto de carga a nivel de flujo y no de segmento. De esta forma, aunque el reparto de carga no es perfecto, el impacto en capas superiores se minimiza.

Algoritmo distribuido y empleando información local

El conjunto de rutas óptimo solamente puede ser conocido cuando se conoce con precisión el tráfico de la red. Sabiendo con precisión esta información se pueden utilizar técnicas como la propuesta por Teng y Rouskas [95], en las que un elemento central calcula el conjunto rutas a partir de la demanda. Para poder obtener esta información es necesario el intercambio periódico de medidas de tráfico. Esta alternativa no es siempre posible ya que se corre el riesgo de saturar el canal de control, y es difícil determinar el intervalo óptimo de envío de información. Asimismo se corre el riesgo de contar con información desactualizada. Por tanto, se sigue la aproximación pragmática de MRDV y se asume que cada nodo sólo puede acceder a estadísticas locales, que para un nodo OBS pueden ser el número y tamaño de las ráfagas que circulan por el nodo, la probabilidad de bloqueo, etc.

El cálculo se va a realizar de manera distribuida para que cada nodo pueda emplear la información local en la toma de decisiones. Al ser distribuido, y poder tomarse decisiones localmente, si se emplea un protocolo como JET, hará falta dar un *offset* extra para posibles rutas más largas. Esto se debe a que el *offset* es el tiempo que transcurre desde que se envía el mensaje de control de la ráfaga hasta que se envía la misma. Con un encaminamiento de fuente, el camino es conocido desde el principio, con lo que el tiempo de *offset* es conocido, pero encaminando salto a salto, el tiempo puede variar. Si se emplean arquitecturas como la propuesta por Klinkowski *et al.* en las que el *offset* se da nodo a nodo [56], no existe este problema de que el *offset* que se ha indicado inicialmente no sea suficiente. Otra alternativa es disponer de FDLs en determinados nodos solo para el caso de que el *offset* se agote por tomar caminos alternativos.

Algoritmo con histéresis

Uno de los criterios de partida es que el algoritmo sea estable, para lo cual se ha decidido que se base en medidas de tráfico en periodos relativamente largos, de decenas de minutos, para tomar la decisión de las rutas a utilizar. A pesar de ello, el tráfico puede oscilar en un umbral, con lo que se podría llegar a una situación en que las rutas cambien cada 15 minutos, por ejemplo, con un tráfico relativamente similar. Para evitar estas oscilaciones, se propone seguir la estrategia de MRDV de emplear un mecanismo de histéresis en el cálculo de las rutas. Para ello, el algoritmo se divide en dos partes, en una de ellas se calcula cuál es el conjunto de rutas que se pueden utilizar dado el nivel de carga (o probabilidad de bloqueo), y en otra cómo se reparte la carga entre estos caminos. Las rutas

se activarán o desactivarán en función de la carga o probabilidad de bloqueo, con un umbral cuando se incrementa la carga, y otro umbral cuando decrece. De esta forma, se consigue una estabilidad en los caminos utilizables.

A modo de resumen, la Tabla 7-1 sintetiza las principales decisiones de diseño del algoritmo propuesto, para el cual se ha escogido el acrónimo AMOR (*Adaptive Multipath OBS Routing*).

Criterio de diseño	Información adicional
Adaptativo al tráfico	Intervalos de cambios, en decenas de segundos
Multicamino	Cuidado con el impacto en capas superiores
Balanceo de carga	Función a partir de carga y probabilidad de bloqueo
Histéresis	Umbral de subida para activar rutas y umbral de bajada para desactivarlas.

Tabla 7-1 Criterios de diseño del algoritmo AMOR

7.2.4 Funcionamiento detallado del algoritmo AMOR

En primer lugar, el algoritmo adaptativo para OBS va a calcular en cada nodo un conjunto de caminos siguiendo una métrica independiente de la carga (ej. número de saltos). Se denomina *C* al coste del camino según la métrica escogida. Así, cada camino *i* tendrá un coste C_i. Localmente, a partir de la información de la probabilidad de bloqueo de ráfagas experimentada en dicho enlace, que depende de las particularidades del nodo y de la carga, se calcula un coste local de cada camino, que se denominará CL_i. Empleando este coste local por camino CL_i (dependiente de la carga), se escoge, salto a salto, caminos alternativos al de menor coste (coste *C*, independiente de la carga).

Funcionamiento del algoritmo paso a paso:

1. Cada nodo calcula el conjunto de rutas de menor coste.
2. Inicialmente, cada nodo emplea la ruta calculada de menor coste para encaminar las ráfagas. En el caso de que haya varias rutas de menor coste, el tráfico se reparte el tráfico a partes iguales entre todas ellas en vez de escogerse una única ruta.
3. Periódicamente se monitoriza el bloqueo de ráfagas en los distintos enlaces de salida.
4. Después de cada monitorización, se calcula un coste local para cada una de las rutas calculadas inicialmente.
5. Tras realizar el cálculo de costes, se eligen los posibles enlaces de salida por los que se pueden encaminar las ráfagas. La elección de un enlace de salida como apto para ser una alternativa al enlace principal se realiza en función de los costes locales calculados de las dos rutas, por lo que una ruta *i* es apta para ser utilizada como alternativa del camino principal *p* si, siendo CL_i el coste local del camino alternativo *i* y siendo CL_p el coste local del camino principal:

$$CL_i < CL_P \tag{7-3}$$

6. La cantidad de carga a repartir por los caminos secundarios dependerá del número y velocidad de las longitudes de onda disponibles en el enlace de salida.

A continuación se describen los parámetros del algoritmo que obtienen un resultado adecuado con OBS, que se han obtenido tras realizar simulaciones con distintas alternativas.

Métrica de los caminos

La métrica de los caminos se calcula de la manera habitual. Por ejemplo, se puede emplear el número de saltos, la distancia o un coste administrativo. Como OBS es muy sensible a las pérdidas de ráfaga en cada salto, se recomienda escoger la métrica del camino de menor número de saltos. La métrica del camino es independiente de la carga o probabilidad de bloqueo. De esta forma, se garantiza una estabilidad de la métrica.

Definición de la función de coste local

Una vez calculados las métricas de los caminos, se calcula un coste local de los mismos. En redes IP, MRDV emplea el nivel de carga del enlace para determinar qué enlaces secundarios son utilizables. En OBS, aunque la carga está directamente relacionada con la probabilidad de bloqueo, una medida de carga no equivale a una medida de probabilidad de bloque. La probabilidad de bloqueo depende del tipo de nodo (con o sin FDL, por ejemplo) y del perfil de tráfico. Por ello, en esta estrategia de encaminamiento se ha elegido la probabilidad de bloqueo del enlace como uno de los parámetros a considerar para calcular el coste del enlace en vez de la carga. De esta forma se reflejan las particularidades de cada nodo, y cómo impactan en OBS. Dependiendo de las capacidades del nodo, se emplea una estimación de la probabilidad de pérdida o una medida real de la misma.

El coste local propuesto C_i (mostrado en la ecuación (7-4)) consta de dos parámetros, la métrica, que como se ha mencionado, se ha escogido que sea el número de saltos (enlaces) desde el nodo actual hasta el destino del camino que sale por el enlace *i*, que se denomina N_i, y un peso W_i (mostrado en la ecuación (7-5)) que tiene en cuenta la estimación (o medida si es posible) de la probabilidad de pérdida de ráfaga en el enlace de salida del camino considerado, que se denomina pb_i, una probabilidad de pérdida de ráfaga mínima (es decir, la probabilidad de pérdida mínima que se considere adecuada para nuestra red en condiciones de baja carga), que se denomina pb_{min}, el peso máximo $Wmax$, y un parámetro *k*, que define la forma de la curva

$$C_i = W_i N_i \tag{7-4}$$

$$W_i = 1 + (Wmax - 1)\left(\frac{\log(pb_i) - \log(pb_{min})}{\log(pb_{min})}\right)^k \tag{7-5}$$

Se introduce el ciclo de histéresis como se ha comentado anteriormente para prevenir inestabilidades en la función de coste local y que haya cambios excesivos de tráfico, para lo cual se utiliza el parámetro *k* multiplicado por -1, como se observa en la ecuación (7-6) que define el peso $Wbajada_i$ para el coste del camino por el enlace *i* cuando desciende la probabilidad de bloqueo en dicho enlace

$$Wbajada_i = 1 + (Wmax - 1)\left(\frac{\log(pb_i) - \log(pb_{min})}{\log(pb_{min})}\right)^{-k} \tag{7-6}$$

Para ver un ejemplo de la función de carga, en la sección de este capítulo donde se realiza la evaluación experimental del algoritmo (sección 0), se muestra en la Figura 7-2 dicha función de carga particularizada para los parámetros empleados en las simulaciones.

7.2.5 Reparto de carga

Una vez se ha elegido qué caminos son utilizables en un determinado instante para un destino hay que decidir cuánta carga (ráfagas) se envían por las distintas rutas. El cálculo del reparto de carga ha de tener en cuenta, por un lado, el número de saltos de los caminos alternativos, para evitar cargar en exceso otros nodos de la red y producir el efecto inverso. Por otro lado, ha de tener en cuenta la carga en cada enlace, para evitar saturar un enlace que ya tiene mucho tráfico. Es decir, aunque se haya escogido un camino alternativo como

utilizable, si este tiene mucho tráfico, solamente se desvia a él una pequeña cantidad de tráfico.Hay que destacar que este reparto de carga es una mejora con respecto a MRDV, en el que las proporciones de reparto de carga son fijas, por lo que, aunque un enlace de salida esté cargado, si ha sido elegido, se envía tráfico en el independientemente de su carga. El proceso de reparto detallado de carga se explica a continuación:

Paso 1: Reparto inicial

Una vez escogidos los caminos utilizables, y no se ha realizado anteriormente un reparto de carga, se realiza un reparto inicial de manera inversamente proporcional al número de saltos de cada camino. Por tanto, denominando P_i al portentaje de tráfico del camino *i*, N_i el número de saltos del camino *i*, N_j el número de saltos de camino *j*, y *d* el número total de caminos utilizables:

$$P_i = \frac{1/N_i}{\sum_{j=1}^{j=d} 1/N_j} \tag{7-7}$$

Paso 2: Reparto progresivo

Una vez que se ha realizado el reparto incial, tras un cierto tiempo de funcionamiento, cambia el estado de la red. Cuando el algoritmo entra en un nuevo ciclo, obtiene las nuevas medidas, y calcula los costes. A partir de estos costes, decide qué enlaces son utizables, como se ha mencionado anteriormente. Ahora, el algoritmo realizará únicamente unas ligeras modificaciones con respecto al paso anterior para evitar oscilaciones bruscas del tráfico. Para ello:

- Si el camino se sigue utizando, su coste local es menor que 0,8 veces el coste local del camino más corto y la carga en el enlace por el que sale el camino es menor que la carga del enlace por el que sale el camino más corto, se propone para recibir un incremento de tráfico.
- Si el camino se sigue utilizando, su coste local es mayor que 1,2 veces el coste local del camino más corto y la carga en el enlace por el que sale el camino es mayor que la carga del enlace por el que sale el camino más corto, se propone para recibir una bajada de tráfico.
- Si el camino es nuevo, se propone para recibir tráfico.
- Si el camino desaparece, se elimina la totalidad del tráfico por dicho enlace (ya que se entiende que si no se puede usar, es mejor hacerlo de inmediato por riesgo de saturación).

A partir de la distribución anterior, con unos coeficientes P_i, se calculan unos nuevos coeficientes. En primer lugar, si desaparece un camino, se reparte su carga de manera proporcional entre el resto de caminos. Luego, para los enlaces que suben/bajan carga, la diferencia entre el nuevo coeficiente y el anterior (ΔP_i) se define en la ecuación (7-8), donde $\rho_{principal}$ es el nivel de carga en el enlace por donde sale el camino principal (el de menor coste), ρ_i, el nivel de carga del enlace por donde sale el camino al que se va a repartir la carga, $N_{principal}$ el número de saltos del camino principal y N_i el número de saltos del camino al que se va a repartir la carga.

$$\Delta P_i = \frac{\rho_{principal} - \rho_i}{N_{principal} + N_i} \tag{7-8}$$

La carga que se utiliza en el algoritmo AMOR (ρ) se define mediante la ecuación (7-9), donde donde T_m es la duración en segundos del intervalo de medida de la carga (fijado por el usuario), $x_{burst}(T_m)$ es la suma del tamaño en bytes de todas las ráfagas que se han

transmitido en el enlace (en todas las longitudes de onda disponibles) donde se media la carga durante el intervalo de medida T_m y B_{lambda}es la tasa binaria de cada longitud de onda en bytes/s y L el número de longitudes de onda en el enlace.

$$\rho_j = \frac{x_{burst}(T_m)}{T_m \cdot L \cdot B_{lambda}} \tag{7-9}$$

Finalmente, se normalizan los coeficientes obtenidos, para repartir toda la carga. De esta forma, la carga se va moviendo progresivamente entre los enlaces, hasta alcanzar una situación de equilibrio. En el siguiente apartado, se analiza mediante simulación el comportamiento del algoritmo.

7.3 Análisis mediante simulación

El objetivo de este apartado es realizar una validación funcional y una evaluación de prestaciones mediante simulación del algoritmo AMOR propuesto. En concreto, se compara con el algoritmo del camino más corto, que se abrevia en este apartado como SP y ECMP (*Equal Cost MultiPath*) [243], por la que se reparten las ráfagas entre todos los caminos más cortos. La comparación se realiza en cuanto a probabilidad de bloqueo de ráfaga, número medio de saltos y retardo medio. Finalmente se analiza el reparto de la carga en la red.

7.3.1 Escenario de simulación

Las simulaciones se han realizado empleando la herramienta ns-2 [244], que se ha modificado para simular nodos OBS. Para realizar las simulaciones se ha escogido la topología de 17 nodos de la red alemana de referencia empleada por Hülsermann y Jäger [245] y mostrada en la Figura 7-1, empleando unas distancias a partir de las distancias por carretera entre las ciudades, que se han recopilado en la Tabla 7-2. El escenario se ha escalado en cuanto a tasa binaria en los enlaces y número de longitudes de onda para poder realizar las simulaciones. De esta forma, se han escogido 8 longitudes de onda por enlace, con 100 Mbps por cada canal. En cada uno de los nodos se ha simulado un nodo OBS con capacidades de nodo de borde, al tener capacidad de ensamblar y desensamblar ráfagas, y capacidades de nodo de transporte, al tener capacidad de conmutar y planificar ráfagas. Los nodos se han equipado con 4 *buffers* basados en FDL, con una granularidad equivalente al tamaño medio de ráfaga.

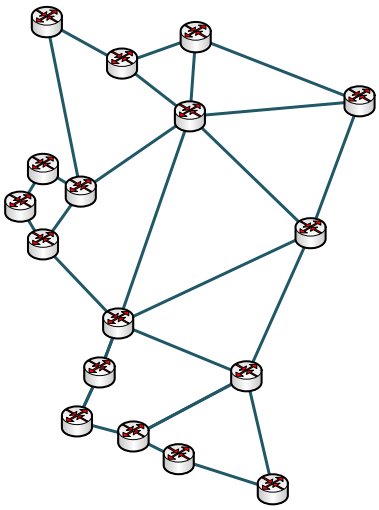


Figura 7-1 Topología de 17 nodos de la red de referencia alemana

Enlace	Distancia(Km)	Enlace	Distancia (Km)
Hamburg-Berlin	293	Hamburg-Hannove	158
Hamburg-Bremen	123	Bremen-Norden	161
Bremen-Hannover	131	Berlin-Hannover	288
Berlin-Leipzig	192	Hannover-Leipzig	265
Hannover-Frankfurt	352	Hannover-Dortmund	214
Dormund-Norden	303	Dortmund-Essen	37
Essen-Düsseldorf	37	Düsseldorf-Köln	38
Köln-Dortmund	96	Köln-Frankfurt	193
Frankfurt-Leipzig	386	Frankfurt-Nürnberg	228
Frankfurt-Mannheim	79	Mannheim-Karlsruhe	67
Karlsruhe-Stuttgart	79	Stuttgart-Ulm	92
Ulm-München	154	München-Nürnberg	169
Nürnberg-Stuttgart	208		

Tabla 7-2 Distancias de red referencia a partir de distancias de carretera

Para simular el tráfico, los nodos generan las ráfagas directamente. Es decir, no se realiza una agregación de tráfico. El proceso de generación de ráfagas es un proceso de Poisson, por lo que el tiempo entre envíos de ráfagas y el tamaño de las ráfagas siguen una distribución exponencial negativa. El tamaño medio de ráfaga se ha fijado para todas las simulaciones en 4000 bytes. Se ha empleado una matriz de tráfico uniforme, por lo que todos los nodos envían tráfico al resto de los nodos con el mismo tiempo entre ráfagas y tamaño medio de ráfagas. En las simulaciones, se ha ido variando el tiempo entre ráfagas. Así, cada valor de tiempo entre ráfagas empleado representa una carga de la red.

A modo de resumen, la Tabla 7-3 recopila los principales parámetros empleados en las simulaciones con respecto a los nodos OBS.

Parámetro	Valor
Tamaño medio de ráfaga	4000 bytes
Retardo básico de FDL	320 μs
Número de FDLs disponibles por canal	4
Tiempo de procesamiento del BCP	2,5 μs
Tiempo de reconfiguración de la matriz óptica	1 μs
Número de longitudes de onda por enlace	8
Ancho de banda de cada longitud de onda	100 Mbps

Tabla 7-3 Resumen de parámetros de la simulación

Con respecto a los parámetros del algoritmo AMOR, en la Figura 7-2 se muestra la función del peso que se emplea para calcular el coste local para distintas probabilidades de bloqueo, tanto en su tramo ascendente, como descendente, para los valores escogidos para las simulaciones, k=3, W_{max} =3 y una probabilidad de bloqueo mínima de 10^{-6}. Se han escogido estos valores tras realizar divesar pruebas con varias combinaciones de parámetros y obtener con ellos el mejor resultado.

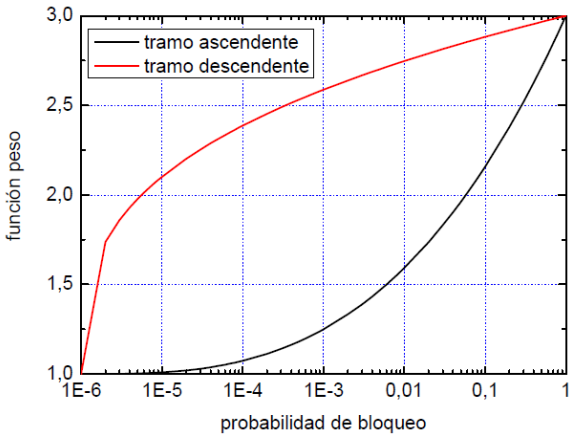


Figura 7-2 Función de peso con k = 3, W_{max} = 3, pb_{min}=10^{-6} mostrando el ciclo de histéresis.

7.3.2 Estudio de la probabilidad de bloqueo

El primer escenario analizado consiste en la red mostrada en la Figura 7-1 en la que los nodos OBS no están equipados con FDLs. En esta red se han comparado tres estrategias, SP (camino más corto), ECMP (múltiples caminos del mismo coste) y el algoritmo que se ha diseñado, AMOR. En primer lugar, se comienza comparando el rendimiento en cuanto a probabilidad de bloqueo de ráfaga. En el caso simulado de nodos sin FDLs, como se muestra en la Figura 7-3, se aprecia que las diferencias entre las distintas estragias son pequeñas, con ligeramente mejor rendimiento en el caso del algoritmo propuesto.

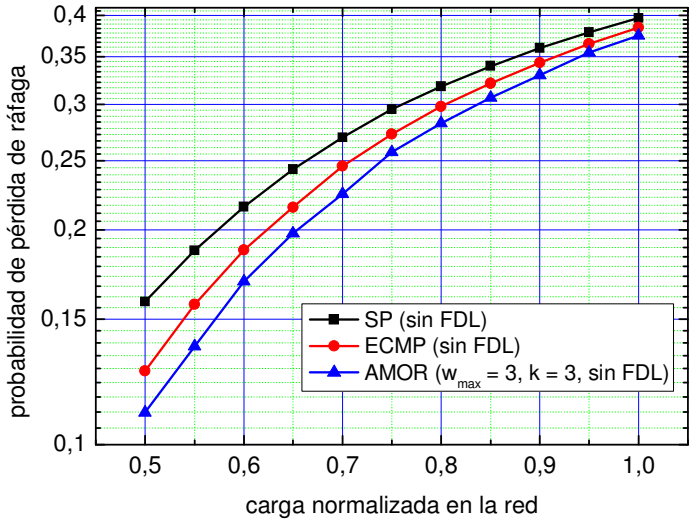


Figura 7-3 Probabilidad de pérdida de ráfaga en caso sin FDLs

A continuación, se estudia un escenario en el que los nodos se han equipado con FDLs. En este caso, como se ve en la Figura 7-4, el algoritmo propuesto consigue bajar, en el caso de una carga de 0,5 por debajo del 5%, la mitad que la estrategia del camino más corto. Se observa también que el simple hecho de repartir la carga entre los caminos del mismo coste (ECMP) mejora con respecto escoger un único camino más corto. Por tanto, con FDLs, el hecho de repartir la carga es beneficioso, tanto con ECMP, como con el algoritmo propuesto, en el escenario analizado en cuanto a probabilidad de bloqueo. Esto se debe a que la ganancia por quitar tráfico de los enlaces más cargados compensa al aumento de bloqueo en los menos cargados.

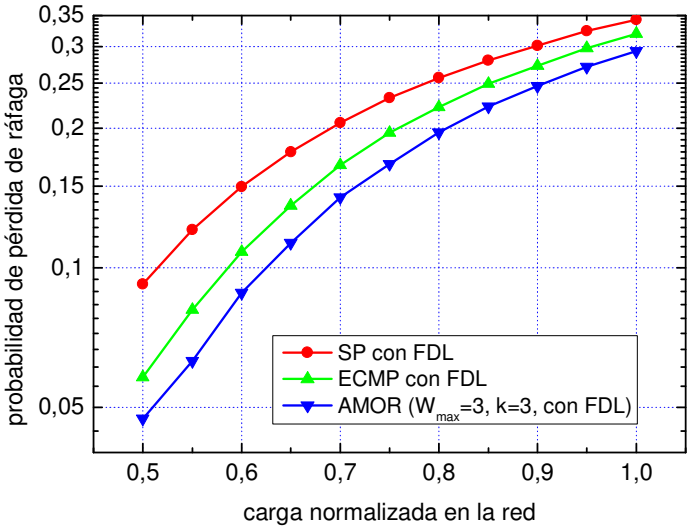


Figura 7-4 Probabilidad de pérdida de ráfaga en caso con FDLs

7.3.3 Análisis del número de saltos y retardo

En el apartado anterior se ha visto que las estrategias multicamino ECMP y AMOR se comportan bien en términos de probabilidad de bloqueo. Los caminos de ECMP (con métrica de coste el número de saltos) para un par origen-destino tienen el mismo número de saltos que el camino escogido con SP. Sin embargo, el número medio de saltos, como se ve en la Figura 5-1 para el caso de nodos sin FDLs, es ligeramente superior en ECMP que en SP. Esto se debe a que, por un lado, como en ECMP se está obteniendo una menor probabilidad de bloqueo, más ráfagas llegan al destino. Por cuantos más nodos pasa una ráfaga, mayor es la probabilidad de que dicha ráfaga se bloquee. Por ello, cuando la probabilidad de bloqueo es menor, el número de enlaces recorridos por ráfaga aumenta. El descenso del número medio a medida que aumenta la carga es lineal en SP y ECMP, en AMOR hay un comportamiento a escalones (debido al comportamiento explicado del algoritmo para habilitar/deshabilitar caminos).

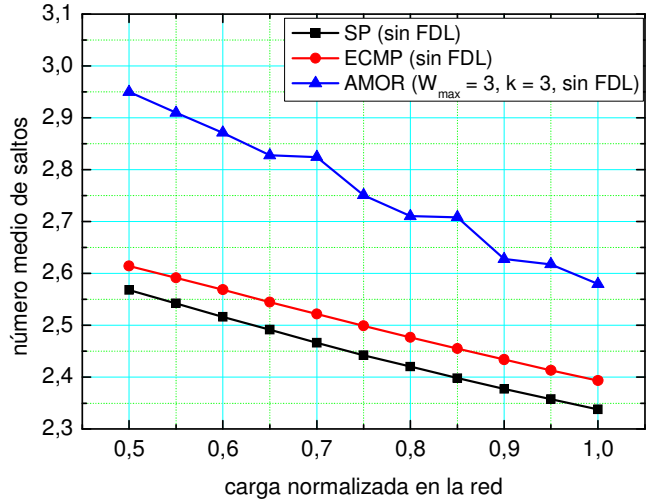


Figura 7-5 Número medio de saltos por ráfaga, caso sin FDL

En el caso del algoritmo propuesto AMOR, además, en algunas ocasiones se emplean rutas más largas, por lo que el número medio de saltos sube con respecto a ECMP (por ejemplo, para carga 0,5 sube de 2,55 hasta 2,95, es decir, un 16%. Como se ha mencionado, como las ráfagas que recorren rutas más largas tienen más posibilidades de perderse, a menor carga, las probabilidades de que al pasar por un nodo se bloquee una ráfaga se reducen, y las ráfagas realizan en media un mayor número de saltos.

Al repertir las simulaciones con nodos con FDL se obtiene un número medio de saltos ligeramente mayor en que en el caso sin FDL, como se aprecia en la Figura 7-6. La razón de esta ligera subida del número medio de saltos por ráfaga es que las ráfagas que recorren mayor número de saltos son las más beneficiadas de la menor probabilidad de bloqueo por nodo.

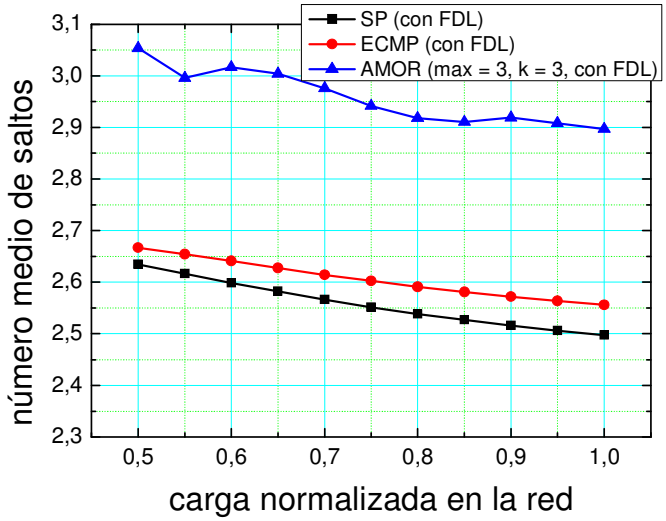


Figura 7-6 Número medio de saltos por ráfaga, caso con FDL

Una de las consecuencias de escoger caminos más largos y aumentar el número de saltos es el aumento del retardo. Sin embargo, hay que recordar que el encaminamiento SP se ha realizado empleando como métrica el menor número de saltos, lo que no implica el menor retardo, ya que cada enlace tiene un retardo de propagación diferente. El retardo de propagación extremo a extremo obtenido para el caso sin FDL puede observarse en la Figura 7-7, donde se muestra que el comportamiento es análogo al número medio de saltos por ráfaga. Como se ha empleado tráfico uniforme, y los enlaces son relativamente cortos (menores de 300 kms), el retardo de propagación medio por ráfaga es pequeño. Incluso en el caso del algoritmo propuesto AMOR sube hasta 1.9 ms con respecto a los 1.5 ms de SP.

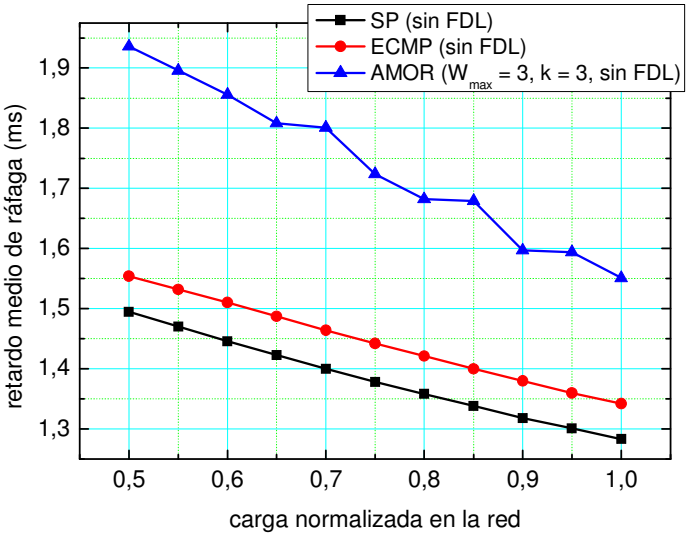


Figura 7-7 Retardo medio de ráfaga (ms) en caso de nodos sin FDLs

En el caso de nodos con FDL, como se observa en la Figura 7-8, el retardo aumenta ligeramente, no solo por el mayor número de saltos por ráfaga, sino porque el menor bloqueo ha sido conseguido a costa de un tiempo en un FDL. En este caso con FDL, la evolución del retardo no tiene un comportamiento líneal en el algoritmo AMOR.

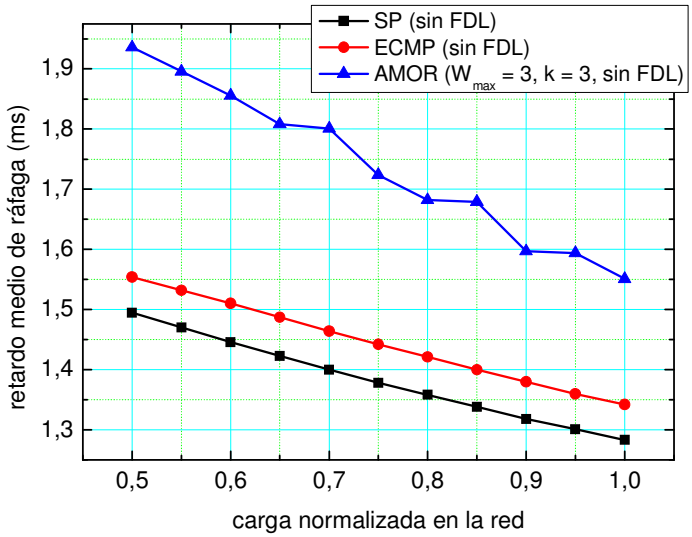


Figura 7-8 Retardo medio de ráfaga (ms) en caso de nodos con FDLs

Análisis de carga en los enlaces

La técnica de encaminamiento que escoge el camino más corto en número de saltos es la que introduce una menor carga en la red, ya que el número de tramos de la red (enlaces) a los que contribuye cada par origen-destino es mínimo. La estrategia ECMP, en principio, introduce la misma carga que la estrategia SP. Sin embargo, al obtenerse con ECMP una menor probabilidad de bloqueo que con SP, hay más ráfagas que progresan, con lo que la carga media en los enlaces red es ligeramente superior a la medida con SP, como se puede observar en la Figura 7-9, que muestra la carga media de los enlaces para las 3 estrategias de encaminamiento analizadas. En concreto, la diferencia de carga es menor del 5% para todas las cargas simuladas.

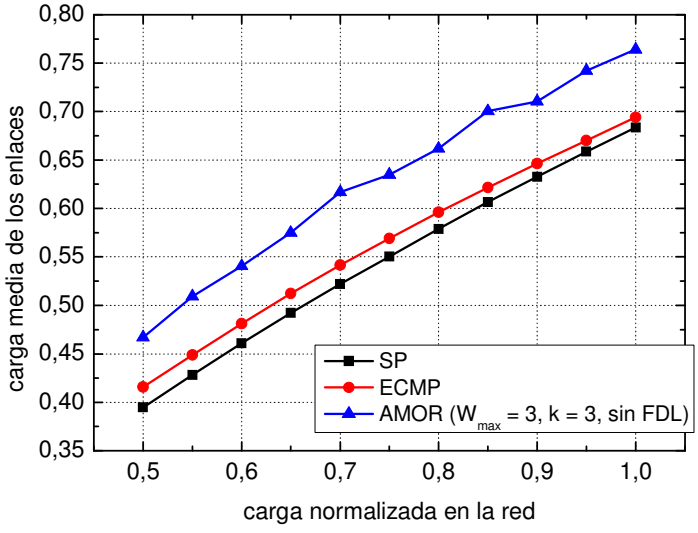


Figura 7-9 Carga media en los enlaces de la red (sin FDLs)

La estrategia AMOR emplea caminos más largos en algunos casos, por lo que introduce una mayor carga en los enlaces al enviar ráfagas por más tramos. Por ejemplo, cuando la carga normalizada introducida en la red es de 0,5, la carga media en los enlaces con SP es de 0,395, mientras que con AMOR la carga media en los enlaces sube a 0,467, es decir se ha incrementado un 18%. Este aumento de carga media en los enlaces se mantiene para todos los niveles de carga, aunque se observan algunos casos en los que el aumento, en términos absolutos es ligeramente superior. Esto se debe al comportamiento del algoritmo,

que habilita nuevos caminos a medida que aumenta la carga. Hay situaciones en las que se superan los umbrales y aparecen nuevos caminos más largos, aumentando proporcionalmente la carga media más que en una situación de tráfico menor. Por ejemplo, la mayor diferencia se obtiene cuando la carga normalizada en la red es 0,85, en la que se pasa de 0,61 a 0,7.

Finalmente, se va a analizar cómo reparten los algoritmos la carga en la red. La estrategia SP, al concentrarse en los caminos más cortos, puede saturar en exceso unos enlaces mientras que otros pueden estar prácticamente vacíos. Para medir cómo distribuye la carga, se mide la varianza de la carga de los distintos enlaces de la red y se representa gráficamente en la Figura 7-10.

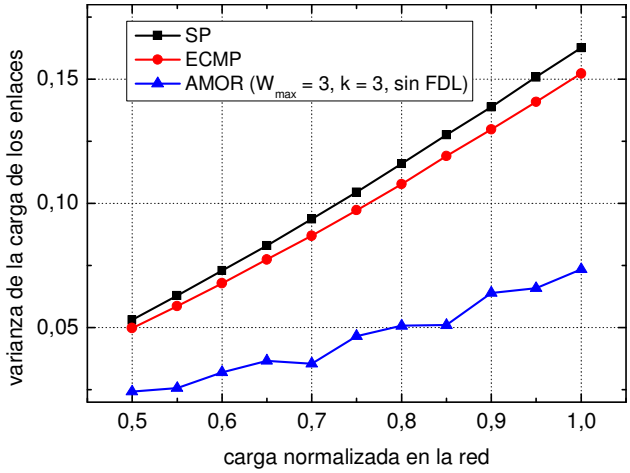


Figura 7-10 Varianza de la carga en los enlaces de la red (sin FDLs)

La gran ventaja del algoritmo AMOR es su habilidad para repartir la carga en la red. Se observa cómo es capaz de mantener la varianza en valores muy bajos. Es decir, es capaz de conseguir una distribución bastante uniforme de la carga. Se observa cómo la varianza en el caso de AMOR crece con la carga normalizada a menor ritmo que en SP o ECMP. Esto se debe a que SP y ECMP al concentrar la carga en pocos enlaces, cuanto más carga haya en la red, más estarán cargados y más diferencias tendrán con el resto de enlaces poco cargados de la red. Sin embargo, con AMOR, a medida que la carga en la red aumenta, se empiezan a habilitar caminos alternativos, que emplean enlaces poco cargados. La apertura de nuevos caminos se produce de manera escalonada, por lo que no se produce un crecimiento lineal, al igual que sí ocurre en el caso de SP y ECMP.

Por otro lado, ECMP, al emplear más enlaces que SP, balance ligeramente la carga. Sin embargo se observa que la diferencia entre las varianzas de SP y ECMP es pequeña. Por tanto AMOR demuestra su capacidad para balancear la carga en la red. De esta forma, AMOR será especialmente útil en aquellos escenarios de red cuanto más desbalanceado esté el tráfico con respecto a la topología. Estas situaciones pueden producirse, por ejemplo, después de la caída de nodos en la red, o bien ante situaciones de aumento inesperado del tráfico. Aunque una red se haya planificado y dimensionado para un tráfico, a medida que pasa el tiempo, el patrón del tráfico puede cambiar. En esos casos, una estrategia como AMOR puede ayudar a repartir el tráfico por toda la red y emplear recursos infrautlizados.

7.4 Conclusiones

En este capítulo se ha propuesto y estudiado una técnica de encaminamiento dinámica multicamino con histéreris que emplea información local para OBS. El punto de partida ha sido la técnica MRDV, diseñada para redes IP. Si se aplica directamente MRDV en una red

OBS, el resultado es muy inestable, y se producen oscilaciones de tráfico. Por tanto, la técnica ha sido rediseñado para OBS, en concreto se ha propuesto una nueva función de coste local y una nueva función de reparto de carga más estable. La técnica propuesta se denomina AMOR, *adaptive multipath OBS routing*.

Los resultados muestran que, en nodos sin FDLs se consiguen únicamente pequeñas mejoras, ya que al emplear caminos más largos, se aumenta la carga en la red, y el hecho de evitar zonas saturadas no compensa el aumento de carga, a costa de un ligero incremento del retardo.

En cambio, la técnica AMOR consigue mejoras en nodos con FDLs, que se benefician más de la reducción de carga a niveles de carga elevadas, y no les penaliza mucho un ligero aumento de carga a cargas bajas. Igualmente, la técnica AMOR conlleva un ligero aumento del retardo.

Capítulo 8

Diseño e implementación de un PCE para WR-OBS

Los estudios realizados en los capítulos anteriores de la tesis se han centrado en una de las alternativas de OBS, específicamente en OBS sin confirmación. En los dos últimos capítulos de la tesis las investigaciones se centran en otra de las alternativas de OBS, en este caso OBS con confirmación. En concreto, los estudios se centran en la arquitectura *Wavelength-Routed*-OBS (WR-OBS).

En primer lugar se realiza un repaso de los elementos de una red WR-OBS, y se describen las principales técnicas de creación de ráfagas. En la literatura se proponen arquitecturas de WR-OBS tanto distribuidas como centralizadas. En este capítulo se propone el empleo de una arquitectura WR-OBS con elementos definidos por los estándares actuales: un cálculo de caminos centralizado basado en *Path Computation Element* (PCE) y un establecimiento de camino óptico distribuido basado en GMPLS. Así, se describe en detalle cómo sería el intercambio de mensajes en dicha arquitectura. Uno de los elementos principales es el mencionado PCE, y en concreto el protocolo *Path Computation Element Protocol* (PCEP). Dicho protocolo está actualmente definido para otros tipos de redes. En este capítulo se describe cómo el protocolo PCEP puede soportar redes WR-OBS, y se proponen las extensiones protocolares necesarias. Después, se realiza el diseño de un PCE para WR-OBS y se describen los módulos necesarios. Finalmente, se realiza un prototipo del PCE, en el que se implementan las extensiones protocolares propuestas. El siguiente capítulo se dedica en su totalidad a la evaluación experimental del prototipo de PCE implementado.

8.1 Arquitectura de red WR-OBS

Las redes WR-OBS se basan en el establecimiento dinámico de *lightpaths* para la transmisión de ráfagas [246] [247]. La gran principal diferencia con el resto de redes OBS es que el establecimiento del camino óptico es asentido, facilitando la garantía de QoS. Por tanto, una WR-OBS combina las características de una red OBS con las de una red de conmutación de circuitos tradicional.

La arquitectura de una red WR-OBS consta de dos tipos de nodos, los nodos de borde (*edge router*), que se encargan de clasificar y agrupar los paquetes en ráfagas, y los nodos de transporte ópticos (*core router*) por los que se transmiten las ráfagas ópticas. De esta forma, los datos se agregan en los nodos de borde para ser transmitidos posteriormente de manera totalmente óptica a través de los nodos de transporte. Ambas funcionalidades, nodo de borde y de transporte, pueden combinarse en un mismo equipo.

Hay tres métodos importantes de formación de ráfagas en WR-OBS [248], FBAT (*Fixed Burst Assembly Timer,* Ráfagas de tiempo de agregación fijo), LBS (*Limited Burst Size*, tamaño de ráfaga limitado) y UBS (*Unlimited Burst Size*, tamaño de ráfaga sin límite), que se describen a continuación.

Ráfagas de tiempo de agregación fijo FBAT

Empleando este método, la ráfaga i_k (donde *i* es el número de ráfaga y *k* es el número de par origen destino) se construye agregando paquetes en el *buffer k* durante un tiempo fijo *T* desde la llegada del primer paquete al *buffer*. Por tanto, en el caso de FBAT, el valor del temporizador *T* y el tiempo de agregación $t_{aggr,k}$ (definido como el tiempo durante el cual se está formando la ráfaga) coinciden. Así, una vez transcurrido el tiempo *T* desde la llegada del primer paquete la ráfaga está completa y comienza la construcción de una nueva ráfaga. Cuando la ráfaga está completa se inicia el proceso de creación del *lightpath*. El funcionamiento exacto de este proceso de creación de *lightpath* dependerá del tipo de arquitectura empleada. Una vez que el *lightpath* está establecido, comienza la transmisión de la ráfaga.

Tamaño de ráfaga limitado LBS

Empleando la técnica LBS, se inicia un temporizador de tiempo fijo de valor *T* en cuanto llega el primer paquete al *buffer k*. Una vez que ha pasado dicho tiempo *T*, se inicia el proceso de creación del *lightpath* (incluyendo los cálculos necesarios y la señalización del mismo). En cuanto se ha establecido el *lightpath* se finaliza la construcción de la ráfaga y se procede a su transmisión. El siguiente paquete que llegue al ensamblador, como la construcción de la ráfaga ya se había completado, inicia la creación de una nueva ráfaga. Por tanto, en el caso de LBS el tiempo de agregación t_{aggrk} es mayor que el valor del temporizador *(T)*.

Tamaño de ráfaga ilimitado UBS

Cuando se emplea la técnica UBS, se inicia un temporizador de tiempo fijo de valor *T* en cuanto llega el primer paquete al *buffer k*. Una vez que ha pasado dicho tiempo *T*, al igual que en LBS, se inicia el proceso de creación del *lightpath* (incluyendo los cálculos necesarios y la señalización del mismo). Cuando se ha establecido el *lightpath* se empieza a transmitir la ráfaga formada por todos los paquetes del *buffer*. Sin embargo, a diferencia de LBS, los paquetes que llegan al buffer *k* mientras se está transmitiendo la ráfaga, se añaden a la ráfaga. De esta forma, el tamaño de la ráfaga en UBS es potencialmente ilimitado, en comparación con LBS y FBAT, en los que el tiempo de construcción de la ráfaga, y por tanto su tamaño, es limitado.

Arquitectura de red WR-OBS con PCE

Una red WR-OBS puede construirse con arquitecturas tanto distribuidas como centralizadas. Esta tesis toma como punto de partida el modelo de WR-OBS propuesto por de Miguel [248], y propone una arquitectura de control de WR-OBS, mostrada en la Figura 8-1, elaborada a partir de elementos de control definidos por los estándares actuales. Así, el elemento funcional dedicado a realizar el cálculo de caminos es el denominado PCE (*Path Computation Element*, elemento de cálculo de caminos), propuesto por el IETF en la RFC 4655 [10]. En concreto este elemento se define como una entidad capaz de calcular un camino de red o ruta a partir de un grafo de red así como aplicar restricciones en dicho cálculo. En el caso de WR-OBS se emplearía un PCE centralizado que tenga el detalle de la ocupación de las longitudes de onda en la red y se comunicaría con los nodos WR-OBS a través del protocolo PCEP (*Path Computation Element Protocol*) [11].

El establecimiento del *lightpath* se realiza de manera distribuida basándose en la arquitectura de control de GMPLS (*Generalizaed Multiprotocol Label Switching*) [75], en concreto empleando el protocolo de reserva *Resource Reservation Protocol-Traffic Engineering* (RSVP-TE).

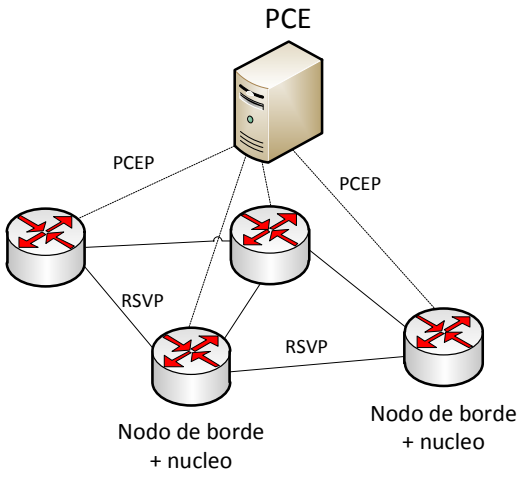


Figura 8-1 Arquitectura de control de WR-OBS

Así, en la arquitectura de control propuesta, el PCE para WR-OBS recibirá peticiones de cálculo de ruta de *lightpaths* de corta duración entre dos nodos. Estas peticiones serán enviadas por los clientes PCC (*Path Computation Client*), ubicados en los nodos WR-OBS. La ruta del *lightpath* consistirá en un conjunto de nodos y fibras junto con la longitud de onda correspondiente. En la nomenclatura de GMPLS (la arquitectura que define el plano de control de las redes de transporte) se denomina λ-LSP (λ-*Label Switched Path*) a la representación en el plano de control de un *lightpath*. De esta forma, el PCE tiene que ejecutar un algoritmo dinámico de encaminamiento y asignación de longitud de onda (*Dynamic Routing and Wavelength Assignment*, DRWA) para calcular dicha secuencia de nodos, fibras y lambdas. Zapata-Beghelli y Bayvel, en el contexto de WR-OBS, recomiendan 3 estrategias [82] dentro de las técnicas DRWA clásicas de la literatura:

- AUR-E (*Adaptive Unconstrained Routing Exhaustive*) [249]: Según Zapata-Beghelli y Bayvel, obtiene los mejores resultados en cuanto a probabilidad de bloqueo, ya que ejecuta el algoritmo de *Dijkstra* en cada petición para cada lambda. Esto hace que sea bastante intensivo en cuanto a CPU [250].
- SP-FF (*Shortest Path First Fit*) [251]: Las rutas están pre-calculadas ejecutando el algoritmo de Dijkstra y solamente se hace en tiempo real la asignación de lambda. Tiene una mayor probabilidad de bloqueo que AUR-E.
- k-SP-FF (*k Shortest Path First Fit*). Se calculan los k caminos más cortos. A continuación, se intenta encontrar una lambda disponible empezando por el camino más corto hasta llegar al más largo. Se escoge el primer camino en el que se pueda asignar la lambda.

Al tratarse de *lightpaths* de corta duración, el tiempo transcurrido entre el instante en el que se ha calculado la ruta de un *lightpath* y el instante de tiempo en el que se establece dicho *lightpath* no es despreciable en comparación con el tiempo que permanece establecido el *lightpath*. Es altamente probable que durante el intervalo de tiempo mencionado (fin del cálculo hasta fin del proceso de establecimiento) puedan llegar nuevas peticiones. Típicamente, en una red basada en PCE, la base de datos de ingeniería de tráfico, en la que se tiene la información, por ejemplo, de la ocupación de los canales en una fibra, se actualiza una vez que se ha establecido el LSP. Por tanto, hay un periodo de tiempo con potenciales inestabilidades. Por otro lado, si la duración de los LSPs es muy corta, el volumen de mensajes de actualización de cada enlace podría ser elevado. Por tanto, el PCE para WR-OBS ha de poder bloquear temporalmente los recursos calculados, asumiendo que el establecimiento se producirá satisfactoriamente, y no asignará dichos recursos, en este caso lambdas en cada enlace de la ruta, a otras peticiones. La duración del bloqueo de los recursos puede indicarse explícitamente en la solicitud, como por ejemplo en el caso del

modo de funcionamiento FBAT explicado anteriormente en el que el tamaño de la ráfaga es conocido. En el caso de que a priori no se conozca la duración del *lightpath*, se pueden bloquear los recursos de manera indefinida (o un tiempo grande) y posteriormente notificar explícitamente el fin del *lightpath*. Este último comportamiento es el caso de LBS y UBS.

Por otro lado, de Miguel propone ofrecer garantías de retardo a las ráfagas [81], de tal forma que, cuando se solicite un camino, el tiempo de respuesta esté acotado, para que se cumplan unas garantías, y el retardo extremo a extremo de la ráfaga esté acotado. Para ello, tanto el nodo, que actúa de *Path Computation Client* (PCC), como el PCE han de ser capaces de calcular el retardo entre ambos. Asimismo, el PCE ha de implementar un mecanismo de reintentos en el caso de que no se haya podido asignar una lambda a una petición.

Funcionamiento detallado con ráfagas de tiempo de agregación fijo FBAT

Anteriormente, al comienzo de este capítulo, se ha descrito el fundamento del método FBAT. A continuación, en la Figura 8-2, se muestra un ejemplo del funcionamiento detallado, particularizado para la arquitectura WR-OBS con PCE+RSVP propuesta, que se explica seguidamente.

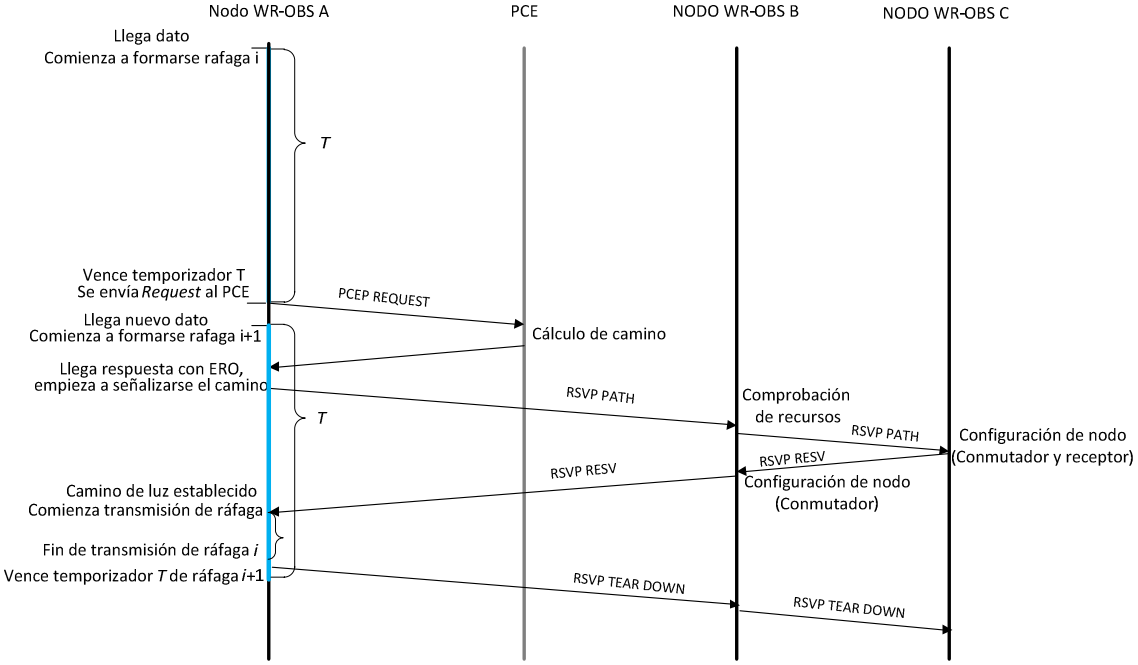


Figura 8-2 Funcionamiento detallado de FBAT con PCE y RSVP

En el ejemplo de la Figura 8-2 hay 3 nodos WR-OBS (denominados A, B y C), que realizan las funcionalidades tanto de nodo de borde como de núcleo. En t_0 llega un paquete al nodo A con destino al nodo C y comienza la construcción de la ráfaga *i*, poniéndose en marcha el temporizador *T*. Una vez transcurrido ese tiempo se da por finalizada la ráfaga, conociéndose en ese instante el tamaño de la ráfaga. En ese momento, t_0+T se envía la petición de cálculo de ruta del *lightpath* unidireccional al PCE, empleando el protocolo PCEP que se describe posteriormente, en el que se indicará, entre otros parámetros, cuánto tiempo ha de bloquear el PCE los recursos del camino (longitud de onda). El PCE calcula la ruta, que consistirá en una secuencia de nodos, interfaces de salida (la combinación nodo-interfaz de salida identifica un enlace de fibra entre dos nodos) y longitud de onda, que se envía al nodo A en un mensaje PCEP de tipo *Response*. Este mensaje contiene un objeto denominado ERO (EXPLICIT ROUTE OBJECT) que contiene los detalles de la ruta, que se podrá utilizar posteriormente a la hora de realizar el establecimiento. El nodo A, tras examinar la ruta, empieza el establecimiento del camino con el protocolo RSVP. Para ello, envía a su siguiente salto (B) un mensaje de tipo *Path*, en el que incluye un objeto

ERO. Este ERO está constituido por el ERO enviado por el PCE quitando el primer salto (ya que no es de interés para el siguiente nodo). El nodo B, tras recibir el mensaje *Path*, comprueba si los recursos en ese instante están libres (un camino entre la fibra de entrada y la de salida para una determinada longitud de onda). En caso de que lo estén, envía el mensaje de tipo *Path* con el ERO recibido menos el último salto realizado. Cuando llega el mensaje *Path* al nodo C, este comprueba que la longitud de onda en recepción está libre y que puede recibir la longitud de onda por el puerto deseado. En caso de que sí haya recursos, los reserva, para que nadie más pueda utilizarlos en ese momento, y envía un mensaje de tipo *Resv* al nodo predecesor en la ruta. Cuando B recibe la respuesta, reserva los recursos, realiza las conexiones y envía el mensaje de respuesta al nodo A. Cuando A recibe la respuesta positiva, prepara al nodo para la transmisión y se comienzan a transmitir los datos. En cuanto ha terminado de transmitir, se envía un mensaje de tipo *Teardown* (también conocido como *ResvTear*) para eliminar el camino recién establecido. En cuanto los nodos reciben el mensaje, se liberan los recursos.

Funcionamiento detallado con tamaño de ráfaga limitado LBS

Un ejemplo del funcionamiento detallado de LBS, particularizado para la arquitectura WR-OBS con PCE y señalización con RSVP, se muestra en la Figura 8-3.

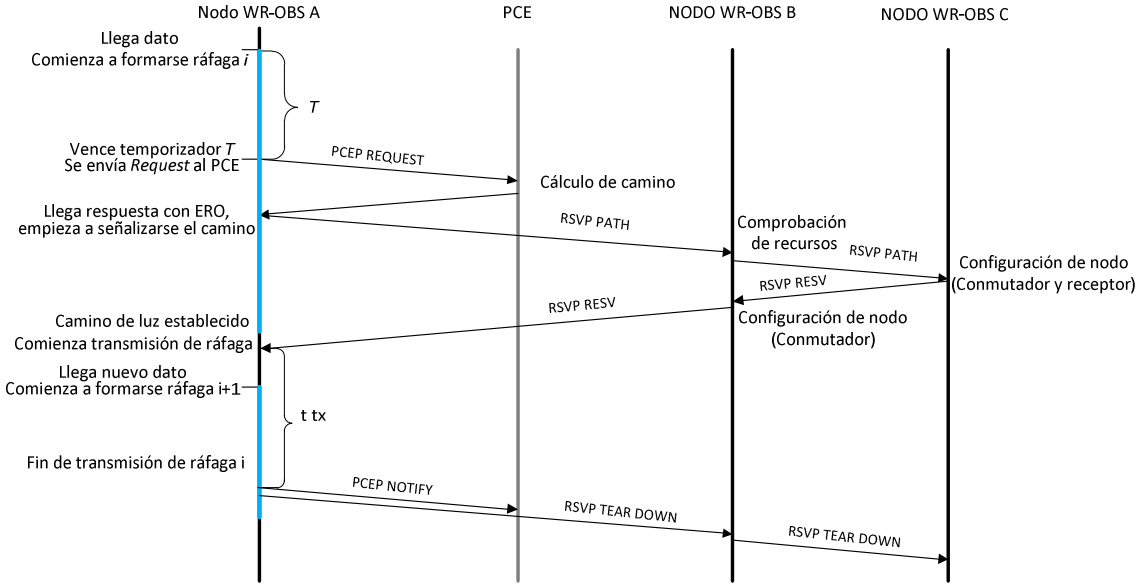


Figura 8-3 Funcionamiento detallado de LBS en arquitectura WR-OBS con PCE y RSVP

Al igual que en el ejemplo anterior, en el ejemplo de la Figura 8-3 hay 3 nodos WR-OBS (A, B y C), que realizan las funcionalidades tanto de nodo de borde como de núcleo. En t_0 llega un paquete al nodo A con destino al nodo C y comienza la construcción de la ráfaga *i*, poniéndose en marcha el temporizador *T*. Una vez transcurrido ese tiempo, en t_0+T, se envía la petición de ruta de *lightpath* unidireccional al PCE con un mensaje de tipo *Request* del protocolo PCEP, en el que, además de preguntar por la ruta del *lightpath,* se indica que hay que bloquear en el PCE los recursos del camino de manera indefinida. El PCE calcula la ruta (secuencia de nodos, enlaces y longitud de onda), la envía al nodo A en un mensaje PCEP de tipo *Response* junto con un identificador del bloqueo de los recursos en el PCE. El nodo A, tras examinar la ruta, empieza el establecimiento del camino con el protocolo RSVP siguiendo el proceso que descrito anteriormente (mensajes *Path* y *Resv*). Cuando A recibe la respuesta confirmando el correcto establecimiento del camino, se finaliza la construcción de la ráfaga y se comienza a transmitir el contenido de la misma. En cuanto ha terminado de transmitir, se envía una notificación al PCE para que desbloquee los recursos mediante un mensaje de tipo *Notify* (mensaje del protocolo PCEP), y se envía un mensaje *Path Tear* (mensaje de liberación de camino del protocolo RSVP) al siguiente

nodo en la ruta (nodo B) para eliminar el camino recién establecido. El mensaje de tipo *Notify* no requiere de confirmación, ya que la fiabilidad de la comunicación está garantizada por el empleo del protocolo de transporte TCP. Finalmente, cuando llega la notificación (mensaje *Notify*) al PCE, este ya tiene los recursos libres para asignárselos a otra petición.

8.2 Protocolo PCEP para WR-OBS

La comunicación entre el cliente de cálculo de caminos (denominado PCC, *Path Computation Client*) y el servidor de cálculo de rutas (denominado PCE, *Path Computation Element*) se realiza mediante el protocolo PCEP (*Path Computation Element Protocol,* protocolo del PCE) [11]. En esta sección se definen las características específicas del protocolo PCEP para el caso en el que el cliente PCC sea un nodo de borde WR-OBS y el PCE sea el encargado de calcular los caminos de una red WR-OBS. Para ello, en primer lugar se realiza un breve recorrido por el estado del arte del protocolo. A continuación se detallan las necesidades específicas de WR-OBS, justificando las elecciones realizadas y describiendo las extensiones necesarias que los estándares actuales no cubren.

Introducción al protocolo PCEP

El protocolo PCEP está especificado en la RFC 5440 [11]. Dicho protocolo se implementa sobre TCP [177], y define un conjunto de mensajes que se intercambian, así como un mecanismo para iniciar la sesión. Los mensajes se componen de una cabecera y un conjunto de objetos. A su vez, los objetos contienen campos de datos y unas estructuras de datos denominadas TLVs (*Type-Length-Value*), similares a los objetos. En la RFC 5440 se define la sintaxis de los mensajes de PCEP. Esta sintaxis define los objetos que hay en cada mensaje, en qué orden han de aparecer los objetos, cuáles de estos son obligatorios y cuáles son opcionales. Los tipos de mensajes que se han definido en el estándar PCEP se muestran a continuación. Entre paréntesis, después del nombre del mensaje, se indica el nombre abreviado del tipo de mensaje acorde al estándar de la RFC 5440.

- ***Open***: Usado en el inicio de sesión PCEP
- ***Keepalive* (*KA*)**: Usado para mantener viva una sesión.
- ***Request* (*PCReq*)**: Mensaje para pedir uno o varios cálculos de camino.
- ***Response* (*PCRep*)**: Mensaje en respuesta a uno o varios cálculos de camino.
- ***Notify* (*PCNtf*)***:* Notificación de un evento especial.
- ***Error* (*PCErr*)**: Mensaje para indicar errores.
- ***Close***: Empleado para cerrar una sesión PCEP.

Inicio de sesión

El inicio de sesión se puede comenzar tanto del lado del cliente PCC, como del lado del servidor PCE, y empieza abriendo una sesión TCP al puerto registrado de PCEP, el 4189. El puerto origen de la conexión en principio estaba restringido también al 4189. Sin embargo, esta restricción se ha relajado tras detectar la imposibilidad de cumplir este requisito en algunos casos [252]. Por tanto, el puerto origen puede ser cualquiera, al igual que en una conexión HTTP. Aunque como se ha mencionado, tanto PCC como PCE pueden iniciar la conexión, típicamente es el PCC el que inicia la conexión, y el PCE el que escucha en el puerto 4189 nuevas conexiones.

En cuanto está establecida la conexión TCP ambos extremos de la conexión envían sendos mensajes del protocolo PCEP de tipo *Open*, en los que indican las características de la sesión. Si las características indicadas en dicho mensaje son aceptadas, se envía un mensaje *Keepalive* como confirmación.

En el inicio de la sesión se pueden indicar las capacidades del PCE. Por ejemplo, se indica la frecuencia con que se van a enviar mensajes de refresco (mensajes de tipo *Keepalive*) al otro extremo y cuál es el temporizador que debe utilizar para decidir si una sesión PCEP ha de cerrarse si no se recibe ningún mensaje. En el inicio de sesión el PCE ha de indicar las distintas funciones objetivo que se soportan [253].

Mantenimiento de la sesión

Una vez establecida la sesión, puede optarse por mantener una sesión persistente. Para ello, cada extremo envía periódicamente mensajes *Keepalive* con el fin de mantener viva la sesión. Si durante un tiempo, cuyo valor viene determinado por el valor de la variable *Deadtimer*, que se negocia al comienzo de la sesión, un extremo no envía ningún mensaje PCEP, la sesión PCEP se cierra. La otra alternativa consiste en que una vez establecida la sesión, esta se mantenga abierta por un tiempo predefinido, sin necesidad de enviar mensajes *Keepalive*, con la posibilidad de cerrar la sesión mediante un mensaje de tipo *Close*.

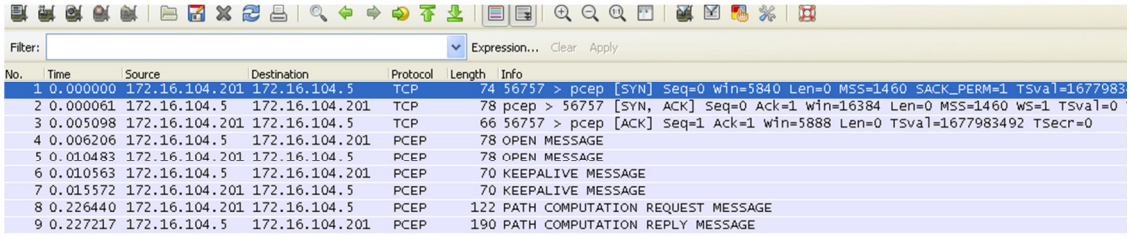
Petición de cálculo

Una vez se ha establecido la sesión PCEP, en cualquier momento el PCC puede enviar una petición de cálculo de camino. Para ello, envía un mensaje de tipo *PCReq*, que contiene un conjunto de objetos para indicar los atributos y restricciones de la petición. En un mensaje *PCReq* se pueden enviar varias peticiones juntas. Para cada petición, se puede indicar, por ejemplo, cuáles son el origen y destino del camino que se solicita, el ancho de banda, la métrica a utilizar, etc. El PCE, una vez hecho el cálculo, envía la respuesta en un mensaje *PCRep*. Dicho mensaje puede contener una o varias respuestas, por lo que el PCE puede optar, en caso de haber recibido varias peticiones, en responderlas en mensajes *PCReq* diferentes, o en uno solo. En cualquier caso, cada petición de camino incluida en un mensaje *PCReq* contiene un identificador, que se empleará en la respuesta para poder asociar dicha respuesta a la petición.

La respuesta puede ser tanto positiva, en la que se incluyen los detalles del camino e información adicional, por ejemplo métricas del mismo, como negativa, en la que se puede incluir la causa por la cual no se ha encontrado un camino válido.

Ejemplo de funcionamiento

Para ilustrar el funcionamiento del protocolo PCEP se muestra en la Figura 8-4 una captura de pantalla del programa *Wireshark* [254] en la que aparece un intercambio de mensajes del protocolo PCEP entre un cliente PCC y un PCE. Tanto el PCC como el PCE se han implementado en este capítulo, como se describe posteriormente. La captura de los mensajes con *Wireshark* se ha realizado en la máquina en la que está ubicado el PCE. Por otro lado, al mostrar la captura del tráfico, se han mostrado los paquetes de TCP que realizan el inicio de la conexión y los mensajes PCEP.


Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.104.201	172.16.104.5	TCP	74	56757 > pcep [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=1677983
2	0.000061	172.16.104.5	172.16.104.201	TCP	78	pcep > 56757 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1460 WS=1 TSval=0
3	0.005098	172.16.104.201	172.16.104.5	TCP	66	56757 > pcep [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=1677983492 TSecr=0
4	0.006206	172.16.104.5	172.16.104.201	PCEP	78	OPEN MESSAGE
5	0.010483	172.16.104.201	172.16.104.5	PCEP	78	OPEN MESSAGE
6	0.010563	172.16.104.5	172.16.104.201	PCEP	70	KEEPALIVE MESSAGE
7	0.015572	172.16.104.201	172.16.104.5	PCEP	70	KEEPALIVE MESSAGE
8	0.226440	172.16.104.201	172.16.104.5	PCEP	122	PATH COMPUTATION REQUEST MESSAGE
9	0.227217	172.16.104.5	172.16.104.201	PCEP	190	PATH COMPUTATION REPLY MESSAGE

Figura 8-4 Captura de un intercambio de mensajes PCEP básico

Para ayudar el entendimiento de la captura, se ha incluido en la Figura 8-5 un diagrama con mensajes PCEP intercambiados, al que se ha añadido el intercambio de segmentos TCP inicial.

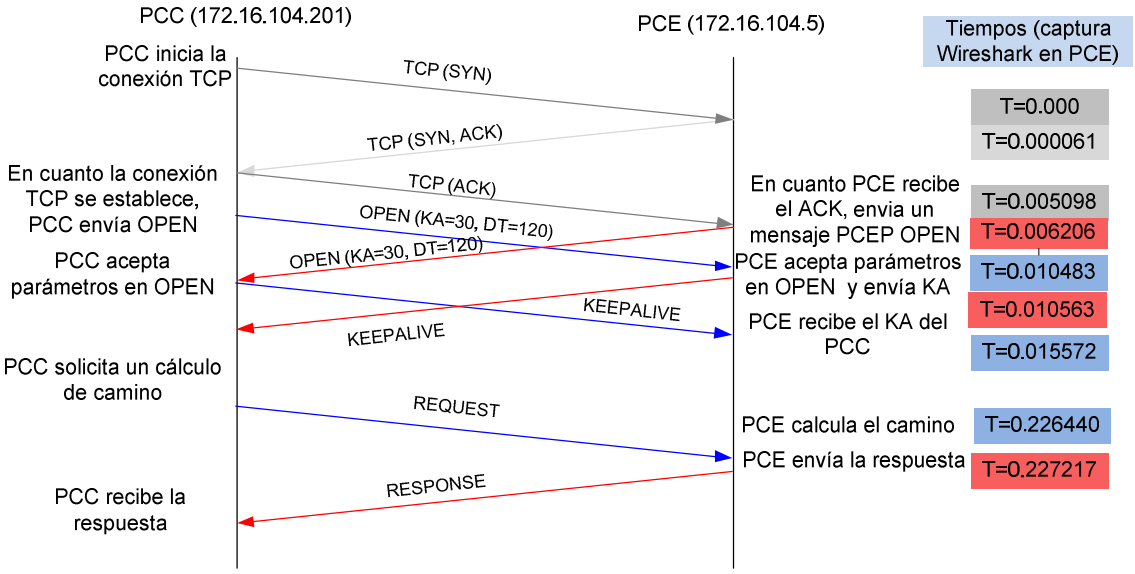


Figura 8-5 Diagrama de un intercambio de mensajes PCEP básico

En el ejemplo mostrado en la Figura 8-4 y la Figura 8-5 hay un PCC, cuya dirección IP es la 172.16.104.201 y un PCE, cuya dirección IP es la 172.16.104.5, y entre ellos hay un RTT de aproximadamente 5 ms, medido a través de un *ping*. En la Figura 8-5 se muestran en gris los segmentos TCP y en azul los mensajes PCEP (que se transportan sobre segmentos TCP que ya no se muestran, al igual que no se muestran sus ACKs) del PCC al PCE, y en rojo los mensajes PCEP del PCE al PCC. A la derecha de la Figura 8-5 se indica el instante en el que se ha capturado el mensaje en el PCE (obtenido a partir de la captura mostrada en la Figura 8-4). En primer lugar el PCC inicia la sesión TCP. Se observa que en cuanto ha llegado al PCC la confirmación del establecimiento de sesión y ha enviado el ACK para completar el *3 way handshake* de TCP, envía un mensaje de tipo *Open*. El PCE a su vez, en cuanto recibe el ACK de TCP envía su mensaje *Open*. Por tanto, los mensajes *Open* de ambos extremos se cruzan por el camino. Como PCC y PCE aceptan los parámetros indicados en los mensajes *Open* respectivos, envían cada uno de ellos un mensaje *KA* (*Keepalive*).

PCEP para WR-OBS

El protocolo PCEP, tal y como está definido en la RFC 5440 [11] y las distintas RFCs que lo complementan, se centra en redes MPLS (*Multi-Protocol Label Switching*). Una red WR-OBS tiene unas características específicas en cuanto a dinamicidad y a recursos solicitados que determinarán las elecciones para particularizar el protocolo PCPEP. A continuación, se analiza la particularización del protocolo PCEP para WR-OBS.

Conexión permanente vs intermitente

Cada nodo de borde de una red WR-OBS realiza, por cada par origen destino, varias peticiones de camino por segundo, suponiendo valores del temporizador en el orden de las decenas de milisegundo. Hay dos formas de operar un PCE, en modo intermitente, en el que se abre una nueva sesión PCEP por cada petición, y en modo permanente, en el que la conexión queda permanentemente establecida y se pueden en cualquier momento peticiones. Empleando el modo intermitente el tiempo necesario para enviar una petición incluye el *3 way handshake* y el intercambio de mensajes *Open*, por lo que como mínimo, se necesita que transcurra 2 veces el RTT (*Round Trip Time*) antes de poder enviar la petición. Este retardo adicional puede ser excesivo. En cambio, en modo permanente, la penalización es únicamente el envío periódico de un mensaje *Keepalive*, típicamente cada 30

segundos, con un impacto mínimo. Por lo tanto, la primera conclusión es que para un entorno WR-OBS se empleará una conexión PCEP permanente.

Identificación de nodos, fibras y longitud de onda

En primer lugar, es necesario poder identificar cada nodo y cada fibra. Esta identificación se empleará tanto en los mensajes de petición para indicar los extremos del camino, como en la respuesta, para indicar la ruta. Como el número de nodos será relativamente pequeño (en el orden de decenas de nodos), se puede emplear una dirección IPv4 de 32 bits para identificar cada nodo. Cada nodo WR-OBS consta de varias fibras de salida, que identificarán con interfaces no numerados [255], mediante el identificador del nodo y un identificador de 32 bits único en el nodo.

Para identificar longitud de onda se empleará el formato de etiqueta definido en la RFC 6205 [256], en la que se especifica la rejilla (*grid*), que puede ser DWDM (*Dense Wavelength Division Multiplexing*) o CWDM (*Coarse Wavelength Division Multiplexing*), el espaciado (*Channel Spacing*, C.S.), cuyo valor más utilizado es 50 GHz y un número *n* con el que puede calcularse la longitud de onda en cuestión. Por ejemplo, para el caso de la rejilla DWDM, la frecuencia se calcula con la fórmula $Frequency\ (\mathrm{THz}) = 193.1\ \mathrm{THz} + n * channel\ spacing\ (\mathrm{THz})$ [257]. El identificador se usa de manera opcional para poder especificar un láser concreto si hay varios disponibles. En esta tesis se emplea el identificador a 0, ya que es suficiente con identificar la lambda que se tiene que emplear. En la Figura 8-6 se muestra la codificación de la etiqueta que representa la longitud de onda. Esta etiqueta, como se verá más adelante, formará parte de la respuesta.

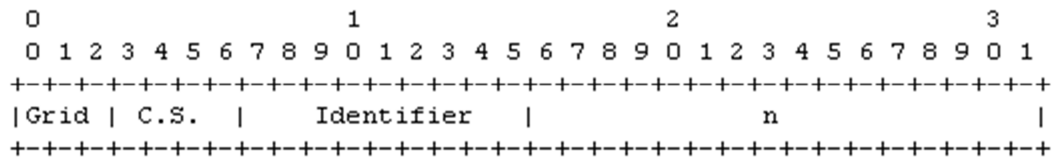


Figura 8-6 Etiqueta que representa la longitud de onda

Formato del mensaje de petición de camino

En el protocolo PCEP se emplea el mensaje de tipo *Request* para solicitar una ruta. Este mensaje se construye a partir de varios objetos, algunos de ellos obligatorios y otros opcionales. Para una red WR-OBS hay que escoger qué objetos de los que define el estándar serán necesarios, y qué nuevos objetos se necesitan.

Objeto REQUEST PARAMETERS (representado como <RP>) [11]: Especifica las características del camino a pedir. Cabe destacar que se solicitan siempre caminos unidireccionales, con lo que el bit B (*bidirectional*) se ha de poner a cero. Por otro lado, la ruta que se solicita será estricta, es decir, tendrá que incluir todos los saltos, por lo que el bit O (*strict/loose*) se pone a 0, tal y como indica la RFC 5440 [11]. Asimismo, en este objeto se incluye un identificador, que será único para cada petición dentro de una sesión PCC-PCE, y sirve para distinguir las peticiones. Este identificador lo ha de generar el PCC, y se empleará en la respuesta del PCE.

Objeto END POINTS (representado como <ENDPOINTS>) [11]: Especifica los extremos del camino solicitado. Se emplea como extremos las direcciones IPv4 de los nodos de borde WR-OBS, por lo que el tipo de objeto END POINTS es IPv4.

Objeto BANDWIDTH (representado como <BW>) [11]: Es un objeto opcional que se usa para indicar el ancho de banda solicitado. Para una red WR-OBS se escoge emplear este objeto, indicando el ancho de banda del *lightpath* solicitado. Nótese que no se solicita un determinado ancho de espectro, sino una cantidad en Mbits/segundo.

Objeto METRIC (representado como <METRIC>) [11]: Es un objeto opcional que puede representar o bien la métrica que se quiere minimizar, por ejemplo métrica de

ingeniería de tráfico, número de saltos…, o bien un límite en una métrica, por ejemplo número máximo de saltos, o simplemente se solicita al PCE que responda con la métrica calculada. En el caso de una red WR-OBS, se pueden escoger varias métricas a minimizar. Unas métricas comúnmente empleadas son minimizar el retardo de extremo a extremo y minimizar el número de saltos. La métrica del número de saltos está especificada en la RFC 5440, pero el minimizar el retardo de extremo a extremo no está estandarizado aún. Se propone utilizar la métrica *delay* propuesta por Dhody y Manral en draft-dhody-pce-pcep-service-aware-02 [258]. Esta métrica se define como latencia extremo a extremo, por lo que se ajusta las necesidades de una red WR-OBS.

Objeto OBJECTIVE FUNCTION (representado como <OF>) [253]: El PCE puede tener implementados distintos algoritmos de cálculo con distintos criterios de optimización. El IETF ha propuesto un objeto opcional denominado OBJECTIVE FUNCTION para que un cliente pueda escoger la función objetivo, es decir, el objetivo de optimización. Para ello, se ha de facilitar en dicho objeto un código, llamado código de función objetivo. En el caso de WR-OBS, se empleará este objeto para escoger el algoritmo implementado de encaminamiento y asignación de longitud de onda (RWA, *Routing and Wavelength Assignment*), como el AUR-*Exhaustive* o el *Dijkstra* con *First Fit* mencionados anteriormente.

Objeto MONITORING (representado como <MON>) [259]: Mediante el objeto opcional MONITORING se pueden pedir al PCE datos de rendimiento, tales como el tiempo de cálculo en el PCE, tiempo máximo de cálculo o el tiempo medio de cálculo. En una red WR-OBS se empleará este objeto únicamente cuando se deseen tomar medidas de rendimiento desde el cliente.

Objeto RESERVATION (representado como <RESERVATION>): Las redes basadas en conmutación de circuitos (MPLS, OTN, WSON…) emplean típicamente un protocolo de estado de enlace, como por ejemplo OSPF, para diseminar la topología, y extensiones de ingeniería de tráfico para, además de la topología diseminar las características de los enlaces, como la ocupación del ancho de banda, los grupos de riesgo, etc.. Las actualizaciones se envían de manera periódica, aunque en ocasiones se realiza el envío de la actualización del enlace en cuanto se ha producido la misma. El PCE se sirve de esta información para alimentar su base de datos de ingeniería de tráfico, sobre la cual se van a establecer los caminos. Estos protocolos están diseñados para redes con baja dinamicidad, en las que el tiempo entre el establecimiento de nuevos caminos está como mínimo en el orden de las decenas de segundo. Sin embargo, en una red WR-OBS la dinamicidad es muy elevada, y en un segundo se pueden establecer y tirar varias conexiones. En el mejor de los casos, la actualización con el envío de información se envía en cuanto se ha configurado el nodo, (por ejemplo tras la llegada del mensaje RESV si se emplea RSVP). Esto implica que, desde que se reserva el recurso, hasta que el PCE se percata de ello, el PCE considera que esos recursos están libres y por tanto pueden ser empleados por otras peticiones. Por tanto, surge la necesidad de poder indicar al PCE en la petición que se desea bloquear los recursos por un tiempo determinado. Sin embargo, el estándar PCEP no contempla actualmente ninguna forma de poder indicar dicho bloqueo. De esta forma, se propone un nuevo objeto, el objeto *Reservation*, cuya presencia indica que se quiere que se bloqueen los recursos calculados en esa petición para que otras peticiones no lo utilicen temporalmente.

El formato del objeto propuesto se muestra en la Figura 8-7, donde *Timer* es un número entero de 32 bits, en milisegundos, *Resource Type* el tipo de recurso a reservar (lambda, *slot* TDM, ancho de banda), un conjunto de *flags* (*flag* S, si se pone a 1, se han de bloquear también los grupos de riesgo compartidos, denominados *Shared Risk Link Group*, SRLG, *flag* N, si se activa, los recursos a bloquear son los nodos de la ruta, y el *flag* L, los recursos a

bloquear son los enlaces completos de la ruta) para poder aumentar el ámbito de los recursos a reservar, y un espacio para TLVs opcionales (para futuras extensiones). Al tener 32 bits para la reserva, y estar esta expresada en ms, se pueden reservar tiempos muy elevados (hasta aproximadamente 50 horas), superiores a los tiempos necesarios en WR-OBS. La extensión propuesta se ha presentado como *draft* en el IETF [260].

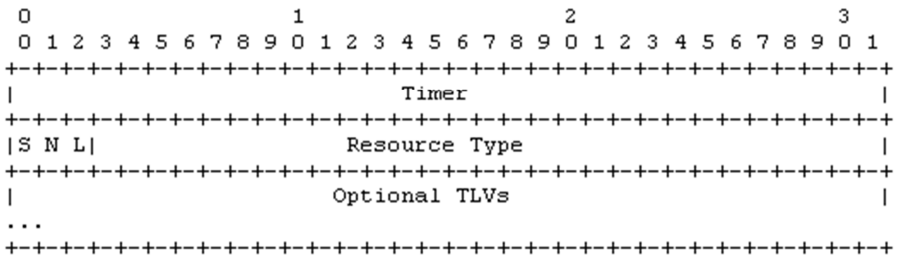


Figura 8-7 Formato del objeto RESERVATION

A continuación, en la Figura 8-8, se muestra una captura de *Wireshark*, con un ejemplo de petición de camino WR-OBS incluyendo los objetos anteriormente descritos. Para poder mostrar correctamente todos los objetos, se ha personalizado el disector (módulo en lenguaje C en el que se programa la lectura de mensajes de un protocolo determinado) del protocolo PCEP en *Wireshark*, incluyendo los elementos que no están en el estándar.

```
Path Computation Element communication Protocol
  PATH COMPUTATION REQUEST MESSAGE Header
      PCEP Version: 1
    + Flags: 0x00
      Message Type: PATH COMPUTATION REQUEST MESSAGE (3)
      Message length: 68
  RP object
      Object Class: RP OBJECT (2)
      Object Type: 1
    + Flags
      Object Length: 12
      Reserved: 0x00
    + Flags: 0x000001
      Requested ID Number: 0x00000004
  END-POINT object
      Object Class: END-POINT OBJECT (4)
      Object Type: 1
    + Flags
      Object Length: 12
      Source IPv4 Address: 172.16.101.1
      Destination IPv4 Address: 172.16.101.5
  BANDWIDTH object
      Object Class: BANDWIDTH OBJECT (5)
      Object Type: 1
    + Flags
      Object Length: 8
      Bandwidth: 10000000000,000000
  METRIC object
      Object Class: METRIC OBJECT (6)
      Object Type: 1
    + Flags
      Object Length: 12
      Reserved: 0
    + Flags: 0x00
      Type: Hop Counts (T=3)
      Metric Value: 0,000000
  OBJECTIVE FUNCTION object
      Object Class: OBJECTIVE FUNCTION OBJECT (OF) (21)
      Object Type: 1
    + Flags
      Object Length: 8
      OF-Code: AUR/Exhaustive (1001)
  RESERVATION object
      Object Class: RESERVATION OBJECT  (160)
      Object Type: 1
    + Flags
      Object Length: 12
      Timer: 10 ms
```

Figura 8-8 Captura de mensaje de PCEP *Request* para WR-OBS

En el ejemplo de la Figura 8-8 se realiza una petición para obtener la ruta de un *lightpath* entre dos nodos cuyas direcciones IP son 172.16.101.1 y 172.16.101.5. En la figura se muestra el contenido de cada uno de los objetos del mensaje *Request*. Por ejemplo, en el objeto REQUEST PARAMETERS (abreviado por *Wireshark* como RP) se puede ver el identificador de la petición, 4, que ha sido escogido por el cliente. El siguiente objeto que se muestra es el de tipo END POINTS, que incluye la dirección IP del nodo origen (*source IPv4 address,* de valor 172.16.101.1) y la dirección del nodo destino (*destination IPv4 Address*, de valor 172.16.101.5), es decir, de los extremos del *lightpath*. Después, en el objeto BANDWIDTH, se indica que se solicita un *lightpath* cuyo ancho de banda sea 10 Gbps. A continuación, en el objeto METRIC se indica la métrica que hay que emplear en el cálculo. En el ejemplo, se menciona que hay que emplear la métrica de número de saltos (*Hop Counts*, con valor estándar 3). Además de la métrica, en el objeto OBJECTIVE FUNCTION se indica que se quiere emplear la función objetivo número 1001 (no estándar), que se corresponde al algoritmo AUR-E. Finalmente, se indica que se quiere pre-reservar durante 10 ms mediante el objeto propuesto RESERVATION.

Formato de la respuesta

Mediante el mensaje Response del protocolo PCEP se puede enviar el resultado de un cálculo de ruta. Para una red WR-OBS se escoge emplear los objetos PCEP estándar, REQUEST PARAMETERS (similar al de la petición), NOPATH cuando no haya camino, METRIC para indicar la métrica del camino calculado, un EXPLICIT ROUTE OBJECT (ERO) para indicar la ruta (que se detalla más adelante), y un nuevo objeto propuesto, el RESERVATION CONF para identificar el bloqueo de recursos.

Objeto REQUEST PARAMETERS (representado como <RP>): Este objeto tiene como función identificar a qué petición pertenece la respuesta. Para ello, contendrá el mismo identificador que se envió en el mensaje *Request*. Hay que recordar que el ámbito de este identificador es la sesión PCC-PCE. En otras sesiones se puede repetir el número. Asimismo, un mismo mensaje Response puede contener respuestas a varias rutas. Para ello, ha de contener varios objetos RP. En el caso de WR-OBS, se decide emplear una única respuesta por mensaje *Response*. Otra alternativa sería agrupar respuestas, pero como se necesita responder cuanto antes, en cuanto se calcule el camino, se envía la respuesta en un mensaje inmediatamente.

Objeto METRIC (representado como <METRIC>): En la respuesta, el objeto METRIC contendrá el valor de una métrica del camino calculado. El PCE puede devolver varias métricas, para lo cual incluirá varios objetos de tipo METRIC. El PCE puede incluir en la respuesta más métricas que las que se han solicitado en la petición, con un carácter meramente informativo. Por ejemplo, se puede configurar el PCE para que por defecto responda siempre con el retardo extremo a extremo del camino calculado.

Objeto NOPATH (representado como <NOPATH>) [11]: Este objeto opcional indica que no se ha encontrado una ruta válida para la petición realizada. En él se indica la causa por la cual no se ha podido encontrar ruta, por ejemplo, si de debe a que no ha podido satisfacer alguna de las restricciones indicadas en la petición.

Objeto RESERVATION CONF (representado como <RESERVATION_CONF>): Mediante el objeto *Reservation* se puede solicitar al PCE que se bloqueen los recursos. Sin embargo, en una red WR-OBS, cuando se emplea LBS o UBS, en el momento de realizar la petición de camino al PCE, no se conoce la duración de la ráfaga (que será el tiempo durante el cual el LSP estará activo), por lo que hay que indicar al PCE un bloque de recursos durante un tiempo alto, y posteriormente cancelar la reserva en cuanto se termine de transmitir la ráfaga. En el caso de LBS, el tiempo de la reserva será superior a la duración de la transmisión de la ráfaga mayor posible. Para poder cancelar la reserva posteriormente, en la respuesta se incluye un identificador de la reserva realizada. Cuando

el nodo quiera cancelar la reserva, porque ya ha terminado de transmitir los datos, enviará una notificación al PCE incluyendo el identificador de la reserva. Se ha propuesto el objeto RESERVATION CONF para indicar que se ha realizado la reserva e incluir el identificador de reserva. Además, se indica el tiempo durante el cual el PCE realiza la reserva. El tiempo puede ser diferente al indicado en la petición, ya que puede estar limitado por política, para evitar ataques malintencionados. Por ejemplo, se puede configurar un tiempo máximo de reserva en el PCE. En la Figura 8-9 se muestra la codificación del objeto propuesto, presentado como *draft* en el IETF [260].

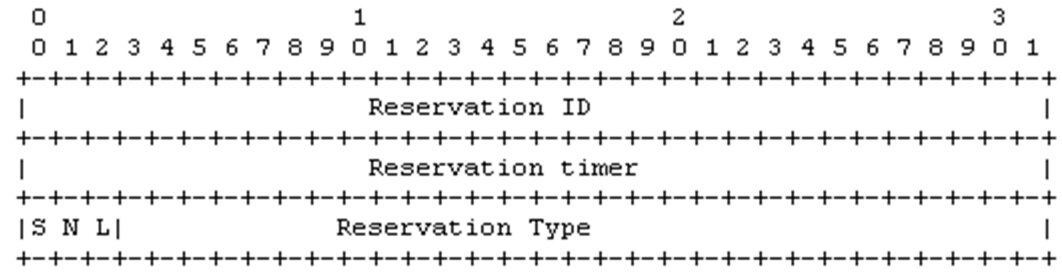


Figura 8-9 Formato del objeto propuesto RESERVATION CONF

Formato de la ruta

La ruta del *lightpath* a establecer se codifica mediante un objeto de tipo ERO (EXPLICIT ROUTE OBJECT). Este objeto ERO se incluye en el mensaje de *Request* cuando se ha encontrado un camino. Su contenido es similar al del objeto ERO de RSVP con extensiones de ingeniería de tráfico (RSVP-TE) [83], y se forma a base de subobjetos ERO. Para el caso de WR-OBS, la elección que se ha realizado es la siguiente:

- Se emplea un subobjeto de tipo *Unnumbered Interface ID Subobject* [255] por cada nodo-fibra de salida. Este subobjeto contiene dos números de 32 bits, el primero de ellos, denominado *Router ID*, identifica el nodo de manera unívoca en toda la red, y el segundo denominado *Interface ID*, que identifica de manera unívoca el interfaz dentro de un nodo. Es decir, este identificador solo tiene significado local, pero la combinación de los dos identificadores es única en la red. El primer subobjeto de la ruta, en el caso de WR-OBS, será siempre un objeto de tipo *Unnumbered Interface ID*, indicando la primera fibra de salida de la ruta.
- Después de cada interfaz no numerada, se propone incluir un subobjeto del tipo *Label ERO*. En este subobjeto, como etiqueta se emplea una etiqueta generalizada, cuyo contenido es el valor de la longitud de onda definido anteriormente (ver Figura 8-6)
- El último nodo se representará como un subobjeto IPv4. Así, el fin de la ruta es fácilmente reconocible.

En la Figura 8-9 se muestra una captura de pantalla con el ejemplo de un objeto del tipo ERO utilizable en WR-OBS. El primer subobjeto que se muestra es de tipo *Unnumbered Interface ID*, que identifica, mediante el campo *Router ID* 172.16.101.1 al primer nodo de la ruta. Con el campo *Interface ID*, en este caso de valor 0, se identifica la fibra de salida. A continuación, se incluye un subobjeto de tipo *Label Control*, que identifica de manera explícita la longitud de onda a utilizar en la fibra indicada anteriormente. Después se indica de nuevo mediante un par de subobjetos el siguiente nodo, fibra de salida y longitud de onda. Finalmente, el último nodo de la ruta del lightpath se indica con un subobjeto de tipo *IPv4 Prefix*.

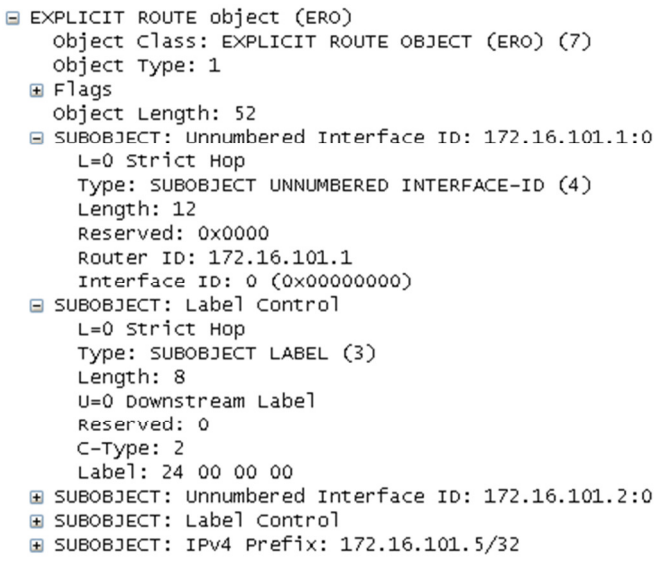


Figura 8-10 Formato de ruta de WR-OBS

Extensión para cancelación de bloqueo de recursos en PCE

Por último, cuando se emplea LBS o UBS es necesario poder cancelar el bloqueo de recursos, para evitar que estos queden inutilizables durante largo tiempo. Para ello se propone un nuevo tipo de notificación, en la que se incluya una TLV cuyo valor sea el identificador de 32 bits que se devuelve en el mensaje *Request* en el objeto RESERVATION CONF. Para evitar una sobrecarga de mensajes, esta notificación de cancelación de bloqueo de reservas no se confirma. No hay que olvidar que PCEP funciona sobre TCP, un protocolo que garantiza la transmisión de los datos.

8.3 Diseño del PCE

En los apartados anteriores se ha visto cómo es la arquitectura WR-OBS con PCE y cómo es el protocolo que comunica a un nodo con el PCE. En esta sección se procede a describir el diseño del PCE para WR-OBS, justificando las principales decisiones. El PCE consta de un conjunto de módulos, mostrados en la Figura 8-11, que se comunican entre sí. Estos módulos corren en paralelo, por lo que se tendrá una aplicación multi-hilo. En este apartado se explica brevemente la funcionalidad de cada módulo.

8.3.1 Servidor de sesiones PCEP

Este módulo se encarga de escuchar en el puerto 4189 y establecer una sesión PCC-PCE por cada conexión TCP entrante.

8.3.2 Sesión PCC-PCE

Este es el módulo encargado de mantener una sesión entre el PCE y un cliente PCC que se le conecte. Este módulo sigue el procedimiento de inicio de sesión explicado en la sección 8.2. Una vez iniciada la sesión, se inician dos procesos, el KA y el DT. El proceso KA se encarga de enviar periódicamente mensajes *Keepalive* para mantener viva la sesión. El proceso DT mantiene el temporizador *DeadTimer*, que se resetea tras la llegada de cualquier mensaje PCEP. Si vence el temporizador, se asume que la sesión está muerta y se finaliza la sesión. Después de arrancar estos dos procesos, la sesión PCC-PCE se dedica a escuchar mensajes PCEP. Cada vez que llega un mensaje PCEP, se decodifica. En caso de que el mensaje sea una petición, se envía a la cola de peticiones. De esta forma, de manera inmediata se siguen escuchando mensajes. Así, no hay que esperar a que se procese la petición requerida en el mensaje para continuar.

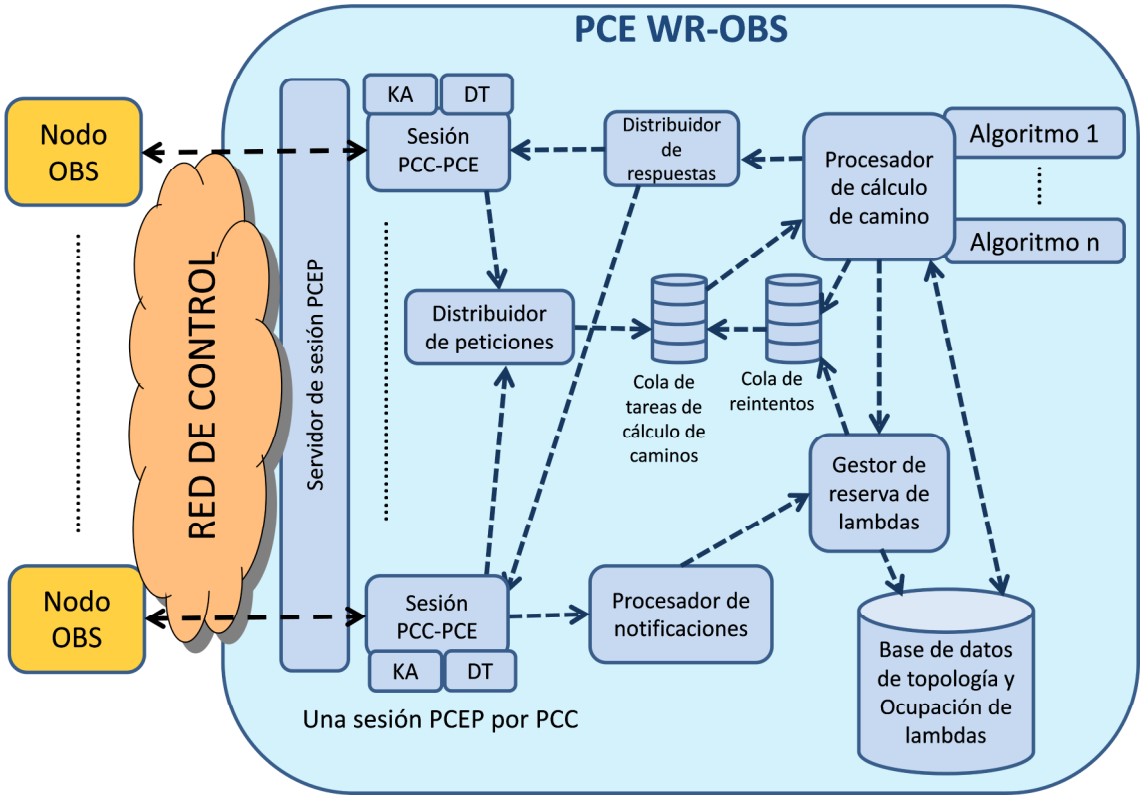


Figura 8-11 Diseño del PCE-WROBS

8.3.3 Distribuidor de peticiones

El distribuidor de peticiones es el encargado de recibir los mensajes de tipo *Request* de las distintas sesiones, desgranarlos (ya que potencialmente un mensaje *Request* puede contener varias peticiones), y crear una tarea de cálculo que se introduce en la cola de tareas de cálculo, común para todas las sesiones PCEP. La tarea de cálculo contiene la petición de camino.

8.3.4 Cola de tareas de cálculo de camino

Como se ha mencionado, cada vez que llega un mensaje de tipo *PCReq*, que contiene una o varias peticiones de cálculo de una ruta de un *lightpath*, las tareas de cálculo individuales se introducen en la cola de tareas de cálculo de camino. Esta cola de tareas es común para todas las sesiones PCEP establecidas con el PCE.

8.3.5 Cola de reintentos

Cuando no se encuentra una combinación de camino y longitud de onda disponible para la petición de cálculo, se puede introducir dicha petición en la cola de reintentos, con la esperanza de que, más adelante, cuando se liberen recursos en la red, se pueda encontrar una combinación de camino y longitud de onda. Kozlovski y Bayvel [261] y de Miguel [248] demuestran en sus estudios los beneficios de volver a planificar una petición de RWA, siempre y cuando el tiempo que se tarda en procesar la petición se mantenga acotado. Así, el distribuidor de peticiones, cuando se liberen recursos en el PCE, procede a mover las tareas de cálculo de la cola de reintentos a la cola de tareas de cálculo de camino si no se ha excedido el tiempo de espera en el PCE para la tarea de cálculo en cuestión.

8.3.6 Procesador de cálculo de caminos

Este es el módulo encargado de realizar los cálculos. A partir de los parámetros de la petición, principalmente la función objetivo, escogerá el algoritmo deseado y lo aplicará.

8.3.7 Base de datos de ingeniería de tráfico

La base de datos de ingeniería de tráfico contiene un grafo con los nodos y enlaces de la red WR-OBS. Cada nodo se identificará con una dirección IP de 32 bits, y cada una de sus fibras de salida con un identificador, también de 32 bits. La información de ingeniería de tráfico que se guarda en los enlaces son dos mapas de bits, uno que representa la ocupación de las lambdas en la red según el conocimiento que tenga el PCE (por ejemplo por OSPF, *Open Shortest Path First*, u otro medio), y otro que representa las lambdas bloqueadas por el PCE en dicho enlace. Hay un bit por cada lambda habilitada en la red. Un valor típico en una red DWDM son 80 lambdas.

8.3.8 Gestor de reservas

El gestor de reservas se encarga de bloquear temporalmente los recursos de una ruta que se ha calculado correctamente. Para ello, actuará directamente sobre la base de datos de ingeniería de tráfico escribiendo en los mapas de bits de lambdas bloqueadas. Asimismo, este módulo se encarga de terminar con el bloqueo, bien sea porque ha vencido el tiempo de la reserva, o bien ha llegado una notificación.

8.4 Implementación del PCE

El elemento de cálculo de caminos se ha implementado en el lenguaje de programación Java, empleando su versión 1.6 [262]. El motivo de dicha elección ha sido en primer lugar su fortaleza para manejar hilos, paralelizar tareas y gestionar la sincronización entre las mismas, en segundo lugar la facilidad para estructurar el código y documentarlo y por último su portabilidad, que nos permitirá su ejecución bajo, Linux, Windows y Mac. De esta forma se implementa un prototipo relativamente complejo y se hace funcionar en múltiples plataformas.

El entorno de desarrollo ha sido Eclipse, en su versión Helios [263]. En la Figura 8-12 se muestra una captura de pantalla de dicho entorno de desarrollo, donde se puede ver la estructura de paquetes implementada. El prototipo de PCE se ha implementado por completo, sin partir de la base de ninguna implementación previa. Únicamente, se ha empleado la ayuda de una biblioteca de grafos, como se explica a continuación.

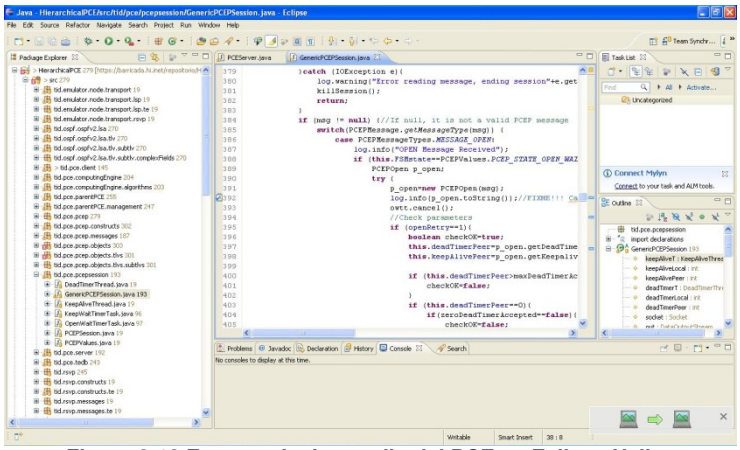


Figura 8-12 Entorno de desarrollo del PCE en Eclipse Helios

8.4.1 Implementación de la biblioteca de codificación/decodificación PCEP

Se ha implementado un conjunto de clases para facilitar la codificación y decodificación de mensajes del protocolo PCEP. La aproximación que se ha seguido es realizar una distinción entre mensajes, objetos, construcciones y TLVs. Para cada una de ellas se ha creado una clase abstracta en la que se han incluido las funciones comunes, como la codificación/decodificación de la cabecera. El protocolo PCEP en muchos casos reutiliza objetos de RSVP-TE [83] y OSPF-TE [90], por lo que asimismo se han implementado los objetos necesarios de estos protocolos.

Mensajes PCEP

Como se ha mencionado anteriormente, el protocolo PCEP se basa en el intercambio de mensajes PCEP, que son un conjunto de bytes, en los que hay una cabecera y un conjunto de objetos, que a su vez pueden contener TLVs (campos de longitud variable). Se ha creado una clase para cada mensaje PCEP, objeto PCEP y TLV. Por tanto, cuando hay que decodificar un mensaje hay que crear una clase del tipo de mensaje correspondiente y pasarle el conjunto de bytes. Para saber qué tipo de mensaje es se ha creado una función estática que a partir de un conjunto de bytes obtiene el tipo de mensaje. De esta forma antes de crear la clase específica, se obtiene el tipo de mensaje.

Cada una de las clases contiene un campo para cada objeto de interés. En el proceso de codificación, a partir del valor de estos objetos y el tipo de mensaje, se crea un conjunto de bytes. Se ha seguido una misma metodología para la codificación en todas las clases de mensajes:

1. Se codifican todos los objetos del mensaje
2. Se calcula la longitud total a partir de la longitud de los objetos más 4 bytes de la cabecera PCEP
3. Se crea un *array* de bytes
4. Se codifica la cabecera
5. Se codifica cada uno de los objetos. Este es un procedimiento recursivo, ya que cada objeto tiene método para codificar.

Algoritmos de cálculo de ruta

Los algoritmos de encaminamiento y asignación de longitud de onda se han implementado tomando como punto de partida la biblioteca jgraphT [264], que cuenta con implementaciones de los algoritmos básicos más conocidos (*Dijkstra*, *Bellman-Ford*, etc.). Se ha realizado la implementación de los algoritmos AUR-E (*Adaptive Unconstrained Routing Exhaustive*) [249] y SP-FF (*Shortest Path First Fit*) [251].

8.5 Conclusiones

En este capítulo se ha descrito una arquitectura WR-OBS con elementos definidos por los estándares actuales, como son el PCE y RSVP. Se ha particularizado el protocolo PCEP para WR-OBS y se han propuesto extensiones protocolares, que además se han implementado. Se ha diseñado un PCE especializado en WR-OBS y se ha prototipado empleando Java. La principal conclusión de este trabajo es la adecuación de la arquitectura de PCE propuesta en el IETF para las redes WR-OBS. En el próximo capítulo se realizara una evaluación de prestaciones del PCE en un entorno WR-OBS emulado.

Capítulo 9

Evaluación experimental del rendimiento de PCE para WR-OBS

En el capítulo anterior se ha estudiado una arquitectura de WR-OBS con elementos y protocolos basados en los estándares actuales. Dentro de esta arquitectura, el PCE es el elemento encargado de realizar el cálculo de rutas, cuyo protocolo de comunicación, denominado PCEP, se ha extendido para soportar WR-OBS. En este contexto, en el capítulo anterior se realizó el diseño de un PCE para WR-OBS. Además del diseño, se realizó un prototipo del mismo. Este capítulo está dedicado a realizar una evaluación experimental de dicho prototipo. Se evalúa su rendimiento en un entorno emulado, midiendo parámetros tales como el retardo de borde, la probabilidad de bloqueo o el tiempo de cálculo. Cabe destacar que no se trata de una simulación, ya que las peticiones al PCE se generan y responden en tiempo real.

En el capítulo, en primer lugar, se describe el entorno de emulación en el cual se realizan las pruebas con el PCE implementado. En concreto, se detalla el funcionamiento del emulador de WR-OBS desarrollado y la metodología de la emulación. En las primeras pruebas se detectó que el protocolo de transporte TCP, empleado para transportar los mensajes del protocolo PCEP (y por tanto las peticiones al PCE), tiene una gran influencia en los resultados. Así pues, se dedica un apartado a explicar dicha influencia, que se debe especialmente al algoritmo de Nagle de TCP, y a continuación se describen los experimentos que se han realizado. Tres parámetros de gran interés en una red WR-OBS son el retardo de borde (debido a su relación con el retardo de extremo a extremo), el número medio de *lightpaths* establecidos y la probabilidad de bloqueo. Para el retardo de borde y el número medio de *lightpaths* se han obtenido expresiones analíticas para poder validar los resultados experimentales. Dichas expresiones analíticas, elaboradas a partir de los estudios realizados por de Miguel [248], requieren de la caracterización de dos procesos, el de llegada de peticiones al PCE y el del tiempo de procesamiento de las peticiones en el PCE. Así, se dedican dos secciones al análisis de sendos procesos. Posteriormente se dedica una sección para cada evaluación experimental, es decir, una sección para la evaluación del retardo de borde, otra para la evaluación del número medio de *lightpaths* y finalmente otra para la evaluación de la probabilidad de bloqueo.

9.1 Entorno experimental emulado

En el capítulo anterior se ha implementado un prototipo de PCE para WR-OBS, capaz de responder en tiempo real a peticiones de cálculo de rutas de *lightpaths*. Para poder evaluar experimentalmente el rendimiento que se obtendría en una red OBS empleando dicho PCE se implementa un entorno experimental emulado. Así, dicho entorno consta de un emulador de red WR-OBS, en el que se emula el proceso de agregación de tráfico y generación de las ráfagas. Dicho emulador se comunica en tiempo real, mediante el

protocolo PCEP extendido, con el PCE implementado y es capaz de tomar distintas estadísticas así como medidas de tiempos.

En este apartado, en primer lugar, se detalla el funcionamiento del emulador de WR-OBS implementado. A continuación se describe el *set-up* experimental concreto que se ha utilizado para los experimentos de este capítulo. Finalmente se muestran las distintas topologías de red WR-OBS que se pueden cargar en el emulador y que se han utilizado en los experimentos.

9.1.1 Emulador de WR-OBS

Una red WR-OBS, como se ha comentado en diversos capítulos de la tesis, consta de un conjunto de nodos de borde, que agregan el tráfico y lo envían en ráfagas a otros nodos de borde a través de los nodos de núcleo. Por tanto, el emulador de red WR-OBS ha de reproducir el comportamiento de los nodos. Una posible alternativa para la emulación es generar el tráfico que se inyecta en cada nodo de borde y realizar la agregación del mismo en tiempo real. Sin embargo, debido a que el objetivo para una red OBS es transmitir datos en el orden de Gbit/s, la generación de tráfico a dichas velocidades y su agregación vía software para generar ráfagas en tiempo real, para varios pares origen destino, tiene unos requisitos de hardware muy elevados. Con el fin de que el emulador sea capaz de funcionar en un entorno con hardware de propósito general, se ha elegido generar eventos en tiempo real que indiquen el vencimiento del temporizador de ensamblado para cada par origen-destino, y calcular analíticamente el tráfico que se generaría en el intervalo de tiempo considerado, que dependerá del tipo de método de generación de ráfaga, LBS o FBAT. De esta forma, no es necesario generar tráfico a alta velocidad por cada par origen-destino. Cuando vence el temporizador de ensamblado, en tiempo real se construye un mensaje del protocolo PCEP de tipo *Request*, cuyos detalles concretos dependerán de si es LBS o FBAT, y se envía al PCE. Para realizar esta comunicación con el PCE se emplea un único cliente PCC (*Path Computation Client*) a compartir entre todos nodos emulados. El empleo de un PCC compartido permite reducir las necesidades de procesamiento en el emulador frente a mantener un PCC por nodo.

El establecimiento del *lightpath* y la transmisión de la ráfaga son asimismo emulados. De esta forma, cuando haya que establecer un *lightpath*, el proceso se mantiene en espera durante el tiempo que tardaría el protocolo RSVP en realizar dicho establecimiento. Para simplificar, se asumirá que dicho tiempo es dos veces el retardo de propagación de la ruta, que será proporcionado por el PCE en la respuesta. Igualmente, para la transmisión, el emulador calcula el tamaño de la ráfaga a partir del tiempo real que ha estado agregando datos (como se ha mencionado, el intervalo de tiempo en el que se agrega depende del mecanismo concreto) y a partir de dicho tamaño obtiene el tiempo que debería estar transmitiendo. La metodología empleada para calcular el tamaño de las ráfagas parte del método de generación de ráfagas de Rostami y Wolisz [265] que se explica en profundidad posteriormente en este apartado.

Con el fin de simplificar el emulador, toda la tarea de mantener la topología y la ocupación de las longitudes de onda recae en el PCE. De esta forma, el PCE puede responder con la ruta del *lightpath* en el caso de que haya encontrado recursos libres, o bien, responder con un objeto NOPATH en el caso de que todos los recursos estén ocupados. Con el fin de mantener la información de ingeniería de tráfico (en este caso, ocupación de longitudes de onda) correctamente actualizada, en el caso de FBAT en la petición se indica la duración de la ráfaga, y en el caso de LBS, el emulador reserva de manera indefinida los recursos del *lightpath* e indica posteriormente, de forma explícita, mediante una notificación que finalice la reserva de recursos. Estos mecanismos se verán con más detalle en el siguiente subapartado, cuando se detalle el funcionamiento del emulador con LBS y FBAT.

Diseño del emulador

En este subapartado se describe brevemente el diseño del emulador de WR-OBS que cumple con la funcionalidad descrita anteriormente. La Figura 9-1 muestra el diseño del entorno emulado, con el emulador WR-OBS a la izquierda, y el PCE a la derecha. El emulador cuenta con un módulo PCC, que mantiene la sesión del protocolo PCEP entre cliente y servidor. Se mantiene un módulo por cada par origen-destino, que se encargará del proceso de agregación de tráfico, preparar la petición al PCE cuando sea oportuno y emular el establecimiento/liberación del *lightpath*. El generador de eventos está encargado de avisar a los módulos par origen-destino cuando vence el temporizador de ensamblado. Este módulo es de vital importancia, ya que la planificación de eventos se ha de realizar en tiempo real y cualquier retraso o adelanto supondrá un error en las medidas. Para ayudar a los módulos par origen-destino a calcular cuánto tráfico han agregado en la ráfaga se incluye un generador de números aleatorios. Por otro lado, los módulos "controlador emulación" y "estadísticas" son los encargados de lanzar y parar la emulación y recopilar las distintas estadísticas respectivamente. El PCE, a la derecha de la figura, se configura mediante un fichero en formato XML en el que se describe en detalle la topología de red.

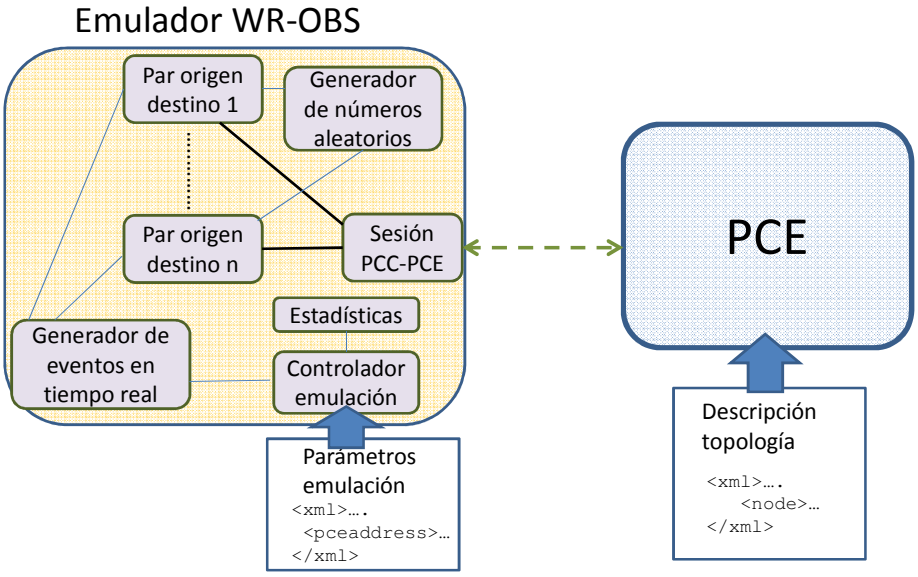


Figura 9-1 Diseño del entorno de emulación WR-OBS y PCE

Emulación del mecanismo de construcción de ráfagas LBS

Una vez que se ha visto el funcionamiento general y el diseño del emulador de WR-OBS, se describe el detalle del funcionamiento de la emulación con el mecanismo de construcción de ráfagas LBS.

En cuanto se inicia la emulación, se abre una sesión mediante el protocolo PCEP entre el PCC y el PCE, que se compartirá entre todos los pares origen-destino. La comunicación se inicia en el lado del PCC, que ha de conocer la dirección del PCE.

A continuación se planifica un evento por cada par origen destino i en el instante de tiempo $T_{0i} + T$, siendo T el valor del temporizador de ensamblado y T_{0i} el instante de tiempo en el que llegaría el primer paquete del par nodo de borde origen-nodo de borde destino i. Para simplificar el procedimiento de inicio, se ordenan los pares origen-destino y la recepción del primer paquete en cada caso se fija a un ms más que el par anterior.

Cuando vence el temporizador, se ejecuta un proceso independiente que se encargará en adelante de esa ráfaga, que se denomina "tarea de fin de temporizador". Dicho proceso crea, en primer lugar, el mensaje *Request* de acuerdo a la especificación del protocolo PCEP. Acto seguido, a través del PCC, que mantiene la sesión PCEP que se comparte entre todos

los pares origen destino, envía dicho mensaje al PCE. En el mensaje se indica que se ha de reservar los recursos de manera indefinida, aunque en la práctica lo que realmente se hace es solicitar reservarlos durante un tiempo muy elevado. Para ello, se hace uso del objeto propuesto RESERVATION, y se incluye un valor de temporizador muy alto (por ejemplo de varias horas). En cuanto se ha enviado el mensaje *Request* al PCE, el proceso se queda bloqueado a la espera de la respuesta del PCE, con un tiempo máximo de espera configurable (que se fija en las pruebas realizadas a 100 ms).

Una vez que ha respondido el PCE con un mensaje de tipo *Response*, el PCC informa de la respuesta y, en el caso de que la respuesta contenga una ruta (en concreto, un objeto de tipo ERO), se emula el establecimiento del LSP. Para simplificar el proceso, el PCE devuelve en el mensaje *Response* un objeto de tipo METRIC que contiene el valor del retardo de propagación del camino calculado. Además, se incluye el objeto RESERVATION_CONF, con un identificador de la reserva de los recursos del *lightpath* en el PCE, para que se puedan liberar más adelante. Se asume que en la red OBS el mensaje RSVP sigue la ruta de datos, y se realiza la simplificación de que el tiempo de establecimiento es de dos veces el valor de la métrica de retardo de propagación que llegó en el objeto METRIC, es decir, se desprecia el tiempo de transmisión, procesamiento y configuración en cada nodo. El proceso se duerme durante dicho tiempo, emulando el establecimiento del *lightpath*.

Una vez ha transcurrido el tiempo de establecimiento, se considera que el camino está utilizable y se da por finalizado el proceso de creación de ráfaga. Justo a continuación se planifica la ejecución de una nueva tarea de fin de temporizador en T ms (el tiempo hasta la llegada del siguiente paquete se considera despreciable) y vuelve a comenzar el ciclo.

Una vez que se ha finalizado la creación de la ráfaga, se puede conocer el tiempo total de agregación, que se obtiene midiendo el tiempo total real transcurrido. A partir del tiempo total de agregación se emplea el modelo de Rostami y Wolisz [265], que se describe a continuación, para calcular el tráfico que se ha recibido en dicho tiempo de agregación para ese par origen-destino. Una vez conocido el tamaño de la ráfaga óptica, se calcula su tiempo de transmisión. Al igual que en el caso del establecimiento, el proceso se duerme durante el tiempo de envío de la ráfaga.

Una vez que ha transcurrido el tiempo de transmisión de la ráfaga, se envía una notificación al PCE indicando que puede liberar el bloqueo de los recursos (longitudes de onda) de la petición realizada anteriormente. Para ello, incluye en el mensaje de notificación el identificador de la reserva contenido en el mensaje de respuesta.

Emulación del mecanismo de construcción de ráfagas FBAT

En este subapartado se detalla la emulación con otro mecanismo de construcción de ráfagas (*Fixed Burst Assembly Timer*). Al igual que en el caso de la emulación de LBS, la emulación de FBAT emplea una única sesión con el PCE compartida entre todos los pares origen destino. De la misma forma que en LBS, se comienza planificando un evento de fin de temporizador por cada par origen destino i en $T_{0i} + T$, siendo T el valor del temporizador de ensamblado y T_{0i} el instante de tiempo en el que llegaría el primer paquete a un par nodo de borde origen-nodo de borde destino.

Cuando vence el temporizador se considera en FBAT que la ráfaga se ha terminado de ensamblar. Por tanto, en cuanto vence el temporizador, se planifica un nuevo evento para indicar el fin del ensamblado de la siguiente ráfaga (por simplicidad, se asume que el tiempo entre el fin del ensamblador de una ráfaga y la llegada de un nuevo paquete para el par origen-destino en cuestión es despreciable).

Cuando la ráfaga se ha terminado de ensamblar, se puede medir el tiempo de agregación, que en teoría debería ser igual al temporizador de ensamblado nominal. Sin

embargo, al tratarse de un proceso funcionando en tiempo real y ejecutándose a la vez que otros procesos habrá unas ligeras diferencias con el valor teórico. Una vez conocido el tiempo de agregación, se puede emplear el método de Rostami y Wolisz [265] para calcular el tamaño de la ráfaga. Acto seguido se crea el mensaje del protocolo PCEP de tipo *Request*. En dicho mensaje se incluye un objeto RESERVATION en el que se indica el tiempo que ha reservar el PCE los recursos del *lightpath*. Como en el caso de FBAT se conoce el tamaño de la ráfaga antes de enviar la petición al PCE, el tiempo de reserva del *lightpath* se ajusta a la duración de la ráfaga. Una vez construido el mensaje, se envía dicho mensaje a través del PCC compartido al PCE.

Una vez llega la respuesta, se realiza una emulación del establecimiento y envío de ráfaga como en el caso de LBS. Sin embargo, al contrario que en LBS, al terminar de enviarse la ráfaga no hay que notificar al PCE, ya que anteriormente se ha indicado al PCE el tiempo de reserva del *lightpath* necesario para realizar la transmisión de la ráfaga.

Modelo de generación de ráfagas

Tal y como se ha mencionado, el tamaño de las ráfagas se calcula a partir del modelo de generación de ráfagas propuesto por Rostami y Wolisz [265]. Dicho modelo proporciona expresiones analíticas para calcular tanto la longitud de las ráfagas como el tiempo entre ráfagas para varios algoritmos. En concreto, el método empleado en esta tesis toma como punto de partida el algoritmo 1 del modelo de Rostami y Wolisz, que se basa en asumir un proceso de llegada de paquetes de *Poisson* al ensamblador.

La descripción del algoritmo particularizado para WR-OBS (tanto FBAT como LBS) para calcular el tamaño de una ráfaga dada una tasa de llegada de paquetes λ (paquetes/s), con paquetes de longitud media μ^{-1} bytes, tamaño mínimo de ráfaga L_{min} y tiempo de agregación medido experimentalmente t_{aggr}, se muestra en la Figura 9-2.

Dados λ, μ, t_{aggr}
$n \leftarrow \text{Poisson}(\lambda \cdot t_{aggr})$
$x \leftarrow \text{Erlang}(n+1, (n+1)\mu^{-1})$
si $x < L_{min}$
    $x \leftarrow L_{min}$
fin si

Figura 9-2 Algoritmo de cálculo del tamaño de ráfaga en bytes (x)

En primer lugar se calcula el número de paquetes recibidos en el intervalo de tiempo de agregación (n). En el caso del emulador empleado en esta tesis, para LBS el valor del intervalo de agregación t_{aggr} se mide experimentalmente en tiempo real, desde que vence el temporizador de ensamblado, hasta que se ha establecido el *lightpath*. En el caso de FBAT, el intervalo de agregación, t_{aggr}, coincide (en teoría) con el valor del temporizador de ensamblado, pero de nuevo también se mide en tiempo real por las razones anteriormente expuestas. Una vez conocido el número de paquetes recibidos, a partir de una caracterización del tamaño de los paquetes, se calcula el número de bytes recibidos (x) en el intervalo de tiempo de agregación.

Implementación del emulador

En este apartado se comentan los aspectos más relevantes en cuanto a la implementación del emulador diseñado y que pueden tener impacto en el rendimiento y los resultados. En primer lugar, el emulador se ha implementado como un proceso multi-hilo en Java 1.6. La planificación de eventos en tiempo real, como ocurre en el caso de planificar el evento de fin de temporizador, se realiza empleando un ejecutor de tareas, en concreto, con la clase *ScheduledThreadPoolExecutor*, que dispone de un *pool* de *threads* para realizar tareas simultáneamente y una precisión en el orden de milisegundos. En concreto, se ha escogido emplear $2K$ *threads*, siendo K el número de pares origen-destino.

La emulación del establecimiento de caminos y transmisión de ráfagas se realiza bloqueando el proceso durante un tiempo determinado. Para realizar la espera se ha empleado el método *sleep*, que, en la implementación de Java empleada, tiene una granularidad de milisegundos.

En el emulador se mide en tiempo real el tiempo de agregación. Para ello, se emplea la función *System.nanotime* de Java a través de la cual se obtiene la medida de tiempo relativa que más precisión puede dar el sistema operativo, y puede usarse para obtener diferencias entre instantes de tiempo. La precisión concreta depende de la máquina en la que se use (típicamente, al menos en el orden del milisegundo).

Para calcular el tamaño de la ráfaga con el método de la Figura 9-2 es necesario disponer de un generador de números aleatorios, una distribución de Poisson y una distribución de Erlang. Para ello, se ha empleado Colt [266], una biblioteca de código abierto para cálculo científico y técnico de alto rendimiento desarrollada por el CERN. En concreto, se emplea el paquete *cern.jet.random* y *cern.jet.random.engine*, en los que están implementados varios generadores de números aleatorios y varias distribuciones. Dentro de los generadores disponibles, se escoge el *Mersene Twister* (MT19937) [267], en su versión de números de 32 bits. Para obtener una distribución de Poisson, se emplea la clase *cern.jet.random.Poisson*, basada en el método de Zechner [268]. Finalmente, la distribución de *Erlang* se obtiene a través de la clase *cern.jet.random.Distributions*.

9.1.2 *Set-up* experimental

Para realizar la evaluación experimental con el emulador WR-OBS y el prototipo de PCE se ha elaborado un *set-up* experimental, que se muestra en la Figura 9-3. Dicho *set-up* consta de dos máquinas unidas por cable UTP Categoría 5 y un *switch Fast Ethernet*. En una de las máquinas, de nombre PCE, con IP 172.16.105.5, se ejecuta el software del prototipo de PCE, que responde peticiones de rutas de *lightpaths* para las ráfagas ópticas en tiempo real. En la otra máquina, de nombre OBSTester, con IP 172.16.104.201, corre el emulador de red WR-OBS que se ha descrito en anteriormente y desde el cual se generan las peticiones al PCE.

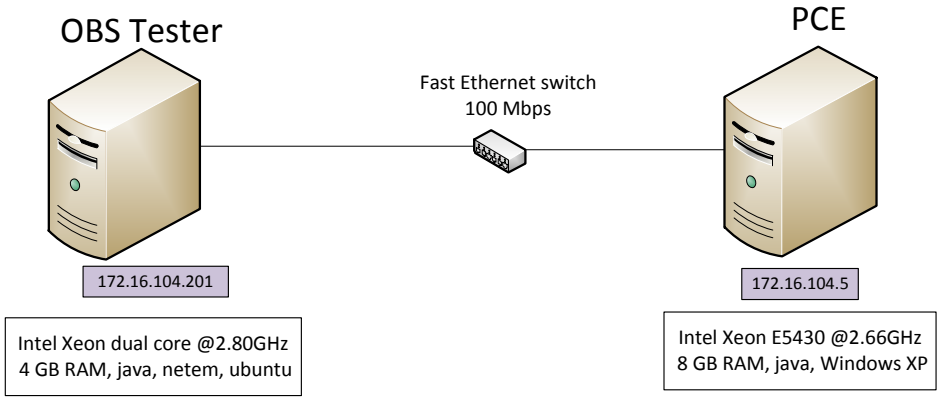


Figura 9-3 *Set-up* experimental PCE para WR-OBS

Se ha utilizado como máquina para el PCE un servidor Intel Xeon E5430 (*Quad Core*) a 2.66 GHz con 8 GB de RAM, y Windows XP. En este servidor corre únicamente el software del PCE, que es capaz de cargar distintas topologías de red para emular distintas redes. En el siguiente subapartado se detallan las topologías de red que se han empleado en los experimentos.

Para ejecutar el emulador de red OBS se ha utilizado una única máquina, con un procesador *Intel Xeon dual core* a 2.80 GHz. En él, como se ha explicado anteriormente, se emula el funcionamiento de los nodos de borde, el proceso de creación de ráfagas con los métodos LBS y FBAT y la reserva del camino.

En un entorno real, entre los nodos WR-OBS y el PCE habrá un cierto retardo. En el experimento, se emula el retardo entre los nodos WR-OBS emulados y el PCE utilizando la herramienta *netem* [269]. Esta herramienta introduce retardos de manera unidireccional, en la cola de salida de la interfaz de red que se escoja, en nuestro caso la interfaz *eth1*. Al emular todos los nodos en una misma máquina, el retardo entre cada nodo WR-OBS y el PCE es el mismo. En el entorno experimental que se ha construido, entre la máquina donde corre el emulador WR-OBS y el PCE hay un RTT inicialmente de aproximadamente 0,1 ms, ya que sólo hay un *switch Fast Ethernet* entre medias. Mediante la herramienta *netem* se introducen 4,9 ms a la salida de la maquina OBSTester. Para introducir dicho retardo, el comando que se ha utilizado es:

```
# tc qdisc add dev eth0 root netem delay 100ms
```

Una vez lanzado *netem*, se comprueba mediante un ping que el retardo se sitúa alrededor de 5 ms, como se muestra en la Figura 9-4.

```
ogondio@autoslocos:~$ ping 172.16.104.5
PING 172.16.104.5 (172.16.104.5) 56(84) bytes of data.
64 bytes from 172.16.104.5: icmp_seq=1 ttl=128 time=5.03 ms
64 bytes from 172.16.104.5: icmp_seq=2 ttl=128 time=5.01 ms
64 bytes from 172.16.104.5: icmp_seq=3 ttl=128 time=5.02 ms
64 bytes from 172.16.104.5: icmp_seq=4 ttl=128 time=5.00 ms
64 bytes from 172.16.104.5: icmp_seq=5 ttl=128 time=5.00 ms
```

Figura 9-4 *Ping* entre máquina OBSTester y PCE

Finalmente, en la máquina del PCE se ha instalado el software *Wireshark* [254] para capturar todos los paquetes que llegan por la interfaz de red (*eth1* en el *set-up*) y poder complementar los experimentos.

9.1.3 Escenarios de red emulados

El prototipo de PCE tiene la capacidad de cargar topologías de red WR-OBS descritas en formato XML. Para los experimentos que se llevan a cabo en este capítulo de la tesis se han escogido 3 topologías de red. En cada una de ellas, los nodos tienen capacidad de actuar de nodo de borde y nodo de núcleo simultáneamente (es decir, pueden generar ráfagas y encaminarlas). En cada experimento concreto se elige qué nodos agregan tráfico, haciendo la tarea de nodo de borde y de núcleo simultáneamente. En cada una de las topologías se ha asignado una dirección IP de 32 bits a cada nodo y un identificador, también de 32 bits a cada fibra. La primera topología que se emplea es la red de ocho nodos mostrada en la Figura 9-5, que sirve como ejemplo de red sencilla. Se empleará en experimentos en los que muestre un caso de complejidad del cálculo de ruta baja.

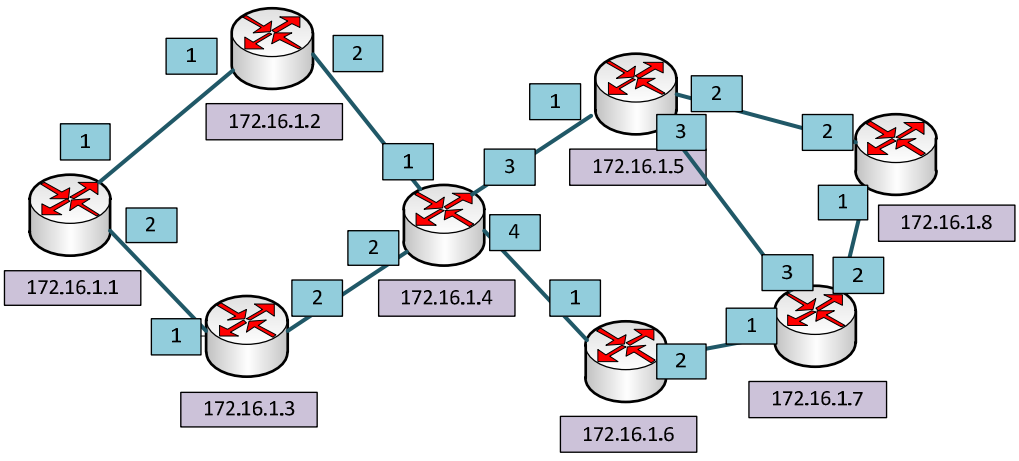


Figura 9-5 Red artificial de 8 nodos

Se han empleado dos configuraciones para los experimentos en los que se emplea la red artificial de 8 nodos. En una de ellas, solo se genera tráfico en 20 pares origen-destino, y en la otra configuración se aumenta a 56 pares origen-destino. La Tabla 9-1 muestra para la configuración de 20 y 56 pares la lista de nodos de borde empleados. La lista completa de pares origen-destino se obtiene con todas las combinaciones entre los nodos de borde de la Tabla 9-1.

Nodos de borde (20 pares origen-destino)	Nodos de borde (56 pares origen-destino)
172.16.1.1	172.16.1.1
172.16.1.2	172.16.1.2
172.16.1.3	172.16.1.3
172.16.1.4	172.16.1.4
172.16.1.5	172.16.1.5
	172.16.1.6
	172.16.1.7
	172.16.1.8

Tabla 9-1 Nodos de borde que originan tráfico tanto en la red artificial de 8 nodos como en la red ARPA-2

Como ejemplo de topología de red real se ha considerado la topología de la red ARPA-2 con 21 nodos [249], que se muestra en la Figura 9-6. Al igual que en el caso anterior, se han considerado 2 configuraciones, con 20 pares origen-destino y 56 pares origen-destino. La lista de pares origen-destino para cada configuración es la misma que en el caso anterior, por lo que se puede emplear la Tabla 9-1 como referencia para saber entre qué nodos se establecen los *lightpaths*.

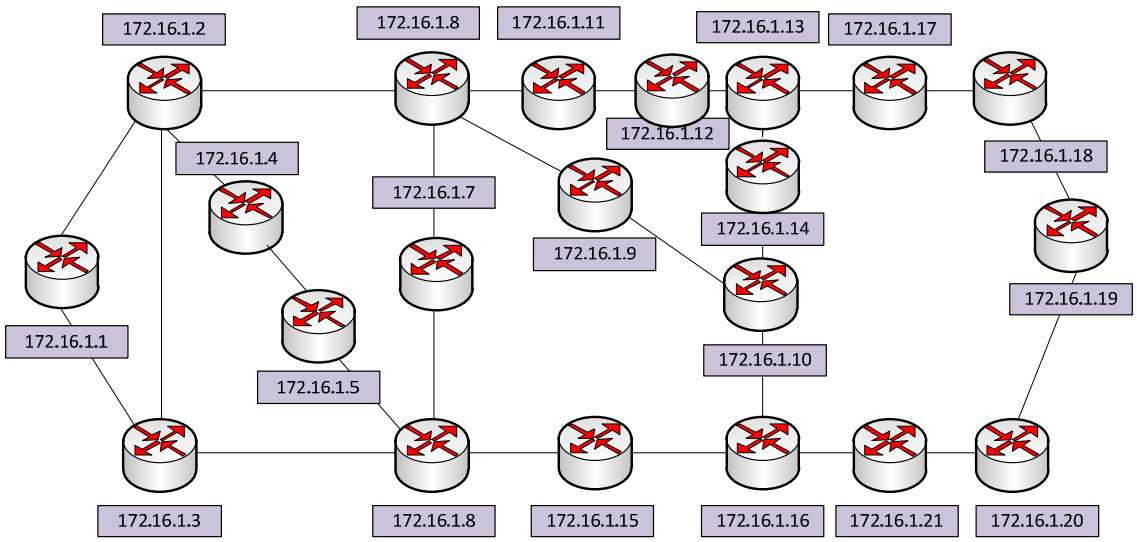


Figura 9-6 Topología de red ARPA-2 de 21 nodos

Por último, para las pruebas experimentales en las que se mide la probabilidad de bloqueo se ha empleado la topología de la red NSFNet de 14 nodos, mostrada en la Figura 9-7. En este caso se han repartido los nodos de borde donde se agrega tráfico por distintas zonas de la red tal y como se muestra en la Tabla 9-2.

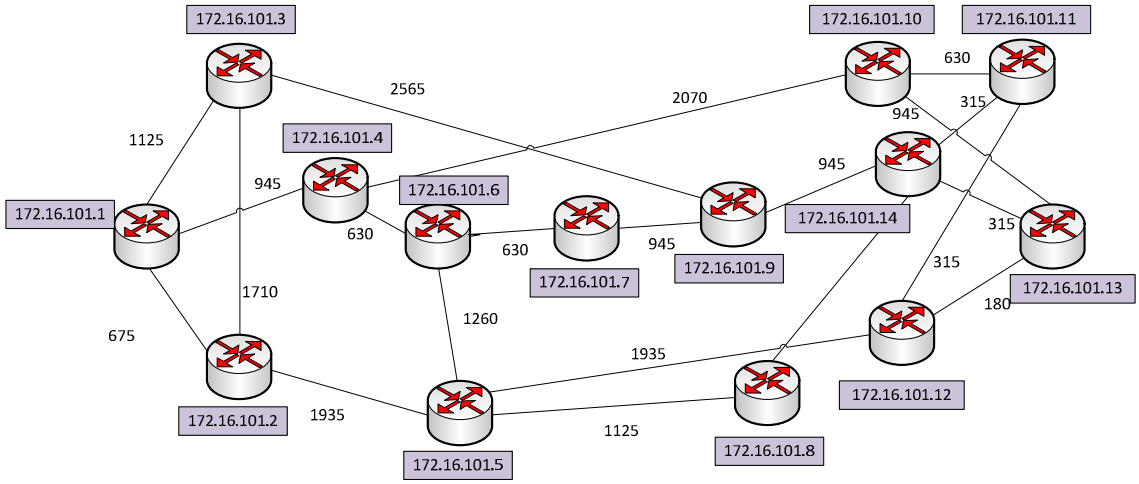


Figura 9-7 Red NSFNet

Nodos de borde
172.16.1.2
172.16.1.3
172.16.1.4
172.16.1.7
172.16.1.8
172.16.1.10
172.16.1.12
172.16.1.14

Tabla 9-2 Nodos de borde que originan tráfico en la red NSFNet

9.2 Impacto del algoritmo de Nagle de TCP en la emulación de WR-OBS con PCE

El protocolo PCEP emplea las facilidades del protocolo de transporte TCP, el cual se ha estudiado con detalle en el Capítulo 3 de la tesis. Así, los *bytes* de los mensajes del protocolo PCEP se escriben en un *buffer* del sistema operativo y TCP se encarga de transportar los datos (*bytes*) existentes en dicho buffer hasta el destino. Por tanto, escribir un mensaje PCEP en el buffer de TCP no implica que el mensaje se envíe de manera inmediata. En este caso, tiene una gran importancia el algoritmo de *Nagle* [270], que trata de reducir el número de paquetes que se envían a la red. Para ello, en vez de enviar inmediatamente los datos disponibles en el *buffer* de TCP, se espera a la llegada de nuevos datos. Así, cuando se escriben los *bytes* del mensaje PCEP en el *buffer*, estos no se envían al momento, sino que se envían o bien cuando hay suficientes datos para llenar un segmento de tamaño máximo (MSS, *Maximum Segment Size*), o bien cuando vence un temporizador, cuyo valor concreto depende del sistema operativo. En cambio, desactivando el algoritmo de Nagle, en cuanto hay datos en el *buffer* de TCP, estos se envían al destino, sin necesidad de esperar a que lleguen más datos.

En el emulador de WR-OBS implementado, así como en el prototipo de PCE, existe la posibilidad de controlar el uso del algoritmo de Nagle. Para ello, a través del *socket* (interfaz de las aplicaciones con TCP) se puede activar la opción *tcp_no_delay*. Cuando la opción *tcp_no_delay* está activa, el algoritmo de Nagle está desactivado y por tanto, los datos se envían sin retrasos adicionales.

Con el fin de comprobar el impacto del algoritmo de Nagle se han realizado una serie de pruebas con el emulador WR-OBS activando y desactivando dicho algoritmo. Cuando se activa el algoritmo de Nagle se observa que el tiempo de respuesta del PCE, medido en el emulador desde que se escribe el mensaje *Request* en el *socket* hasta que se lee el mensaje *Response,* es, en media, 10 ms superior al RTT medido al destino (obtenido a través de un *ping*). Además, se observa que cuando se usa Nagle los mensajes del protocolo PCEP se acumulan (se observan grupos de 10 peticiones en un mismo segmento TCP). Para ilustrar dicho comportamiento, en la Figura 9-8 se muestra una captura de *Wireshark* con el ejemplo de un segmento TCP que contiene múltiples mensajes PCEP (en concreto son PCEP *Requests*) que se ha obtenido con el algoritmo de Nagle activado. De hecho, el paquete capturado tiene 1298 bytes. En cambio, si el algoritmo de Nagle se desactiva, las peticiones PCEP tienden a viajar solas en un único segmento TCP (aun desactivando, hay ocasiones en que los datos se escriben de manera tan seguida que se acumulan, por lo que previsiblemente, el sistema operativo tiene algún temporizador interno además del propuesto por Nagle) y no se introduce apenas retardo adicional con respecto al tiempo medido en el *ping*.

En resumen, el retardo se minimiza desactivando el algoritmo de Nagle de TCP, pero se introduce una mayor cantidad de paquetes en la red, aumentando la sobrecarga. En el caso de WR-OBS, el retardo es un factor crítico, por lo que, a priori, conviene desactivar dicha opción. En los experimentos que se realizan en este capítulo se activa y desactiva dicha opción para ver el impacto en el rendimiento de la red WR-OBS, principalmente en cuanto a retardo de borde, como se verá en el apartado 9.4.

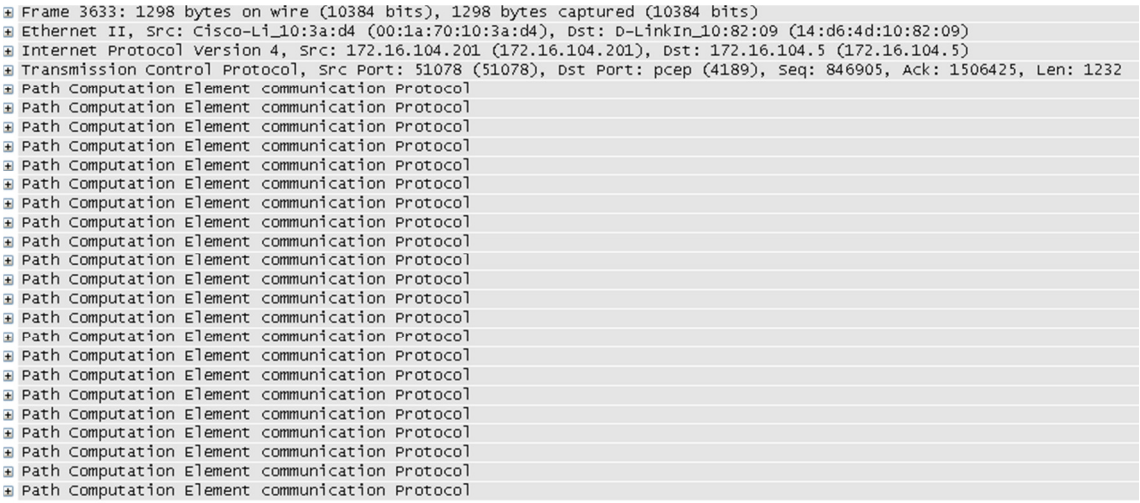


Figura 9-8 Segmento TCP con múltiples mensajes PCEP obtenido con el algoritmo de Nagle activado.

9.3 Caracterización de los procesos de llegada de peticiones y tiempo de procesamiento en el PCE

En los apartados anteriores se han descrito los detalles del entorno emulado en el que se lleva a cabo la evaluación experimental de la arquitectura WR-OBS con PCE. El resto del capítulo está dedicado a mostrar los resultados de dicha evaluación, que tiene el objetivo de medir parámetros de rendimiento de interés en redes WR-OBS. Así, en el apartado 9.4 se evalúa el retado de borde (t_{edge}), definido como el tiempo de espera del primer paquete de una ráfaga desde su llegada a un *buffer* del *edge router* hasta su transmisión. Con el objetivo de validar los resultados experimentales, se incluye además un estudio analítico del retardo de borde, particularizado para el entorno emulado, elaborado a partir de los estudios realizados por de Miguel [248]. Las expresiones analíticas del retardo de

borde requieren de la caracterización de los procesos de llegada de peticiones al PCE y del tiempo de procesamiento en el PCE. Por esta razón, se dedica este apartado a llevar a cabo un análisis de dichos procesos y realizar una caracterización de los mismos. Después, en el apartado 9.5 se mide en el entorno emulado el número medio de *lightpaths* establecido, que da una idea de la eficiencia de WR-OBS frente a una red de conmutación de circuitos. Al igual que en el caso del retardo de borde, se obtienen expresiones analíticas para el número medio de *lightpaths* establecido, que también requieren de la caracterización de los procesos de llegada de peticiones y del tiempo de cálculo.

9.3.1 Análisis del proceso de llegada de peticiones al PCE

El proceso de llegada de los mensajes de tipo PCEP *Request* influye en el tiempo que tarda el PCE en responder una petición. Esto se debe a que el PCE procesa las peticiones de manera secuencial y las peticiones de cálculo de ruta que lleguen mientras el PCE está procesando una petición son encoladas. Así, si los mensajes llegan en ráfagas, como sucede en el ejemplo que se mostró en la Figura 9-8, muchas peticiones han de esperar en la cola del PCE. Sin embargo, si los mensajes de tipo *Request* llegan al PCE de manera espaciada las peticiones apenas están encoladas, ya que el PCE es suficientemente rápido como para acabar con una petición antes de que llegue la siguiente.

En el apartado 9.4, que se verá posteriormente, la expresión analítica del del retardo de borde necesita el valor de la media y la varianza del tiempo del proceso de llegada de peticiones al PCE (ya que se modela el PCE como un sistema de colas). A continuación se realiza un conjunto de pruebas con el fin de obtener la caracterización de dicho proceso. En concreto, se estudia el proceso de llegada de peticiones cuando se emplea la técnica de ensamblado de ráfagas LBS y cuando se emplea la técnica FBAT. Como anteriormente se ha visto que el algoritmo de Nagle de TCP tiende a concentrar los mensajes que se envían desde el emulador WR-OBS al PCE, se estudia un caso en el que se active dicho algoritmo y se compara con el mismo caso en el que el algoritmo está desactivado.

Escenario de emulación del análisis del proceso de llegada de peticiones al PCE

En los experimentos que se muestran a continuación se ha empleado en el entorno emulado la topología de la red ARPA-2, mostrada en la Figura 9-6, y se han escogido N = 8 nodos de borde (recopilados en la Tabla 9-2) y 80 longitudes de onda por enlace. Se envía tráfico entre todos esos nodos de borde, por lo que hay un total de $K = N(N-1) = 56$ pares origen-destino, y se emplea una matriz de tráfico uniforme. La carga de tráfico normalizada por cada par origen-destino $\nu_{s\text{-}d}$ se define en la ecuación (9-1), donde $B_{in,s\text{-}d}$ es la tasa binaria media de llegada de datos al *buffer* asociado al par *s-d*, B_{core} la tasa binaria del *lightpath,* $\lambda_{s\text{-}d}$ la tasa media de llegada de paquetes al *buffer* asociado al par *s-d*, y μ^{-1} el tamaño medio de un paquete.

$$\nu_{s-d} = \frac{B_{in,s-d}}{B_{core}} = \frac{\lambda_{s-d}}{B_{core}\mu} \tag{9-1}$$

Se ha empleado, para todas las emulaciones de este capítulo de la tesis, un tamaño de medio de paquete de 485,6 bytes [265] y una tasa binaria de *lightpath* de 10 Gbps. En cada caso concreto emulado se escoge el valor del temporizador de ensamblado T y se configura el uso o no del algoritmo de Nagle de TCP. Por otro lado, para todos los casos emulados en este apartado se ha fijado una carga de tráfico uniforme $\nu \equiv \nu_{s\text{-}d} = 0{,}2\ \forall s,d$.

El PCE emplea el algoritmo *Adaptive Unconstrained Routing – Exhaustive* (AUR-E) [249] para realizar el cálculo de la ruta del *lightpath*. AUR-E realiza una ejecución del algoritmo de *Dijskstra* (en el prototipo se ha utilizado la implementación de la librería *JGrapht* [264]) por cada longitud de onda para encontrar la ruta más corta disponible (si existe) en cada una de esas lambdas seleccionando la más corta de todas ellas.

El criterio de parada de las pruebas realizadas en este apartado de análisis del proceso de llegada de peticiones al PCE es que o bien ha convergido el análisis del tiempo medio de respuesta del PCE [271] o bien se ha llegado a un tiempo de emulación (real) de 40 minutos.

En los experimentos llevados a cabo en este apartado se captura en la máquina PCE todo el tráfico con la herramienta *Wireshark* [254]. Posteriormente, se filtra la captura de tráfico por mensaje PCEP, en este caso, por mensajes de tipo *Request*. Después se obtiene a través del propio *Wireshark* el tiempo entre paquetes capturados que contienen mensajes de tipo *Request*. Hay casos en los que un mismo paquete capturado contiene varios mensajes PCEP (como se ha explicado en el apartado 9.2). En estos casos, se cuenta como cero el tiempo entre mensajes *Request*. La diferencia entre el número de total de mensajes de tipo *Request* (conocido a través de *Wireshark*) y el número de paquetes capturados que contienen al menos un mensaje de tipo *Request* (también conocido a través de *Wireshark*) es el número de ocasiones en las que el tiempo entre *Request* es cero. El resto de los valores del tiempo entre *Request* se obtiene de las medidas de los tiempos entre paquetes capturados que contienen mensajes de tipo *Request*.

Análisis con el mecanismo de construcción de ráfagas LBS y algoritmo de Nagle desactivado

Cuando se emplea el mecanismo de construcción de ráfagas LBS, una vez que vence el temporizador de ensamblado, se envía un mensaje *Request* al PCE. Dicho temporizador se vuelve a activar cuando finaliza el proceso de formación de la ráfaga. La ráfaga se completa después de que el PCE haya contestado y se haya establecido el *lightpath*. Así, el tiempo que transcurre entre el envío de dos mensajes *Request* no es fijo, y depende de cuánto tarde el PCE en contestar y del tiempo de establecimiento (el cual se puede considerar prácticamente constante para un mismo par origen-destino).

Se ha realizado un experimento con LBS siguiendo la metodología descrita anteriormente en la que se ha escogido un temporizador de ensamblado *T* con valor cero. Tanto en la máquina OBSTester donde corre el emulador de WR-OBS como en la máquina PCE se ha desactivado el algoritmo de Nagle de TCP para que los que los mensajes PCEP se envíen inmediatamente. En la Figura 9-9 se muestra la función de distribución acumulada del tiempo entre peticiones (en ms) a partir de los resultados del experimento.

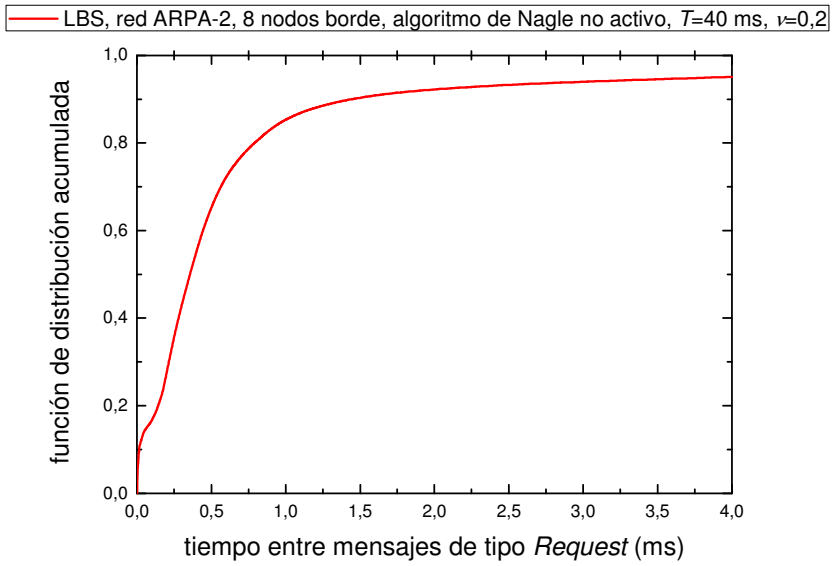


Figura 9-9 Función de distribución acumulada del tiempo entre mensajes *Request*, LBS 56 pares, T=0

Hay que destacar que aproximadamente un 10% de los mensajes *Request* llega en un mismo segmento TCP que otro mensaje *Request*. Tal y como se ha mencionado anteriormente, en esos casos se ha considerado cero el tiempo entre petición para el todos los *Request* presente en el segmento TCP excepto el primer mensaje. En el histograma del tiempo entre *Request*, mostrado en la Figura 9-10, se observa un pico cerca de cero, que coincide con los mencionados mensajes *Request* que llegan en el mismo segmento TCP y los mensajes que llegan prácticamente seguidos (aun en segmentos TCP diferentes).

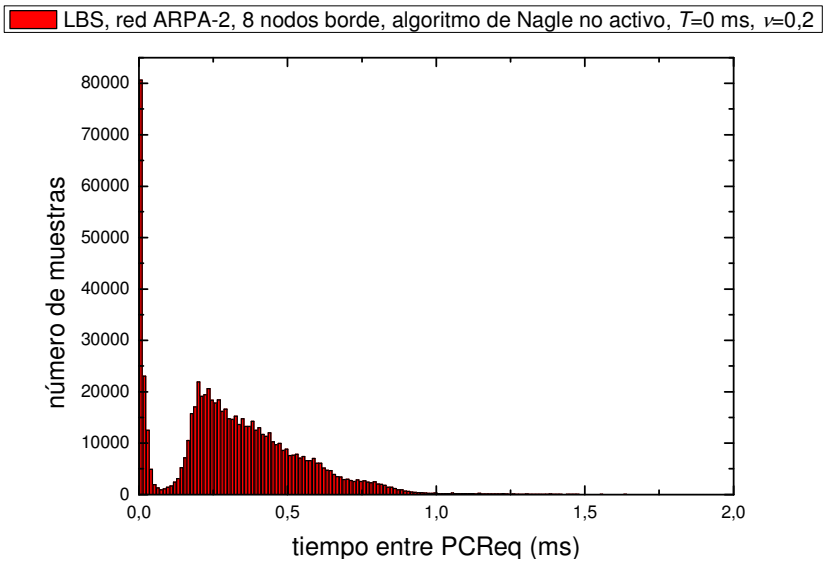


Figura 9-10 Histograma del tiempo entre *Request*, LBS, 56 pares origen destino, *T*=0 ms

Se realiza un nuevo experimento aumentando el valor del temporizador de ensamblado *T* hasta 40 ms. En la distribución de probabilidad acumulada de la Figura 9-11 se observa un comportamiento similar al caso anterior y también hay un 10% de mensajes de tipo *Request* que llegan en el mismo segmento TCP.

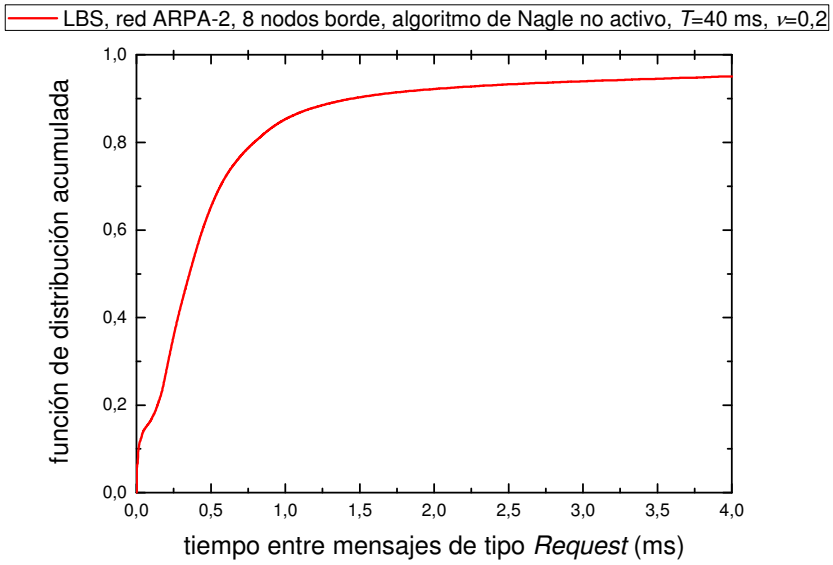


Figura 9-11 Función de distribución acumulada del tiempo entre *Request*, LBS, 8 nodos de borde, *T*=40 ms

Uno de los objetivos de las pruebas de este apartado es la obtención de la media y la varianza del proceso de llegada entre mensajes de tipo *Request*. Estos datos se recopilan en la Tabla 9-3 para los dos casos emulados. La principal diferencia entre ambos casos es que la media del tiempo entre *Request* cuando T vale 40 ms se dobla con respecto a cuando T vale 0 ms, pasando de 0,4122 ms a 0,9475 ms y la varianza aumenta de 1,4121 a 5,7366 ms^2.

Caracterización tiempo entre *Request*	*T*=0 ms	*T*=40 ms
Media	0,4122 ms	0,9475 ms
Varianza	1,4121 ms^2	5,7366 ms^2

Tabla 9-3 Caracterización del tiempo entre mensajes PCEP *Request*, LBS, 8 nodos de borde, *T*=0 ms, *T*=40 ms

Activación de algoritmo de Nagle con LBS

A continuación se realiza un experimento con el fin de observar el comportamiento del proceso del tiempo entre *Request* cuando se activa el algoritmo de Nagle de TCP. El experimento se realiza fijando el valor del temporizador de ensamblado *T* a 0 ms (como la primera prueba realizada). En este experimento se obtiene que el 91% de los mensajes de tipo *Request* llegan en un segmento TCP en el que ya había otro mensaje *Request*. Esto se debe a la influencia del algoritmo de Nagle, como se anticipó en el análisis del apartado 9.2, por la que los mensajes PCEP se agrupan en los segmentos TCP. La Figura 9-12 muestra la distribución de probabilidad acumulada para el tiempo entre *Request* de este experimento.

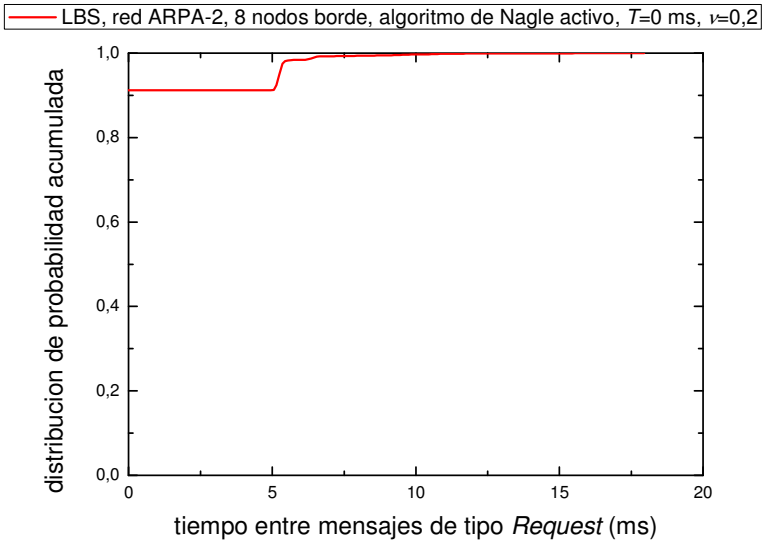


Figura 9-12 Distribución de probabilidad acumulada del tiempo entre mensajes PCEP *Request* activando Nagle

Para observar el grado de agregación de mensajes PCEP de tipo *Request* en segmentos TCP cuando se activa el algoritmo de Nagle se calcula, a partir de los datos capturados en *Wireshark*, el histograma de mensajes *Request* por segmento TCP (Figura 9-13).

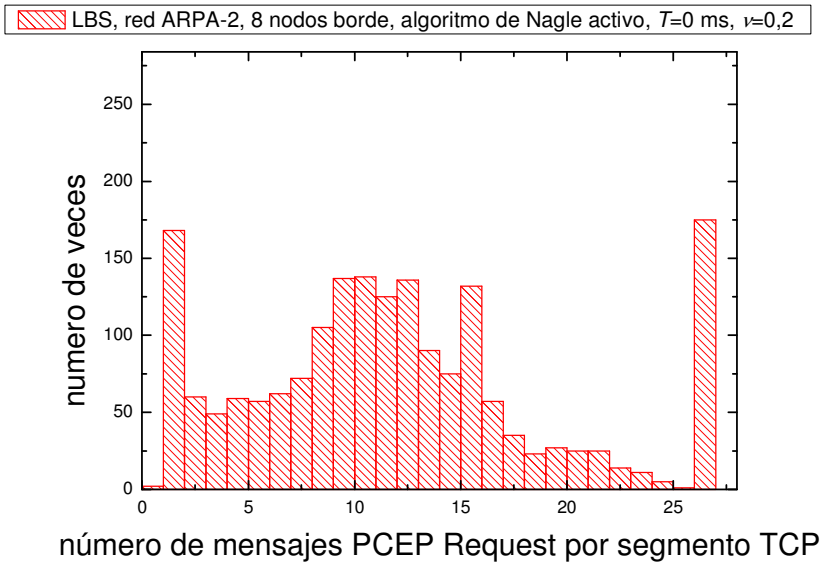


Figura 9-13 Histograma del número de mensajes PCEP *Request* por segmento TCP con LBS y *T*=0

El hecho de que se emplee una única sesión PCEP para las peticiones de todos los nodos WR-OBS emulados (y por tanto, una única conexión TCP compartida) contribuye a la alta concentración de mensajes *Request* por segmento TCP como se observa en el histograma de la Figura 9-13.

Análisis con el mecanismo de construcción de ráfagas FBAT

Cuando se emplea el mecanismo de construcción de ráfagas FBAT, los mensajes de tipo *Request* se envían desde cada nodo WR-OBS en intervalos de tiempo prácticamente constantes de valor igual al valor del temporizador de ensamblado *T*. Se ha realizado una prueba con FBAT, con el algoritmo de Nagle desactivado, en la que se ha escogido un valor de temporizador de ensamblado tal que sea parecido al tiempo de agregación de ráfaga medido en las pruebas anteriores al emplear LBS y temporizador de ensamblado *T* de 40 ms. El valor del temporizador de ensamblado de FBAT que cumple (aproximadamente) con estas características es un valor de 50 ms. La Figura 9-14 muestra una comparativa con las distribuciones del tiempo entre mensajes de tipo *Request* para LBS con temporizador T de 40 ms y FBAT con temporizador *T* de 50 ms.

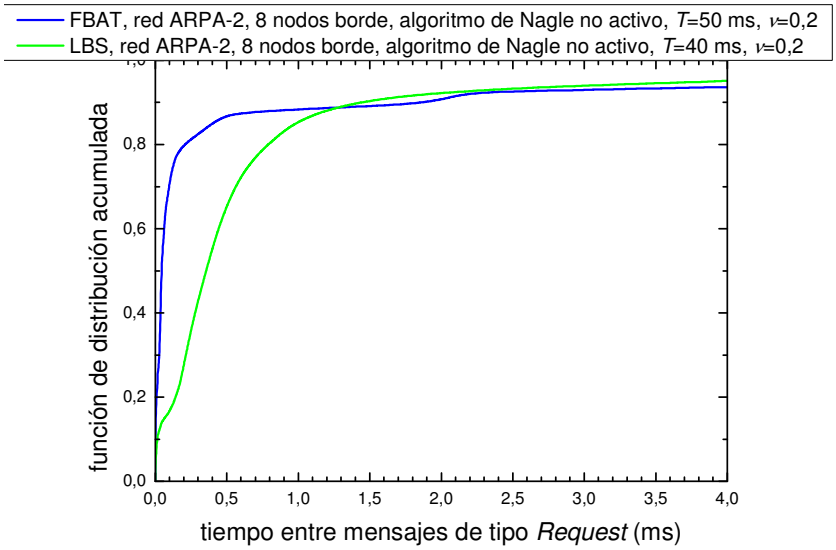


Figura 9-14 Comparativa de la función de distribución acumulada del tiempo entre *Request* de FBAT y LBS

Se observa claramente en la Figura 9-14 que el mecanismo FBAT concentra las llegadas de mensajes de tipo *Request* al PCE (aproximadamente el 80% de los casos el tiempo entre mensajes de tipo *Request* es menor de 0,2 ms), mientras que en LBS las llegadas de mensajes de tipo *Request* al PCE se espacian más. Este hecho se corrobora con la obtención de la media y la varianza del tiempo entre mensajes de tipo *Request* que se muestra en la Tabla 9-4. Se aprecia que, a pesar de que la media en ambos casos es similar (apenas hay un 10% de diferencia entre las medias), la varianza es 3 veces mayor en FBAT que en LBS.

Caracterización del tiempo entre *Request*	LBS (*T*=40 ms)	FBAT (*T*=50 ms)
Media	0,4122 ms	0,4577 ms
Varianza	1,4121 ms^2	4,2807 ms^2

Tabla 9-4 Media y varianza del tiempo entre *Request* para LBS (*T*=40 ms) y FBAT (*T*=50 ms)

9.3.2 Análisis del tiempo de cálculo en el PCE

El tiempo total de procesamiento es el tiempo que transcurre desde que el PCE recibe un mensaje de tipo *Request* hasta que envía la respuesta con el resultado del cálculo. Para obtener el tiempo de procesamiento completo hay que seguir con detalle qué ocurre en el PCE. Cuando llega un mensaje *Request* con una petición de cálculo de camino al PCE, se procesa dicho mensaje y se introducen en una cola las peticiones de cálculo que contenga.

El procesador de peticiones se encarga de leer de esa cola, escoger el algoritmo a emplear según los datos de la petición concreta, ejecutar dicho algoritmo, realizar las reservas necesarias, generar el mensaje PCEP de respuesta (PCEP *Response*, que se abrevia indistintamente como *Response* o *PCRep*) y enviarlo al destino (escribiendo en un *socket*). De esta forma, hay dos componentes principales en el tiempo de procesamiento, por un lado, como el PCE solamente procesa una petición a la vez, el tiempo de encolado, es decir, el tiempo que se pasa una petición de cálculo de *lightpath* en la cola de peticiones, y por otro lado el tiempo de cálculo, que se define como el tiempo que transcurre desde que el PCE extrae la petición de cálculo de camino de la cola de peticiones hasta que escribe el mensaje *Response* en el socket de TCP. En el análisis teórico del retardo de borde que se realiza posteriormente en el apartado 9.4.1 una de las entradas necesarias es la caracterización del tiempo de cálculo, para poder obtener el tiempo de procesamiento en el PCE. En este apartado se realiza un análisis del tiempo de cálculo de una petición de cálculo de ruta de *lightpath* en el PCE y, al igual que con el análisis del proceso del tiempo entre mensajes de tipo *Request*, se obtiene la caracterización del proceso del tiempo de cálculo.

El tiempo de cálculo va a depender del algoritmo a emplear, la topología de red, las restricciones que lleve asociada cada petición, la velocidad del procesador, el número de procesadores y la actividad en la máquina durante el tiempo que está procesando la petición.

Análisis experimental

En el experimento que se realiza para caracterizar el tiempo de cálculo se ha empleado en el entorno emulado la topología de la red ARPA-2, mostrada en la Figura 9-6, y se han escogido 8 nodos de borde para generar y recibir tráfico (recopilados en la Tabla 9-2) y 80 longitudes de onda por enlace. El PCE emplea el algoritmo AUR-E [249] para realizar el cálculo de la ruta del *lightpath*. Al igual que en el apartado anterior, se envía tráfico entre todos esos nodos de borde, por lo que hay un total de 56 pares origen-destino, y se emplea una matriz de tráfico uniforme. Se fija la carga de tráfico normalizada ν a 0,2 según la definición de la ecuación (9-1) y los mismos valores que en el apartado anterior. El criterio de parada de las pruebas realizadas en este apartado de análisis del proceso de llegada de peticiones al PCE es que haya convergido el análisis del tiempo medio de cálculo del PCE. En este caso, se realiza un experimento en el que se ha fijado como método de construcción de ráfagas LBS con temporizador T=0.

El histograma con la distribución del tiempo de procesamiento se muestra en la Figura 9-15, donde se observa que el tiempo de cálculo oscila en la mayoría de las ocasiones aproximadamente entre 200 y 800 μs.

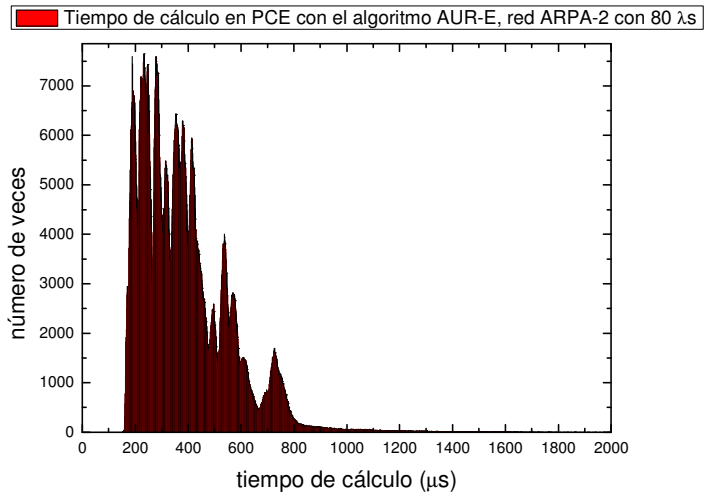


Figura 9-15 Distribución del tiempo de cálculo con algoritmo AUR-E y red ARPA-2 de 80 lambdas

Sin embargo, hay un pequeño porcentaje de veces, inferior al 1% en el que hay valores por encima de 800 μs, incluso hay algún caso puntual por encima de los 2 ms. Esto se debe a que, al ser una implementación en Java, hay ocasiones en las que el recolector de basura de Java (*Garbage Collector*, GC) entra en acción parando el resto de procesos. En otras ocasiones, el PCE tiene realizar otras tareas simultáneamente, como recibir mensajes o eliminar reservas. En media, el tiempo de cálculo es de 385,9 μs. La caracterización del tiempo de cálculo se muestra en la Tabla 9-5.

Caracterización tiempo de cálculo con algoritmo AUR-E y red ARPA-2 con 80 lambdas	
Media	0,3859 ms
Varianza	0,0321 ms^2

Tabla 9-5 Media y varianza del tiempo de cálculo

9.4 Análisis del retardo de borde medio

Un parámetro importante para medir el rendimiento de una red WR-OBS es el denominado retardo de borde (t_{edge}), definido como el tiempo de espera del primer paquete de una ráfaga desde su llegada a un *buffer* del *edge router* hasta su transmisión [248]. En este capítulo se obtiene en el entorno emulado experimental medidas en tiempo real del retardo de borde y se contrastan con los modelos teóricos del retardo de borde elaborados por de Miguel [248], que han sido adaptados para tener en cuenta las particularidades de la arquitectura con PCE y RSVP y las condiciones específicas del *testbed* experimental. El análisis se realiza para los mecanismos de construcción de ráfagas LBS y FBAT.

9.4.1 Estudio analítico del retardo de borde medio.

En este apartado se muestra el análisis matemático realizado para obtener estimaciones del retardo de borde medio a partir del modelo teórico del retardo de borde desarrollado por de Miguel [248].

El PCE recibe, en media, una petición de cálculo de ruta cada $\bar{t}_{aggr}$ (tiempo medio durante el cual se está ensamblando una ráfaga). Por tanto, en una red WR-OBS con K pares origen-destino, la tasa de llegadas al PCE es de $K/\bar{t}_{aggr}$. Suponiendo que el tiempo medio de cálculo de una petición en el PCE es $\bar{t}_c$ y que en el PCE un único procesador se encarga del cálculo, es necesario que se verifique, para que el sistema sea estable, la condición mostrada en (9-2).

$$\bar{t}_{aggr} > K\bar{t}_c \tag{9-2}$$

El retardo medio de borde está determinado, tanto para LBS como FBAT, en la arquitectura WR-OBS estudiada por la suma del valor del temporizador T, el tiempo que tarde en llegar el mensaje *Request* al PCE ($t_{PCC-PCE}$), el tiempo que tiene el PCE encolado el mensaje de petición de cálculo de ruta ($\bar{t}_q$), el tiempo de cálculo de la ruta ($\bar{t}_c$), el tiempo que tarda en enviarse la respuesta del PCE al cliente PCC ($t_{PCE-PCC}$) y el tiempo que tarda en establecerse el *lightpath* mediante RSVP (t_{RSVP}). Así $\bar{t}_{edge} = T + t_{PCC-PCE} + t_{PCE-PCC} + \bar{t}_q + \bar{t}_c + t_{RSVP}$. Para simplicar, el tiempo de envío de los mensajes PCEP ($t_{PCC-PCE}$ y $t_{PCE-PCC}$) se suponen constantes e iguales al retardo de propagación, aunque la implementación real es más compleja, ya que hay un *buffer* de TCP, un conjunto de nodos WR-OBS compartiendo la sesión PCEP, además de la propia dinámica de TCP para enviar los datos, lo que hará que realmente esos tiempos no sean constantes. Según de Miguel [248], un nodo de control con una cola para los trabajos puede modelarse como un sistema G/G/1, por lo que puede emplearse la fórmula de Allen-Cunneen [272], que proporciona una aproximación del tiempo medio de espera en cola en estos sistemas en función del

coeficiente de variación del proceso de llegadas (C_A), en este caso el proceso de llegadas de mensajes de tipo *Request* al PCE, y del coeficiente de variación del tiempo de servicio (C_B), en este caso el tiempo de cálculo de la ruta del *lightpath*. Al emplear esta formula para obtener $\bar{t}_q$, se obtiene

$$\bar{t}_{edge} = T + t_{PCC-PCE} + \bar{t}_c + t_{PCE-PCC} + t_{RSVP} + \frac{K\bar{t}_c{}^2(C_A^2 + C_B^2)}{2(\bar{t}_{aggr} - K\bar{t}_c)} \quad (9\text{-}3)$$

El coeficiente de variación (tanto C_A para el proceso de llegadas como C_B para el tiempo de cálculo) se obtiene dividiendo la desviación estándar (raíz cuadrada de la varianza) entre la media. En el apartado 9.3.1 se obtuvo, para unos experimentos con LBS y FBAT, la media y varianza de del tiempo entre mensajes de tipo *Request*. En el apartado 9.3.2 se obtuvo, para un algoritmo concreto (AUR-E) y una red concreta (ARPA-2) la media y varianza del tiempo de cálculo.

A continuación, se particulariza la ecuación (9-3) que calcula el retardo de borde para los métodos de construcción de ráfagas LBS y FBAT

Estudio analítico del retardo de borde medio para LBS

Empleando LBS el retardo de borde y el tiempo de agregación coinciden, ya que se finaliza el proceso de formación de ráfaga una vez que el camino está establecido y listo para poder empezar a transmitir los datos de la ráfaga. Así, $\bar{t}_{aggr} = \bar{t}_{edge}$, con lo que sustituyendo en (9-3), se tiene:

$$\bar{t}_{edge} = \frac{A + K\bar{t}_c + \sqrt{(K\bar{t}_c - A)^2 + K\bar{t}_c{}^2 C}}{2} \quad (9\text{-}4)$$

Siendo $A = T + t_{PCC-PCE} + \bar{t}_c + t_{PCE-PCC} + t_{RSVP}$ y siendo $C = C_A^2 + C_B^2$.

Estudio analítico del retardo de borde para FBAT

Empleado FBAT, el tiempo de agregación es siempre igual a *T* para todos los pares origen destino. Así, como $\bar{t}_{aggr} = T$, sustituyendo en (9-3), se obtiene la ecuación (9-5) para el retardo de borde en FBAT.

$$\bar{t}_{edge} = T + t_{PCC-PCE} + \bar{t}_c + t_{PCE-PCC} + t_{RSVP} + \frac{K\bar{t}_c{}^2(C_A^2 + C_B^2)}{2(T - K\bar{t}_c)} \quad (9\text{-}5)$$

Posteriormente, se representan, junto con los resultados experimentales, los valores teóricos a partir de (9-4) y (9-5), con C_A y C_B obtenido a partir de la caracterización realizada del tiempo de cálculo y el proceso de llegadas de los mensajes *Request* al PCE. Para LBS, empleando la media y varianza del proceso de llegadas de la Tabla 9-3 y la media y varianza del tiempo de cálculo de la Tabla 9-5, se obtiene *C*=8,5 para *T*=0 y *C*=6,5 para *T*=40. Se emplea para mostrar los resultados de LBS el mayor de los valores anteriores (*C*=8,5). Para FBAT, se emplea el valor de la Tabla 9-4 para la media y varianza del proceso de llegada, obteniendo un valor de *C* igual a 20,6.

9.4.2 Descripción de los experimentos

Los experimentos que se realizan en este apartado de la tesis tienen el objetivo de medir el retardo de borde medio. Uno de los aspectos con mucha influencia en el retardo de borde es el tiempo de cálculo en el PCE. Por ello, se han elaborado dos casos principales con respecto al tiempo de cálculo. El primero de ellos, denominado "tiempo de cálculo pequeño", ilustra una situación en la que se emplea un algoritmo de complejidad baja en una red de un número pequeño de nodos. En este caso, en el entorno emulado se emplea

la topología de la red artificial de 8 nodos mostrada en la Figura 9-5 del apartado 9.1.3 y en el PCE se emplea para calcular la ruta del *lightpath* el algoritmo del camino más corto (de aquí en adelante se abrevia como SP, *Shortest Path*), en concreto se emplea una implementación del algoritmo de Dijkstra. El segundo caso, denominado "tiempo de cálculo alto", se ha empleado en el entorno emulado la topología de la red ARPA-2, mostrada en la Figura 9-6, con 80 lambdas por enlace y el PCE emplea el algoritmo AUR-E para realizar el cálculo de rutas. La otra componente que tiene influencia en el retardo de borde es el tiempo de encolado, que depende, entre otros factores, del número de pares origen-destino *K*. Así, para cada uno de los casos anteriores, se realiza una prueba con *K*=20 pares, para lo cual se escogen 4 nodos de borde según la columna izquierda de la Tabla 9-1 y emplea una matriz de tráfico uniforme, y otra prueba con *K*=56 pares origen-destino, para lo cual se escogen 8 nodos de borde según la columna izquierda de la Tabla 9-1 y emplea una matriz de tráfico uniforme. Las direcciones de los nodos de borde son válidas para las dos topologías empleadas. Para todos los casos emulados en este apartado se ha fijado una carga de tráfico normalizada con respecto a la velocidad del *lightpath* por par origen destino ν de 0,2, según la definición de carga de la ecuación (9-1). Asimismo, para todas las emulaciones de este capítulo de la tesis un tamaño de medio de paquete de 485,6 bytes [265] y una tasa binaria de *lightpath* de 10 Gbps. Para simiplificar, el tiempo de establecimiento del lightpath se ha fijado a 5 ms, independientemente de la ruta.

Se realizan experimentos tanto con LBS como con FBAT. En cada emulación se configura el uso o no del algoritmo de Nagle de TCP. En cada caso estudiado se realiza un barrido por varios valores del temporizador de ensamblado T y se realiza una emulación para cada valor del temporizador. La duración de cada uno de los experimentos realizados en este apartado viene dada por varios factores: se emula hasta que se cumpla que el análisis del retardo de borde medio haya convergido, y hayan transcurrido como mínimo 40 minutos reales de experimento. En muchos de los experimentos, con las condiciones mencionadas, se han generado en torno a 2 millones de ráfagas.

Las medidas del retardo de borde se han realizado con la ayuda del método *System.nanotime* de Java 1.6 que proporciona el valor del reloj del sistema operativo con mayor precisión. En concreto, tiene una precisión de nanosegundos y se puede utilizar para medir el tiempo entre dos instantes. Así primero se consulta el valor del reloj en cuanto empieza el ensamblado de la ráfaga y posteriormente cuando la emulación del establecimiento del *lightpath* termina. La diferencia entre ambos valores es el retardo de borde. El resultado que se obtiene para cada emulación es la media de todas las muestras del retardo de borde tomadas.

9.4.3 Análisis de los resultados

A continuación se describen los resultados de los experimentos realizados y se realiza un análisis de los mismos.

Caso LBS con 56 pares origen-destino y tiempo de cálculo pequeño

En primer lugar se realiza un experimento con LBS para representar un caso en el que el tiempo de cálculo en el PCE sea muy pequeño (empleo del algoritmo SP y uso de la red de 8 nodos). Los resultados experimentales, mostrados en la Figura 9-16, se han obtenido para dos casos, activando y desactivando el algoritmo de Nagle de TCP y muestran el retardo de borde para distintos valores del temporizador de ensamblado *T*. En los experimentos se ha medido también el tiempo de cálculo medio ($\bar{t}_c$) en el PCE y se ha obtenido un valor de 0,071 ms. Este valor medio es diferente al obtenido en la caracterización del tiempo de cálculo realizado en la sección 9.3.2 ya que el algoritmo de cálculo empleado y la topología sobre la que se realiza el cálculo son diferentes. En la Figura 9-16 se representa junto con los resultados experimentales el valor obtenido con el

modelo analítico. Para obtener los valores del modelo analítico se han utilizado como datos de entrada un coeficiente *C* de 8,5 (obtenido a partir de la caracterización de los procesos de llegada de *Request* y tiempo de cálculo de la sección 9.3.2, aunque las condiciones no sean realmente las mismas) y como tiempo de cálculo medio $\bar{t}_c$ el valor medido experimentalmente (0,071 ms).

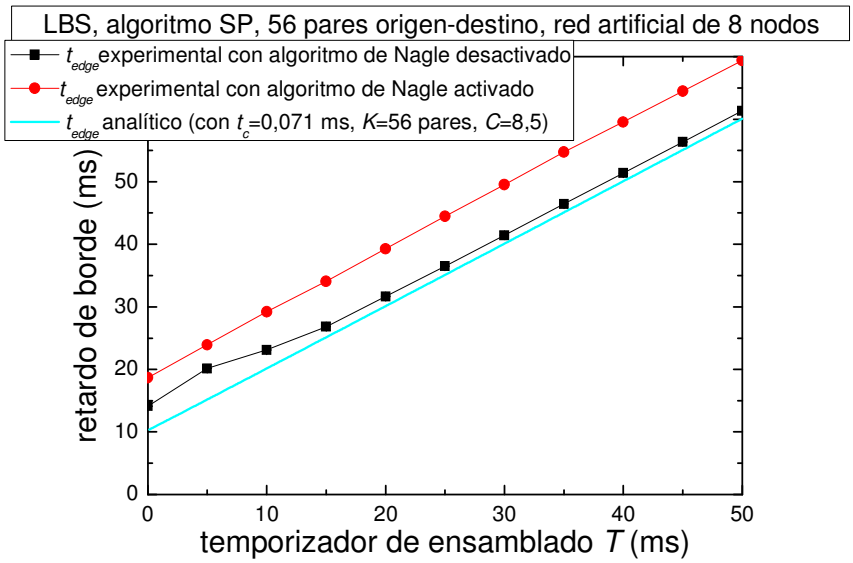


Figura 9-16 Retardo de borde medio para LBS con algoritmo SP en red de 8 nodos y 56 pares origen destino

Se observa claramente que cuando se desactiva el algoritmo de Nagle, el retardo de borde que se mide experimentalmente y el que se obtiene aplicando el modelo analítico prácticamente coinciden. De hecho, los valores experimentales (con Nagle desactivado) sólo se alejan de manera significativa en los experimentos en los que el temporizador es inferior a 10 ms. En cambio, con el algoritmo de Nagle activo, el retardo se incrementa en unos 10 ms con respecto al valor del modelo analítico para todos los valores del temporizador de ensamblado.

Caso LBS con 20 pares origen-destino y tiempo de cálculo pequeño

En este caso, se ha realizado un experimento similar al anterior, pero reduciendo el número de pares origen destino a 20, escogiendo únicamente 5 nodos de borde.

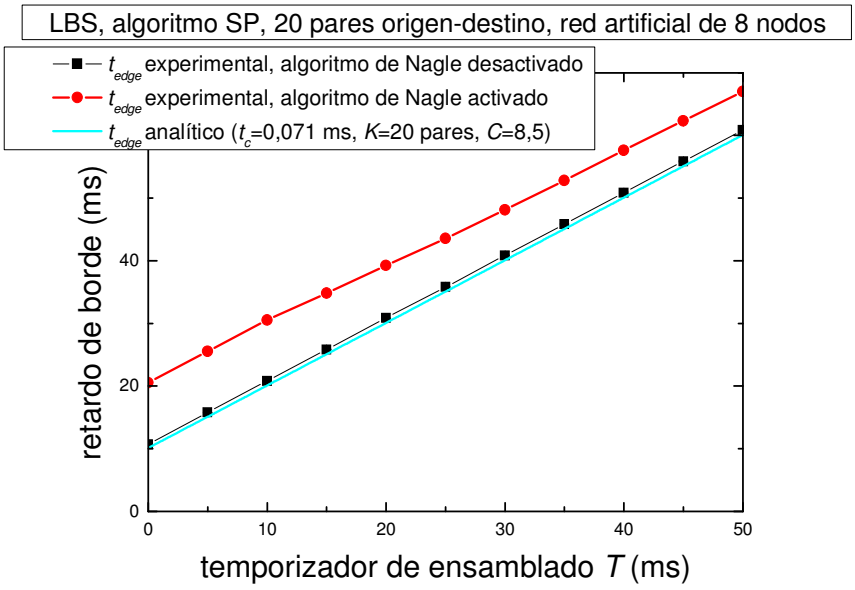


Figura 9-17 Retardo de borde medio para LBS con SP en red de 8 nodos y 20 pares origen destino

El resultado experimental, mostrado en la Figura 9-17, cuando se desactiva el algoritmo de Nagle, sigue fielmente al modelo analítico (alimentado con los valores del coeficiente C=8,5 obtenido de la caracterización de la sección 9.3.2 y $\bar{t}_c$=0,071 ms medido en los experimentos anteriores que empleaban el mismo algoritmo) para todos los valores. Al haber únicamente 20 pares origen destino, el emulador de red WR-OBS está muy poco sobrecargado, y no se producen desviaciones inesperadas aun con un tiempo entre peticiones muy bajo. En cambio, el en experimento anterior que se emplearon 56 pares origen destino, cuando el tiempo entre peticiones era bajo, los resultados se apartaban ligeramente de lo esperado. En el caso de que se active el algoritmo de Nagle, al igual que en el caso anterior con K=56 pares origen destino, los resultados experimentales son 10 ms mayores que cuando se desactiva el algoritmo.

Caso LBS con 56 pares origen-destino y tiempo de cálculo alto

En esta ocasión se estudia, también para LBS, un caso con tiempo de cálculo alto, para lo cual en el entorno emulado se emplea la topología de la red ARPA-2 (Figura 9-6) y se utiliza el algoritmo AUR-E con 80 lambdas. En concreto, el tiempo de cálculo medio, como ya se anticipó en la caracterización del tiempo de cálculo en el apartado 9.3.2, es de 0,3859 ms. En la Figura 9-18 se muestran los resultados experimentales tanto activando como desactivando el algoritmo de Nagle. En la Figura 9-18 también se muestra el valor obtenido con el modelo analítico, en este caso empleado dos valores de C.

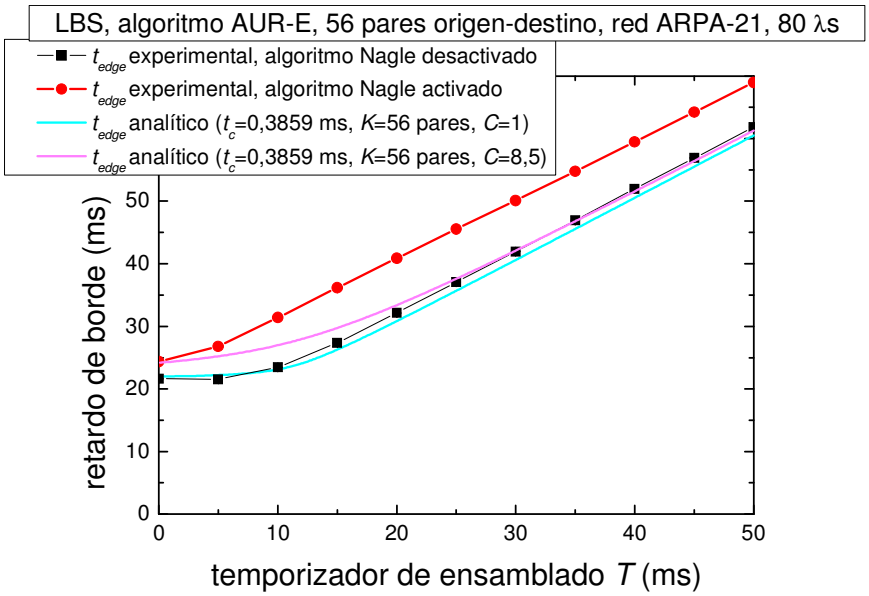


Figura 9-18 Retardo de borde medio para LBS con AUR-E, red ARPA-2 de 21 nodos, 80 lambdas y 56 pares

En primer lugar se ha obtenido el valor analítico empleando C=8,5 (obtenido con los datos de las medias y varianzas de los procesos de llegada de mensajes *Request* y del tiempo de cálculo del apartado 9.3, en concreto con los valores de la Tabla 9-3 y la Tabla 9-5). Al representar el valor analítico con C=8,5 se observa que para valores del temporizador por debajo de 20 ms se sobreestima el retardo de borde, siendo el modelo demasiado pesimista. En la Figura 9-18 se muestra también el resultado del modelo analítico probando con un valor de C=1, obtenido con una caracterización del proceso de llegada de mensajes *Request* en la que la varianza fuese menor. Se observa que especialmente para valores por debajo de 20 ms modelo analítico con C=1 se comporta perfectamente, mientras que para valores del temporizador por encima de 20 ms, sigue a los resultados experimentales muy de cerca (1 ms de diferencia). En el caso analizado, en el que el tiempo de procesamiento en el PCE ya es significativo, se puede apreciar cómo el retardo de borde ya no desciende para valores bajos del temporizador de ensamblado T, tal y como predice el modelo teórico.

Caso LBS con 20 pares origen-destino y tiempo de cálculo alto

En el experimento anterior se comprobó que para valores del temporizador de ensamblado *T* por debajo de 10 ms, el retardo de borde medio se mantenía en el mismo valor, sin bajar más, debido a la carga en el PCE que provocaba que las peticiones se encolaran. En el nuevo experimento, se reduce el número de pares origen destino a 20, manteniendo el mismo algoritmo de cálculo de caminos. En este caso, el PCE puede atender las peticiones sin apenas encolar, ya que, como se muestra en la Figura 9-6, el retardo de borde no para de bajar para todos los valores del temporizador de ensamblado *T*. Se llega a medir un retardo de borde de 10 ms para el caso de *T*=0, mostrando el buen comportamiento del emulador, especialmente con pocos pares origen-destino. En este caso, el retardo de borde obtenido con el modelo analítico, empleando el valor de C=8,5, sigue perfectamente los resultados experimentales en los que el algoritmo de Nagle está desactivado.

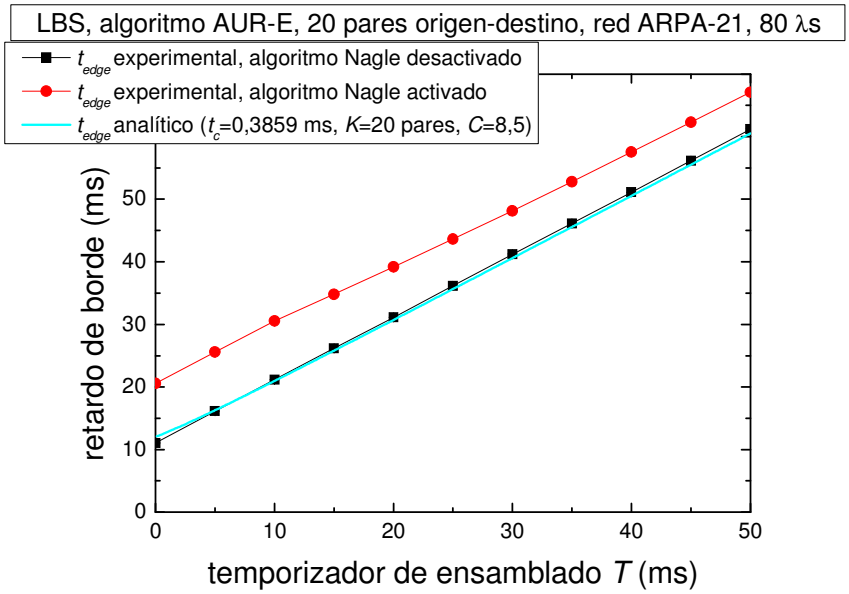


Figura 9-19 Retardo de borde medio para LBS con AUR-E, red ARPA-2 de 21 nodos, 80 lambdas y 20 pares

Análisis del retardo de borde en FBAT

Una vez se ha realizado el análisis del retardo de borde medio en LBS, a partir de este punto se realiza el análisis del retardo de borde medio con FBAT. Se va a emplear la misma metodología, por lo que se empieza el análisis con un caso de tiempo de cálculo pequeño, primero con 56 pares origen-destino y después con 20 pares, y a continuación se estudia el caso con tiempo de cálculo alto, también primero con 56 pares y después con 20.

Caso FBAT con 56 pares origen-destino y tiempo de cálculo pequeño

Los medidas experimentales del retardo de borde para distintos valores del temporizador de ensamblado *T* se muestran en la Figura 9-20, tanto para el caso en el que se activa el algoritmo de Nagle de TCP, como para el que se desactiva dicho algoritmo. Junto con los valores experimentales se muestra el valor del retardo de borde obtenido empleando el modelo analítico con *C*=20,6 (obtenido a partir de la media y varianza del proceso de llegada de mensajes *Request* en FBAT de la Tabla 9-4 y la caracterización del tiempo de cálculo de la Tabla 9-5). Aunque en la Figura 9-20 se muestran resultados experimentales a partir de un temporizador de ensamblado de 10 ms se han realizado experimentos con un valor menor del temporizador. Sin embargo, en esos casos con el valor del temporizador por debajo de 10 ms, la red no es estable y el emulador WR-OBS implementado no dispone de *threads* suficientes para mantener la tasa de envío de peticiones al PCE. No se muestra ningún valor en la Figura 9-20 para los casos en los que

la red no es estable y el emulador WR-OBS no puede mantener el temporizador de ensamblado fijado.

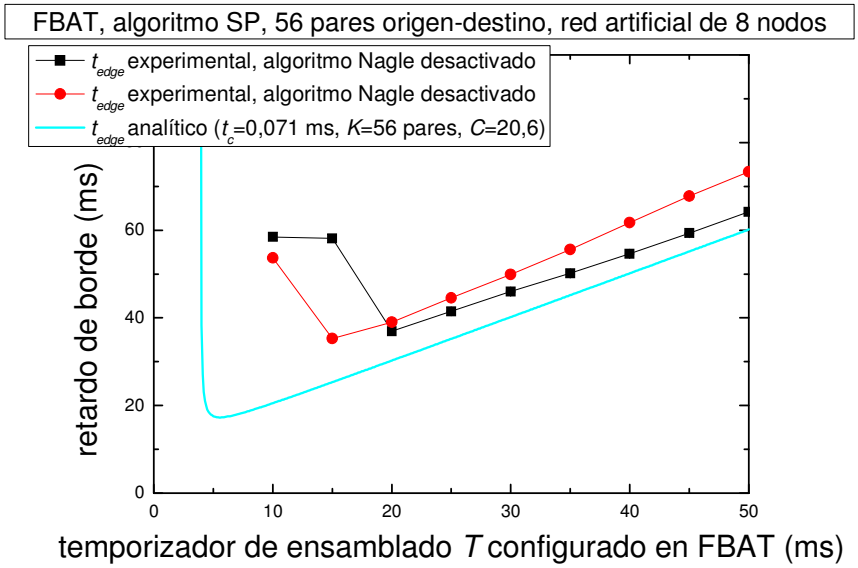


Figura 9-20 Retardo de borde medio para FBAT con SP en red de 8 nodos, y 56 pares

El modelo analítico en FBAT predice que el retardo de borde medio disminuye hasta cierto punto donde la red es inestable. Los resultados experimentales, como se aprecia en la Figura 9-20, siguen la dicha tendencia pero la inestabilidad ocurre en otro punto. El motivo, como se ha comentado anteriormente, es que emulador WR-OBS agota los *threads* y no es capaz de superar situaciones de congestión en la que haya muchas peticiones enviadas al PCE sin contestar. Por otro lado, el modelo analítico se ajusta más al caso en el que se desactiva el algoritmo de Nagle, por las razones ya comentadas en experimentos anteriores.

Caso FBAT con 20 pares origen-destino y tiempo de cálculo pequeño

Se ha observado que, con 56 pares, se produce una inestabilidad del retardo de borde en el entorno emulado con temporizadores de ensamblado por debajo de 20 ms. En este caso, se repite el experimento anterior reduciendo el número de pares origen destino a 20.

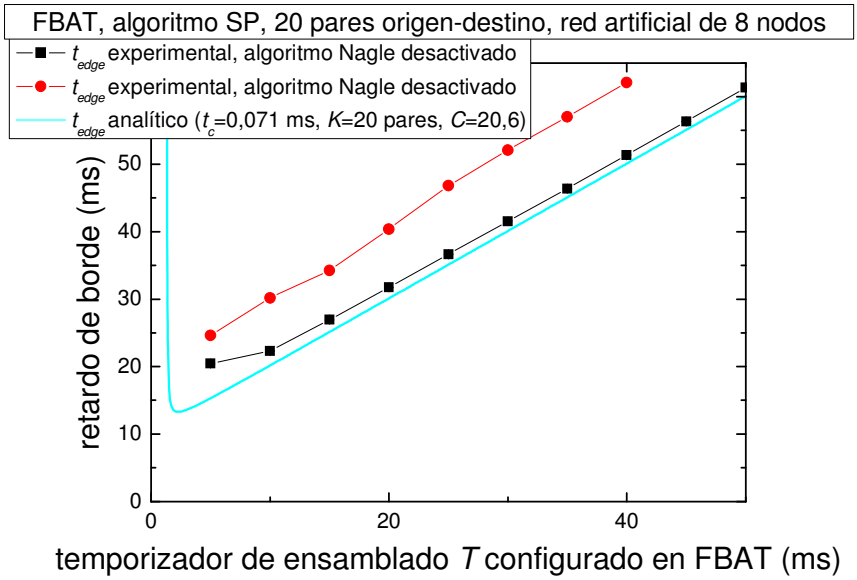


Figura 9-21 Retardo de borde medio para FBAT con 20 pares, con SP en red de 8 nodos

En este caso el retardo de borde medio experimental, mostrado en la Figura 9-21, es estable a partir de un temporizador de ensamblado de 5 ms. Dicho valor está cerca del mínimo teórico del retardo de borde medio que predice el modelo analítico, alrededor de 1 ms. Por tanto, con menos pares origen-destino que gestionar, el emulador WR-OBS tiene menos problemas para mantener la tasa de envío de mensajes *Request* al ritmo que marca el temporizador de ensamblado. Para temporizadores muy bajos, entre 1 y 5 ms, el entorno emulado sigue siendo inestable, incapaz de alcanzar los valores de retardo de borde del modelo analítico. Asimismo, el modelo se ajusta en gran medida al caso en el que se desactiva el algoritmo de Nagle. Cuando se activa el algoritmo de Nagle, el retardo de borde obtenido es aproximadamente 10 ms mayor, por las razones ya comentadas.

Caso FBAT con 56 pares origen-destino y tiempo de cálculo alto

En este caso se analiza qué ocurre cuando el tiempo de cálculo es alto. Para ello, se emplea el algoritmo AUR-E en el PCE y se utiliza la red ARPA-2 de la Figura 9-6 con 80 lambdas. Los resultados experimentales, para el caso en el que se emplean 56 pares origen destino, se muestran en la Figura 9-22 junto con los valores del retardo de borde obtenidos con el modelo analítico (con el valor de C=20,6 escogido para FBAT y el tiempo medio de cálculo de la Tabla 9-5). Por debajo de 25 ms se han obtenido resultados inestables, por lo que no se muestran en la Figura 9-22.

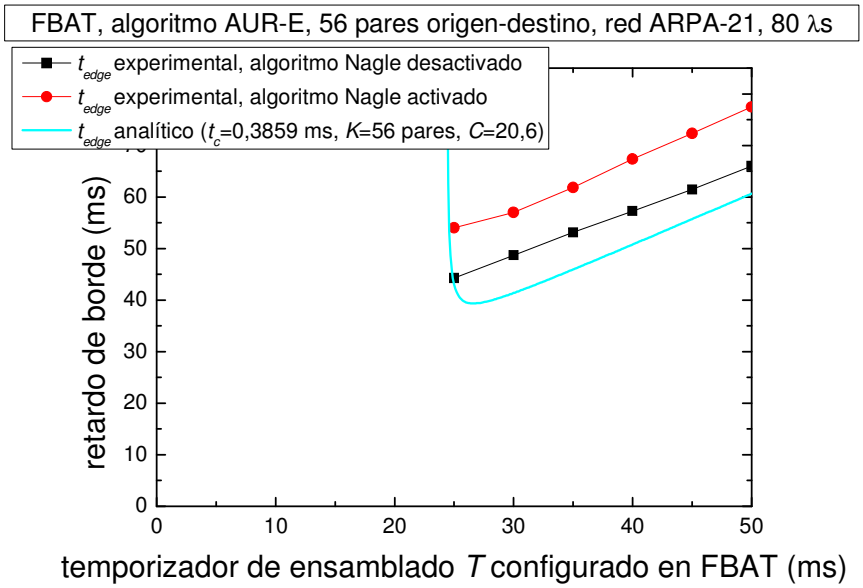


Figura 9-22 Retardo de borde medio para FBAT con *K* = 56 pares, con AUR-E en red ARPA-2

En el caso que se ha emulado, como se observa en la Figura 9-22, los resultados experimentales siguen la tendencia del modelo analítico y el punto donde la red es inestable prácticamente coincide. Aunque los resultados experimentales siguen la tendencia del modelo analítico, cuando se desactiva el algoritmo de Nagle, el resultado experimental es aproximadamente 5 ms mayor que el valor del modelo analítico, mientras que activando el algoritmo de Nagle de TCP, la diferencia sube a 18 ms.

Caso FBAT con 20 pares origen-destino y tiempo de cálculo alto

Para finalizar el estudio del retardo de borde medio en FBAT, se repite el experimento anterior, en el que se estudió el caso del tiempo de cálculo alto con 56 pares origen-destino, pero con un número de pares origen destino menor, 20. Los resultados experimentales del retardo de borde medio así como los valores del modelo analítico se muestran en la Figura 9-23. En esta ocasión, los resultados estables se obtienen a partir de un temporizador de ensamblado en FBAT de 10 ms.

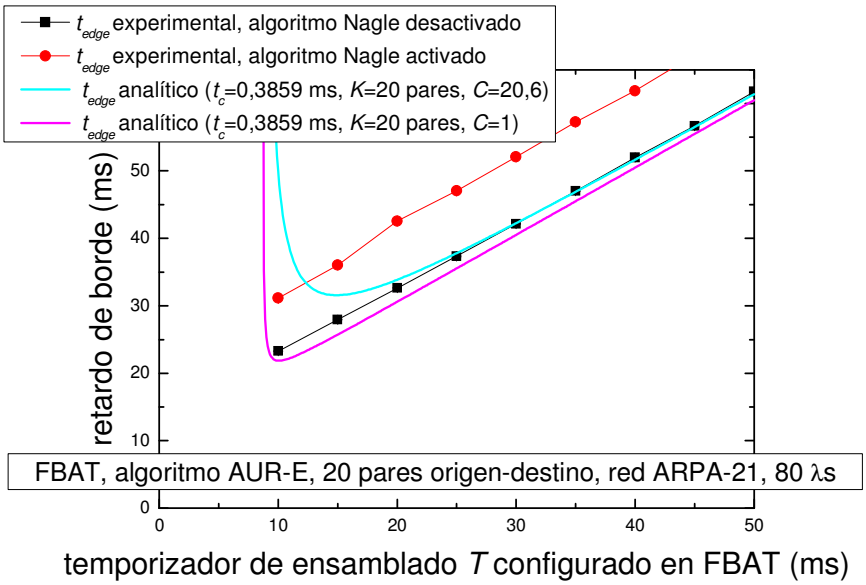


Figura 9-23 Retardo de borde medio para FBAT con *K* = 20 pares, algoritmo AUR-E en red ARPA-2

En este caso, los valores del modelo analítico (con el coeficiente C=20,6 que se ha empleado a lo largo del apartado, línea azul en la Figura 9-23) coinciden con los resultados experimentales a partir de un valor del temporizador de 20 ms. Para valores menores del temporizador, el modelo analítico predice un incremento del retardo de borde medio hasta que a red es inestable aproximadamente por debajo de 10 ms de temporizador de ensamblado. Sin embargo, los resultados experimentales muestran que el retardo de borde medio sigue descendiendo hasta llegar a los 10 ms. Por debajo de los 10 ms, los experimentos muestran inestabilidades. Por tanto, el punto de inestabilidad de la red coincide entre modelo analítico y experimentos. Se ha probado a introducir en el modelo analítico un coeficiente *C* menor, que represente un proceso de llegadas de mensajes *Request* con una varianza menor. Cabe destacar que el proceso de llegadas en FBAT se caracterizó para un temporizador de ensamblado de 40 ms y 56 pares origen destino. Por tanto, cuando el temporizador de ensamblado es bajo y además hay 20 pares origen-destino, la caracterización puede cambiar. Así, en la Figura 9-23 se ha incluido la curva de valores del modelo analítico empleando *C*=1. En esta ocasión, el modelo analítico con *C*=1 y los resultados experimentales desactivando el algoritmo de Nagle tienen un comportamiento similar, con apenas un par de milisegundos de diferencia entre ambos.

9.5 Análisis del número medio de *lightpaths*

Una de las principales ventajas de WR-OBS frente a las redes de conmutación de circuitos estática (OCS, *Optical Circuit Switching*), en las que se establecen *lightpaths* entre cada par de nodos entre los que hay demanda de tráfico, es la compartición de recursos. Así, para cursar el mismo tráfico, el número de recursos necesarios en WR-OBS es menor que en OCS. Un parámetro que nos ayuda a medir el rendimiento de WR-OBS en cuanto a uso de recursos es el número medio de *lightpaths* establecidos entre cada par origen-destino. En el caso de una red OCS estática, habrá un *lightpath* establecido entre cada par origen-destino en el que haya tráfico, independientemente de la carga (partiendo del supuesto de que el tráfico no excede la capacidad de un *lightpath*, en cuyo caso se necesitaría más de un *lightpath* para un par origen destino). En el caso de una red WR-OBS, el número de *lightpaths* establecidos varía en el tiempo. Si el tráfico ofrecido puede cursarse con un número medio de *lightpaths* por par origen-destino menor que uno, entonces la WR-OBS se comporta de una manera más eficiente que OCS. En esta sección se analiza, primero de manera teórica,

particularizando para el escenario emulado, y a continuación de manera experimental en el entorno emulado descrito anteriormente el número medio de *lightpaths* para distintos casos.

9.5.1 Análisis teórico del número medio de *lightpaths*

El análisis de número medio de *lightpaths* establecidos se realiza en primer lugar de manera genérica para WR-OBS y después se particulariza para cada mecanismo de construcción de ráfagas y a las particularidades del entorno emulado.

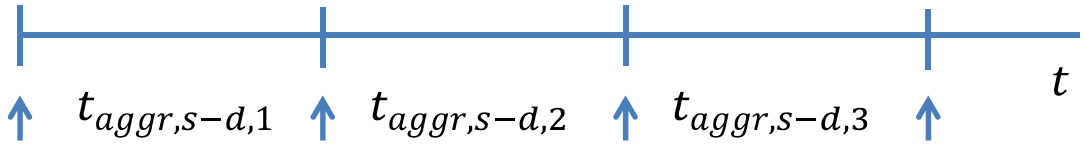


Figura 9-24 Ciclos de agregación en WR-OBS

En una red WR-OBS, para un par origen-destino, el tiempo que transcurre entre el fin de formación de una ráfaga y el comienzo de la siguiente se puede considerar despreciable [248]. De esta forma, para un determinado par *s-d* hay unos ciclos consecutivos de agregación de paquetes en ráfagas, como se muestra en la Figura 9-24. En cada ciclo de agregación *i* llegan $t_{aggr,s-d,i}B_{in,s-d,i}$ bits, donde $t_{aggr,s-d,i}$ es el tiempo de agregación de ráfagas para el par *s-d* en el ciclo de agregación *i*, y es $B_{in,s-d,i}$ es la tasa binaria con la que llegan los datos al *buffer* en el que se forma la ráfaga asociada al par *s-d* en ese ciclo. Puesto que una vez establecido el *lightpath* los datos se transmiten a una tasa B_{core}, el tiempo que estará establecido dicho *lightpath* para enviar la ráfaga será:

$$t_{WHT,s-d,i} = \frac{t_{aggr,s-d,i}B_{in,s-d,i}}{B_{core}} + t_{ovh,s-d,i} \tag{9-6}$$

donde $t_{ovh,s-d,i}$ es la sobrecarga (es decir, el tiempo durante el cual los recursos del *lightpath* están reservados pero no se está transmitiendo por él).

Aplicando la definición de carga normalizada para un par origen-destino (ν_{s-d}) de la ecuación (9-1), se obtiene que el tiempo en el que estará establecido el *lightpath* es:

$$t_{WHT,s-d,i} = \nu_{s-d,i}t_{aggr,s-d,i} + t_{ovh,s-d,i} \tag{9-7}$$

Así, el número medio de *lighpaths* establecidos en la red asociados a un par *s-d* será por tanto:

$$\bar{\rho}_{s-d} = \frac{\sum_i \left(\nu_{s-d,i}t_{aggr,s-d,i} + t_{ovh,s-d,i}\right)}{\sum_i t_{aggr,s-d,i}} \tag{9-8}$$

Suponiendo que el número de bits que llegan en cada intervalo *i* al *buffer* es constante, y que por tanto $\nu_{s-d,i} \equiv \nu_{s-d}$, entonces el número medio de *lightpaths* establecidos en la red asociados a un par *s-d* será:

$$\bar{\rho}_{s-d} = \nu_{s-d} + \frac{\bar{t}_{ovh,s-d}}{\bar{t}_{aggr,s-d}} \tag{9-9}$$

A pesar de las limitaciones de dicha suposición, éste será el modelo que se utilizará para estimar dicho parámetro.

Particularización para el caso de LBS

En el caso de de LBS, el tiempo medio de agregación para un par origen-destino *s-d* $\bar{t}_{aggr,s-d}$ coincide con el retardo de borde $\bar{t}_{edge,s-d}$. Por otro lado, como el *lightpath* queda reservado en cuanto se ha calculado el camino, la sobrecarga viene dada por el tiempo de transmisión del mensaje *Response* del PCE al nodo *s*, $t_{PCE-PCC}$ y el tiempo que tarda en reservarse el *lightpath*, que en el entorno emulado es $t_{RSVP} = 2 \cdot t_{prop,s-d}$, donde $t_{prop,s-d}$

es el tiempo de propagación entre el nodo origen y el nodo destino por la ruta calculada por el PCE. De esta forma se tiene que el número medio de *lightpaths* por par origen-destino en LBS es

$$\bar{\rho}_{LBS} = \nu_{s-d} + \frac{t_{PCE-PCC} + 2 \cdot t_{prop,s-d}}{\bar{t}_{edge,s-d}} \tag{9-10}$$

Hay que recordar que el retardo de borde para LBS se puede obtener a partir de la ecuación (9-4). Por tanto, el número medio de *lightpaths* ($\bar{\rho}_{LBS}$) se puede obtener con la expresión (9-11).

$$\bar{\rho}_{LBS} = \nu_{s-d} + \frac{t_{PCE-PCC} + 2 \cdot t_{prop,s-d}}{0{,}5 \cdot \left(A + K\bar{t}_c + \sqrt{(K\bar{t}_c - A)^2 + K{\bar{t}_c}^2 C} \right)} \tag{9-11}$$

Nótese que al suponer tráfico uniforme, el número medio de *lightpaths* en la red (suponiendo que no hay bloqueo) es $\bar{\rho}_{LBS}K$.

Particularización para FBAT

Cuando se emplea FBAT, el tiempo medio de agregación para un par origen-destino *s-d* es igual al temporizador de ensamblado *T*. La sobrecarga es el tiempo de transmisión del mensaje *Response* del PCE al nodo *s*, $t_{PCE-PCC}$ y el tiempo que tarda en reservarse el *lightpath*, que en el entorno emulado es $t_{RSVP} = 2 \cdot t_{prop\_s-d}$, donde t_{prop_s-d} es el tiempo de propagación entre el nodo origen y el nodo destino por la ruta calculada por el PCE. De esta forma, se tiene que el número medio de *lightpaths* por par origen-destino en FBAT se obtiene mediante la ecuación (9-12).

$$\bar{\rho}_{FBAT} = \nu_{s\_d} + \frac{t_{PCE\_PCC} + 2 \cdot t_{prop\_s-d}}{T} \tag{9-12}$$

9.5.2 Análisis de resultados experimentales

Se ha realizado un conjunto de experimentos donde se ha medido el número medio de *lightpaths* para un par origen-destino concreto. Para ello, se ha anotado en el emulador de WR-OBS, para el par origen-destino a examinar, el instante de tiempo en el que recibe el mensaje *Response* del PCE y el instante en el que se ha liberado el *lightpath*. Periódicamente, cada 500 ms se calcula el número medio de *lightpaths* en dicho intervalo a partir de los datos almacenados y se guarda. Una vez que el análisis del número medio de *lightpaths* converge, se considera el dato como válido.

Al igual que en los experimentos del retardo de borde, se han elaborado dos casos principales con respecto al tiempo de cálculo, el denominado "tiempo de cálculo pequeño", en el que se emplea la topología de la red artificial de 8 nodos mostrada en la Figura 9-5 del apartado 9.1.3 y en el PCE se emplea para calcular la ruta del *lightpath* el algoritmo del camino más corto (SP, *Shortest Path*) y el denominado "tiempo de cálculo alto", con la topología de la red ARPA-2, mostrada en la Figura 9-6, con 80 lambdas por enlace y el PCE emplea el algoritmo AUR-E. De manera similar, se realizan unas prueba con *K*=56 pares y otras con *K*=20 pares.

La mayor particularidad de esta prueba es que se ha emulado una situación de recursos infinitos, por lo que la reserva de *lightpaths* en el PCE no llega a realizarse, para que no haya bloqueo de ráfaga. Asimismo, se ha simplificado el proceso de establecimiento de tal forma que todos los establecimientos duran 5 ms ($t_{prop\_s\_d}$ = 2,5 ms), independientemente de la respuesta con el PCE. Por otro lado, se asume un tráfico uniforme. Así, como todos los tiempos de propagación son iguales, el número medio de *lightpaths* por par s-d (ρ) es el mismo para todos los pares *s-d*. Se realizan experimentos tanto con LBS como con FBAT

y se configura el uso o no del algoritmo de Nagle de TCP. En cada caso estudiado se realiza un barrido por varios valores del temporizador de ensamblado T y de carga. La duración de cada uno de los experimentos realizados en este apartado viene dada por varios factores: se emula hasta que se cumpla que el análisis del número medio de *lightpaths* haya convergido.

Caso LBS con 56 pares origen-destino y tiempo de cálculo pequeño

El primer experimento para obtener el número medio de *lightpaths* por par s-d, ρ, se realiza con el mecanismo de construcción de ráfagas LBS, con 56 pares origen-destino con la red artificial de 8 nodos y el algoritmo SP. En la Figura 9-25 se muestra ρ en función del valor del temporizador T, tanto el medido experimentalmente como el valor del análisis teórico, alimentado con los valores del retardo de borde medio obtenidos en el apartado anterior para una carga ν de 0,2.

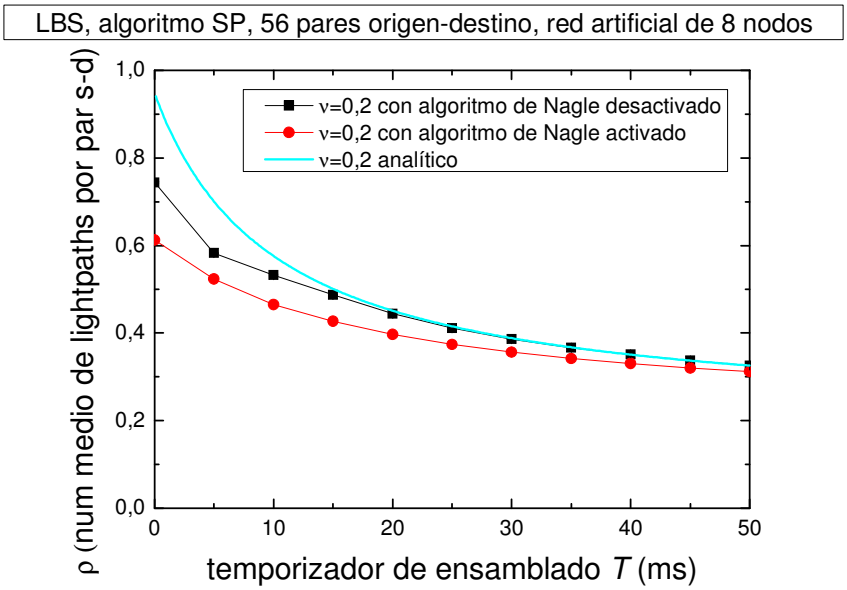


Figura 9-25 ρ para ν=0,2, con LBS, 56 pares, SP en red de 8 nodos, experimental y teórico

Se observa en la Figura 9-25 cómo los resultados experimentales siguen a la curva del modelo analítico a partir de un valor de temporizador de ensamblado de 15 ms. Para valores pequeños del temporizador, los resultados experimentales se alejan del modelo. Tal y como era de esperar, se se activa el algoritmo de Nagle, ρ es diferente, en este caso menor, sobre todo para temporizadores bajos. Es decir, activando el algoritmo de Nagle se logra un mejor uso de los recursos (ρ menor para la misma carga) a costa de un retardo de borde mayor.

Se repite la prueba para distintos volúmenes de carga normalizada ν. El resultado se muestra en la Figura 9-26, en la que se muestra un conjunto de curvas con los valores del modelo analítico del número medio de *lightpaths* (ρ) en función de la carga junto con puntos (los símbolos cuadrado, triángulo y círculo) en los que se muestra el resultado de cada experimento, en los que se ha desactivado el algoritmo de Nagle. En la Figura 9-26 se aprecia que hasta la carga 0,6 el número medio de *lightpaths* (ρ) permanece por debajo de 1 para temporizadores mayores de 10 ms. Sin embargo, con una carga alta como es 0,8 el número medio de *lightpaths* es siempre superior a uno. Por otro lado, se comprueba como el modelo se ajusta muy bien para todas las cargas que se han emulado, con solo un poco de desvío en el caso de temporizadores de ensamblado bajos. El desvío se hereda de la desviación del retardo de borde con respecto al modelo analítico con temporizadores bajos, como se mostró en la Figura 9-16.

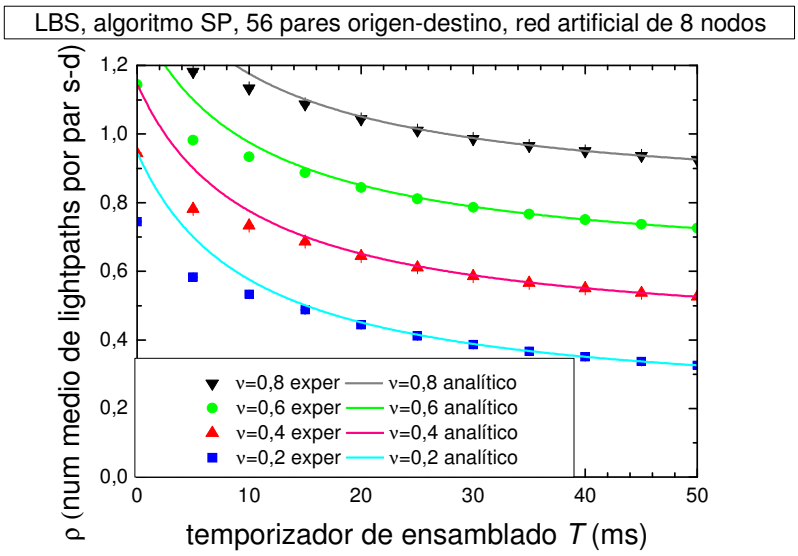


Figura 9-26 *ρ* para distintos valores de carga (*ν*), con LBS, 56 pares origen destino, SP, Nagle desactivado

Caso LBS con 20 pares origen-destino y tiempo de cálculo pequeño

A continuación se reduce el número de pares a 20, para examinar un escenario en el que el cliente no tenga ningún potencial problema de rendimiento. El resultado se muestra en la Figura 9-27. En ese caso el resultado experimental y teórico encajan perfectamente. Para valores de los temporizadores bajos (entre 0-10 ms), el número medio de *lightpaths* es relativamente alto, siendo WR-OBS eficiente únicamente en cargas bajas para dichos temporizadores. Se ve cómo, para temporizadores mayores la eficiencia aumenta (a costa de un mayor retardo de borde).

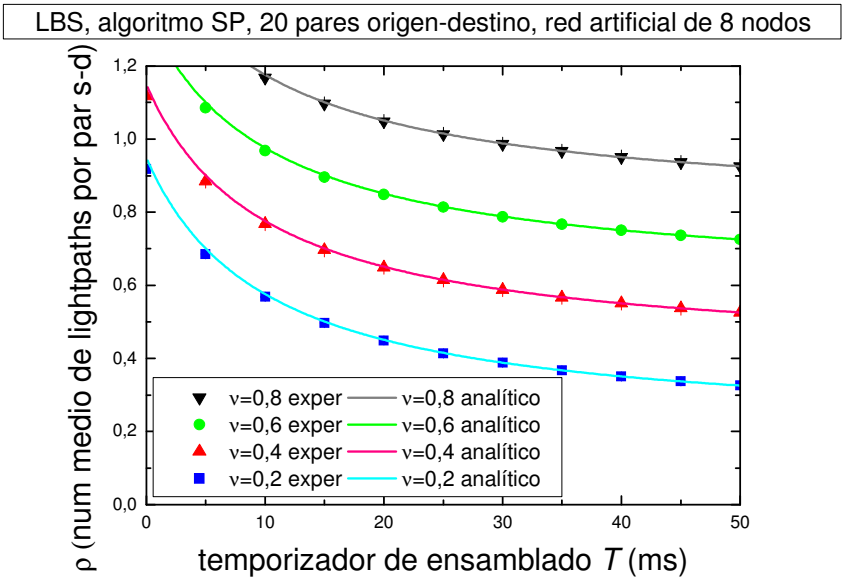


Figura 9-27 *ρ* para distintos valores de carga (*ν*), con LBS, 20 pares origen destino, SP, Nagle desactivado

Caso LBS con 56 pares origen-destino y tiempo de cálculo alto

Se repiten las pruebas anteriores con un algoritmo más complejo para tener un tiempo de cálculo elevado. Los resultados, con el algoritmo de Nagle de TCP desactivado, se muestran para 56 pares origen destino en la Figura 9-28. El resultado experimental en el entorno emulado coincide con los valores del modelo analítico del número medio de *lightpaths*. La principal diferencia con respecto al caso del algoritmo de cálculo sencillo es que, al tardar más en calcular el PCE (y por tanto en responder), el retardo de borde ya no

desciende más si se disminuye el temporizador y el número medio de *lightpaths* permanece prácticamente constante.

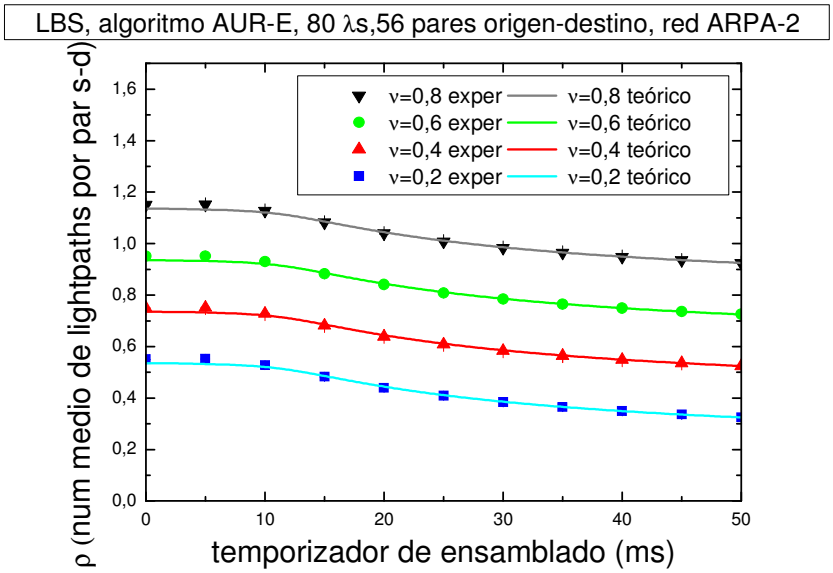


Figura 9-28 ρ para distintos valores de carga (ν), con LBS, 56 pares origen destino, AUR-E, Nagle desactivado

Caso LBS con 20 pares origen-destino y tiempo de cálculo alto

Si el número de pares es bajo, 20, aunque el tiempo de cálculo sea alto, el retardo de borde sigue bajando para temporizadores pequeños (de hecho hasta T=0 sigue bajando), por lo que el número medio de *lightpaths* aumenta al disminuir el temporizador (Figura 9-29) al igual que en los resultados con el algoritmo sencillo, cuyos resultados se mostraron en la Figura 9-26 y Figura 9-27.

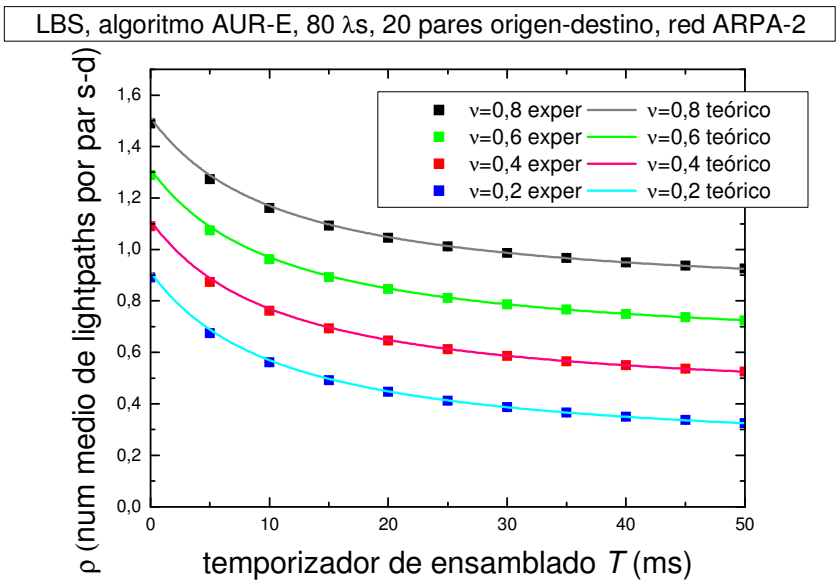


Figura 9-29 ρ para distintos valores de carga (ν), con LBS, 20 pares origen destino, AUR-E, Nagle desactivado

Caso FBAT con 56 pares origen-destino y tiempo de cálculo pequeño

En este apartado se analiza el número medio de *lightpaths*, ρ, para FBAT. En este caso, el modelo teórico solo se separa para el valor más bajo del temporizador de ensamblado de 10 ms, como se aprecia en la Figura 9-30. Esto se debe a que, para ese valor la red se empieza a volver inestable y el emulador WR-OBS tiene problemas con los *threads* disponibles, como se comentó en el análisis del retardo de borde, lo que repercute en un número medio de *lightpaths* menor que el teórico para ese valor de T. Si se comparan los

resultados de FBAT y LBS, para la misma carga, un caso en el que se obtiene un retardo de borde parecido (por ejemplo, un retardo de borde de aproximadamente 50 ms se obtiene con T=40 ms tanto en LBS como FBAT), la utilización es similar. La principal diferencia entre LBS y FBAT radica en que con FBAT no se consigue bajar de un retardo de borde de 30 ms, mientras que en LBS se puede bajar hasta 10 ms en el entorno emulado.

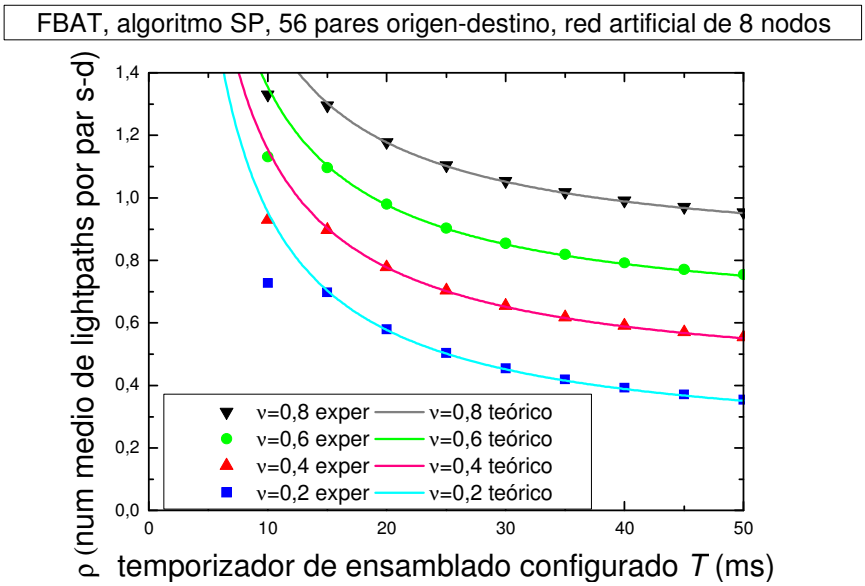


Figura 9-30 ρ para distintos valores de v, FBAT, 56 pares, AUR-E, algoritmo de Nagle desactivado

Caso FBAT con 20 pares origen-destino y tiempo de cálculo pequeño

En este este escenario se realizan las mismas pruebas experimentales que en el punto anterior, pero reduciendo el número de pares origen destino a 20. Los resultados experimentales, como se muestra en la Figura 9-31, siguen fielmente las curvas del modelo analítico. Como el número de pares origen destino es pequeño (20), el emulador no tiene problemas de rendimiento con temporizadores de ensamblado bajos. Por debajo de 10 ms la red no es estable y no se muestran ningún resultado para valores de T. En este caso, para cargas de hasta 0,6 el número de *lightpaths* por par origen-destino es inferior a uno, es decir, es eficiente en comparación con una red OCS.

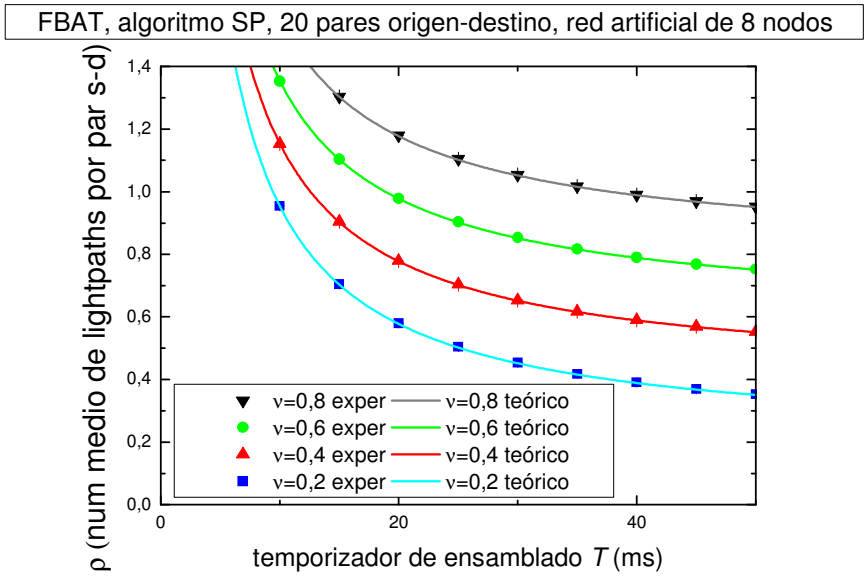


Figura 9-31 ρ para distintos valores de v, FBAT, 20 pares, AUR-E, algoritmo de Nagle desactivado

9.6 Análisis experimental de la probabilidad de bloqueo

En los experimentos realizados en las secciones anteriores se ha realizado realizando la simplificación de que no hay falta de recursos y, para ello, no se actualizan los recursos consumidos en el PCE. En esta sección del capítulo, se estudia un escenario en el que el número de recursos es finito, y por tanto hay probabilidad de bloqueo no nula. Es decir, cuando se solicite una ruta para un *lightpath* al PCE existe la posibilidad de que no haya recursos libres.

Los experimentos realizados en esta sección emplean la topología de red NSFNet, mostrada en la Figura 9-7, en la que hay unos nodos que actúan de nodos de borde e inyectan tráfico (la lista de nodos de borde se mostró en la Tabla 9-2) y otros nodos únicamente conmutan *lightpaths*. El tiempo de establecimiento del camino en estos experimentos viene determinado por el retardo de propagación del mismo, que se incluye como métrica en la respuesta del PCE. El criterio de parada de cada experimento realizado es que el análisis de la probabilidad de bloqueo media haya convergido, y hayan transcurrido como mínimo 40 minutos de experimento. De esta forma, en muchos casos se generan en torno a 2 millones de ráfagas. En cada experimento se varía el número de longitudes de onda disponibles, así como la carga normalizada ν y el valor del temporizador de ensamblado T.

9.6.1 Probabilidad de bloqueo en LBS

En este apartado se obtiene experimentalmente la probabilidad de bloqueo para distintos valores de carga normalizada ν y temporizador de ensamblado T en LBS. Se comienza con un escenario con recursos escasos, 4 lambdas, y se sube hasta 8 lambdas.

El primer experimento se ha realizado limitando los recursos a 4 longitudes de onda disponibles por nodo, y se han realizado pruebas con el temporizador de ensamblado T a 0, 10 y 20 ms. Con tan solo 4 lambdas disponibles, la probabilidad de bloqueo es alta, como puede observarse en los resultados mostrados en la Figura 9-32, incluso para cargas bajas, con todos los valores del temporizador. Con el valor del temporizador más alto de los analizados, T=20 ms, se consiguen los mejores resultados en cuanto a probabilidad de bloqueo, con poca diferencia, ya que los recursos son muy escasos.

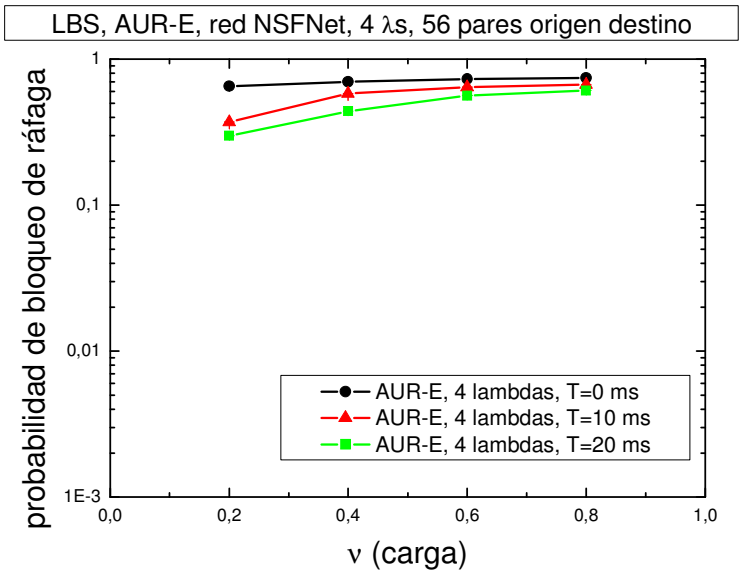


Figura 9-32 Probabilidad de bloqueo en LBS, variando la carga, en NSFNet con 4 lambdas

En el siguiente bloque de experimentos, se aumentan los recursos a 6 longitudes de onda por enlace. Los resultados se muestra en la Figura 9-33, donde se aprecia una

reducción alta de la probabilidad de bloqueo con respecto al caso anterior, ya que hay más recursos, especialmente para temporizadores altos y cargas bajas. Se puede observar cómo para carga 0,2 y T=20 ms, se obtiene una probabilidad de bloqueo alrededor de 10^{-5}, un valor adecuado para la operación de una red. En el resto de casos, los valores siguen siendo elevados, por encima del 10% en la mayor parte de los experimentos, excepto para T=10 ms y carga 0,2 en la que se obtiene un 2% aproximadamente.

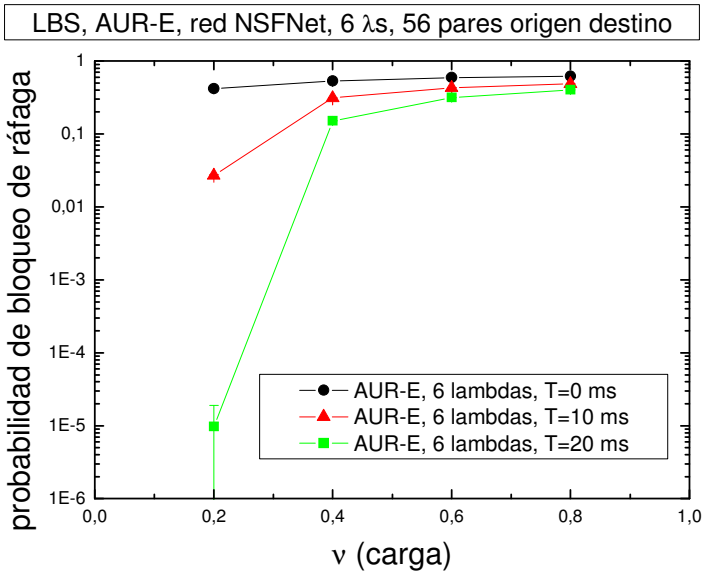


Figura 9-33 Probabilidad de bloqueo en LBS, variando la carga, en NSFNet con 6 lambdas

Por último, se ha elevado de nuevo el número de recursos a 8 longitudes de onda por enlace. Los resultados, recogidos en la Figura 9-34, muestran que, para T = 20 ms, se obtienen resultados de bloqueo inferiores a 10^{-4} hasta carga 0,4, y con T=10 ms, ya se consigue para carga baja, 0,2, un valor por debajo de 10^{-4}.

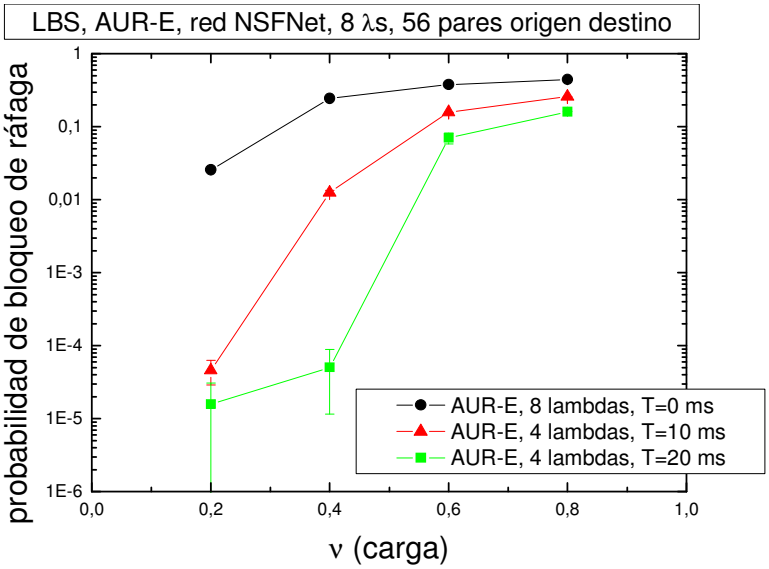


Figura 9-34 Probabilidad de bloqueo en LBS, variando la carga, en NSFNet con 8 lambdas

Los resultados anteriores se han mostrado con una curva para cada valor del temporizador de ensamblado *T*, sacando como conclusión que un valor de *T* alto (20 ms) obtiene los mejores resultados en cuanto a bloqueo. Los mismos resultados obtenidos se han organizado de forma diferente y se han obtenido curvas para cada número de lambdas. En la Figura 9-35 se muestran los resultados fijando *T*=0, en la Figura 9-36 T=10 ms, y en la Figura 9-38 T=20ms. En estas gráficas se puede comprobar que 4 y 6 lambdas son insuficientes para lograr un bloqueo aceptable en la gran mayoría de los casos.

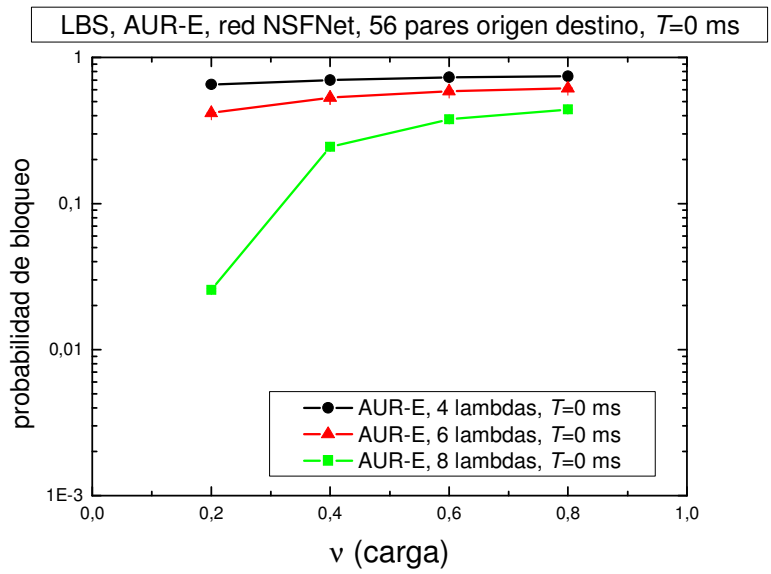


Figura 9-35 Probabilidad de bloqueo en LBS, variando la carga, en NSFNet con *T*=0

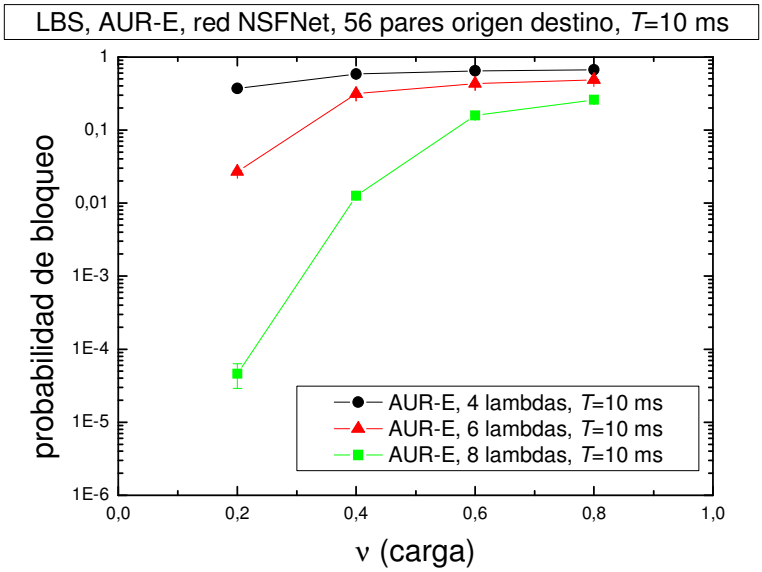


Figura 9-36 Probabilidad de bloqueo en LBS, variando la carga, en NSFNet con *T*=10 ms

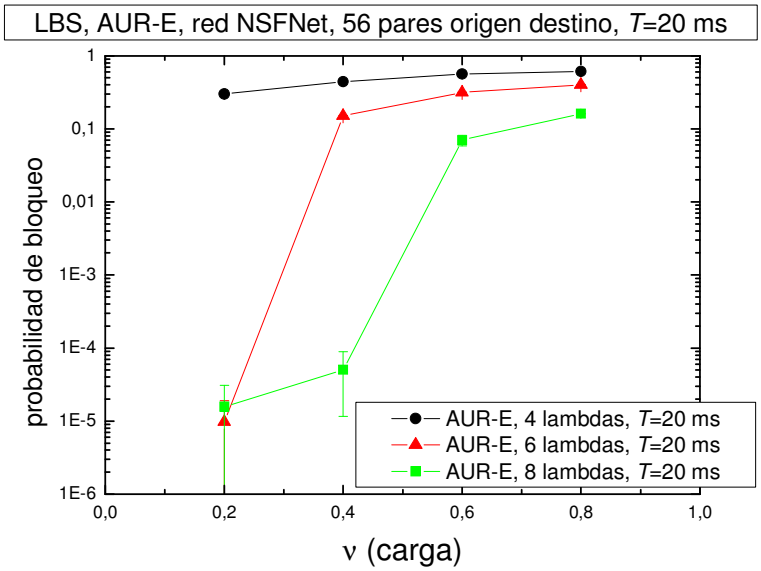


Figura 9-37 Probabilidad de bloqueo en LBS, variando la carga, en NSFNet con *T*=20 ms

9.6.2 Análisis de probabilidad de bloqueo en FBAT

Finalmente, se realiza un grupo de experimentos con el mecanismo FBAT. En este caso, el objetivo es poder comparar FBAT con LBS en escenarios similares. Se ha escogido, al igual que en el apartado anterior, un escenario con el algoritmo de cálculo más intensivo, el AUR-E, se ha fijado *T*=20 ms y se han fijado 3 posibilidades, 4, 6 y 8 longitudes de onda por enlace. Los resultados en cuanto a probabilidad de bloqueo se muestran en la Figura 9-38. Comparado con LBS, con el mismo valor de temporizador, se obtiene un bloqueo ligeramente inferior con 4 longitudes de onda, mientras que con recursos suficientes (6 lambdas y 8 lambdas), el bloqueo es ligeramente superior. Esto se debe a que en FBAT las peticiones llegan de una manera más seguida al PCE, por lo que la probabilidad de que una petición se encuentre con los recursos ocupados es mayor. Con 8 longitudes de onda, para cargas bajas se obtiene una probabilidad de bloqueo muy baja, aunque superior a la de LBS (donde se alcanzaba 10^{-5} frente a 10^{-4} en FBAT).

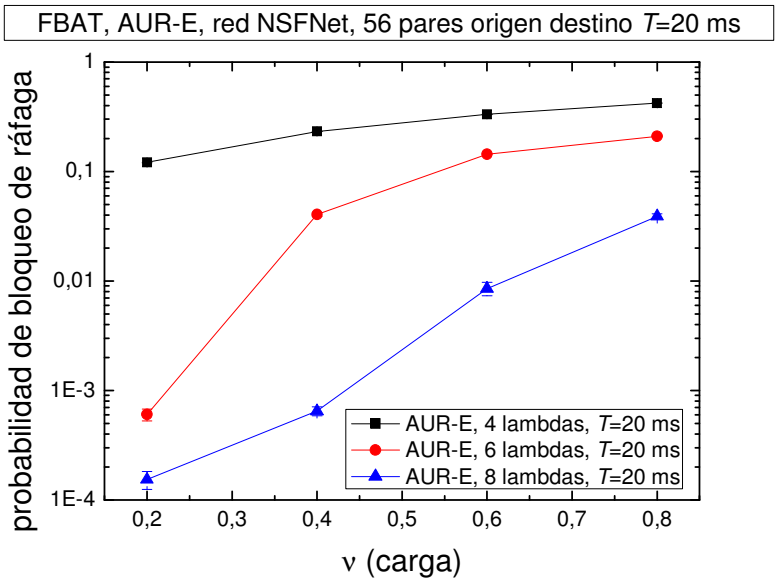


Figura 9-38 Probabilidad de bloqueo en FBAT, variando la carga, en NSFNet con *T*=20 ms

9.7 Análisis experimental del mecanismo de reintentos

En las pruebas del apartado anterior se ha visto que se pueden producir bloqueos cuando no se encuentra una combinación de camino y longitud de onda disponible para la petición de cálculo. Kozlovski y Bayvel [261] y de Miguel [248] demuestran en sus estudios los beneficios de volver a planificar una petición de RWA, siempre y cuando el tiempo que se tarda en procesar la petición se mantenga acotado.

Por esta razón, en el prototipo de PCE, cuyo diseño se mostró la Figura 8-11 del capítulo anterior, se ha implementado un mecanismo básico de reintentos para reducir la probabilidad de bloqueo. Para ello, cuando llega un mensaje *Request* al PCE, se le adjunta una marca de tiempo. Posteriormente, en el caso de que el algoritmo de cálculo de ruta y asignación de longitud de onda no encuentre una longitud de onda disponible por falta de recursos, se mira el tiempo que lleva dicha petición en el PCE a partir de la marca de tiempo mencionada. Si el tiempo es inferior que un límite configurable, la petición se introduce en la cola de reintentos (ver Figura 8-11). En cuanto el PCE detecte que han liberado longitudes de onda en algún enlace de la red, se pasan todas las peticiones de la cola de reintentos a la cola de tareas de cálculo.

En este apartado se realiza un experimento con LBS, 6 longitudes de onda y T = 10 ms, con y sin reintentos en el escenario de la red NSFNet con 56 pares origen-destino (similar al escenario del apartado 9.6). El límite de tiempo para que una petición pueda insertarse en la cola de reintentos se ha fijado a 10 ms. En la Figura 9-39 se muestra la probabilidad de bloqueo obtenida en el experimento. Puede observarse claramente cómo se reduce la probabilidad de bloqueo en un orden de magnitud. Además, se han realizado un conjunto de experimentos varios con otras configuraciones, en las que se corrobora el descenso de un orden de magnitud, tanto para LBS como FBAT. De hecho, se han logrado para cargas bajas y 8 lambdas con probabilidad de bloqueo inferior a 10^{-6}.

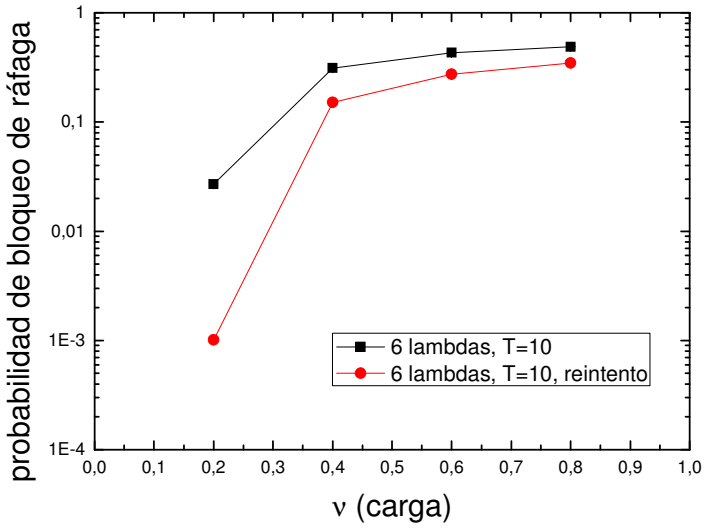


Figura 9-39 Probabilidad de bloqueo en FBAT, variando la carga, en NSFNet con *T*=10 ms

9.8 Conclusiones

En este capítulo se verificado la funcionalidad del PCE diseñado e implementado para una arquitectura WR-OBS, así como las extensiones al protocolo PCEP propuestas. Con el fin de realizar una evaluación experimental del rendimiento de una arquitectura WR-OBS con PCE, se ha implementado un entorno emulado, que se ha validado experimentalmente.

Se ha descubierto que TCP tiene un gran impacto en el rendimiento de la arquitectura de WR-OBS con PCE. Si se desactiva el algoritmo de Nagle, los mensajes PCEP, en vez de enviarse directamente en cuanto se escriben en el *socket*, esperan durante un tiempo determinado por si llegan más mensajes, lo que provoca que, por un lado, el tiempo de envío del mensaje al PCE aumente, y por otro que los mensajes enviados desde el emulador WR-OBS se acumulen y viajen en ráfagas. Por ejemplo, en uno de los experimentos realizados, con el algoritmo de Nagle activo, el 90% de los mensajes PCEP se transporta en un segmento TCP en el que hay otros mensajes PCEP.

El rendimiento obtenido en el entorno emulado se ha evaluado en cuanto a retardo de borde, número medio de *lightpaths* y probabilidad de bloqueo. Además, para el retardo de borde y número medio de *lightpaths* se han elaborado sendos modelos analíticos, particularizados tanto para LBS como para FBAT. Para alimentar los modelos analíticos se ha caracterizado el proceso de llegadas de mensajes *Request* al PCE, así como el tiempo de cálculo del PCE. Se ha comprobado que los resultados experimentales siguen las tendencias marcadas por los modelos analíticos. Solamente en algunos casos puntuales se han observado divergencias significativas. Se ha comprobado que la caracterización realizada de los procesos de llegada de mensajes *Request* y tiempo de cálculo puede emplearse como entrada de los modelos analíticos.

Se ha comprobado, como se ha visto a lo largo de la tesis, que TCP tiene una gran influencia en los resultados. En este caso, el algoritmo de Nagle juega un papel clave en el retardo de borde. Cuando se activa dicho algoritmo (por defecto está activado en los sistemas operativos modernos), se obtienen, para la mayoría de los casos, 10 ms más que si se desactiva.

La principal conclusión de este capítulo final es que la arquitectura de WR-OBS con PCE y RSVP es viable. Además, simplemente con un prototipo software funcionando sobre una arquitectura PC de propósito general se ha logrado un rendimiento adecuado, teniendo en cuenta que las pruebas experimentales son emulaciones en tiempo real. Como ejemplo del buen rendimiento del PCE y el entorno emulado, se han logrado tiempos de retardo de borde tan bajos como 10 ms y tiempos medios de cálculo por debajo del medio milisegundo para un algoritmo computacionalmente extensivo como el AUR-E.

Capítulo 10

Conclusiones y líneas futuras de investigación

10.1 Conclusiones

En esta tesis se ha realizado en primer lugar una profunda revisión del estado del arte de la tecnología OBS. Se ha constatado una clara evolución en esta tecnología, empezando por los estudios teóricos, primeros *testbeds* con sus pruebas de concepto, experimentos más complejos y finalmente las primeras implementaciones comerciales y su estandarización. La aparición de unos primeros productos comerciales, aún limitados a entornos metropolitanos, junto con los primeros paso hacia la estandarización, el los que participan las principales operadoras y fabricantes mundiales, confirma el interés de la industria por OBS.

Esta tesis ha realizado aportaciones al estado del arte en el estudio del comportamiento de flujos de datos que empleen el protocolo de transporte TCP sobre una red OBS. Ha propuesto y evaluado una técnica de encaminamiento para OBS y finalmente ha diseñado e implementado un elemento de cálculo de rutas basado en PCE para redes de conmutación de ráfagas OBS con encaminamiento por longitud de onda (WR-OBS). A continuación, se detallan las conclusiones de los distintos temas evaluados en esta tesis.

TCP sobre OBS

TCP en redes OBS se enfrenta al problema clásico de las redes de alta velocidad, al que se le añade una penalización por el retardo introducido en el proceso de ensamblado y un problema debido al descarte de ráfagas, que repercute en la pérdida de segmentos TCP. Mediante la simulación con O PNET *Modeler*, se han analizado cualitativamente los distintos escenarios de pérdidas, en los que se ha descrito con detalle el comportamiento de TCP. Los escenarios en los que la tasa de transmisión de TCP se ve más afectada son aquellos en los que TCP necesita esperar al vencimiento del temporizador de retransmisión. En el caso de pérdidas de múltiples segmentos, empleando asentimientos selectivos, TCP es capaz de realizar esa recuperación en caso de pérdidas múltiples como si de una pérdida simple se tratase. Los estudios tanto teóricos como mediante simulación o experimentales de la literatura han logrado estudiar con detalle la situación con un solo flujo. Asimismo, dichos estudios únicamente consideran las opciones estándar de TCP. En este sentido, en esta tesis se ha comparado de manera cualitativa y mediante simulación con OPNET Modeler el *throughput* de TCP sobre OBS con y sin la utilización del asentimiento retardado, una opción empleada habitualmente en redes asimétricas. El análisis ha señalado dos aspectos negativos y uno positivo cuando se utiliza dicho algoritmo: un aspecto negativo es un tiempo mayor para recuperarse de una pérdida de ráfaga. Otro aspecto negativo es una reducción de la ganancia de correlación, debido a una transmisión no ideal de los segmentos de TCP, que hacen que una ventana de TCP se distribuya en varias ráfagas. El aspecto positivo es la mayor facilidad para TCP de recuperarse de una pérdida de ráfaga empleando asentimientos selectivos, al no concentrarse en cada ráfaga toda la ventana.

Por tanto, si los clientes TCP, es decir, los usuarios que navegan, descargan, etc., tienen un acceso simétrico, conviene no utilizar el algoritmo de asentimiento retardado. A partir de una relación de ancho de banda de subida frente a ancho de banda de bajada de 1 a 30, se podría enviar un asentimiento por segmento sin que se resintiera el rendimiento de la conexión. Teniendo en cuenta que hay más tráfico, se podría establecer una relación aproximada de 1 a 10. A partir de esta relación de asimetría, se puede considerar que el asentimiento retardado deja de tener su beneficio y, por tanto, conviene desactivarlo. Otra opción es usar la técnica *de byte counting*, por la que se incrementa la ventana no por el número de asentimientos recibidos sino por el número de bytes. Sin embargo, esta opción prácticamente no se ha implementado, a pesar de que el IETF recomiende su uso.

Además de estudiar el impacto de un aspecto concreto de TCP, se ha añadido tráfico a un ensamblador OBS, y se ha comparado el comportamiento de TCP sobre OBS con y sin tráfico de fondo. Se ha comprobado que, si un flujo TCP se ensambla con flujos adicionales, se produce una fragmentación de la transmisión continua de la ráfaga, lo que lleva a una disminución del número de segmentos por ráfaga. Esto lleva a que cuando se estudia un flujo TCP junto con tráfico adicional, la distinción entre fuentes medias y rápidas se desvanece y se observa un comportamiento parecido en todos los casos. Esto se debe a que al introducir tráfico adicional, la transmisión de segmentos TCP se fragmenta en distintas ráfagas, y el número medio de segmentos por ráfaga disminuye.

Cuando se estudia TCP considerando múltiples fuentes de tráfico en una red OBS, se constata que una pérdida de ráfaga no es siempre tan negativa como se pensaba en estudios previos, al contener la mayoría de ráfagas pocos segmentos. De esta forma, empleando asentimientos selectivos, TCP puede recuperarse rápidamente de la mayor parte de pérdidas de ráfagas en OBS. Para las probabilidades de pérdida de ráfaga empleadas, TCP obtiene un buen rendimiento, sin ser penalizado en exceso por dichas pérdidas.

Modelo periódico de TCP sobre OBS

La tesis ha presentado un modelo periódico de TCP sobre OBS con el que obtener el rendimiento aproximado de un flujo que atraviesa una red OBS a partir de una serie de parámetros de la red, como son el ancho de banda del enlace de cuello de botella, el temporizador de ensamblado, el *Round Trip Time* (RTT) y el tamaño máximo de segmento (MSS).

Se ha comprobado que, asumiendo un pequeño error, se puede aplicar la suposición de que el número medio de segmentos por ráfaga sea constante. Para ello hay que aplicar un pequeño factor de corrección, que se ha denominado factor de partición de ráfagas α al modelo periódico, cuyo valor se ha estimado mediante simulación y que, para los casos estudiados, varía entre 1.3-1.5 para flujos medios y 1.5-4 para flujos rápidos dependiendo de la ventana. Aplicando este factor al modelo periódico, se puede obtener el rendimiento de una conexión TCP en OBS.

Aunque el modelo parte de conexiones TCP de duración infinita y un comportamiento periódico, se ha evaluado el rendimiento en un escenario con transferencias de ficheros con un tamaño finito, y se ha comprobado el buen comportamiento del modelo, especialmente en las zonas de probabilidad de pérdida de interés, entre 10^{-4} y 10^{-3}.

Con probabilidades de pérdida baja, el modelo sobreestima el rendimiento, ya que asume una recuperación sencilla, y no mediante temporizador.

Empleando el factor de partición, se obtienen resultados con muy poco error con respecto a la simulación. Aunque no se puede obtener el factor de partición a priori de manera exacta, se obtienen mediante simulación unas recomendaciones para el uso de dicho factor.

La principal crítica que se puede realizar al modelo es que el número de segmentos por ráfaga se ha comprobado mediante simulación que no es realmente una constante, y a medida que aumenta la ventana, este parece tener una tendencia a descender. Por tanto, una posible extensión del modelo pasaría por la introducción de un comportamiento no lineal en el número de segmentos por ráfaga según la ventana.

La principal ventaja del modelo es que, independientemente del factor de partición, para probabilidades de pérdida de ráfaga de 10^{-4}, valor que se puede afirmar como normal en una red no congestionada, el modelo obtiene un valor preciso en todos los casos simulados.

Sincronización de flujos TCP en OBS

Esta tesis ha estudiado el efecto de sincronización TCP que se da en situaciones en las que varias conexiones TCP comparten un enlace y los mecanismos de control de congestión de los clientes y servidores TCP involucrados en la transmisión reaccionan al mismo tiempo. Este efecto hace que el perfil del tráfico presente una mayor variación que en un caso en el que no hubiera sincronización y, de este modo, el ancho de banda que hay que provisionar en un enlace cuando hay sincronización TCP es mayor para obtener el mismo *throughput* medio. Por tanto, cuanto mayor sea la sincronización, menos eficiente es el uso de los recursos disponibles.

En una red de conmutación de ráfagas ópticas, la sincronización de flujos TCP está presente, ya que las ráfagas contienen segmentos de varios flujos, con lo que una pérdida implica la reducción de la tasa de transmisión de varias fuentes simultáneamente y su consecuente sincronización. Para medir el impacto de la sincronización, la tesis propone un índice de sincronización y un método para calcular la capacidad mínima necesaria para poder transportar los flujos TCP obteniendo el mismo rendimiento. Esta capacidad mínima se obtiene a partir de medidas de tráfico e indica cuál es el sobredimensionado necesario, que se define como el ancho de banda por encima de la media que se ha de proveer en un enlace y que es capaz de soportar los picos de tráfico.

Se ha analizado el dimensionado de capacidad y el impacto de la sincronización mediante simulación con OMNeT++ en el caso de flujos medios en los que la relación entre el ancho de banda de acceso y el valor del temporizador de ensamblado es tal que en una ráfaga pueden viajar varios segmentos de un mismo flujo, y en el caso de flujos rápidos en los que la relación entre el ancho de banda de acceso y el valor del temporizador de ensamblado es tal que el número de segmentos TCP que potencialmente caben en una ráfaga es lo suficientemente alto como para que TCP en la mayoría de las pérdidas de ráfaga deba recuperarse con el temporizador de retransmisión. Se han considerado dos escenarios, un escenario de caso peor, en el que hay una sincronización total, en el cual todos los flujos tienen exactamente el mismo RTT, y un escenario pragmático, en el que cada flujo tiene un RTT diferente. En cada escenario se estudian dos técnicas de ensamblado propuestas en la literatura, MF, en la que se emplea un único ensamblador para todos los flujos, y la técnica PF, en la que se emplea un ensamblador para cada flujo. Esta última técnica, PF servirá para obtener la cota inferior del sobredimensionado, ya que proporciona el mejor caso posible en cuanto a sincronización.

Además, con el objetivo de reducir las necesidades de ancho de banda, la tesis propone y estudia una técnica de ensamblado por la cual varios flujos con el mismo origen-destino comparten varias colas.

Los resultados del estudio demuestran que tener una cola única para todos los segmentos de un par origen-destino conduce a un aumento del ancho de banda a sobredimensionar, comparado con la situación ideal de una cola para cada flujo. Por ejemplo, en el escenario analizado, en el caso peor de sincronización total, se necesita hasta

un 126% de sobredimensionado. En el escenario pragmático, con RTTs variados, la sincronización disminuye, necesitándose un sobredimensionado del 66%, aún lejos del 20% del caso ideal. Cuando se pasa a compartir varias colas, se consigue reducir significativamente la varianza así como la necesidad de ancho de banda. Así, con 2 colas, en el escenario pragmático, se reduce el sobredimensionado necesario al 47%, y si se emplean 4 colas se reduce aún más hasta el 32%, ya cerca del caso ideal. Por tanto, empleando varias colas se reduce significativamente la necesidad de ancho de banda, sin aumentar en exceso la complejidad del ensamblado en OBS.

Encaminamiento multicamino en OBS

Uno de los principales problemas identificados en OBS es la contienda entre ráfagas, que puede dar lugar a la pérdida de las ráfagas, o si se emplean líneas de retardo a un aumento del retardo. Si se envían muchas ráfagas por un enlace, la probabilidad de pérdida de ráfaga en dicho enlace puede aumentar drásticamente, perjudicando la probabilidad de pérdida de ráfaga global en toda la red. Un buen encaminamiento del tráfico, como demuestran los estudios de optimización de la literatura, puede reducir dicha probabilidad de pérdida. Sin embargo, obtener toda la información para llevar a cabo esta optimización en un entorno real es muy complicado. Esta tesis realiza una propuesta de algoritmo de encaminamiento en tiempo real multicamino que emplea únicamente información local, obtenible en el nodo donde se produce la decisión de encaminamiento de la ráfaga. Para ello, se toma como punto de partida las ideas de la técnica de encaminamiento multicamino para redes IP denominada MRDV y se diseña una técnica específica para OBS, denominada AMOR (*Adaptive Multipath OBS Routing*).

Mediante simulación en ns2, en la tesis se evalúa el rendimiento de la técnica diseñada y se compara con el encaminamiento por el camino más corto y con ECMP (encaminamiento multicamino con el mismo coste), que reparte el tráfico entre todos los caminos con el mismo coste. Los resultados muestran que, cuando se usan nodos OBS sin FDL, se obtienen únicamente ligeras mejoras, ya que al emplear caminos más largos, se aumenta la carga en la red, y el hecho de evitar zonas saturadas no compensa el aumento de carga, a costa de un ligero incremento del retardo. En cambio, la técnica AMOR consigue mejoras significativas en nodos con FDLs, que se benefician en mayor medida de la reducción de carga cuando esta es alta, y no sufren apenas penalizaciones por un ligero aumento de carga, cuando ésta es baja. Como contraprestación, la técnica AMOR conlleva un ligero aumento del retardo. El estudio demuestra que el reparto de carga, aunque no sea perfecto por el hecho de emplear información local, y no global como sería lo óptimo (pero complicado de realizar), es beneficioso y se puede implementar fácilmente.

PCE para WR-OBS

Finalmente, se ha analizado una arquitectura WR-OBS elaborada a partir elementos definidos por los estándares actuales, como son el elemento de cálculo de caminos (PCE) y el protocolo de reserva de recursos (RSVP). Actualmente, el PCE está únicamente especificado para redes MPLS, aunque está empezando a especificarse para otra tipo de redes. Sin embargo, no hay ninguna propuesta en la literatura que lo proponga para WR-OBS. En este tesis se ha particularizado el protocolo PCEP (el protocolo de comunicación con el PCE) para redes WR-OBS y se han propuesto extensiones protocolares, que además se han implementado y validado experimentalmente. Por ejemplo, una de las extensiones propuestas permite reservar los recursos calculados en el PCE durante un tiempo determinado, así como cancelar dicha reserva en cualquier momento.

En la tesis se ha realizado el diseño de un PCE especializado en WR-OBS y se ha implementado un prototipado del mismo empleando Java 1.6 que responde en tiempo real peticiones PCEP *Request*. Se ha implementado el estándar PCEP junto con las extensiones

protocolares propuestas. Para ayudar a validar el PCE diseñado y la arquitectura propuesta se ha implementado un emulador de WR-OBS. Dicho emulador replica el comportamiento de nodos WR-OBS con dos mecanismos de construcción de ráfagas: LBS y FBAT.

Se ha caracterizado el proceso de llegadas de mensajes PCEP *Request* al PCE empleando LBS y FBAT. En la tesis se ha descubierto que un parámetro de TCP, denominado *tcp_no_delay,* mediante el cual se desactiva/activa el algoritmo de *Nagle*, tiene un gran impacto en el rendimiento. Si se desactiva el algoritmo de Nagle, los mensajes PCEP, en vez de enviarse directamente en cuanto se escriben en el socket, esperan durante un tiempo determinado por si llegan más mensajes, lo que provoca que, por un lado, el tiempo de envío del mensaje al PCE aumente, y por otro que los mensajes enviados desde el emulador WR-OBS se acumulen y viajen en ráfagas. Por ejemplo, en una de los experimentos realizados, con el algoritmo de Nagle activo, el 90% de los mensajes PCEP se transporta en un segmento TCP en el que hay otros mensajes PCEP.

En la tesis se ha caracterizado el tiempo de procesamiento en el PCE. Por ejemplo, en la caracterización empleando el algoritmo AUR-E en la red NSFNet de 21 nodos en una máquina con 4 núcleos y 2,87 GHz, se ha detectado que en la gran mayoría de los casos el tiempo de procesamiento está comprendido en un intervalo entre 200 μs y 800 μs. Sin embargo, en un pequeño porcentaje, inferior al 1%, el tiempo de procesamiento sube a varios ms, y en muy contadas ocasiones, a decenas de ms. Estos tiempos pueden corresponderse a ocasiones en las que el sistema operativo realiza otras tareas, el recolector de basura de Java (*Garbage Collector*) entra en acción o la escritura en el socket se ralentiza. Estos tiempos elevados, al ser muy puntuales, apenas afectan al tiempo medio de procesamiento, pero sí influyen en la varianza.

El rendimiento obtenido en el entorno emulado se ha evaluado en cuanto a retardo de borde, número medio de *lightpaths* y probabilidad de bloqueo. Se han elaborado modelos analíticos para el retardo de borde y el número medio de *lightpaths* particularizados para el entorno emulado. Se han validado los resultados experimentales con los valores de los análisis teóricos. Los resultados teóricos y experimentales coinciden en la gran mayoría de las ocasiones. Únicamente cuando se emula un elevado número de pares origen-destino y se emplean valores de temporizador de ensamblado bajos, condiciones en las cuales el emulador necesita enviar en tiempo real varias peticiones por milisegundo, el entorno emulado no ha sido capaz de alcanzar el rendimiento previsto por el modelo teórico.

Se ha implementado un mecanismo de reintentos en el PCE, de tal forma que si para una petición existe ruta (topológicamente hablando), pero no hay longitud de onda disponible, se pasa la petición a una cola de reintentos. Cuando se liberan recursos en la red, si el tiempo que lleva la petición en el PCE no supera un umbral, vuelve a pasar a la cola de peticiones. Se ha comprobado en los experimentos realizados con el emulador y el PCE que empleando el mecanismo de reintentos se reduce la probabilidad de bloqueo, con un mínimo incremento del retardo.

La principal conclusión de los experimentos realizados con el prototipo de PCE y el emulador WR-OBS diseñados e implementados en la tesis es que la arquitectura de WR-OBS con PCE es viable y realizable con elementos disponibles en el estado del arte. Por tanto, esta arquitectura podría implementarse en un entorno hardware (no emulado) para lograr un rendimiento mayor, acorde al esperado en una red operando tráfico real.

10.2 Líneas futuras de investigación

En este apartado se proponen ideas adicionales de puntos de la tesis que requieren más investigación, así como líneas nuevas de investigación a partir de las tecnologías estudiadas en la tesis.

OBS y *flexi-grid*

Esta tesis se ha centrado en el estudio de la arquitectura OBS, tanto en su versión con señalización sin confirmación, como en su versión WR-OBS. Reciente, ha surgido el concepto de *flexi-grid*. En las redes WDM tradicionales el espectro está dividido en unas rejillas de tamaño fijo (en el caso más habitual, de 50 GHz), por lo que todos los canales ópticos han de emplear la misma rejilla. Con el concepto de rejilla variable (*flexi-grid*), cada canal puede ocupar un ancho de espectro diferente (con una granularidad de 12,5 GHz), lo que permite emplear formatos de modulación con mejor eficiencia espectral. Una evolución natural de la tecnología OBS consiste en emplearse en una red de rejilla flexible, por lo que se pueden emplear dos variables, tiempo y espectro, para poder aumentar la eficiencia de la red. De esta forma, OBS y *flexi-grid* se convierten en tecnologías complementarias y aparece una nueva línea de investigación dedicada a la arquitectura, diseño y operación de *flexi-grid* con OBS.

TCP sobre OBS

Esta tesis ha centrado el estudio de TCP sobre OBS en las versiones actuales de TCP, con los parámetros y opciones esperados para los próximos años. Sin embargo, el protocolo TCP continúa evolucionando constantemente para adaptarse a las tecnologías dominantes, y en ocasiones se incluyen nuevos algoritmos y mejoras. Por tanto, es necesario continuar estudiando la influencia de TCP sobre OBS, y futuras tecnologías de red, con los protocolos de transporte en versión experimental, que aun no siendo el estándar, son el embrión de futuros estándares. De esta forma, las nuevas tecnologías de red estarán diseñadas de acorde al tráfico que circulará por ellas.

El modelo periódico de TCP sobre OBS desarrollado en esta tesis detectó un aspecto clave en el rendimiento de la tasa de transmisión de una conexión TCP, el proceso de formación de ráfagas a partir del ensamblado de los segmentos TCP y cómo los segmentos de los flujos TCP se reparten entre las distintas ráfagas. El proceso es complejo, y muy dependiente de escenario concreto. Como se ha visto, dependiendo del RTT de cada flujo, del resto de tramos de red que compartan, el número de segmentos por ráfaga varía enormemente. Esta tesis ha proporcionado unas cotas analíticas y unos resultados de simulación. Por tanto, una línea futura de investigación es continuar avanzando en el modelado del número de segmentos por ráfaga para distintos condiciones, y encontrar cotas más próximas a los valores obtenidos en la simulación, así como encontrar una relación en entre el número medio de segmentos por ráfaga y el resto de parámetros.

Esta tesis ha estudiado TCP con un cierto número de flujos de larga duración, así como con tráfico de fondo. En cuanto a la sincronización de TCP, sería necesario investigar en profundidad escenarios con un mayor número de flujos, y detectar cual es la influencia de aumentar este número de flujos, y la influencia de las técnicas de ensamblado.

La metodología del estudio de TCP sobre OBS llevada a cabo en esta tesis es aplicable, no solamente a OBS, sino a cualquier tecnología de red. Por lo tanto, líneas futuras de investigación se centrarán en el impacto de TCP sobre nuevas tecnologías. Un ejemplo interesante es estudiar cómo afecta a TCP una aplicación dinámica de *flex-igrid* en la que el ancho de banda del canal se ajuste de manera dinámica.

PCE y WR-OBS

Se ha diseñado y desarrollado un primer prototipo de PCE para WR-OBS. Un aspecto importante en una red la posibilidad de ofrecer distintas calidades de servicio. Un aspecto clave dentro de las redes WR-OBS son las garantías de retardo y calidad de servicio. El prototipo de PCE cuenta con un mecanismo básico de reintentos, con un tiempo máximo de permanencia en el PCE fijo por petición de camino. Una línea de investigación es el

diseño y evaluación de estrategias de reintento que tengan en cuenta el tiempo que lleva cada petición en la red.

Otra línea futura de trabajo es la extensión del entorno emulado WR-OBS, para que cada nodo pueda tener un retardo de transmisión diferente.

En la tesis se han empleado dos algoritmos, el camino más corto (*Dijkstra*) y AUR-E. Próximos pasos consisten en tener en cuenta distintos algoritmos, evaluando el compromiso entre rapidez (menor retardo de borde) y probabilidad de bloqueo.

En la tesis se ha evaluado el PCE para WR-OBS en una red de tamaño nacional. En los próximos pasos, se va a evaluar el PCE para WR-OBS en entornos metropolitanos, ideales para WR-OBS, con distancias reducidas entre nodos.

En la tesis se ha estudiado una arquitectura en la que el PCE está centralizado. Otra línea de investigación es considerar estrategias con arquitecturas distribuidas, o parcialmente distribuidas. Así, se tiene un posible escenario con varios PCEs cooperantes (situados en media a menor distancia que un único PCE), en los que, por ejemplo, cada PCE sería responsable de un conjunto de longitudes de onda. En este escenario, el precio por reducir la latencia al PCE, sería previsiblemente un bloqueo ligeramente mayor.

La tesis se ha centrado en emular el comportamiento de PCE para WR-OBS con las técnicas FBAT y LBS. Por tanto, una clara línea de investigación es estudiar un escenario de PCE para WR-OBS con UBS.

Por último, otra línea de investigación es estudiar la escalabilidad de la solución de PCE para WR-OBS, incluyendo medidas de tráfico y un estudio analítico en el que a partir del protocolo PCEP actual, se conozcan los límites de rendimiento.

10.3 Listado de publicaciones

Los estudios llevados a cabo en esta tesis han dado lugar a un conjunto de publicaciones en distintos congresos y revistas. A continuación se muestra un listado de las publicaciones relacionadas con la tesis hasta el momento de redacción de la memoria:

[273] **O. González**, I. de Miguel, N. Merayo, P. Fernández, R. M. Lorenzo y E. J. Abril, "The impact of delayed. ACK in TCP flows in OBS networks", en *Proceedings of 10th European Conference on Network and Optical Communications (NOC 2005)*, Londres, pp. 367-374, julio 2005.

[274] **O. González de Dios**, I. de Miguel, V. López Alvarez, R.J. Durán, N. Merayo y J.F. Lobo Poyo, "Estudio y simulación de TCP en redes de conmutación óptica de ráfagas (OBS)", en XV Jornadas Telecom I+ D, Madrid, noviembre 2005.

[242] J. Aracil, **O. Gonzalez de Dios** y J.P. Fernandez-Palacios, "Routing strategies for OBS networks based on MRDV", en *Proceedings of ICTON 2005*, Barcelona, pp. 5-8, volumen 1, 2005.

[275] **O.Gonzalez**, C.M. Gauger, D. Sass, M.L. Garcia-Osma, A.J. Elizondo-Armengol, B. Puype, I. Lievens, P. Demeester, G. Rétvári, P. Fodor, J. Tapolcai, M. Düser, "Traffic Modelling and Traffic Engineering for Next Generation Transport Networks - results from the NOBEL project", en *Proceedings of 10th European Conference on Network and Optical Communications (NOC 2005)*, Londres, julio 2005

[276] J. F. Lobo Poyo, J.P. Fernández-Palacios Giménez, A. Ferreiro Olivo, **O. González de Dios** y J. Aracil, "Escenarios de Evolución de la Red Metropolitana", en *XV Jornadas Telecom I+ D*, Madrid, Noviembre 2005.

[277] **O. González de Dios**, I. de Miguel y V. López, "Performance evaluation of TCP over OBS considering background traffic", en *Proceedings of 10th Conference on Optical Network Design and Modelling*, Copenhague, mayo 2006.

[278] J. Aracil, N. Akar, S. Bjornstad, M. Casoni, K. Christodoulopoulos, D. Careglio,J. Fdez-Palacios, C. Gauger, **O. Gonzalez de Dios**, G. Hu, E. Karasan, M. Klinkowski, D. Morato, R. Nejabati, H. Overby, C. Raffaelli, D. Simeonidou, N. Stol, G. Tosi-Beleffi, K. Vlachos "Optical Burst. Switching - A Tutorial from e-Photon/One" en Proceedings of IEEE International Conference on Communications (ICC 2006), Istambul, 2006.

[279] J.A. Hernández, J. Aracil, V. López, J. Fernández Palacios y **O González de Dios**, "A resilience-based comparative study between Optical Burst Switching and Optical Circuit Switching technologies," en *Proceedings of 8th IEEE International Conference on Transparent Optical Networks (ICTON)*, Nottingham, junio 2006.

[280] F. Callegati, J. Aracil, L. Wosinska, N. Andriolli, D. Careglio, A. Giorgetti, J. Fdez-Palacios, C. Gauger, M. Klinkowski, **O. Gonzalez de Dios**, G. Hu, E. Karasan, F. Matera, H. Overby, C. Raffaelli, L. Rea, N. Sengezer, M. Tornatore, K. Vlachos, "Research on Optical Core Networks in the e-Photon/ONe Network of Excellence", en *25th Conference on Computer Communications (INFOCOM)*, Barcelona, abril 2006.

[281] J.M. Finochietto, J. Aracil, A. Ferreiro, J.P. Fernández-Palacios Giménez y **O. González de Dios**, "Migration Strategies Toward All Optical Metropolitan Access Rings", *Journal of Lightwave Technology*, vol. 25, no. 8, pp. 1918-1930, agosto 2007.

[92] **O. González de Dios**, M. Klinkowski, C. García Argos, D. Careglio y J. Solé-Pareta, "Performance analysis of routing algorithms for optical burst switching", en *Proceedings of 11th IFIP Working Conference on Optical Network Design and Modelling (ONDM)* , Atenas, pp. 211-220, mayo 2007.

[282] C. García Argos, **O. González De Dios** y J. Aracil, "Adaptive Multi-Path Routing for OBS Networks", en *9th International Conference on Transparent Optical Networks ICTON '07*, pp. 299-302, Roma, Julio 2007.

[70] K. Ramantas, K. Vlachos, **O. González De Dios** y C. Raffaelli, "Window-based burst assembly scheme for TCP traffic over OBS", Journal of Optical Networking, vol. 7, no. 5, pp. 487-495 , 2008.

[49] **O. González De Dios**, A.M. Guidotti, C. Raffaelli, K. Ramantas, and K. Vlachos, "On transmission control protocol synchronization in optical burst switching," *Photonic Network Communications*, vol. 18, 2009, pp. 323-333.

[283] S. Gunreben y **O. González De Dios**, "Why deterministic traffic shows the highest reordering ratio", en *Workshop on Optical Burst Switching*, Madrid, Septiembre 2009.

[234] J. Aracil, J. A. Hernández, A. J. Elizondo, R. Duque, y **O. Gonzalez de Dios**, "On Local CAC Schemes for Scalability of High-speed Networks", *Journal of Networks*, vol. 5, no. 2, pp. 225-229, febrero 2010.

[284] **O. Gonzalez de Dios**, F.J. Jimenez Chico y F. Muñoz del Nuevo, "Benefits of limited context awareness in stateless PCE", en *Proceedings of OFC/NFOEC 2011*, Los Angeles, CA, paper OThI4, marzo 2011.

[285] F. Paolucci, **O. González de Dios**, R. Casellas, S. Duhovnikov, P. Castoldi, R. Muñoz y R. Martínez, "Experimenting Hierarchical PCE Architecture in a Distributed Multi-Platform Control Plane Testbed", en *Proceedings of Optical Fiber Communication Conference (OFC) 2012*, Los Angeles, paper OM3G.3, marzo 2012..

[286] **O. González de Dios**, I. de Miguel, R. J. Durán, J. C. Aguado, N. Merayo, P. Fernández, "Impact of TCP Synchronization on Capacity Dimensioning of Optical Burst Switched (OBS) Links", en *Proceedings of NOC 2012*, Vilanova, Junio 2012.

10.4 Contribuciones a estándar

Parte de los resultados de las investigaciones llevadas a cabo en esta tesis, además de divulgarse en revistas y congresos, se están transfiriendo a la industria a través de contribuciones a los organismos de estandarización. De esta forma, se puede completar el ciclo de una investigación y obtener un mayor impacto. Son dos los principales organismos de estandarización donde se pueden aplicar los resultados de esta tesis, el IETF (*Internet Engineering Task Force*) [287], para todos los temas relacionados con el control de las redes, principalmente a través de los grupos de trabajo CCAMP (*Common Control and Management Protocol*) y PCE (*Path Computation Element*), y la Unión Internacional de Telecomunicaciones (ITU-T) [22], para los temas de plano de datos y arquitectura de redes de transporte ópticas. A continuación se enumeran los aspectos trabajados en esta tesis que se han empezado a transferir a los estándares.

En la tesis se han realizado extensiones del protocolo PCEP para la arquitectura WR-OBS. Para ello, han sido necesarias extensiones para el soporte de GMPLS, RWA y reserva de recursos. Estas extensiones se han llevado al organismo de estandarización IETF al grupo de trabajo PCE. En concreto, las contibuciones realizadas con las extensiones a PCEP son:

[260] **O. Gonzalez de Dios** (Ed.), R. Casellas, C. Margaria, Y. Lee y F. Zhang "PCEP Extensions for Temporary Reservation of Computed Path Resources and Support for Limited Context State in PCE, http://tools.ietf.org/html/draft-gonzalezdedios-pce-reservation-state-01, marzo 2012

[288] C. Margaria (Ed), **O. Gonzalez de Dios** (Ed.), F. Zhang (Ed). "PCEP extensions for GMPLS", draft-ietf-pce-gmpls-pcep-extensions-05. Este documento se presentó como draft individual en varias ocasiones (la primera de ellas en Marzo de 2010), y fue aceptado por el IETF y se convirtió en documento de trabajo. Se espera que sea RFC a finales de 2012. Versión actual, marzo 2012.

Se ha realizado una contribución para el framework de redes OBS en el contexto de GMPLS:

[25] **O. Gonzalez de Dios** (Ed.), G. Bernini, G. Zervas, M. Basham. "Framework for GMPLS and path computation support of sub-wavelength switching optical networks", draft-gonzalezdedios-subwavelength-framework-00, marzo de 2011.

Asimismo, a partir de las investigaciones sobre OBS llevadas a cabo en la tesis, se ha participado en la definición de OBS en la Unión Internacional de Telecomunicaciones (UIT-T), en el grupo de trabajo SG-15, encargado de las redes de transporte.

Capítulo 11

Listado de Acrónimos

AAP: *Adaptive Assembly Period*
ABL: *Average Burst Loss*
ABT-IT: *ATM block transfer with Immediate Transmission*
ABO: *Average Buffer Occupancy*
ACK: *Acknowledgement*
AIMD: *Additive Increase Multiplicative Decrease*
ALU: *Average Link Utilization*
AMOR: *Adaptive Multipath OBS Routing*
ARQ: *Automatic Retransmission Request*
AUR-E: *Adaptive Unconstrained Routing Exhaustive*
BCP: *Burst Control Packet*
BIC: *Binary Increase Congestion*
BP: *Bypass Path*
CLDR: *Contention-based Limited Deflection Routing*
CS: *Channel Spacing*
CWDM: *Coarse Wavelength Division Multiplexing*
DFL: *Delayed First Loss*
DRWA: *Dynamic Routing and Wavelength Assignment*
DWDM: *Dense Wavelength Division Multiplexing*
DT-BA: *Dynamic Time based Burst Assembly*
DSL: *Digital Subscriber Line*
DSLAM: *Digital Subscriber Line Access Multiplexer*
DVD: *Digital Versatile Disc*
ECMP: *Equal Cost Multi-Path*
E-JIT: *Enhanced Just In Time*
E-OBS: *offset-time Emulated OBS*
ERD: *Expected Route Distance*
ERO: *Explicit Route Object*
FBAT: *Fixed Burst Assembly Timer*
FB: *Feed-Back*
FDL: *Fiber Delay Line*
FEC: *Forward Error Correction*
FF: *Feed-Forward*

FTP: *File Transfer Protocol*
GMPLS: *Generalized Multi-Protocol Label Switching*
HSTCP: *High Speed TCP*
HTTP: *HyperText Transfer Protocol*
IBG: *In-Burst Generation*
IETF: *Internet Engineering Task Force*
IGP: *Interior Gateway Protocol*
IS-IS: *Intermediate System-Intermediate System*
ITU: *International Telecommunication Union*
ITU-T: *ITU's Telecommunication Standardization Sector*
IP: *Internet Protocol*
JET: *Just Enough Time*
JIT: *Just In Time*
k-SPFF: *k Shortest Path First Fit*
LAUC: *Latest Available Unscheduled Channel*
LAUC-VF: *Latest Available Unscheduled Channel with Void Filling*
LC: *Liquid Crystal*
LCOS: *Liquid Crystal On Silicon*
LBS: *Limited Burst Size*
LER: *Label Edge Router*
LP: *Linear Programming*
LSP: *Label Switched Path*
MAN: *Metropolitan Area Network*
MCL: *Maximum Congested Link*
MEC: *Maximum End to end Congested path*
MEMS: *Micro-Electro-Mechanical System*
MFL: *Multi-Frequency Laser*
MILP: *Mixed Integer Linear Programming*
Min-EV: *Minimum Ending Void*
Min-SV: *Minimum Starting Void*
MPLS: *Multi-Protocol Label Switching*
MF: *Mixed Flow*
MQ: *Multi Queue*
MRDV: *Multi-path Routing with Dynamic Variance*
MSS: *Maximum Segment Size*
NACK: *Negative Acknowledgement*
NCR: *Non Congestion Robustness*
OBS: *Optical Burst Switching*
OBTN: *Optical Burst Transport Network*
OBG: *Out of Burst Generation*
OCS: *Optical Circuit Switching*
OLT: *Optical Line Terminal*

OPS: *Optical Packet Switching*
OPST: *Optical Packet Switch & Transport*
OSPF: *Open Shortest Path First*
OSPF-TE: *Open Shortest Path First Traffic Engineering*
PBS: *Polarization Beam Splitter*
PCC: *Path Computation Client*
PCE: *Path Computation Element*
PCEP: *Path Computation Element Protocol*
PF: *Per Flow*
PLZT: *Lead Lanthanum Zirconate Titanate*
PR: *Packet Reordering*
RAM: *Random-Access Memory*
RED: *Random Early Detection*
RFC: *Request For Comments*
RIP: *Routing Information Protocol*
RLDRS: *Reinforcement Learning-based Deflection Routing Scheme*
ROADM: *Reconfigurable Optical Add-Drop Multiplexer*
RSVP-TE: *Resource Reservation Protocol-Traffic Engineering*
RP: *Request Parameters*
RTT: *Round Trip Time*
SACK: *Selective Acknowledgement*
SAN: *Storage Area Network*
SOA: *Semiconductor Optical Amplifier*
SOA-MZI: *SOA-Match Zender Interferometer*
SP: *Shortest Path*
SP-DR: *Shortest Path Deflection Routing*
SP-FF: *Shortest Path First Fit*
SP-PRDR: *Shortest Path Prioritized Random Deflection Routing*
STCP: *Scalable TCP*
SCORE: *Store-and-forward COtention-Resolution*
TAG: *Tell and Go*
TAW: *Tell and Wait*
TCP: *Transmission Control Protocol*
UBS: *Unlimited Burst Size*
VoD: *Video on Demand*
WR-OBS: *Wavelength Routed Optical Burst Switching*
WS: *Wavelength Switching*
WSON: *Wavelength Switched Optical Network*

Capítulo 12

Bibliografía

[1] C. Alierta, "Sustaining growth in challenging times", en *Mobile World Congress*, 2009, http://info.telefonica.es/mwc/pdf/PR_3GSM_v20.pdf.

[2] Cisco Systems, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010–2015", febrero 2011.

[3] Cisco Systems, "Cisco Visual Networking Index: Forecast and Methodology, 2008-2013", junio 2009.

[4] B. Mukherjee, *Optical WDM Networks* New York, Springer, 2006.

[5] P. Sterling, "The need for Next Generation ROADM Networks", Heavy Reading, septiembre 2010.

[6] C. Qiao y M. Yoo, "Optical burst switching (OBS)–a new paradigm for an Optical Internet", *Journal of High Speed Networks*, vol. 8, no. 1, pp. 69–84, marzo 1999.

[7] D.J. Blumenthal y M. Usami, "Tutorial on Optical Signal Processing: The roadmap towards high-speed optical Packet/Burst Switching", en *Proceedings of European Conference on Optical Communications (ECOC)*, Viena, Septiembre 2009.

[8] R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1994.

[9] Internet Engineering Task Force. [Online]. www.ietf.org

[10] A. Farrel y J. P. Vasseur, "A Path Computation Element (PCE)-Based Architecture", *RFC 4655*, agosto 2006.

[11] J.P. Vasseur y J.L Le Roux, "Path Computation Element (PCE) Communication Protocol", *RFC 5440*, marzo 2009.

[12] OPNET Modeler. [Online]. http://www.opnet.com/

[13] OMNeT++ Network Simulation Framework. [Online]. http://www.omnetpp.org/

[14] M. Salsi, C. Koebele, P. Tran, H. Mardoyan, S. Bigo y G. Charlet, "80x100-Gbit/s transmission over 9,000km using erbium-doped fibre repeaters only", en *Proceedings of 36th European Conference and Exhibition on Optical Communication*, Turin, septiembre 2010.

[15] J. Yu, Z. Dong y N. Chi, "Generation, Transmission and Coherent Detection of 11.2 Tb/s (112x100Gb/s) Single Source Optical OFDM Superchannel", en *Proceedings of Optical Fiber Communications (OFC)*, Los Angeles, postdeadline paper A6, marzo 2011.

[16] R. Ramaswami y K. Sivarajan, *Optical Networks: A Practical Perspective*, 3rd ed., Morgan Kaufmann, julio 2009.

[17] M. Pickavet, R. Van Caenegem, S. Demeyer, P. Audenaert y D. Colle, "Energy

footprint of ICT", en *Proceedings of , Broadband Europe* , Antwerpen, diciembre 2007.

[18] G. Epps, D. Tsiang y T. Boures, "System Power Challenges", en *Cisco Routing Research Seminar*, agosto 2006.

[19] E. Bonetto, L. Chiaraviglio, D. Cuda, G. A. Gavilanes y F. Neri, "Optical Technologies Can Improve the Energy Efficiency of Networks ", en *Proceedings of 35th European Conference on Optical Communication*, Viena, Septiembre 2009.

[20] P. Roorda y B. Collings, "Evolution to Colorless and Directionless ROADM Architectures", en *Proceedings of Optical Fiber Communication Conference (OFC)*, San Diego, paper NWE2, febrero 2008.

[21] S. Perrin, "The need for next-generation ROADM Network", HeavyReading , junio 2010.

[22] International Telecommunication Union (ITU). [Online]. http://www.itu.int

[23] ITU, "Spectral grids for WDM applications: DWDM frequency grid", *G.694.1*, junio 2002.

[24] J. S. Turner, "Terabit burst switching", *Journal of High Speed Networks*, vol. 8, no. 1, pp. 3-16, marzo 1999.

[25] O. Gonzalez de Dios, G. Bernini, G. Zervas y M. Basham, "Framework for GMPLS and path computation support of sub-wavelength switching optical networks", *draft-gonzalezdedios-subwavelength-framework-00*, marzo 2011.

[26] Xtera. (2011) Sub Wavelength Switching Solution for Metropolitan Networks. [Online]. http://www.xtera.com/content/solutions/metropolitan-networks/sub-wavelength-switching-solution

[27] S. Yao, B. Mukherjee y S. Dixit, "Advances toward Optical Subwavelength Switching", en *IP over WDM: Building the Next-Generation Optical Internet*, S. Dixit, Ed. Hoboken, NJ, USA, John Wiley & Sons, cap. 6, enero 2004.

[28] ITU-T, "Generic Functional Architecture of Transport Networks", *ITU-T Recommendation G.805*, noviembre 2005.

[29] J.E. Simsarian, P. Bernasconi, J. Gripp, M. C. Larson, D.T. Neilson y J. Sinsky, "Fast tunable lasers for optical routers and networks", en *Proceedings of Conference on Lasers and Electro-Optics and Quantum Electronics and Laser Science Conference (CLEO/QELS)*, Long Beach, mayo 2006.

[30] G. S. Zervas *et al.*, "Multi-Granular Optical Cross-Connect: Design, Analysis, and Demonstration", *Journal of Optical Communications and Networking*, vol. 1, no. 1, pp. 69-84, 2009 junio.

[31] J. Gripp, J.E. Simsarian, J.D. LeGrange, P.G. Bernasconi y D.T. Neilson, "Architectures, Components, and Subsystems for Future Optical Packet Switches", *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 16, no. 5, pp. 1394 - 1404, septiembre-octubre 2010.

[32] D. Van Thourhout, P. Bernasconi, B. Miller, W. Yang, L. Zhang, N. Sauer y L. Stulz, "High-power digitally tunable laser with integrated star coupler", *IET Electronic Letters*, vol. 39, no. 4, pp. 370–371, febrero 2003.

[33] E. Simsarian, A. Bhardwaj, J. Gripp, K. Sherman, Yikai Su, C. Webb, Liming Zhang y M. Zirngibl, "Fast switching characteristics of a widely tunable laser transmitter", *IEEE Photonic Technology Letters*, vol. 15, no. 8, pp. 1038–1040, agosto 2003.

[34] J.E. Simsarian, M.C. Larson, H.E. Garrett, Hong Xu y T.A. Strand, "Less than 5-ns wavelength switching with an SG-DBR laser", *IEEE Photonic Technology Letters*, vol. 18, no. 4, pp. 565–567, febrero 2006.

[35] K. Vlachos, "Optical Switch Fabrics and their application", en *Enabling Optical Internet with Advanced Network Technolgies*, J. Aracil y F. Callegati, Eds., Springer, cap. 6, 2009.

[36] P. Perrier, "Optical cross connects - Part 2: enabling technologies", *Fiber optics online (http://www.fiberopticsonline.com)*, septiembre 2000.

[37] G. Papadimitriou, C. Papazoglou y A. Pomportsis, "Optical Switching: Switch Fabrics, Techniques and Architectures", *IEEE/OSA Journal of Lightwave Technology*, vol. 21, no. 2, pp. 384–390, febrero 2003.

[38] A. Neukermans y R. Ramaswami, "MEMS Technology for Optical Networking Applications", *IEEE Communications Magazine*, vol. 10, no. 1, pp. 62–69, 2001.

[39] JDSU - JDS Uniphase corporation - Optical Products. [Online]. http://www.jdsu.com

[40] A. Yariv, *Optical Electronics in Modern Communications*, 5th ed., Oxford University Press.

[41] S. Y. Kim, S. B. Lee, J. Chung, S. Y. Kim, I. J. Park, J. Jeong y S. S. Choi, "Highly stable optical add/drop multiplexer using polarization beam splitters and fiber Bragg gratings", *IEEE Photonics Technology Letters*, vol. 9, no. 8, pp. 1119-1121, agosto 1997.

[42] Finisar. [Online]. http://www.finisar.com/

[43] I. Baldine, M. Cassada, A. Bragg y G. Karmous-Edwards, "Just-in-time optical burst switching implementation in the ATDnet all-optical networking testbed", en *Proceedings of IEEE GLOBECOM*, vol. 5, pp. 2777–2781, diciembre 2003.

[44] A. Sahara, R. Kasahara, E. Yamazaki, S. Aisawa y M. Koga, "The demonstration of congestion-controlled optical burst switching network utilizing two-way signaling - field trial in JGN II testbed", en *Proceedings of Optical Fiber Communication Conference (OFC)*, Anaheim, paper OFA7, marzo 2005.

[45] K. Kitayama, M. Koga, H. Morikawa, S. Hara y M. Kawai, "Optical burst switching network testbed in Japan", en *Proceedings of Optical Fiber Communication Conference (OFC)*, Anaheim, paper OWC3, marzo 2005.

[46] Y. Sun, T. Hashiguchi, V. Q. Minh, X. Wang, H. Morikawa y T. Aoyama, "Design and implementation of an optical burst-switched network testbed", *IEEE Communications Magazine*, vol. 43, no. 11, pp. 48-55, noviembre 2005.

[47] A. Al Amin *et al.*, "Development of an Optical-Burst Switching Node Testbed and Demonstration of Multibit Rate Optical Burst Forwarding", *IEEE/OSA Journal of Lightwave Technology*, vol. 27, no. 16, pp. 3466-3475, agosto 2009.

[48] E. Rosen, A. Viswanathan y R. Callon, "Multiprotocol Label Switching Architecture", *RFC 3031*, enero 2001.

[49] O. González De Dios, A.M. Guidotti, C. Raffaelli, K. Ramantas y K. Vlachos, "On transmission control protocol synchronization in optical burst switching", *Photonic Network Communications*, vol. 18, no. 3, pp. 323-333, diciembre 2009.

[50] A. Detti y M. Listanti, "Impact of Segments Aggregation on TCP Reno Flows in Optical Burst Switching Networks", en *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 3, New York, pp. 1803-1812, 2002.

[51] X. Yu, C. Qiao, Y. Liu y D. Towsley, "Performance Evaluation of TCP Implementations in OBS Network", CSE Dept., SUNY, Buffalo, Technical Report 2003-13, 2003.

[52] X. Yu, C. Qiao y Y. Liu, "TCP Implementations and False Time Out Detection in OBS Networks", en *Proceedings of the 23rd Conference of the IEEE Communications Society (INFOCOM)*, Hong Kong, pp. 774-784, marzo 2004.

[53] M. Düser y P. Bayvel, "Modelling of Optical Burst-Switched Packet Networks", en *Proceedings of European Conference on Optical Communications (ECOC)*, Munich, paper 1.3, septiembre 2000.

[54] L. Berger, "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", *RFC 3473*, enero 2003.

[55] Y. Chen, C. Qiao y X. Yu, "Optical Burst Switching: A new area in Optical Networking Research", *IEEE Network*, vol. 18, no. 3, pp. 16 - 23 , junio 2004.

[56] M. Klinkowski, D. Careglio, J. Solé-Pareta y M. Marciniak, "Performance Overview of the Offset Time Emulated OBS Network Architecture", *Journal of Lightwave Technology*, vol. 27, no. 4, pp. 2751 - 2764, julio 2009.

[57] A. Ge, F. Callegati y L. S. Tamil, "On Optical Burst Switching and Self-Similar Traffic", *IEEE Communications letters*, vol. 4, no. 3, pp. 98-100, marzo 2000.

[58] E. Kozlovski, M. Düser, I. de Miguel y P. Bayvel, "Analysis of Burst Scheduling for Dynamic Wavelength Assignment in Optical Burst-Switched networks", en *Proceedings of the 14th Annual Meeting of the IEEE Lasers and Electro-Optics Society (LEOS)*, vol. 1, San Diego, pp. 161 - 162, 2001.

[59] X., Li, J. Cao y Y., Qiao, C. Chen, "Assembling TCP/IP Packets in Optical Burst Switched Networks", en *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, vol. 3, Taipei, pp. 2808 - 2812, noviembre 2002.

[60] M. de Vega Rodrigo y J. Gotz, "An analytical study of optical burst switching aggregation strategies", en *Proceedings of the 3rd international Workshop on OBS (WOBS)*, San Jose, 2004.

[61] I. de Miguel, M. Düser y P. Bayvel, "Traffic Load Bounds for Optical Burst-Switched Networks with Dynamic Wavelength Allocation", en *Proceedings of the IFIP TC6 Fifth Working Conference on Optical Network Design and Modeling (ONDM)*, Viena, pp. 209 - 226, febrero 2001.

[62] A. Zapata y P. Bayvel, "Impact of burst aggregation schemes on delay in optical burst switched networks", en *Proceedings of the 16th Annual Meeting of the IEEE Lasers and Electro-Optics Society (LEOS)*, vol. 1, Tucson, pp. 57-58, octubre 2003.

[63] V.M. Vokkarane, K. Haridoss y J. P. Jue, "Threshold-Based Burst Assembly Policies for QoS Support in Optical Burst-Switched Networks", en *Proceedings od SPIE Optical Networking and Communication Conference (OptiComm)*, vol. 4874, Boston, pp. 125-136, 2002.

[64] G. Hu, K. Dolzer y C. Gauger, "Does burst assembly really reduce the self-similarity?", en *Proceedings of Optical Fiber Communications Conference (OFC)*, vol. 1, Atlanta, pp. 124 - 126, marzo 2003.

[65] S. Oh y Kang. M., "A Burst Assembly Algorithm in Optical Burst Switching Networks", en *Proceedings of Optical Fiber Communication Conference and Exhibit (OFC)*,

Anaheim, pp. 771–773, marzo 2002.

[66] X. Yu, Y. Chen y C. Qiao, "Study of Traffic Statistics of Assembled Burst Traffic in Optical Burst Switched Networks", en *Proceedings of Optical Networking and Communications Conference - OptiComm*, Boston, pp. 149-159, julio 2002.

[67] D. Kang, W.H. Yang, J.Y. Lee y Y.C. Kim, "Dynamic Time based Burst Assemble scheme to reduce energy consumption in OBS network with Low-Power Idle", en *Proceedings of the 16th Asia-Pacific Conference on Communications (APOC)*, Auckland, pp. 183-187, noviembre 2010.

[68] R. Nakkeeran, C. Balaji, N. Gujju, Balakumaran K. y M.K. Sasidharan, "Enhanced Burst Assembly Mechanism with Hybrid Signalling Scheme for Optical Burst Switched Networks", en *Proceedings of the 15th International Conference on Advanced Computing and Communication (ADCOM)*, Guwahati, pp. 727-732, diciembre 2007.

[69] M. Düser, I. de Miguel y P. Bayvel, "Timescale analysis for wavelength-routed optical burst-switched (WR-OBS) networks", en *Proceedings of Optical Fiber Communication Conference (OFC)*, Anaheim, paper WG7, marzo 2002.

[70] K. Ramantas, K. Vlachos, O. González De Dios y C. Raffaelli, "Window-based burst assembly scheme for TCP traffic over OBS", *Journal of Optical Networking*, vol. 7, no. 5, pp. 487-495 , 2008.

[71] R. Rajaduray, D. J. Blumenthal y S. Ovadia, "Impact of Burst Assembly Parameters on Edge Router Latency in an Optical Burst Switching Network", en *Proceedings of 16th Annual Meeting of the IEEE Lasers and Electro-Optics Society (LEOS)*, vol. 1, Tucson, pp. 55-56, octubre 2003.

[72] A. Zalesky, E. W. M. Wong, M. Zukerman, H. L. Vu y R. S. Tucker, "Performance Analysis of an OBS Edge Router", *IEEE Photonics Technology Letters*, vol. 16, no. 2, pp. 695–697, febrero 2004.

[73] T. Hashiguchi, X. Wang, H. Morikawa y T. Aoyama, "Burst Assembly Mechanism with Delay Reduction for OBS Networks ", en *Proceedings of Conference on the Optical Internet (COIN)*, Melbourne, pp. 664-666, julio 2003.

[74] E. Mannie, "Generalized Multi-Protocol Label Switching (GMPLS) Architecture", *RFC 3945*, octubre 2004.

[75] A. Farrel y I. Bryskin, *GMPLS: Architecture and Applications*, Elsevier, 2006.

[76] A. Zapata, M. Düser, J. Spencer, P. Bayvel, I. de Miguel, D. Breuer, N. Hanik y A. Gladisch, "Next-Generation 100-Gigabit Metro Ethernet (100 GbME) Using Multiwavelength Optical Rings", *Journal of Lightwave Technology*, vol. 22, no. 11, pp. 2420 - 2434, noviembre 2004.

[77] E. Varvarigos y V. Sharma, "The Ready-To-Go Virtual Circuit Protocol: A loss-Free Protocol for Multigigabit Networks Using FIFO Buffers", *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 705–718, octubre 1997.

[78] I. Baldine, G. N. Rouskas, H. G. Perros y D. Stevenson, "JumpStart: A Just-In-Time Signaling Architecture for WDM Burst-Switched Networks", *IEEE Communications Magazine*, vol. 40, no. 2, pp. 82–89, febrero 2002.

[79] M. Yoo, C. Qiao y S. Dixit, "QoS Performance of Optical Burst Switching in IP-Over-WDM Networks", *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 2062-2071, octubre 2000.

[80] J.J.P.C. Rodrigues y M. Freire, "Performance Assessment of Enhanced Just-in-Time Protocol in OBS Networks Taking into Account Control Packet Processing and Optical Switch Configuration Times", en *Proceedings of 22nd International Conference on Advanced Information Networking and Applications (AINA)*, Gino-wan City, pp. 434-439, marzo 2008.

[81] I. de Miguel, E. Kozlovski y P. Bayvel, "Provision of end-to-end delay guarantees in wavelength-routed optical burst-switched networks", en *Proceedings of IFIP 6th Working Conference on Optical Networks Design and Modelling (ONDM)*, vol. 1, Turin, pp. 85–100, febrero 2002.

[82] A. Zapata y P. Bayvel, "Dynamic Versus Static Wavelength-Routed Optical Networks", *IEEE/OSA Journal of Lightwave Technology*, vol. 26, no. 20, pp. 3403 - 3415, octubre 2008.

[83] D. Awduche, L. Berger, D. Gan, Li T., V. Srinivasan y G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", *RFC 3209*, diciembre 2001.

[84] J. S. Turner, "Managing Bandwidth in ATM Networks with Bursty Traffic", *IEEE Network*, vol. 6, no. 5, pp. 50–58, septiembre 1992.

[85] P. E. Boyer y D. P. Tranchier, "A Reservation Principle with Applications to the ATM Traffic Control", *Computer Networks and ISDN Systems*, vol. 24, no. 4, pp. 321–334, mayo 1992.

[86] J. Y. Wei y R. I. McFarland, "Just-In-Time Signaling for WDM Optical Burst Switching Networks", *IEEE/OSA Journal of Lightwave Technology*, vol. 18, no. 12, pp. 2019–2037, diciembre 2000.

[87] G. Malkin, "RIP Version 2", *RFC 2453*, noviembre 1998.

[88] J. Moy, "OSPF Version 2", *RFC 2328*, abril 1998.

[89] D. Oran, "OSI IS-IS Intra-domain Routing Protocol", *RFC 1142*, 1990 febrero.

[90] D. Katz, K. Kompella y D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", *RFC 3630*, septiembre 2003.

[91] F. Zhang, Y. Lee, J., Bernstein, G. Han y Xu Y., "OSPF Extensions in Support of Routing and Wavelength Assignment (RWA) in Wavelength Switched Optical Networks (WSONs)", *draft-zhang-ccamp-rwa-wson-routing-ospf-03* , marzo 2010.

[92] O. González de Dios, M. Klinkowski, C. García Argos, D. Careglio y J. Solé-Pareta, "Performance analysis of routing algorithms for optical burst switching", en *Proceedings of 11th IFIP Working Conference on Optical Network Design and Modelling (ONDM)* , Atenas, pp. 211-220, mayo 2007.

[93] M. Klinkowski, J. Pedro, D. Careglio, M. Pióro, J. Pires, P. Monteiro y J. Solé-Pareta, "An overview of routing methods in optical burst switching networks", *Optical Switching and Networking*, vol. 7, no. 2, pp. 41-53, abril 2010.

[94] E. Hyytia y L. Nieminen, "Linear program formulation for routing problem in OBS networks", en *Proceedings of the 9th IEEE Symposium on Computers and Communications (ISCC)*, Alexandria, pp. 252 - 257 , junio 2004.

[95] J. Teng y G.N. Rouskas, "Routing path optimization in optical burst switched networks", en *Proceedings of the 9th Conference on Optical Network Design and Modeling (ONDM)*, Milan, febrero 2005.

[96] A.L. Barradas y M.C.R. Medeiros, "Pre-Planned Optical Burst Switching Routing

Strategies", en *Proceedings of the 2nd ICTON Mediterranean Winter (ICTON-MW)*, Marrakesh, diciembre 2008.

[97] Y. Du, T. Pu, H. Zhang y Y. Guo, "Adaptive load balancing routing algorithm for optical burst-switching networks", en *Proceedings of Optical Fiber Communication Conference (OFC)*, Anaheim, paper OThF7, marzo 2006.

[98] M. Klinkowski, F. Herrero, D. Careglio y J. Sole-Pareta, "Adaptive routing algorithms for optical packet switching networks", en *Proceedings of Conference on Optical Network Design and Modeling (ONDM)*, Milan, pp. 235 - 241, febrero 2005.

[99] G.R.V. Thodime, V.M. Vokkarane y J.P. Jue, "Dynamic congestion-based load balanced routing in optical burst-switched networks", en *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, San Francisco, pp. 2628 - 2632, diciembre 2003.

[100] M. Klinkowski, D. Careglio y J. Sole-Pareta, "Reactive and proactive routing in labelled optical burst switching networks ", *IET Communications*, vol. 3, no. 3, pp. 454-464, marzo 2009.

[101] K. Kitayama, S. Arakawa, S. Matsuo, M. Murata, M. Notomi, R. Takahashi y Y. Itaya, "All-optical RAM-based buffer for packet switch ", en *Proceedings of Photonics in Switching*, San Francisco, pp. 5-6, agosto 2007.

[102] Y. Xiong, M. Vandenhoute y H. Cankaya, "Design and analysis of optical burst-switched networks", en *Proceedings of SPIE Conference on All Optical Networking: Architecture, Control, Management Issues*, vol. 3843, Boston, pp. 112–119, septiembre 1999.

[103] Y. Xiong, M. Vandenhoute y H. C. Cankaya, "Control architecture in optical burst-switched WDM networks", *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 10, pp. 1838-1851, octubre 2000.

[104] J. Xu, C. Qiao, J. Li y G. Xu, "Efficient channel scheduling algorithms in optical burst switched networks", en *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 3, San Francisco, pp. 2268 - 2278, marzo 2003.

[105] G.M. Li, Z.K. Sun y S.H. Geng, "Dimensioning fiber delay lines in optical burst switching networks", en *Proceedings of 5th International IEEE ICST Conference on Communications and Networking in China (CHINACOM)*, Beijing, pp. 1-5, 2010.

[106] Y. Lee, N. Kim, J. Ahn y M. Kang, "Fiber Delay Line Based Loss Differentiation in OBS Networks", en *Proceedings of the Joint International Conference on Optical Internet and the 32nd Australian Conference on Optical Fibre Technology (COIN-ACOFT)*, Melbourne, junio 2007.

[107] C.M. Gauger, "Dimensioning of FDL Buffers for Optical Burst Switching Nodes", en *Proceedings of the 5th IFIP Optical Network Design and Modeling (ONDM)*, Turin, febrero 2002.

[108] D. Yang, S. Xu y S. Wang, "Edge buffer based contention resolution scheme in optical burst switching networks", en *Proceedings of Asia-Pacific Conference Communications (APCC)*, Bangkok, pp. 461 - 464 , octubre 2007.

[109] V. Vokkarane, J. Jue y S. Sitaraman, "Burst segmentation: an approach for reducing packet loss in optical burst switched networks", en *Proceedings of IEEE International Conference on Communications (ICC)*, vol. 5, New York, pp. 2673-2677, mayo 2002.

[110] A. Abid, F. M. Abbou y H. T. Ewe, "Effective Implementation of Burst Segmentation Techniques in OBS Networks", *International Journal of Information Technology (IJIT)*, vol. 3, no. 4, pp. 231-236, octubre 2006.

[111] O. Pedrola, S. Rumley, D. Careglio, M. Klinkowski, P. Pedroso, J. Solé-Pareta y C. Gaumier, "A performance survey on deflection routing techniques for OBS networks", en *Proceedings of 11th IEEE International Conference on Transparent Optical Networks (ICTON)*, Island of São Miguel, pp. 147-152, junio 2009.

[112] A. Belbekkouche, A. Hafid y M. Gendreau, "A Reinforcement Learning-based Deflection Routing Scheme for Buffer-less OBS Networks", en *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, 2008.

[113] C. Cameron, A. Zalesky y M. Zukerman, "Prioritized Deflection Routing in Optical Burst Switching Networks", *IEICE Transactions on Communication* , vol. 88, no. 5, pp. 1861-1867, 2005.

[114] S. Lee, K. Sriram, H. Kim y J. Song, "Contention-Based Limited Deflection Routing Protocol in Optical Burst-Switched Networks", *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 8, pp. 1596-1611, agosto 2005.

[115] T. Coutelen, H. Elbiaze, B. Jaumard y A. Metnani, "Impact of Network Topology and Traffic Load Balancing on Deflection Routing Efficiency", en *Proceedings of International Conference on Optical Communication Systems and Networks (IASTED OCSN)*, Banff, 2005.

[116] R. Takahashi, T. Nakahara, K. Takahata, H. Takenouchi, T. Yasui, N. Kondo y H. Suzuki, "Photonic random access memory for 40-Gb/s 16-b burst optical packets", *IEEE Photonics Technology Letters*, vol. 16, no. 4, p. 1185, abril 2004.

[117] N. Pleros, D. Apostolopoulos, D. Petrantonakis, C. Stamatiadis y H. Avramopoulos, "All-optical static RAM cell with read/write functionality at 5 Gb/s", en *Proceedings of 34th European Conference on Optical Communication (ECOC)*, Bruselas, paper We.2.C.5, septiembre 2008.

[118] K. Kitayama, T. Kubo, R. Takahashi, S. Matsuo, S. Arakawa, M. Murata, M. Notomi, K. Nozaki y K. Kato, "All-optical RAM Buffer Subsystem Demonstrator", en *Proceedings of Optical Fiber Communication Conference (OFC)*, Los Angeles, paper OMK1, marzo 2011.

[119] Memory RAM Speed - Access Time, Megahertz (MHz), Bytes Per Second. [Online]. http://www.computermemoryupgrade.net/measuring-ram-speed.html

[120] M. Notomi, A. Shinya, S. Mitsugi, G. Kira, E. Kuramochi y T. Tanabe, "Optical bistable switching of Si high-Q photonic-crystal nanocavities", *Optical Express*, vol. 13, no. 7, pp. 2678-2687, abril 2005.

[121] K. Nozaki, A. Shinya, T. Tanabe, S. Matsuo, T. Sato, T. Kakitsuka, E. Kuramochi, H. Taniyama y M. Notomi, "Extremely low power nanophotonic devices based on photonic crystals", en *Proceedings of Photonics in Switching*, Monterey, paper PWE1, julio 2010.

[122] E.F. Burmeister, D.J. Blumenthal y J.E. Bowers, "A comparison of optical buffering technologies", *Optical Switching and Networking*, vol. 5, no. 1, pp. 10-18, marzo 2008.

[123] R. Tucker, P.C. Ku y C. Chang-Hasnain, "Slow-light optical buffers: capabilities and fundamental limitations", *IEEE/OSA Journal of Lightwave Technology*, vol. 23, no. 12, pp. 4046–4066, diciembre 2005.

[124] N. Beheshti, Y. Ganjali, R. Rajaduray, D. Blumenthal y N. McKeown, "Buffer sizing in all-optical packet switches", en *Proceedings of Optical Fiber Communication Conference (OFC)*, Anaheim, paper OThF8, marzo 2006.

[125] E. F. Burmeister, J. P. Mack, H. N. Poulsen, J. Klamkin, L. A. Coldren, D. J. Blumenthal y J. E. Bowers, "SOA gate array recirculating buffer with fiber delay loop", *Optics Express*, vol. 16, no. 12, pp. 8451-8456, mayo 2008.

[126] J. Bannister, F. Borgonovo, L. Fratta y M. Gerla, "A Performance Model of Deflection Routing in Multibuffer Networks with Nonuniform Traffic", *IEEE Transactions on Networking*, vol. 3, no. 5, pp. 509-520, octubre 1995.

[127] A. Bononi, F. Forghieri y P. R. Prucnal, "Analysis of One-Buffer Deflection Routing in Ultra-Fast Optical Mesh Networks", en *Proceedings of IEEE INFOCOM* , pp. 303-311, 1993.

[128] C.F. Hsu, T.L. Liu y N.F. Huang, "Performance Analysis of Deflection Routing in Optical Burst-Switched Networks", en *Proceedings of IEEE INFOCOM*, pp. 66-73, 2002.

[129] N. Ogino y H. Tanaka, "Deflection Routing for Optical Bursts Considering Possibility of Contention at Downstream Nodes", *IEICE Transactions on Communications*, vol. E88-B, no. 9, pp. 3660-3667, septiembre 2005.

[130] H.T. Lin, W. R. Chang y T.S. Lin, "On burst-header contention-resolution in OBS networks", *Photonic Network Communications*, vol. 16, no. 2, pp. 155-168, octubre 2008.

[131] Q. Zhang, V.M. Vokkarane, Y. Wang y J.P. Jue, "Evaluation of Burst Retransmission in Optical Burst-Switched Networks", en *Proceedings of 2nd International Conference on Broadband Networks (BroadNets)*, Boston, pp. 276 - 282, octubre 2005.

[132] S. Arima, T. Tachibana, Y. Kaji y S. Kasahara, "FEC-Based Reliable Transmission for Multiple Bursts in OBS Networks", *IEICE Transactions on Communications*, vol. 90, no. 12, diciembre 2007.

[133] X. Huang, V. M. Vokkarane y J. P. Jue, "Burst Cloning: A Proactive Scheme to Reduce Data Loss in Optical Burst-Switched Networks", en *Proceedings of IEEE International Conference on Communications (ICC)*, Seoul, pp. 1673 – 1677, mayo 2005.

[134] J. Sullivan, N. Charbonneau y V.M. Vokkarane, "Performance Evaluation of TCP over Optical Burst Switched (OBS) Networks using Coordinated Burst Cloning and Forward Segment Redundancy", en *Proceedings of IEEE Global Communications Conference (GLOBECOM)*, Miami, diciembre 2010.

[135] R. Braden, D. Clark y S. Shenker, "Integrated Services in the Internet Architecture: an Overview", *RFC 1633*, junio 1994.

[136] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang y W. Weiss, "An Architecture for Differentiated Services", *RFC 2475* , diciembre 1998.

[137] R. Braden, L. Zhang, S. Berson, S. Herzog y S. Jamin, "Resource ReSerVation Protocol (RSVP)", *RFC 2205*, septiembre 1997.

[138] V.M. Vokkarane, Q. Zhang, J.P. Jue y B. Chen, "Generalized burst assembly and scheduling techniques for QoS support in optical burst-switched networks", en *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, Taipei, p. 2747 , noviembre 2002.

[139] Y. Wang, D. Liu y D. Luo, "The Research on Assembly Algorithm Supporting QoS

in OBS Networks", en *Proceedings of 2nd International Workshop on Intelligent Systems and Applications (ISA)*, Wuhan , mayo 2010.

[140] S. Askar, G. Zervas, D. K. Hunter y D. Simeonidou, "Classified Cloning for QoS Provisioning in OBS Networks", en *European Conference on Optical Communications (ECOC)*, Turin, septiembre 2010.

[141] M.A. González Ortega, A. Suárez-González, J. C. López Ardao y C. López-García, "Loss Differentiation in Full Wavelength Conversion Capable OBS Networks by Burst Cloning", *IEEE Communications Letters*, vol. 15, no. 1, pp. 85-87, enero 2011.

[142] M. Thachayani y R. Nakkeeran, "Path differentiated QoS provisioning scheme for OBS networks", en *Proceedings of 2010 International Conference on Computing Communication and Networking Technologies (ICCCNT)*, TamilNadu, julio 2010.

[143] F. Sabir, Z. Ul-Haq y S. M. H. Zaidi, "An independent wavelength assignment, QoS Differentiation Scheme, in DWDM OBS Networks, for zero high priority burst loss", en *Proceedings of International Symposium on High capacity Optical Networks and Enabling Technologies (HONET)*, Cairo, pp. 274 - 279, diciembre 2010.

[144] V. M. Vokkarane y J.P. Jue, "Prioritized Routing and Burst Segmentation for QoS in Optical Burst-Switched Networks", en *Proceedings of Optical Fiber Communication Conference (OFC)*, Anaheim, p. 221, marzo 2002.

[145] H. Guo, Z. Lan, J. Wu, Z. Gao, X. Li, J. Lin y Y. Ji, "A testbed for optical burst switching network", en *Proceedings of Optical Fiber Communication Conference (OFC)*, Anaheim, paper OFA6, marzo 2005.

[146] W. Jian, Z. Wei y W. Minxue, "Optical Burst Switching Network Testbed", en *Optical Network Design and Modeling, Lecture Notes in Computer Science.*, Springer Berlin / Heidelberg, vol. 4534, pp. 166-175, 2007.

[147] W. Zhang, J. Wu, K. Xu y J.T. Lin, "TCP performance experiment on OBS network testbed", en *Proceedings of the Optical Fiber Communication Conference (OFC)*, Anaheim, paper OThF1, marzo 2006.

[148] W. Zhang, J. Wu, J. Lin, W. Minxue y S. Jindan, "TCP Performance Experiment on LOBS Network Testbed", en *Proceedings of the 11th international IFIP TC6 conference on Optical network design and modeling (ONDM)*, Atenas, pp. 186-193, mayo 2007.

[149] J. Wu, Y.W. Yin, S.R. Cai, X.B. Hong y J.T. Lin, "Experimental Performance Evaluation of High Speed TCPs in Traffic-Driven LOBS Network Testbed", en *Proceedings of the Conference on Optical Fiber communication (OFC)*, San Diego, paper OThB6, febrero 2008.

[150] A. Nawaz, X.B. Hong, J. Wu y J.T. Lin, "Impact of OBS and TCP parameters on instantaneous behavior of TCP congestion window on LOBS network testbed", en *Proceedings of Optical Fiber Communication Conference (OFC)*, San Diego, paper OMO2, marzo 2009.

[151] X. Hong, H. Guo, J. Wu, Y. Yin, L. Liu, Y. Zuo, K. Xu, J. Lin y T. Tsuritani, "Testbed of OBS/GMPLS interworking ", en *Proceedings of 6th International Conference on Broadband Communications, Networks, and Systems (BROADNETS)*, Madrid, septiembre 2009.

[152] T. Tanemura, A. Al Amin y Y. Nakano, "Development and Field Demonstration of an Optical Burst Switching Testbed with PLZT Optical Matrix Switch", en *Proceedings of International Conference on Photonics in Switching*, Pisa, septiembre 2009.

[153] IST PHOSPHORUS. [Online]. http://www.ist-phosphorus.eu/

[154] G. Zervas, R. Nejabati, Zhuoran Wang, D. Simeonidou, Siyuan Yu y M. O'Mahony, "A Fully Functional Application-aware Optical Burst Switched Network Test-bed", en *Proceedings of Conference on Optical Fiber Communication and the National Fiber Optic Engineers Conference (OFC/NFOEC)*, Anaheim, paper OWC2, marzo 2007.

[155] LightReading. (octubre 2008) Matisse demos EtherBurst. [Online]. http://www.lightreading.com/document.asp?doc_id=165306

[156] Network Strategy Partners LLC, "Total Cost of Ownership Analysis of Matisse Networks Etherburst Optical Switch", whitepaper http://0299d3f.netsolhost.com/NewPages/EtherBurst.pdf, abril 2007.

[157] J. McCann. Matisse is gone, European Telecom blog. [Online]. http://mccanntelecom.com/matisse-is-gone

[158] Intune networks. [Online]. http://www.intunenetworks.com

[159] Lightreading. (agosto 2009) OBS: The Pipes Are Calling. [Online]. http://www.lightreading.com/document.asp?doc_id=180392

[160] Lighreading. (julio 2010) Intune Gets Another €3M. [Online]. http://www.lightreading.com/document.asp?doc_id=193978

[161] Lightreading. (mayo 2011) Intune goes commercial. [Online]. http://www.lightreading.com/document.asp?doc_id=208305

[162] J. Dunne, T. Farrell y J. Shields, "Optical Packet Switch and Transport: A New Metro Platform to Reduce Costs and Power by 50% to 75% While Simultaneously Increasing Deterministic Performance Levels", en *Proceedings of the 11th International Conference on Transparent Optical Networks (ICTON)*, Island of São Miguel, paper Mo.D4.4, junio 2009.

[163] N. Deng, "Optical Burst Transport Network – Integrating Multi-Granular Switching on the Optical Layer", Huawei, marzo 2011, anunciado en http://www.ofcnfoec.org/Home/Exhibit-Displays-and-Activities/New-Show-Floor-Activities.aspx.

[164] lightreading. (marzo 2012) Huawei Strives for Optical Respect. [Online]. http://www.lightreading.com/document.asp?doc_id=218487

[165] Jose Nazario. (2012 enero) The megaupload Shutdown Effect. [Online]. http://ddos.arbornetworks.com/2012/01/the-megaupload-shutdown-effect-2/

[166] Megaupload. [Online]. http://www.megaupload.com

[167] Rapidshare. [Online]. www.rapidshare.com

[168] Emule. [Online]. www.emule.org

[169] Bittorrent. [Online]. www.bittorrent.com

[170] B. Cohen. (enero 2008) The BitTorrent Protocol Specification. [Online]. http://bittorrent.org/beps/bep_0003.html

[171] Youtube. [Online]. www.youtube.com

[172] P. Gil, M. Arlittz, Z. Li y A. Mahantix, "YouTube Traffic Characterization: A View From the Edge", en *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, San Diego, pp. 15-28, octubre 2007.

[173] J., Dusi, M. Wolfgang y K.C. Claffy, "Estimating Routing Symmetry on Single Links by Passive Flow Measurements", en *Proceedings of the 6th International Wireless Communications Mobile Computing Conference (IWCMC)*, Caen, pp. 473-478, junio 2010.

[174] CAIDA, The Cooperative Association for Internet Data Analysis. [Online]. www.caida.org

[175] Internet Engineering TaskForce. [Online]. www.ietf.org

[176] IETF. TCP Maintenance and Minor Extensions Working Group. [Online]. http://tools.ietf.org/wg/tcpm/

[177] J. Postel, "Transmission Control Protocol", *RFC 793,* septiembre 1981.

[178] M. Duke, R. Braden, W. Eddy y E. Blanton, "A Roadmap for Transmission Control Protocol (TCP) Specification Documents", *RFC 4614*, septiembre 2006.

[179] M. Welzl y W. Eddy, "Congestion Control in the RFC Series", *RFC 5783*, febrero 2010.

[180] M. Allman, V. Paxson y W. Stevens, "TCP Congestion Control", *RFC 2581*, abril 1999.

[181] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, To. Herbert, A. Agarwal, A. Jain y N. Sutin, "“An argument for increasing TCP's initial congestion window”. ", *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 3, pp. 26-33, junio 2010.

[182] J. Chu, N. Dukkipati, Y. Cheng y M. Mathis. (octubre 2011) Increasing TCP's Initial Window. [Online]. http://tools.ietf.org/html/draft-ietf-tcpm-initcwnd-00

[183] M. Hassan y R. Jain, *High Performance TCP/IP Networking: Concepts, Issues, and Solutions*, Prentice-Hall, 2003.

[184] M. Mathis, J. Mahdavi, S. Floyd y A. Romanow, "TCP Selective Acknowledgment Options", *RFC 2018*, octubre 1996.

[185] S. Floyd, J. Mahdavi, M. Mathis y M. Podolsky, "An Extension to the Selective Acknowledgement (SACK) Option for TCP", *2883*, julio 2000.

[186] E., Allman, M., Fall, K. Blanton y L. Wang, "A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP", *RFC 3517*, abril 2003.

[187] L. Xu, K. Harfoush y I. Rhee, "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks", en *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 4, Hong Kong, pp. 2514 - 2524, marzo 2004.

[188] A. Medina, M. Allman y S. Floyd, "Measuring the evolution of transport protocols in the internet", *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 2, pp. 37-52, abril 2005.

[189] S.R. Angajala, "A survey on various TCP protocols for High Speed Networks", *GESJ: Computer Science and Telecommunications*, vol. 29, no. 6, pp. 76-80, junio 2010.

[190] D. X. Wei, C. Jin, S. H. Low y S. Hegde, "FAST TCP: Motivation, Architecture, Algorithms, Performance", *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1246-1259, diciembre 2006.

[191] S. Floyd, "HighSpeed TCP for Large Congestion Windows", *RFC 3649*, diciembre 2003.

[192] T. Kelly, "Scalable TCP: Improving performance in high-speed wide area networks", *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 2, pp. 83-91, abril 2003.

[193] D.J. Leith y R.N. Shorten, "H-TCP Protocol for High-Speed Long-Distance Networks.", en *Proceedings of the 2nd Workshop on Protocols for Fast Long Distance Networks*, Argonne, p. , febrero 2004.

[194] S. Ha, I. Rhee y L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP", *ACM SIGOPS Operating Systems Review - Research and developments in the Linux kernel*, vol. 42, no. 5, pp. 64-74, julio 2008.

[195] M. Allman, "TCP Congestion Control with Appropriate Byte Counting (ABC)", *RFC 3465*, febrero 2003.

[196] I. F. Akyildiz, G. Morabito y S. Palazzo, "TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks", *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, junio 2001.

[197] S. Bhandarkar, A.L.N. Reddy, M. Allman y E. Blanton, "The Robustness of TCP to Non-Congestion Events", *RFC 4653*, agosto 2006.

[198] S. Bohacek, J.P. Hespanha, J. Lee, C. Lim y Obrac K., "TCP-PR: TCP for persistent packet reordering", en *Proceedings. 23rd International Conference on Distributed Computing Systems*, pp. 222-231, mayo 2003.

[199] S. Landstrom, H. Ekstrom, L. Larzon y R. Ludwig, "TCP-Aix: making TCP robust to reordering and delay variations", Luelea University of Technology 2006.

[200] Y. Tian, K. Xu y N. Ansari, "TCP in Wireless Environments: Problems and Solutions", *EEE Communications Magazine*, vol. 43, no. 3, pp. S27-S32, marzo 2005.

[201] L. Brakmo y L. Peterson, "TCP Vegas: End to End Congestion Avoidance on a Global Internet", *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465-1480, octubre 1995.

[202] C. P. Fu y S. C. Liew, "TCP Veno: TCP Enhancement for Transmission over Wireless Access Networks", *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 2, pp. 216-228, febrero 2004.

[203] R. Wang, K. Yamada, M. Y. Sanadidi y M. Gerla, "TCP with sender-side intelligence to handle dynamic, large, leaky pipes", *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 235-248, febrero 2005.

[204] V. Jacobson, R. Braden y D. Borman, "TCP Extensions for High Performance", *RFC 1323*, mayo 1992.

[205] A. Detti y M. Listanti, "Amplification effects of the send rate of TCP connection through an optical burst switching network", *Optical Switching and Networking*, vol. 2, no. 1, pp. 49–69, mayo 2005.

[206] Padhye, J., V. Firoiu, D.F. Towsley y J.F. Kurose, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation", *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 133 - 145, abril 2000.

[207] R. Pleich, de Vega Rodrigo, M. y J. Goetz, "Performance of TCP over Optical Burst Switching Networks", en *Proceedings of European Conference on Optical Communications (ECOC)* , vol. 4, Glasgow, pp. 883-884, septiembre 2005.

[208] S. Chi, Y. Yin, J. Wu, X. Hongm y J. Lin, "Experimental evaluation of TCP performance over OBS network with burst retransmission", en *Proceedings of the 7th*

International Conference on Optical Internet (COIN), Tokio, octubre 2008.

[209] R. Braden, "Requirements for Internet Hosts -- Communication Layers", *RFC 1122*, octubre 1989.

[210] H. Balakrishnan, V. N. Padmanabhan y R. H. Katz, "The Effects of Asymmetry on TCP Performance", en *Proceedings of ACM/IEEE Mobicom*, Budapest, septiembre 1997.

[211] P. Golden, H. Dedieu y K. S. Jacobsen, Eds., *Fundamentals of DSL Technology*, Auerbach Publications, julio 2004.

[212] M. Allman, "On the Generation and Use of TCP Acknowledgments", *ACM SIGCOMM Computer Communication Review*, pp. 4-21, octubre 1998.

[213] S. Gowda, R. K. Shenai, K. M. Sivalingam y H. C. Cankaya, "Performance Evaluation of TCP over Optical Burst-Switched (OBS) WDM Networks", en *Proceedings of IEEE International Communications Conference (ICC)*, Anchorage, pp. 1433–1437, 2003.

[214] B. Ryu y S. Lowen, "Fractal Traffic Models for Internet Simulation", en *Proceedings of the 5th IEEE Symposium on Computer Communications (ISCC)*, Antibes-Juan les Pins, pp. 200-206, julio 2000.

[215] F. Azimi y P. Bertok, "An Analytical Model of TCP Flow in Multi-hop Wireless Networks", en *35th Annual IEEE Conference on Local Computer Networks (LCN'10)*, Denver, 2010.

[216] S. Utsumi, S. M. S. Zabir y N. Shiratori, "TCP-Cherry for satellite IP networks: Analytical model and performance evaluation", *Computer Communications*, vol. 32, no. 12, pp. 1377-1383, julio 2009.

[217] S. Floyd, "TCP and Explicit Congestion Notification", *ACM SIGCOMM Computer Communication Review* , vol. 24, no. 5, pp. 8-23, octubre 1994.

[218] M. Mathis, J. Semke, J. Mahdavi y T. Ott, "The macroscopic behavior of the TCP congestion avoidance algorithm", *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 3, pp. 67 - 82, julio 1997.

[219] Z. Chen, T. Bu, M. Ammar y D. Towsley, "Comments on Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation", *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 451 - 453, abril 2006.

[220] B. Sikdar, S. Kalyanaraman y K. S. Vastola, "Analytic Models for the Latency and Steady-State Throughput of TCP Tahoe, Reno, and SACK", *IEEE/ACM Transactions on Networking*, vol. 11, no. 6, pp. 959-971, diciembre 2003.

[221] K. Zhou, K. L. Yeung y V. O. K. Li, "Throughput Modelling of TCP With Slow Start and Fast Recovery", en *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, St Louis, pp. 261-265, 2005.

[222] W. Xu, Y. Xu, X. Wu y K. Ou, "Modeling TCP Sack Steady State Performance in Lossy Networks", en *Proceedings of the 25th International Conference on Information Networking (ICOIN)*, Kuala Lumpur, pp. 278 – 283, enero 2011.

[223] K. Zhou, K. L. Yeung y V. O. K. Li, "On Performance Modeling of TCP New-reno", en *Proceedings of IEEE Global Telecommunications Conference (GLOBECOM)*, Washington, pp. 2650 - 2654, noviembre 2007.

[224] N. Parvez, A. Mahanti y C. Williamson, "An Analytic Throughput Model for TCP NewReno", *IEEE/ACM Transactions on Networking*, vol. 18, no. 2, pp. 448-461, abril

2010.

[225] S. Fortin-Parisia y B. Sericola, "A Markov model of TCP throughput, goodput and slow start", *Performance Evaluation*, vol. 58, no. 2-3, pp. 89-108, noviembre 2004.

[226] Matlab. [Online]. http://www.mathworks.com/products/matlab

[227] L. Qiu, Y. Zhang y S. Keshav, "Understanding the performance of many TCP flows", *Computer Networks*, vol. 37, no. 3–4, pp. 277–306, noviembre 2001.

[228] G. Raina, D. Towsley y D. Wischik, "Part II: Control theory for buffer sizing", *ACM/SIGCOMM Computer Communication Review* , vol. 35, no. 3, pp. 79–82, julio 2005.

[229] H. Han, C.V. Hollot, D. Towsley y Y. Chait, "Synchronization of TCP flow in networks with small droptail buffers", en *Proceedings of the 44th IEEE Conference on Decision and Control*, Sevilla, pp. 6762–6767, diciembre 2005.

[230] H. Han, C.V. Hollot, Y. Chait y V. Misra, "TCP networks stabilized by buffer-based AQMs", en *Proceedings of IEEE INFOCOM* , vol. 2, Hong Kong, pp. 964–974, marzo 2004.

[231] T. Bonald, M. May y J.-C. Bolot, "Analytic evaluation of RED performance", en *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000)*, vol. 3, pp. 1415–1424, marzo 2000.

[232] L. Le, J. Aikat, K. Jeffay y F.D. Smith, "The Effects of Active Queue Management and Explicit Congestion Notification on Web Performance", *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1217-1230, diciembre 2007.

[233] A. Pras, L. Nieuwenhuis, R. van de Meent y M. Mandjes, "Dimensioning network links: a new look at equivalent bandwidth", *IEEE Network*, vol. 23, no. 2, pp. 5-10, marzo 2009.

[234] J. Aracil, J. A. Hernández, A. J. Elizondo, R. Duque y O. Gonzalez de Dios, "On Local CAC Schemes for Scalability of High-speed Networks", *Journal of Networks*, vol. 5, no. 2, pp. 225-229, febrero 2010.

[235] M. Casoni, A.M. Guidotti y C. Raffaelli, "Multiple TCP flow performance study over OBS networks", en *Proceedings of NOC*, Kista, pp. 602-609, 2007.

[236] Y. Zhang, S. Wang, L. Li y S. Xu, "Throughput Evaluations and Improvements of High Speed TCPs over OBS Networks", en *Proceedings of International Conference on Information Technology and Computer Science (ITCS)*, vol. 2, Kiev, pp. 283-286, julio 2009.

[237] K. Papagiannaki, N. Taft, S. Bhattacharyya, P. Thiran, K. Salamatian y C. Diot, "A pragmatic definition of elephants in internet backbone traffic", en *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, Marseille, pp. 35-47, noviembre 2002.

[238] Internet2 NetFlow Statistics. [Online]. http://netflow.internet2.edu/

[239] R.V.D. Meent, A. Pras, M. Mandjes, H.V.D. Berg y L.J.M. Nieuwenhuis, "Traffic Measurements for Link Dimensioning - A Case Study", en *Proceedings of DSOM*, pp. 106-117, 2003.

[240] J. Perelló, S. Gunreben y S. Spadaro, "A Quantitative Evaluation of Reordering in OBS Networks and its Impact on TCP Performance", en *Proceedings of the IFIP Working Conference on Optical Network Design and Modelling (ONDM)*, Vilanova i la Geltru, marzo 2008.

[241] F. J. Ramón Salguero, J. Enríquez Gabeiras, J. Andrés Colás y A. Molíns Jiménez, "Multipath Routing with Dynamic Variance", COST 279 Technical Report TD02043, 2002.

[242] J. Aracil, O. Gonzalez de Dios y J.P. Fernandez-Palacios, "Routing strategies for OBS networks based on MRDV", en *Proceedings of ICTON 2005*, Barcelona , pp. 5-8, volumen 1, 2005.

[243] C. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm", *RFC 2992*, noviembre 2000.

[244] The Network Simulator - ns-2. [Online]. http://www.isi.edu/nsnam/ns/

[245] R. Hülsermann y M. Jäger, "Evaluation of a Shared Backup Approach for Optical Transport Networks", en *28th European Conference on Optical Communication (ECOC)*, Copenhagen, septiembre 2002.

[246] S. Baroni, P. Bayvel, R.J. Gibbens y S. K. Korotky, "Analysis and design of resilient multifiber wavelength-routed optical transport networks", *Journal of Lightwave Technology*, vol. 17, no. 5, pp. 743-58, mayo 1999.

[247] M. Düser y P. Bayvel, "Analysis of a Dynamically Wavelength-Routed Optical Burst Switched Network Architecture", *Journal of Lightwave Technology*, vol. 20, no. 4, pp. 574-585, abril 2002.

[248] I. de Miguel, "Diseño y Modelado de Redes Ópticas con Encaminamiento por Longitud de Onda y Conmutación de Ráfagas", *Tesis Doctoral*, Universidad de Valladolid, mayo 2002.

[249] A. Mokhtar y M. Azizoglu, "Adaptive wavelength routing in all-optical networks", *IEEE/ACM Transactions on Networking*, vol. 6, no. 2, pp. 197 - 206, abril 1998.

[250] M. Düser, A. Zapata y P. Bayvel, "Investigation of the scalability of dynamic wavelength-routed optical networks", *Journal of Optical Networking*, vol. 3, no. 9, pp. 667-686, 2004.

[251] I. Chlamtac, A. Ganz y G. Karmi, "Lightpath communications: An approach to high bandwidth optical WAN's", *IEEE Transactions on Communications*, vol. 40, no. 7, pp. 1171–1182, julio 1992.

[252] R. Casellas. (agosto 2011) Errata ID: 2940. [Online]. http://www.rfc-editor.org/errata_search.php?rfc=5440

[253] J.L. Le Roux, J.P. Vasseur y Y. Lee, "Encoding of Objective Functions in the Path Computation Element Communication Protocol (PCEP)", *RFC 5541*, Junio 2009.

[254] Wireshark. [Online]. http://www.wireshark.org/

[255] K. Kompella y Y. Rekhter, "Signalling Unnumbered Links in Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE)", *RFC 3477*, enero 2003.

[256] T. Otani y D. Li, "Generalized Labels for Lambda-Switch-Capable (LSC) Label Switching Routers", *RFC 6205*, marzo 2011.

[257] ITU-T, "Recommendation G.994.2 Spectral grids for WDM applications: DWDM frequency grid", junio 2002.

[258] D. Dhody y V. Manral, "Extensions to the Path Computation Element Communication Protocol (PCEP) to compute service aware Label Switched Path (LSP)", *draft-dhody-pce-pcep-service-aware-02*, diciembre 2011.

[259] JP. Vasseur, JL. Le Roux y Y. Ikejiri, "A Set of Monitoring Tools for Path Computation Element (PCE)-Based Architecture", *RFC 5886*, junio 2010.

[260] O. Gonzalez de Dios, R. Casellas, C. Margaria, Y. Lee y F. Zhang, "PCEP Extensions for Temporary Reservation of Computed Path Resources and Support for Limited Context State in PCE", *draft-gonzalezdedios-pce-reservation-state-01*, marzo 2012.

[261] E. Kozlovski y P. Bayvel, "QoS Performance of WR-OBS Network Architecture", en *6th Working Conference on Optical Network Design and Modelling*, Turin, pp. 101-116, 2002.

[262] Oracle. Java™ Platform, Standard Edition 6 API Specification. [Online]. http://docs.oracle.com/javase/6/docs/api/

[263] Eclipse.org - Helios Simultaneous Release. [Online]. http://www.eclipse.org/helios/

[264] JGraphT Java graph library. [Online]. www.jgrapht.org

[265] A. Rostami y A. Wolisz, "Modeling and Synthesis of Traffic in Optical Burst-Switched Networks", *IEEE/OSA Journal of Lightwave Technology*, vol. 25, no. 10, pp. 2942-2952, octubre 2007.

[266] CERN. COLT: Open Source Libraries for High Performance Scientific and Technical Computing in Java. [Online]. http://acs.lbl.gov/software/colt/

[267] M. Matsumoto y T. Nishimura, "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator", *ACM Transactions on Modeling and Computer Simulation*, vol. 8, No. 1, no. 1, pp. 3 - 30., enero 1998.

[268] H. Zechner, "Efficient sampling from continuous and discrete unimodal distributions", *Tesis doctoral, Technical University Graz, Austria*, 1994.

[269] The Linux Foundation. Netem. [Online]. http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

[270] J. Nagle, "Congestion Control in IP/TCP Internetworks", *RFC 896*, enero 1984.

[271] A.M. Law y W. Kelton, *Simulation Modeling & Analysis*, 2nd ed., McGraw-Hill, 1991.

[272] G. Bolch, S. Greiner, H de Meer y K.S. Trivedi, *Queing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*, John Wiley & Sons, Inc, 1998.

[273] O. González, I. de Miguel, N. Merayo, P. Fernández, R. M. Lorenzo y E. J. Abril, "The impact of delayed. ACK in TCP flows in OBS networks", en *Proceedings of 10th European Conference on Network and Optical Communications (NOC 2005)*, Londres, pp. 367-374, julio 2005.

[274] O. González de Dios, I. de Miguel, V. López Alvarez, R.J. Durán, N. Merayo y J.F. Lobo Poyo, "Estudio y simulación de TCP en redes de conmutación óptica de ráfagas (OBS)", en *XV Jornadas Telecom I+ D*, Madrid, noviembre 2005.

[275] O. Gonzalez *et al.*, "Traffic Modelling and Traffic Engineering for Next Generation Transport Networks - results from the NOBEL project", en *Proceedings of 10th European Conference on Network and Optical Communications (NOC 2005)*, Londres, julio 2005.

[276] J. F. Lobo Poyo, J.P. Fernández-Palacios Giménez, A. Ferreiro Olivo, O. González de Dios y J. Aracil, "Escenarios de Evolución de la Red Metropolitana", en *XV Jornadas Telecom I+ D*, Madrid, Noviembre 2005.

[277] O. González de Dios, I. de Miguel y V. López, "Performance evaluation of TCP over

OBS considering background traffic", en *Proceedings of 10th Conference on Optical Network Design and Modelling*, Copenhague, mayo 2006.

[278] J. Aracil *et al.*, "Optical Burst Switching - A Tutorial from e-Photon/One", en *Proceedings of IEEE International Conference on Communications (ICC 2006)*, Istambul, 2006.

[279] J.A. Hernández, J. Aracil, V. López, J. Fernández Palacios y O González de Dios, "A resilience-based comparative study between Optical Burst Switching and Optical Circuit Switching technologies," en *Proceedings of 8th IEEE International Conference on Transparent Optical Networks (ICTON)*, Nottingham, junio 2006.

[280] F. Callegati *et al.*, "Research on Optical Core Networks in the e-Photon/ONe Network of Excellence", en *25th Conference on Computer Communications (INFOCOM)*, Barcelona, abril 2006.

[281] J.M. Finochietto, J. Aracil, A. Ferreiro, J.P. Fernández-Palacios Giménez y O. González de Dios, "Migration Strategies Toward All Optical Metropolitan Access Rings", *Journal of Lightwave Technology*, vol. 25, no. 8, pp. 1918-1930, agosto 2007.

[282] C. García Argos, O. González de Dios y J. Aracil, "Adaptive Multi-Path Routing for OBS Networks", en *9th International Conference on Transparent Optical Networks* , Roma, pp. 299-302, julio 2007.

[283] S. Gunreben y O. González De Dios, "Why deterministic traffic shows the highest reordering ratio", en *Workshop on Optical Burst Switching*, Madrid, Septiembre 2009.

[284] O. Gonzalez de Dios, F.J. Jimenez Chico y F. Muñoz del Nuevo, "Benefits of limited context awareness in stateless PCE", en *Proceedings of OFC/NFOEC 2011*, Los Angeles, CA, paper OThI4, marzo 2011.

[285] F. Paolucci, O. González de Dios, R. Casellas, S. Duhovnikov, P. Castoldi, R. Muñoz y R. Martínez, "Experimenting Hierarchical PCE Architecture in a Distributed Multi-Platform Control Plane Testbed", en *Proceedings of Optical Fiber Communication Conference (OFC) 2012*, Los Angeles, paper OM3G.3, marzo 2012.

[286] O. González de Dios, I. de Miguel, R. J. Durán, J. C. Aguado, N. Merayo y P. Fernández, "Impact of TCP Synchronization on Capacity Dimensioning of Optical Burst Switched (OBS) Links", en *Proceedings of 17th European Conference on Network and Optical Communications (NOC 2012)*, Vilanova i la Geltrú, junio 2012.

[287] P. Hoffman y P. Harris, "The Tao of IETF: A Novice's Guide to the Internet Engineering Task Force", *RFC 4677*, septiembre 2006.

[288] C. Margaria, O. Gonzalez de Dios y F. Zhang, "PCEP extensions for GMPLS", *draft-ietf-pce-gmpls-pcep-extensions-05*, marzo 2012.