Table of Contents

Figures

Tables

A la memoria de mis abuelos

Acknowledgements

Although the current section is normally considered a cliché as far as the Graduation Thesis and Master of Research Thesis reports are concerned, it is very important for me to dedicate a part of this work to the acknowledgements. Thus, although I have personally thank many people and institutions, which have supported me at all times throughout the development of this Master of Research Thesis, I would like to leave in this report an official proof of my acknowledgement to them.

In the first place, thanks Dr Rubén M. Lorenzo Toledo for putting your trust on me at the moment of offering to me the participation in the European project CHRON, offer that I practically did not doubt in spite of being aware of the important effort that this new stage would suppose from my side. At last, the great working team that forms the Optical Communications Group in the Telecommunications Engineering School at the University of Valladolid, at which I extend my acknowledgements, has made that everything has been much easier I expected and important achievements are taking place.

I have devoted a great deal of hours for the elaboration of this work in the Lab 28 at the ETSIT, so I also want to thank all the members for their company and support, both on the professional and on the personal side. Many people have gone through this laboratory and if I start to name all of them, I would run the risk of forgetting someone, so definitely… Thanks everyone!

At this point, I would like to thank my tutors, Nacho and Ramón, for all its dedication. When reading the orientation guide for the elaboration of a Master of Research Thesis I found that this work was a full and complete responsibility on the student side. However, I would like to clarify that without the support ant the expert recommendations of my tutors this work would have never been as successful as it is at moment.

Then, I also want to show my acknowledgement to all the professors, leaded by the coordinator Yannis Dimitriadis, of the Information and Communication Technologies Research Master at the University of Valladolid, to which this work belongs and I am very proud to share with you.

Last but not least, this work has been supported by the CHRON project with funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 258644.

David.

Agradecimientos

Aunque la presente sección suele estar identificada como un tópico dentro de las memorias correspondientes a los Proyectos Fin de Carrera y los Trabajo Fin de Máster, para mi es muy importante el hecho de dedicar una parte de este trabajo a los agradecimientos, y por ello aunque a muchas personas e instituciones que me han ayudado durante la elaboración de este Trabajo Fin de Máster se lo he agradecido personalmente, me gustaría dejar constancia en este documento de mi profundo agradecimiento hacia las mismas.

En primer lugar, agradezco al Dr. Rubén M. Lorenzo Toledo su confianza puesta en mi en el momento de ofrecerme la participación en el proyecto europeo CHRON, oferta que prácticamente no dudé ni un segundo aun siendo consciente del importante esfuerzo que esta nueva etapa iba a suponer en mi. Al final con un buen equipo de trabajo formado por el Grupo de Comunicaciones Ópticas (GCO) de la Escuela de Ingenieros de Telecomunicación de la Universidad de Valladolid, al que hago extensibles mis agradecimientos, todo ha sido mucho más fácil de lo que me esperaba y se están consiguiendo importantes cosas.

La mayor parte del presente trabajo la he realizado en el Laboratorio 28 de la ETSIT, por ello quiero agradecer la compañía y apoyo, tanto profesional como personal, a todos los integrantes del mismo. Por dicho laboratorio han pasado muchas personas y si me pongo a nombrar a todos, sin ninguna duda correría el riesgo de dejarme algún nombre por el camino, por lo que en definitiva… ¡Gracias a todos!

Los que me conocen saben que me gusta el mundo taurino, y voy a permitirme el lujo de cambiar de tercio para agradecer la dedicación a mis dos tutores, Nacho y Ramón. En la guía de orientación del Trabajo Fin de Máster leí que la elaboración de dicho trabajo era una responsabilidad plena del alumno, pero quiero dejar claro que sin el apoyo y las expertas recomendaciones de mis dos tutores este trabajo no hubiese sido tan fructífero como es ahora.

Antes de terminar, también quiero mostrar mi agradecimiento a todo el profesorado, comandado por el coordinador Yannis Dimitriadis, del Master de Investigación en Tecnologías de la Información y la Comunicación de la Universidad de Valladolid, y al que pertenece el presente trabajo del que estoy muy orgulloso de compartir con usted.

Por último destacar que este trabajo ha sido apoyado por el proyecto CHRON a través de financiación del Séptimo Programa Marco de la Comunidad Europea [FP7/2007-2013] bajo el acuerdo de subvención nº 258644.

David.

Resumen

A continuación se presenta un resumen de cada uno de los capítulos que componen el presente Trabajo Fin de Máster.

1 Capítulo I – Introducción

Este capítulo introductorio presenta los principales aspectos que han motivado el desarrollo del presente trabajo y los objetivos que fueron definidos para ser cumplidos a lo largo de la elaboración de este Trabajo Fin de Máster con el fin de responder a la pregunta de investigación que originó esta contribución: ¿Es posible adaptar los mecanismos de control y gestión actuales a las redes ópticas heterogéneas emergentes?

Actualmente las redes ópticas se enfrentan a crecientes niveles de heterogeneidad que varían desde diferentes tipos de servicios de usuario a amplias configuraciones de la capa física. De esta forma, los operadores de red están haciendo un importante esfuerzo para superar el desafío de soportar una gran cantidad de servicios con diferentes requisitos en términos de calidad de servicio. Por tanto, un factor clave de las redes ópticas heterogéneas es cómo controlar y gestionar eficientemente los recursos de red a la vez que se satisfacen las demandas de usuario y se cumplen los requisitos de calidad.

El proyecto CHRON explora el potencial de la cognición para un control y una gestión eficiente de las mencionadas redes y propone una red óptica, cognitiva, heterogénea y reconfigurable que observa, actúa, aprende y optimiza su rendimiento, teniendo en cuenta su grado de heterogeneidad con respecto a la calidad de servicio y las técnicas de transmisión y conmutación.

CHRON es un proyecto europeo financiado por el séptimo programa marco bajo la temática “Internet del Futuro”. El consorcio lo forman 6 socios: Universidad de Valladolid (UVa), España; Technical University of Denmark (DTU), Dinamarca; Huawei Technologies Duesseldorf GmbH (HWDU), Alemania; Telekomunikacja Polska SA (TP SA), Polonia; Research and Education Laboratory in Information Technologies (AIT), Grecia; y Center of REsearch And Telecommunication Experimentation for NETworked communities (CREATE-NET), Italia. El proyecto empezó en Julio de 2010 y tiene una duración de 36 meses con un presupuesto de 3.15 M€.

El equipo de la UVa lidera el paquete de trabajo 3 en el que se define el sistema de decisión cognitivo, núcleo de la arquitectura de red de CHRON, que se implementa mediante el desarrollo de algoritmos de decisión cognitivos. Además, la UVa participa en el paquete de trabajo 5 donde se evalúa el rendimiento de la red CHRON con herramientas de simulación y emulación. Es por tanto en este segundo conjunto de tareas donde se encuadra el presente Trabajo Fin de Máster, cuyo principal objetivo es estudiar cómo adaptar los mecanismos actuales tanto de control como de gestión a las emergentes redes ópticas heterogéneas. En particular, este trabajo se centra en

GMPLS (*Generalized Multi-Protocol Label Switching*) para el control de la red, y en SNMP (*Simple Network Management Protocol*) y en sistemas basados en XML (*eXtensible Markup Language*) para la gestión de la misma. También se han definido arquitecturas de red con los correspondientes interfaces y se ha llevado a cabo un importante estudio del estado del arte referente a las arquitecturas cognitivas. Finalmente, el entorno de emulación DRAGON servirá como escenario de pruebas para medir la adaptabilidad de las técnicas de control y gestión a las necesidades cognitivas de CHRON.

Una vez definido el objetivo general se especifican los siguientes objetivos particulares:

- Introducir el marco de trabajo del proyecto CHRON y la arquitectura de red centralizada en la que se basa el resto del trabajo.
- Llevar a cabo una revisión de la literatura existente respecto a las aproximaciones de control y gestión en redes ópticas de transporte.
- Definir un lenguaje de especificación cognitivo para codificar los mensajes intercambiados entre el plano de control y gestión y el sistema de decisión cognitivo de la arquitectura CHRON.
- Analizar la aproximación de control basada en GMPLS y su adaptabilidad a la red de CHRON.
- Comparar las aproximaciones de gestión basadas en SNMP y XML mediante la definición de casos de uso considerados en el proyecto CHRON.
- Descubrir las funcionalidades del emulador DRAGON para llevar a cabo la emulación de redes ópticas de transporte.
- Emular una red real controlada por GMPLS en el emulador DRAGON mediante el establecimiento de LSPs (*Labeled Switched Paths*) y poniendo en práctica los resultados de la etapa de estudio del estado del arte.
- Emular una red real que utilice SNMP en el emulador DRAGON implementando casos de usos considerados en el proyecto CHRON.
- Construir un entorno de emulación DRAGON donde tanto las aproximaciones de control como de gestión en redes ópticas estén presentes.
- Conseguir el máximo conocimiento sobre aproximaciones de control y gestión en redes ópticas en un escenario emulado para afrontar las tareas de emulación venideras en el marco del proyecto CHRON.

En lo que respecta a la estructura, este Trabajo Fin de Máster se divide en cinco capítulos además de los espacios dedicados a las referencias (Capítulo VI), acrónimos y abreviaturas (Capítulo VII) y anexos (Capítulo VIII).

En primer lugar, el Capítulo I – Introducción, trata las secciones referentes a la motivación y los objetivos, y la estructura del informe.

Después, el Capítulo II – Arquitectura de Red de CHRON, presenta una introducción del marco de trabajo del proyecto CHRON poniendo atención en la arquitectura de red centralizada. Después, se trata el plano de control y gestión de CHRON presentando una visión general del lenguaje de decisión cognitivo, la interfaz entre el sistema de monitorización de la red y el sistema de decisión cognitivo, y el emulador DRAGON.

En tercer lugar, el Capítulo III - Aproximaciones de Control y Gestión en CHRON presenta, por una parte, la aproximación de control basada en GMPLS mediante la

presentación de la evolución de MPLS a GMPLS y analizando los distintos protocolos de encaminamiento y señalización que forman GMPLS. Por otra parte, se presentan las aproximaciones de gestión basadas en SNMP y XML. En lo que respecta a SNMP, dicha sección consiste de una visión general del protocolo en sí, una presentación de los procedimientos para gestionar la información y una descripción de los casos de uso considerados en CHRON. Después de una comparativa de las dos aproximaciones de gestión se presenta el protocolo NETCONF como gestión basada en XML mediante una descripción de la implementación de los casos de uso mencionados anteriormente. Finalmente se presentan algunos detalles de una aproximación de gestión híbrida.

En lo que respecta al Capítulo IV – Emulación de Redes Ópticas de Transporte, éste se divide en dos partes, la emulación DRAGON de GMPLS y la emulación DRAGON de SNMP. En el primer escenario la red *Deutsche Telekom* has sido emulada como una red real controlada por GMPLS, aunque previamente se presentan los elementos de software, la estructura, los procesos y las herramientas que conforman el emulador DRAGON. En lo que concierne a la emulación de SNMP, se han implementado los casos de uso considerados en el proyecto CHRON haciendo uso de la librería de Linux AGENT++/SNMP++.

Por último, el Capítulo V – Conclusiones presenta las principales conclusiones, futuras líneas de investigación y trabajos relacionados para completar esta contribución.

2 Capítulo II – Arquitectura de Red de CHRON

El proyecto CHRON aborda el desafío de controlar y gestionar la que se conoce como la próxima generación de redes ópticas heterogéneas. Para ello mediante el uso de cognición se pretende proporcionar decisiones efectivas en el momento de: encaminar nuevas demandas de tráfico a través de conexiones ópticas (*lightpaths*) existentes, nuevos *lightpaths* o ejecutando un proceso de configuración de la topología virtual; asignar recursos, técnicas de transmisión y conmutación, formatos de modulación, tasas de transmisión, etc.; y para asegurar un funcionamiento eficiente energéticamente hablando.

Todas las decisiones las toma el sistema de decisión cognitivo, que a su vez tiene en cuenta los requisitos de calidad de servicio (QoS) y de transmisión (QoT) de las distintas demandas de tráfico. De esta forma, éste el núcleo de arquitectura centralizada de CHRON e interactúa con el sistema de monitorización de la red (NMonS), que proporciona información sobre el estado del tráfico y medidas de calidad de transmisión óptica; y con el sistema de protocolos GMPLS para poder llevar a cabo las decisiones tomadas y para difundir la información de monitorización por la red.

La arquitectura de CHRON se divide en tres planos: plano de conocimiento (KP), plano de gestión y control (CMP) y plano físico (PHY).

En primer lugar el plano de conocimiento está principalmente representado por el CDS, que tiene una arquitectura modular dividida en cinco módulos: *Traffic Grooming* (TG), *Virtual Topology Design* (VTD), *Routing & Wavelength Assignment / Routing Modulation Level & Spectrum Allocation* (RWA/RMLSA), *QoT Estimator* and *Network Planner & Decision Maker* (NPDM). Cada uno de los módulos tiene asociada una base de conocimiento, que está relacionada con el propio proceso cognitivo a través de un módulo de aprendizaje.

Respecto al segundo plano, el CMP está compuesto por tres sistemas, el sistema de comunicación (CS), el sistema de protocolos GMPLS (GMPLS PS) y el sistema de monitorización de la red (NMonS). El CS se encarga de garantizar un correcto intercambio de mensajes, bien entre módulos del mismo nodo o entre módulos del nodo de control y los nodos normales. En referencia al GMPLS PS, es el sistema responsable de recuperar y compartir la información relativa a la disponibilidad de recursos y configuración, y de permitir el transporte y la posterior ejecución de las instrucciones provenientes del CDS. Sus componentes son: *Topology Server*, *Topology and Configuration database* (TC-db), TE Agent, RSVP-TE, OSPF-TE y CAC (*Control Access Channel*)/RM (*Resource Manager*). Finalmente, en lo con que concierne al tercer sistema, el NMonS es el sistema responsable de interactuar con la capa física para obtener los datos de los sistemas de monitorización y entregárselos al CDS para que los tenga en cuenta en sus evaluaciones. Los componentes del NMonS son: NMonS Server, NMonS database (NMonS-db) y NMonS Agent.

Para finalizar con los distintos planos, PHY cubre tanto la capa física como la capa de datos. Este plano consiste de un conjunto de dispositivos físicos que se encargan de garantizar las funcionalidades del plano de datos y de monitorizar aquellos parámetros considerados como relevantes dentro de los servicios proporcionados.

Una vez presentada la arquitectura de CHRON, el resto del capítulo enfatiza en el plano de control y gestión (CMP) y todas las acciones que desempeña cuando llega la petición de un nuevo *lightpath* a la red.

Otra tarea dentro del desarrollo de este Trabajo Fin de Máster ha sido la definición de un lenguaje de especificación cognitivo (CSL) basado en XML para codificar aquellos mensajes intercambiados entre el CDS y el CMP, y viceversa. Además, otro importante problema ha sido afrontado en este trabajo, la definición de la manera en la que el CDS es capaz de percibir las condiciones actuales de la red desde el NMonS.

El Capítulo II finaliza con una breve introducción del emulador DRAGON, que será la herramienta usada para testear los resultados de investigación obtenidos en torno a las aproximaciones de control y gestión en redes ópticas.

3 Capítulo III – Aproximaciones de Control y Gestión en CHRON

Aproximación de control basada en GMPLS

En este capítulo, se presentan, antes de entrar en los detalles técnicos, algunas pinceladas históricas del protocolo MPLS, considerado como referencia dentro de los protocolos actuales de control en redes ópticas.

MPLS combina las ventajas del encaminamiento a nivel de red con la rápida conmutación a nivel de enlace. Para tal fin, emplea una pequeña etiqueta de longitud fija que permite conseguir unas mejores prestaciones en las redes de transporte de paquetes IP. Esta etiqueta se asigna a cada paquete en base a la dirección de destino, los parámetros del tipo de servicio, la red virtual correspondiente o cualquier otro criterio.

En lo que respecta a la arquitectura de MPLS y a los elementos de la red, se proporciona una descripción técnica con el objetivo de entender el funcionamiento del protocolo. Concretamente se describen los conceptos de *Label*, *Label Switched Router* (LSR), *Forwarding Equivalence Class* (FEC), MPLS *Header*, *Label Switched Path* (LSP) y *Label Information Base* (LIB).

A continuación se resume la operación de transmisión básica de MPLS. En el momento en que un paquete llega a la red MPLS, después de ser examinado por el nodo de entrada (*Ingress* LSR) se le asigna una FEC en función de la dirección de destino y la calidad de servicio requerida. Así, el paquete no será examinado por los nodos intermedios ya que éstos solamente identificarán la etiqueta de entrada, consultarán la base de datos LIB y sustituirán dicha etiqueta por la correspondiente etiqueta de salida. Finalmente, el paquete alcanzará el nodo de salida (*Egress* LSR), el cual eliminará la cabecera MPLS y lo renviará por encaminamiento convencional en base a la dirección IP de destino.

Según la información anterior, uno puede rápidamente pensar en las ventajas de MPLS sobre el encaminamiento convencional. Así, en el escenario MPLS la elección de los caminos se basa en una sencilla etiqueta que se añade a cada paquete que llega a la red. Por tanto, solamente los nodos extremos de la red MPLS, que conectan una red tradicional con dicha red MPLS, examinará la cabecera IP. El tratamiento de los paquetes por los nodos intermedios resulta mucho más sencillo y requiere de mucho menos tiempo que la técnica *Longest Prefix Match*, lo que supone de mejores prestaciones respecto al encaminamiento IP tradicional.

Continuando con la evolución de los protocolos de control, el primer intento de modificar MPLS para operar sobre las redes ópticas fue MPλS (*Multi-Protocol Lambda Switching*). En este protocolo la longitud de onda se usa como etiqueta. Aunque MPλS no tiene todas las características de MPLS, hay una destacada correspondencia entre los LSRs y los conmutadores de longitud de onda, y entre una LSP y un camino óptico (*lightpath*). Sin embargo, MPλS sólo proporciona soluciones para escenarios de conmutación por longitud de onda, por lo que otros tipos de conmutación están fuera del alcance de dicho protocolo. De esta forma, GMPLS (*Generalized Multi-Protocol*

Label Switching) apareció como una nueva solución para abarcar todos los tipos de conmutación de una manera generalizada.

El concepto de generalización unido a GMPLS consiste en una modificación de la etiqueta de MPLS descrita anteriormente. Esta etiqueta pasa a poder informar del número de ranura temporal, el identificador de longitud de onda o el número de fibra dentro de un conjunto de fibras dentro del mismo cable. En contraposición a MPLS, en el caso de las etiquetas de GMPLS, además de representar el tipo de tráfico también representan los recursos de la red.

Entre las novedades de GMPLS descritas en el Capítulo III están las siguientes: separación entre el plano de control y el plano de datos, la etiqueta generalizada, el establecimiento de LSPs bidireccionales, la característica de *Forwarding Adjacency* (FA) y el agrupamiento de enlaces, el establecimiento de LSPs inter-área, y las extensiones de administración y seguridad.

Respecto a los protocolos que forman GMPLS, por una parte están los protocolos de encaminamiento, que se encargan de difundir la topología de la red y las características TE. Además, son responsables de calcular rutas entre dos puntos de la red bajo determinadas condiciones para la transmisión de los datos. Los protocolos de estado de enlace usados normalmente en GMPLS son OSPF-TE (*Open Shortest Path First-Traffic Engineering*) o IS-IS-TE (*Intermediate System to Intermediate System-Traffic Engineering*).

Por otra parte, el segundo conjunto de protocolos son los protocolos de señalización, que se encargan de distribuir las etiquetas entre los LSRs, reservando los recursos físicos de un LSP para llevar a cabo la transmisión de datos a la vez que se garantiza una específica calidad de servicio, manteniendo la conectividad de los enlaces, y detectando y notificando errores para el restablecimiento de LSPs. En este caso, el IETF definió dos protocolos de señalización para MPLS, que han sido extendidos para ser implementados en GMPLS. Estos protocolos son RSVP-TE (*Resource reSerVation Protocol - Traffic Engineering*) y CR-LDP (*Constrained-Based Routing Label Distribution Protocol*).

Este Trabajo Fin de Master se centra en los protocolos OSPF-TE y RSVP-TE ya que son los usados en el emulador DRAGON para llevar a cabo el encaminamiento y la señalización, respectivamente.

RSVP-TE define siete tipos de mensajes que se explican en detalle a lo largo del capítulo: *Path*, *Resv*, *PathErr* (*Path Error*), *ResvErr* (*Reserve Error*), *PathTear*, *ResvTear* y *ResvConf* (*Reserve Confirm*). Los dos primeros mensajes, *Path* y *Resv*, se usan para pedir y reservar una sesión de señalización. Luego, los mensajes *PathTear* y *ResvTear* se usan para finaliza dicha sesión; *PathErr* y *ResvErr* son mensajes de notificación de errores, y finalmente, *ResvConf* se envía opcionalmente para confirmar una reserva.

La sección dedicada a RSVP-TE termina con una introducción de nuevos objetos para soportar las funcionalidades de GMPLS: *Record Route Object* (RRO), *Explicit Route Object* (ERO), *Session Attribute Object* (SOA), *Generalized Label request object*, *Label Set* (LS) *object*, *Suggested Label* (SL) *object*, *Upstream Label Set* (ULS) and *Upstream Label object* (UL).

Aproximación de gestión basada en SNMP

La segunda sección del Capítulo III empieza con un análisis del potencial de SNMP (*Simple Network Management Protocol*) para llevar a cabo tareas de gestión en la arquitectura centralizada de CHRON. En primer lugar, se presenta una visión general de SNMP. Luego se estudia un par de casos de uso, que tratan dos escenarios considerados en el proyecto, poniendo atención en la estructura de los mensajes SNMP y en la gestión de la información.

Tres son los aspectos clave a tener en cuenta al usar SNMP: *Management Information Base* (MIB), *Structure Management Information* (SMI) y el protocolo SNMP. En general, el modelo de arquitectura SNMP es una colección de estaciones de gestión de red que monitorizan y controlan elementos de la red, y agentes ubicados en dicho elementos, que realizan las acciones solicitadas por las estaciones de gestión.

Los mensajes SNMP se pueden dividir en varios grupos. Por ejemplo, *GetRequest*, *GetNextRequest* y *GetBulkRequest* se envían desde un gestor (*manager*) a un agente para pedir datos referidos a una instancia en la MIB, la instancia siguiente en una cierta lista, o cierta información almacenada en bloques, respectivamente. El mensaje *InformRequest* se envía entre dos gestores para compartir información. Luego, el mensaje *SetRequest* se envía de un gestor a un agente para fijar el valor de un objeto en la MIB del agente. Respecto al mensaje *Response*, éste se envía desde un agente a un gestor para responder a una petición previa. Finalmente, el mensaje *Trap* se envía desde un agente a un gestor para notifica un evento excepcional.

Una vez presentados los aspectos teóricos y técnicos de SNMP se han definido dos casos de uso para su implementación en la arquitectura centralizada de CHRON. En primer lugar, un monitor percibe que el nivel de relación señal a ruido óptica (OSNR) baja por debajo de un umbral prefijado y se lo notifica al sistema de decisión cognitivo mediante el envío de un *Trap*. En el segundo caso de uso, el CDS solicitará (a través del manager SNMP) el valor de OSNR al agente SNMP.

Aproximación de gestión basada en XML

XML (*eXtensible Markup Language*) y sus tecnologías subyacentes ofrecen oportunidades para superar algunos de los contratiempos de SNMP. En esta sección, primero se presenta una comparativa entre SNMP y XML para mostrar que el uso de XML en la gestión de una red se considera como una solución prometedora. Sin, embargo, la mayoría de los dispositivos de red actualmente desplegados están equipados solamente con agentes SNMP. Por lo tanto, para facilitar el proceso de migración de las redes actuales se consideran las aproximaciones híbridas que permiten una coexistencia de agentes SNMP y XML.

Después, como ejemplo de una aproximación de gestión totalmente basada en XML se presenta el protocolo NETCONF para mostrar las fortalezas de dicha alternativa. NETCONF define un simple mecanismo a través del cual un dispositivo de red puede ser configurado, dichos datos de configuración y estado se pueden obtener, y nuevos datos se pueden enviar al dispositivo en cuestión.

En lo que respecta a la gestión de la información, existen distintos tipos de lenguajes de modelado tales como *Document Type Definition* (DTD), XML *Schema Definition* (XSD), *Relax New Generation* (RelaxNG) y YANG.

Además, los dos casos de uso considerados en el análisis de SNMP se estudian aquí de nuevo para explicar la interacción entre los diferentes módulos de la arquitectura de CHRON.

Finalmente, esta sección y por ende este capítulo termina con posibles combinaciones de manager y agentes en arquitecturas de gestión híbridas, y la puesta en marcha en dicho escenario de los casos de uso estudiados.

4 Capítulo IV – Emulación de Redes Ópticas de Transporte

Con el emulador DRAGON es posible implementar un entorno de emulación en una red Ethernet donde los LSPs extremo a extremo son establecidos mediante VLANs (*Virtual Local Area Network*), que permiten llevar a cabo pruebas en redes lógicamente independientes de una manera similar a los *lightpaths* en las redes ópticas de transporte.

El paquete de software DRAGON permite llevar a cabo las tareas del plano de control de una red. La arquitectura de este plano de control consiste de cuatro elementos: *Client System Agent* (CSA), *Network Aware Resource Broker* (NARB), V*irtual Label Switching Router* (VLSR) y *Application Specific Topology Builder* (ASTB). El funcionamiento de cada uno de estos componentes se explica en detalle a lo largo de este capítulo.

En lo que respecta a los protocolos GMPLS de encaminamiento de DRAGON, este emulador usa el protocolo de encaminamiento OSPF-TE y el protocolo de señalización RSVP-TE para las tareas de control. Aparte de los componentes software, DRAGON está formado por un conjunto de procesos demonios de Linux (Zebra, DRAGON, RSVP-TE y OSPF-TE) que se ejecutan en segundo plano para llevar a cabo las funcionalidades de GMPLS.

El software de DRAGON ha sido diseñado para ejecutarse en varios PCs Unix conectados mutuamente y comunicados a través de sockets y túneles GRE (*Generic Routing Encapsulation*). Para emular una topología de red en una única máquina física DRAGON propone la técnica *User Mode Linux* (UML). De esta manera, es factible emular una topología de red en un solo PC ejecutando UML como un proceso de usuario en una máquina. Luego, cada uno de los procesos que pertenecen al Kernel UML serán máquinas virtuales que representan a cada uno de los elementos en la topología de la red.

Por otro lado, DRAGON recibe soporte de otra herramienta denominada VNE (*Virtual Network Experiment*) así como de la sintaxis de XML para la configuración de la topología de la red. Este hecho lleva a DRAGON a una herramienta flexible y rápidamente editable en la que el usuario puede iniciar y detener los distintos nodos e interactuar con ellos para el establecimiento de LSPs o el cálculo de rutas como si se tratase de una red óptica real.

Para entender la estructura del emulador DRAGON se ha llevado a cabo un estudio inicial con una topología sencilla antes de pasar a emular la red Deutsche Telekom (DT). Luego, las pruebas llevadas a cabo en esta red se pueden resumir en la creación de LSPs tanto unidireccionales como bidireccionales, la manipulación del objeto ERO para que el servidor NARB pueda encaminar en base a una ruta estricta, y finalmente el análisis de los mensajes RSVP-TE intercambiados.

En cuanto a la segunda sección de este capítulo, después de la presentación teórica de SNMP en la segunda sección del Capítulo III, en este punto se presenta un entorno de emulación SNMP integrado en DRAGON como aproximación de gestión para una red óptica. Para conseguir un escenario en el GMPLS y SNMP convivan juntos, el sistema de archivos del emulador DRAGON ha sido modificado mediante la instalación de la librería AGENT++/SNMP++.

En lo que respecta a las pruebas de emulación SNMP, en primer lugar, un escenario *Trap* SNMP se expone como implementación del caso de uso 1 considerado en CHRON para SNMP. Después, un escenario de petición/respuesta se presenta para entender mejor el caso de uso 2 considerado en CHRON y descrito, como el anterior, en el Capítulo III.

5 Capítulo V – Conclusiones

En este trabajo se han estudiado y emulado estrategias de control y gestión en redes ópticas heterogéneas dentro del marco de trabajo del proyecto europeo CHRON. Así, después de introducir el proyecto CHRON y su arquitectura de red centralizada, una revisión exhaustiva de la literatura ha proporcionado importante conocimiento, en términos de aproximaciones de control y gestión en redes ópticas de transporte, para ser puesto en práctica en un entorno de emulación.

En este entorno de emulación, el pilar ha sido el emulador DRAGON a través del cual se ha llevado a cabo la emulación del control y la gestión de la red Deutsche Telekom. En base a esta fase es importante destacar que la manipulación y la extensión de este emulador han sido tareas de arduo trabajo y por tanto un importante desafío para el autor.

Dentro del marco del proyecto CHRON, la principal contribución ha sido conseguir un importante conocimiento y experiencia en las aproximaciones de control y gestión en redes ópticas, para de esta manera afrontar con éxito la fase de pruebas (*testbed*) del proyecto.

Como posibles líneas de investigación futuras cabe destacar la posibilidad de extender los protocolos de GMPLS, OSPF-TE (*Open Shortest Path First – Traffic Engineering*) y RSVP-TE (*Resource reSerVation Protocol – Traffic Engineering*), para cumplir las necesidades de CHRON a la hora de controlar y gestionar una red óptica cognitiva, heterogénea y reconfigurable. Además, debido a que algunas redes basan su control en el protocolo PCEP (*Path Computation Element Protocol*), la integración de esta aproximación en el emulador DRAGON podría ser un importante trabajo futuro.

Finalmente, aspectos como la protección y restauración ópticas podrían ser otros dos puntos para continuar el presente trabajo.

Chapter I – Introduction

1 Introduction

This first chapter presents those aspects that have motivated the development of the current work and those objectives that were defined to be met throughout this Master of Research Thesis lifetime with the aim of answering the research question that overflies this contribution: *Is it possible to adapt the current control and management mechanisms to the emerging heterogeneous optical networks?*

In the first place, an introduction is presented to put the reader into the context of CHRON project since this work has been undertaken within the framework of this project. Then, the main objectives of this Master of Research Thesis are listed to focus on the main contributions of the work here presented.

1.1 Motivation and Objectives

Optical networks are nowadays facing with increased levels of heterogeneity, which range from different type of user services to wide physical layer configurations. Thus, network operators are doing their best to cope with the challenge of supporting a great deal of services with different requirements in terms of Quality of Service (QoS), by means of optical transport networks which cover different transmission technologies (modulation formats, data rates, etc.) and different switching paradigms (semi-static/dynamic). Hence, a key factor of these highly heterogeneous optical networks is how to efficiently control and manage network resources whilst fulfilling user demands and meeting QoS requirements.

The CHRON project explores the potential of cognition for an efficient control and management of the aforementioned heterogeneous networks. A cognitive network is defined as "*a network with a process that can perceive current network conditions, and then plan, decide, and act on those conditions. The network can learn from these adaptions and use them to make future decisions, all while taking into account end-to-end goals*" [1]. In addition, these networks rely on the application of machine learning techniques, i.e., algorithms/processes that "*improve its performance through experience gained over a period of time without complete information about the environment in which it operates*" [1]. In this way, the use of cognition in optical networks can offer much more flexibility to telecom operators by tuning physical layer components characteristics and networking layer parameters.

Another important advantage that arises from the presence of cognition is based on a potential decrease of energy consumption. In accordance with the "*Green Future Internet*" paradigm, a cognitive process in the network can take notice of the power usage and availability of resources when deciding how to handle new traffic demands according to the current network status provided by the monitoring system, and to configure the devices properly to reduce the consumption.

In a word, for these reasons, the EU-funded CHRON project proposes a Cognitive Heterogeneous Reconfigurable Optical Network, which observes, acts, learns and

optimizes its performance, taking into consideration its degree of heterogeneity with respect to QoS, transmission and switching techniques.

At moment, once presented the project in which this work is embedded, it is time the reader had the first contact with the motivation of this Master of Research Thesis to sail through this report with a much clearer vision.

1.1.1. Motivation

The CHRON project is an EU-funded FP7 project under the theme '*The Network of the Future* '. The Consortium is formed by 6 partners: University of Valladolid (UVa), Spain; Technical University of Denmark (DTU), Denmark; Huawei Technologies Duesseldorf GmbH (HWDU), Germany; Telekomunikacja Polska SA (TP SA), Poland; Research and Education Laboratory in Information Technologies (AIT), Greece; and Center of REsearch And Telecommunication Experimentation for NETworked communities (CREATE-NET), Italy. The CHRON project started in July 2010 and has a lifetime of 36 months with a budget of 3, 15 M€.

UVa leaders the work-package 3 in which the Cognitive Decision System (CDS), core of the CHRON network architecture, is defined and implemented by developing cognitive decision algorithms. In addition, UVa takes part in the work-package 5 where the performance of the CHRON network will be evaluated by means of both simulation and emulation tools.

In light of the above, it is in this latter set of assigned tasks where the current Master of Research Thesis is placed. The main objective of this Thesis, which belongs to the Master in Information and Communication Technologies (ICT) at the University of Valladolid, is to study how to adapt both the control and management current network mechanisms to the emerging heterogeneous optical networks. In particular, special attention has been paid to the GMPLS (Generalized Multi-Protocol Label Switching) and SNMP (Simple Network Management Protocol) protocols as well as to the XML (eXtensible Markup Language) based management. Moreover, some network architectures and the corresponding interfaces have been defined to make feasible the communication among the control and management planes and the cognitive decision system included in the CHRON architecture [2]. Although it is not one of the objectives of this work, an important state of the art of cognitive network architectures has been realized and further information can be found in [2].

Finally, the DRAGON emulation environment will serve as proof scenario to measure the adaptability of the control and management techniques to the CHRON cognitive needs [3].

1.1.2. Objectives

After presenting the main objective of this Master of Research Thesis in the previous epigraph, this sub-section contains the specific objectives targeted with this work:

- To introduce the CHRON project framework and the centralized network architecture on which the rest of the work is focused.

- To undertake a review in the existent literature regarding the control and management approaches in optical transport networks.
- To define a cognitive specification language to encode the messages exchanged between the control and management plane and the cognitive decision system of CHRON architecture.
- To analyse the control approach based on GMPLS protocols and its suitability to the CHRON network.
- To compare the SNMP-based and XML-based management approaches in optical transport networks by defining use cases considered in CHRON project.
- To discover the DRAGON emulator functionalities to carry out the emulation of optical transport networks.
- To emulate a real network controlled by GMPLS in the DRAGON emulator by establishing LSPs and putting in practice the results of the state-of-the-art stage.
- To emulate a real network managed by SNMP in the DRAGON emulator by implementing a number of use cases considered in the CHRON project.
- To build a DRAGON emulation environment where both control and management approaches in optical networks are present all in one.
- To get the maximum know-how of the control and management approaches in optical networks in an emulated scenario to face the incoming CHRON testbed/emulation tasks successfully.

1.2 Structure of the Master of Research Thesis

All introductory issues considered, the current Master of Research Thesis is structured as follows. As a whole the report is divided into five chapters plus the sections devoted to references (Chapter 6), acronyms and abbreviations (Chapter 7) and annexes (Chapter 8).

Firstly, the current **Chapter I - Introduction** has covered the sections that bring together the motivation issues and objectives of this Master of Research Thesis, and the structure of the report.

Secondly, **Chapter II – CHRON Network Architecture** presents an introduction of the CHRON project framework by paying attention to the centralized network architecture. Then, the CHRON control and management plane is treated in detail by introducing the CHRON cognitive specification language, the feedback system communication between the Network Monitoring System (NMonS) and the CDS, and the DRAGON emulator. Although this chapter does not reflect the main contribution of this work, an important effort has been done to define the CHRON network architecture, both in layer and modular formats, and the cognitive specification language to encode the messages exchanged among different modules.

In the third place, **Chapter III – Control and Management Approaches in CHRON** gathers, on the one hand, the control management approach based on GMPLS by presenting the evolution from MPLS to GMPLS and analysing the protocols that form GMPLS. On the other hand, the management approaches based on SNMP and XML are presented. As far as the former approach is concerned, the SNMP section consists of a general overview of the protocol itself, a presentation of the procedures to manage the information and a description of the use cases considered in CHRON. Then, in regards with the XML-based approach, after comparing both approaches, the NETCONF protocol is presented as fully XML-based management and the use cases considered in CHRON are described before briefly introducing hybrid approaches.

As for **Chapter IV – Emulation of Optical Transport Networks**, it is divided into two parts, the DRAGON GMPLS emulation and the DRAGON SNMP emulation in CHRON. In the former emulation environment the Deutsche Telekom network has been emulated as a real network controlled by GMPLS. At the beginning of this section the software elements, structure, processes and tools associated with DRAGON have been presented to show the emulator functionalities. Concerning the other emulation environment, the same network has been emulated to implement the SNMP use cases defined in CHRON project. This chapter constitutes the main contribution of this Master of Research Thesis, which is expected to be useful during the testbed/emulation stage of CHRON project.

Finally, **Chapter V – Conclusions and Future Research Lines** puts forward with the main conclusions, future research lines and related works to complete this report.

Chapter II – CHRON Network Architecture

2.1 CHRON Project Framework

2.2 CHRON Control and Management Plane

2.3 Conclusions

2 CHRON Network Architecture

This second chapter firstly focuses on the introduction of the CHRON project framework to prepare the reader to know this project. In the second section, it emphasizes on the Control and Management Plane (CMP) that constitutes the basis of this work when studying the control and management approaches in optical networks.

2.1 Overview of Optical Networks

Technologies around the optical networks field have been improving steadily in recent years by means of systems capable of providing the huge amount of bandwidth requested by the current application services. WDM (Wavelength Division Multiplexing) technology has been deployed in long-distance telecommunication networks on a point-to-point basis. In this scenario, the optical signal was initially converted back to electronics domain at each node. Due to the high costs of electronic switching and processing costs important research work is being done to achieve systems capable of switching data entirely in the optical domain.

In the aforementioned scenario the Optical Cross Connects (OXCs) play an important role as it is in charge of switching an optical signal from an input to a concrete output. With the configuration of these devices across a network one can establish all-optical connections, or lightpaths, between source and destination nodes. Thus, data carried on these lightpaths avoid electronic conversion and processing at intermediate nodes, therefore alleviating the electronic bottleneck.

Following with the presentation of more concepts related to optical networks, the problem of finding a route for a lightpath and assigning a wavelength to the lightpath is often referred to as the Routing and Wavelength Assignment problem (RWA). The objective of the problem is to route lightpaths and assign wavelengths in a manner which minimizes the amount of resources that are used in the network.

Another important concept is the virtual topology. First, the physical topology of a network is the set of nodes and optical links among the different nodes, which can be nonadjacent. In this way, once established the optical paths (lightpaths), the set of nodes and the optical paths form the virtual topology of a network. Further information regarding optical networks can be found in [4] and [5].

2.2 CHRON Project Framework

The CHRON project focuses on optical transport networks that are evolving as a result of the need to accommodate for the increasing demands in capacity, variety of services and emerging new optical transmission technologies. The high degree of heterogeneity means for carriers increased complexity in planning, implementing, and operating their networks. The CHRON project proposes a strategy to efficiently control the network by cognitively taken decisions on the most efficient use of its resources to

match the best transmission and switching technology to satisfy the end-to-end service requirements.

CHRON focuses on a reconfigurable transport optical network, where a reconfigurable virtual topology by means of optical connections (lightpaths) is established, and which can accommodate new traffic demands by routing them through existing lightpaths, establishing new lightpaths (which can be later torn down if no longer required) or reconfiguring the virtual topology to better adapt to current network conditions.

According to the explanation given in the previous section, the CHRON project addresses the challenge of controlling and managing the next generation of heterogeneous optical networks supporting the Future Internet. In particular, a CHRON network should be able to provided effective decisions, by relying on cognition:

- To route new traffic demands, either through existing optical connections (lightpaths), through new lightpaths or by triggering a reconfiguration process of the virtual topology (i.e., re-arranging existing connections);
- To assign resources, not only wavelengths, but also the most appropriate transmission/switching techniques, modulation formats, bit rates, etc.;
- To ensure energy-efficient operation.

All the decisions will be made while taking into account the Quality of Service (QoS) and the Quality of Transmission (QoT) requirements of the demands.

Judging from the definition of cognitive networks given before [1], those decisions must be made by taking into consideration current status and knowledge acquired from actions in the past. Therefore, the core element of the CHRON architecture is the Cognitive Decision System (CDS). Such a system interacts with the Network Monitoring System (NMonS), which provides traffic status and optical quality of transmission measurements, and with the GMPLS Protocols System (GMPLS PS) to implement the decisions made by the CDS and to disseminate the monitored information (Figure 2.1).

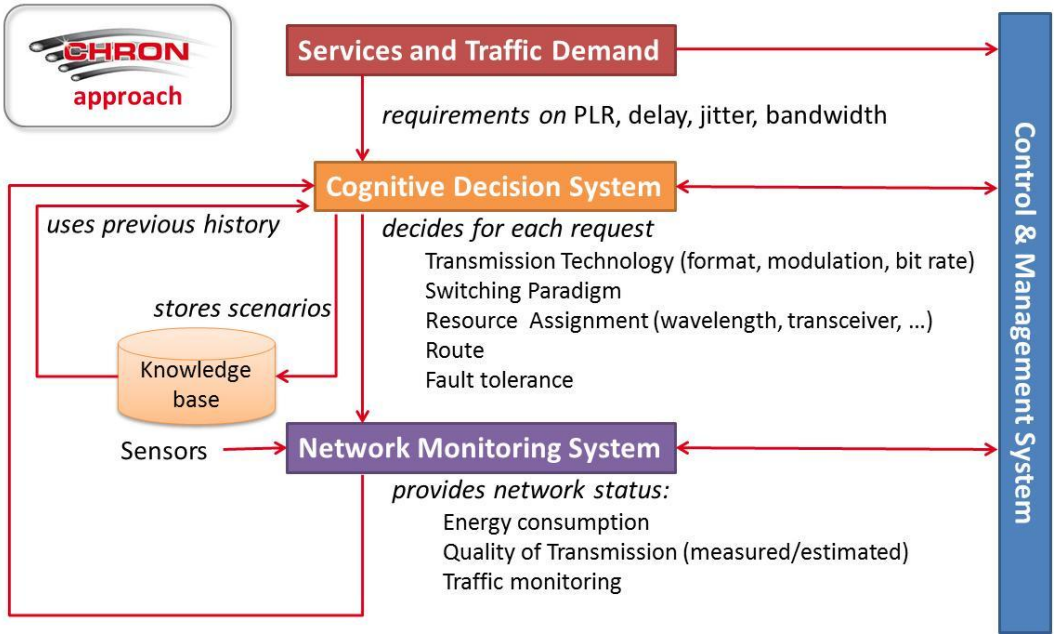

Figure 2.1 CHRON conceptual architecture

2.2.1. CHRON Centralized Network Architecture

In the CHRON framework two types of network architectures are considered, the centralized and the distributed one. The former not only involves centralized control but also centralized cognition. The latter, presents both distributed control and distributed cognition. Both architectures were presented in [6], but according to the information exposed in [7], the final chosen architecture to exploit the cognition advantages is the **centralized** one.

In the centralized approach, all the decisions, which may be triggered by the CDS itself or by incoming user requests, are made by a single CDS instance running on a single node, the control node (Figure 2.2). The set of physical devices enabling the communication and the devices managing and monitoring them on each node form the PHYsical plane (PHY), where all the decisions made by CDS are actually implemented. Then, between the CDS and the PHY, the Control and Management Plane (CMP) takes part of the architecture for providing the information required by CDS to correctly make its decisions and for dispatching those decisions to the involved nodes.

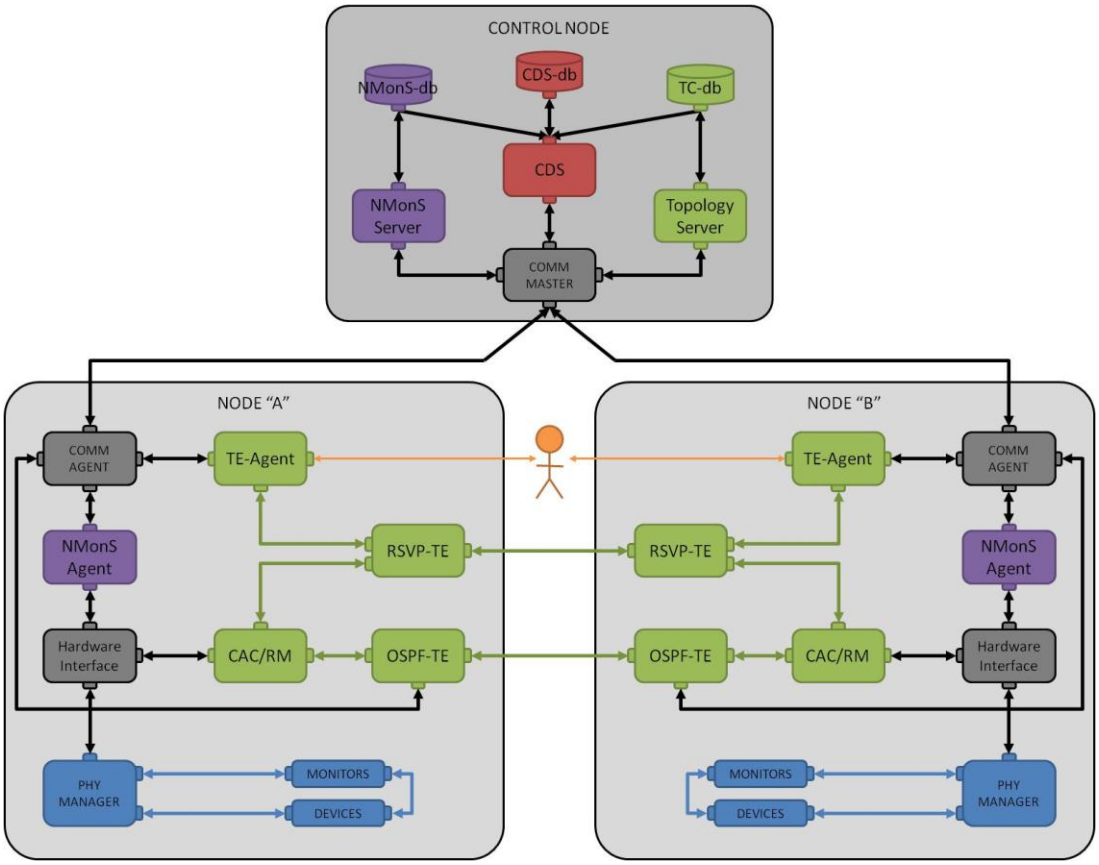

Figure 2.2 CHRON centralized network architecture [7]

In light of the above, three different planes can be distinguished in the CHRON network architecture:

- Knowledge Plane (KP)
- Control and Management Plane (CMP) [8]
- Physical Plane (Physical Layer + Data Plane) (PHY)

Both the blocks and the information contained in Figure 2.2 are later explained when presenting the components of the different planes in the epigraphs of this sub-section.

After the previous planes distinction and emphasizing on the fact that the cognitive networks typically perform better in cross-layer architectures and multi-objective optimization scenarios [1], the previous planes have been defined in accordance with the layered view architecture presented in Figure 2.3 [9]. This figure presents the same information that the previous one, the CHRON centralized network architecture in a different way by distinguishing the horizontal layers and the interfaces among them.

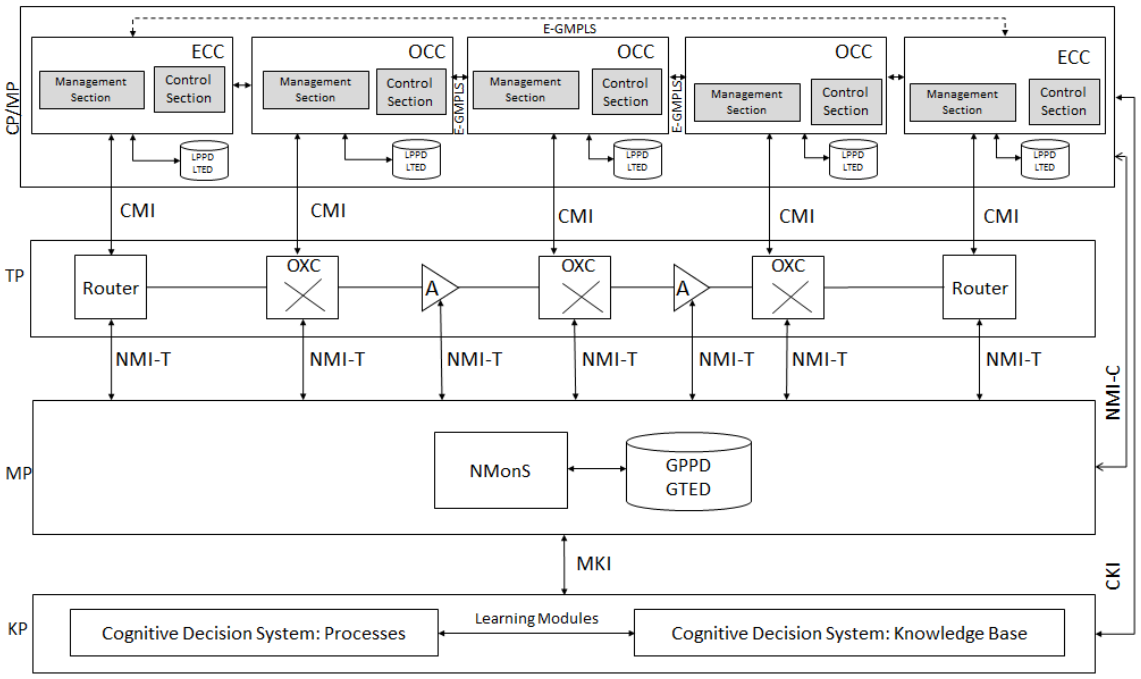

Figure 2.3 CHRON layered view centralized architecture [9]

The content of Figure 2.3 is explained throughout the explanation of the different planes: KP, CMP and PHY (equivalent to TP). However, the meaning of the different acronyms is presented below (further information in [9]):

A – Amplifier
CMI – Connection controller and Management Interface
CP – Control Plane
GPPD – General Physical Parameter Database
GTED – Global Traffic Engineering Database
KP – Knowledge Plane
LPPD – Local Physical Parameter Database
LTED – Local Traffic Engineering Database
MIB – Management Information Base
MKI – Management-Knowledge planes Interface
MP – Management Plane
NMI-C – Network Management Interface with Control plane
NMI-T – Network Management Interface with Transport plane
NMonS – Network Monitoring System

OCC – Optical Connection Controller
OXC – Optical Cross-Connect
TP – Transport Plane

Once presented the CHRON network architecture it is time to delve into the different systems that form the previous planes.

2.2.1.1. Knowledge Plane (KP)

As far as the Knowledge Plane is concerned, it is practically composed by the Cognitive Decision System (CDS) and its Knowledge Bases (KBs), which are also called CDS databases (CDS-db) (red boxes in Figure 2.2). The complete architecture of the CDS in the CHRON centralized network scenario consists of three parallel layers: Cognitive Processes, Learning Modules and Knowledge Bases, with the latter two layers composing the Knowledge Engineering Subsystem (Figure 2.4).

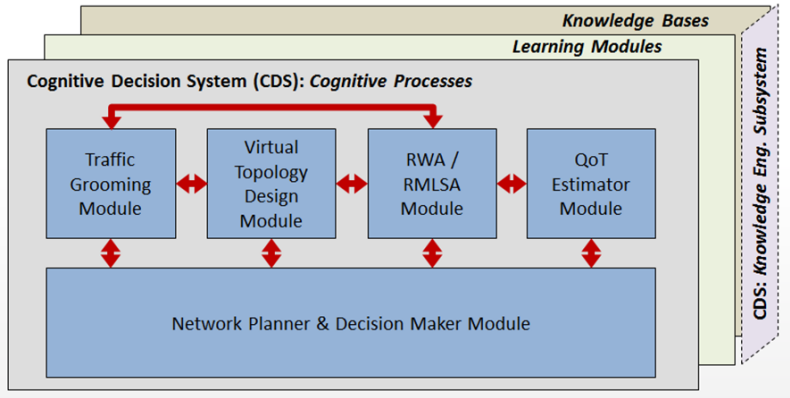

Figure 2.4 CHRON Cognitive Decision System architecture in the centralized scenario [2]

As shown in figure above (Figure 2.4), the CDS presents a modular structure divided into five cognitive modules that are briefly explained below.

- **Traffic Grooming (TG) module:** It is in charge of routing non-optical traffic demands (e.g., TDM label-switched paths (LSPs) through existing lightpaths composing the virtual topology).
- **Virtual Topology Design (VTD) module**: It is in charge of (re)designing the virtual topology (i.e., the set of lightpaths) to be established in the network. This module is mainly used for optimizing network performance by rearranging existing connections.
- **RWA/RMLSA module:** In networks following the ITU-T grid, it solves the Routing and Wavelength Assignment (RWA) problem (as well as determining the modulation level). In elastic networks, where channels do not necessarily comply with the ITU-T grid, it solves the Routing, Modulation Level and Spectrum Allocation (RMLSA) problem.
- **QoT Estimator module:** It makes a prediction of the quality of transmission (QoT) of new lightpaths to be established in the network (as well as the impact on existing connections when undertaking a new one). Thus, the establishment of impairment-aware optical connections relies on this module. It should be noticed that this module provides estimates (which are useful to discard connections where a low QoT is expected or that would disrupt existing ones),

but once a new lightpath is established, the QoT is verified, and the result of this verification will be used to improve the behaviour of the module for future QoT estimations.

- **Network Planner and Decision Maker (NPDM) module:** This module receives user requests and handles them. It is in charge, for instance, of deciding whether a non-optical traffic demand should be directly routed on the virtual topology or a new lightpath should be established (and which parameters, e.g., bitrate, should employ). It also determines whether the virtual topology or the spectrum allocated to connections should be optimized. In order to solve those tasks it coordinates the operation of the other modules relying on their results. The network planner communicates the actions to be performed to the network nodes through control plane protocols and handles the information received from the network monitoring system.

Every module of the presented below has an associated knowledge base (Figure 2.5), which is linked to the cognitive process itself by means of a learning module. These knowledge bases are located in the control node within the database named CDS-db (Figure 2.2).

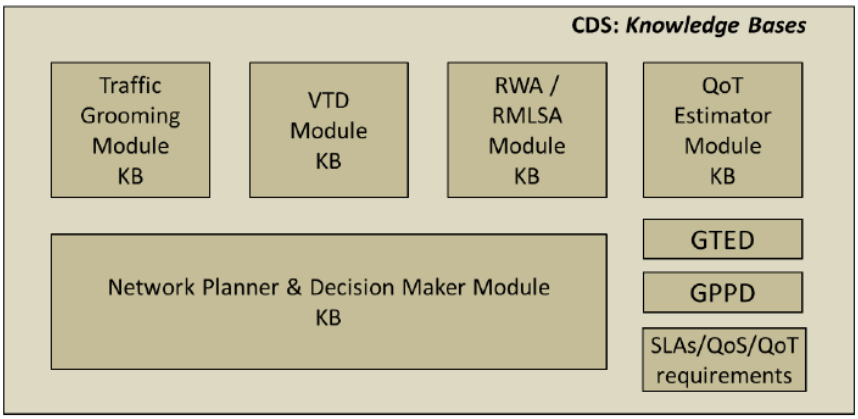

Figure 2.5 Knowledge bases in the CHRON centralized architecture [2]

In addition to the knowledge bases related to cognition, the CDS needs to access to other generic databases. Those contain information associated with the SLAs (Service Level Agreements)/QoS/QoT requirements, GTED and GPPD. According to Figure 2.2, the information related to the two former issues, requirements and traffic status is stored in the NMonS-db and that information regarding physical topology of the network is stored in the TC-db.

2.2.1.2. Control and Management Plane (CMP)

The CMP is in turn composed by 3 systems, the Communication System (CS), the GMPLS Protocols System (GMPLS PS) and the Network Monitoring System (NMonS).

In the first place, the CS is in charge of granting the correct message exchange between either modules of the same node (intra-node communication) or normal and control nodes modules (inter-node communication) (grey boxes in Figure 2.2). This system is composed by three modules.

- **COMM Master:** This module manages on the Control Node side the communications occurring between CDS and other local modules, and the messages exchanges between control node and normal node.
- **COMM Agent:** It manages locally on each node the communications with the control node. It does not permit the communication between normal node local modules, as this task is undertaken by GMPLS PS.
- **Hardware Interface:** This module enables the communication between PHY Manager and both the GMPLS PS (to receive devices configuration requests and report if the operation was successful or not) and the NMonS (processing the request for monitoring information and related answers).

Secondly, the GMPLS PS is the system responsible for retrieving and sharing the information regarding nodes resources availability and configuration, and for supporting the transport and execution of the CDS instructions among the involved nodes (green boxes in Figure 2.2). This system is composed by the following modules.

- **Topology Server:** This module, which is running on the control node, is in charge of coordinating the retrieval of the updated information regarding the current status of the network, in terms of resource availability and network configuration. In addition, it may ask OSPF-TE modules running on each normal node in the network for updates regarding network status.
- **Topology and Configuration database (TC-db):** This database, located in the control node, is used by the Topology Server to store its retrieved information about network configuration and available resources. It can be accessed by CDS to have updated information when making decisions.
- **TE Agent:** This modules works as an interface between CDS/external users and GMPLS PS. It handles incoming requests and takes adequate actions consequently (forwarding the request to CDS or triggering RSVP-TE operations according to the received instructions). After the feedback received from the RSVP-TE module, the TE-Agent may produce a response to be sent to the original requester or notify it of errors occurred during RSVP-TE activity.
- **RSVP-TE:** An instance of this module is running on each normal node to implement the signalling protocol (by sending the appropriate messages to all the affected nodes according to the RSVP-TE protocol) while following the instructions received at source node from the TE-Agent module.
- **OSPF-TE:** An instance of this module is running on each normal node to implement the OSPF-TE protocol. Since, the CDS is in charge of evaluating the route for the new LSP, the OSPF-TE module undertakes the task of sharing/disseminating the information regarding the resource availability and network configuration. It may reply to the Topology Server after an update request or directly notify it after a configuration change.
- **CAC/RM:** This module, running on each normal node in the network, keeps track of the changes in resources availability and network configuration. It is queried by the RSVP-TE module to verify the local resources availability and to allocate/free physical resources. Thus, CAC/RM translates the RSVP-TE requests in instructions for the PHY manager. In case of success, it notifies the RSVP-TE and an update is sent to the OSPF-TE to disseminate the configuration change within the network.

In the third place, the NMonS is the system liable for interacting with the physical layer to retrieve the data coming from the monitoring devices and to deliver them to the CDS to be taken into consideration when taking over evaluations (purple boxes in Figure 2.2). This system is composed by three modules.

- **NMonS Server:** This module, running on the control node, configures the NMonS Agent running on each normal node by following the instructions of the CDS and taking into account the information in NMonS-db, which it updates.
- **NMonS database (NMonS-db):** This database is managed by the NMonS Server and it stores all the information regarding monitored parameters collected from the normal nodes. It can be accessed by the CDS to get the information needed when making decisions.
- **NMonS Agent:** This module, running on normal nodes, receives queries from the NMonS Server and gets the information from the physical devices. It also acts as a bridge by passing the information traps that goes from the PHY Manager to the NMonS Server/CDS.

2.2.1.3. Physical Plane (PHY)

This plane covers both the physical and the data layers. It consists of a set of physical devices in charge of granting data plane functionalities and monitoring some specific parameters which are considered relevant in order to evaluate the services provided.

The main element of this system is the PHY Manager module, which works as an interface between GMPLS PS and NMonS on CMP side, and devices and monitors on PHY side. The operation of this module can be summarized as follows. On the one hand, the CAC/RM module may ask the PHY Manager, via the Hardware Interface module, to configure a concrete physical device and the former module will wait for the answer of the latter. On the other hand, the NMonS Agent module may ask the PHY Manager, via the Hardware Interface module, to query a given monitoring parameter and wait for the answer.

2.3 CHRON Control and Management Plane

After presenting the different planes that form the CHRON network architecture it is noteworthy to mention that the Control and Management Plane (CMP) is the focal point of this Master of Research Thesis. The other planes have been presented to facilitate the understanding as well as to have a general overview of the CHRON architecture.

To start with the CMP activities (Figure 2.6), the OSPF-TE module (M10) implements the OSPF-TE protocol [10], which although generally carries out the routing, in case of CHRON all the decisions concerning routing are made by the CDS (M1), and a lightweight implementation of OSPF-TE has been considered. Thus, this module is basically used as a mean to disseminate the resources availability of the nodes and the links, and the configuration parameters of the devices inside the nodes. This information allows the Topology Server module (M2) to build up a database (C) containing the description of the current status of the network.

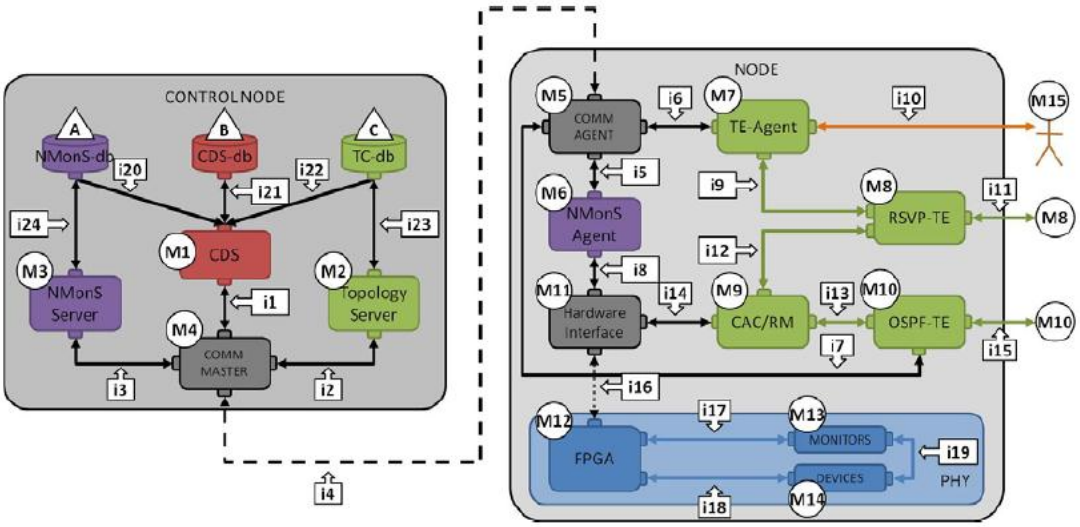

Figure 2.6. CHRON network architecture (functional blocks) [11]

The Topology and Configuration database (C) together with the NMonS-db (A) constitute the information sources for the CDS. The latter takes part of the monitoring system where in each node of the network the NMonS Agent module (M6) collects information about monitored parameters. This agent also queries the physical layer and, once the information is received, it formats and sends it to the control node. The NMonS Server module (M3) retrieves all the information sent by the NMonS agents and stores it in the before mentioned database. Moreover, the CDS module may receive alarms/traps from the NMonS Agents when changes occur at the physical layer.

The OSPF-TE module and the NMonS Agent communicate respectively with the Topology Server and the NMonS Server via the Communication System (CS). Due to this system was presented in the previous section, at this point the focus is on the communication mechanisms. In the CHRON architecture, all the communications between a COMM module and another module (external to the CS) are made via XML files, whilst the communications between two COMM modules may be performed by

encapsulating the XML messages in a SOAP packages or by using the SNMP protocol [11].

Taking everything into account, for each lightpath request the TE-Agent module of each node dispatches it to the CDS through the CS. The CDS performs its choice/decision and provides the information about the lightpath to be reserved. An XML description comprising the transmitter and the receiver information, as well as the switching patterns at the intermediate Optical Cross Connects (OXCs) is sent via the CS to the source node of the connection request. Then, the TE Agent module triggers the reservation process sending the command to the RSVP-TE module (M8).

The RSVP-TE module implements the RSVP-TE protocol [12] and is in charge of the signalling process, i.e. the process of exchanging messages to set up, modify or terminate a lightpath. Furthermore, for all the operations related to local resources the RSVP-TE module exchanges messages with the CAC/RM module (M9), which in turn translates the requests into instructions to be sent down to the physical layer through the Hardware Interface module (M11). This Hardware Interface module may receive requests from the NMonS Agent or from the CAC/RM, and then it will translate them into instructions for the PHY Manager that control all the physical devices. Once the devices reply, the Hardware Interface redirects the information to the requesting modules. Thus when the reservation phase ends, the source node of the connection request sends an acknowledgement to the CDS via the CS, and in turn, the CDS informs the source of the connection to start the transmission. The changes made during the reservation process by the CAC/RM are also notified to the OSPF-TE module that disseminates them within the network.

2.3.1. CHRON Cognitive Specification Language

In the CHRON framework the Cognitive Specification Language (CSL) acts as an interface between the Cognitive Decision System (CDS) and the CMP (Control and Management Plane). Thus, another task of this Master of Research Thesis has been the definition of a CSL based on XML to encode messages from the CDS to the CMP and vice versa. Those messages are related to traffic requests, monitoring information and decisions made by the CDS.

In addition to the interface role, the CSL is also valid for other encoding purposes within the CHRON architecture such as:

- *Network description*: The CSL provides a formal way of defining the network topology and its physical elements and features.
- *Services and QoS/QoR requirements*: The CSL also provides a formal way of relating services to their associated Quality of Service (QoS) and Quality of Resilience (QoR) requirements.
- *Lightpath and non-optical LSP requests*: The CSL is used to define the requests from the CMP module to the CDS to deal with lightpath and non-optical LSP requests. These requests may also include the requirements in terms of QoS/QoR that we have just mentioned in the previous item.

- *CDS decisions and network commands*: The CSL is used to encode the decisions made by the CDS in order to be communicated to the CMP, either directly or translated into network commands.
- *Network monitoring requests and data*: The CSL also specifies how to encode both requests and responses regarding optical performance monitoring and traffic monitoring data.

In this work, only the last point is considered due to it is strongly related to the control and management approaches. The XML messages defined for this communication are exposed in Annex 1 (Optical Performance Monitoring) and Annex 2 (Traffic Monitoring).

2.3.2. Feedback System between NMonS and CDS

Apart from the encoding procedures presented in the previous epigraph, another important problem tackled in this Thesis has been the definition of the way in which the Cognitive Decision System (CDS) is able to perceive current conditions by means of the Network Monitoring System (NMonS). At this point, like an introduction of the information presented later in Chapter IV, a way to encode the monitoring information has been defined, and then an exhaustive analysis of the tentative protocols required for transporting the information has been undertaken. In particular, three alternatives will be considered in the mentioned analysis: SNMP, XML-based management approach and a hybrid approach to support migration between both scenarios.

2.3.3. The DRAGON Emulator

In addition to all the points treated in this chapter, all the research results concerning the control and management approaches will be proved under the functionality of the DRAGON (Dynamic Resource Allocation via GMPLS Optical Networks) emulator [3], which apart from emulating the control plane, it has been modified to realized management tasks based on the SNMP approach such as it has exposed in Chapter IV.

2.4 Conclusions

This chapter has been devoted to presenting the CHRON network architecture in order to understand the remaining chapters. Special attention has been paid to the CHRON centralized network architecture since this is the type of architecture considered when emulating a real network.

Then, the focal point has been the Control and Management Plane (CMP) defined in the CHRON architecture as over this plane rest all the research results and emulation tests undertaken in this Master of Research Thesis. Thus, once presented this plane, the content presented later will be totally identified together with the terminology employed.

Finally, innovative aspects and challenges turn around this work. In particular, the DRAGON emulation scenario (testbed) permits undertaking tests in the most similar way to the operation in a real network. In case of CHRON project, an emulation stage will take place prior to putting in practice the exploitation and standardization results carried out throughout the project lifetime. Hence, the work developed and reflected in this contribution has an added value since it focuses on the previous steps to the commercial application of the research results.

Chapter III – Control and Management Approaches in CHRON

3 Control and Management Approaches in CHRON

In this third chapter the main results of the state of the art study followed at the initial stage of this work is presented to both complete and improve this contribution. In the first place, the evolution from the MPLS (Multi-Protocol Label Switching) to GMPLS is presented by dealing with MPLS, MPλS and GMPLS control grounds. Then, a study of the SNMP management approach is exposed by putting forward with a general overview, the structure of SNMP messages, the information management and the definition of the SNMP use cases considered in CHRON. Finally, a similar presentation of the XML management approach completes this chapter. Moreover, a comparison between both management approaches and the presentation of hybrid management approaches provide important research results.

3.1 Control Approach based on GMPLS

New telecommunications services arisen in the last years as well as the increase on bandwidth demand have awakened an important interest in optical networks. This networks structure consists of three transport technologies: the synchronous optical network, SONET (Synchronous Optical NETwork); the synchronous digital hierarchy, SDH (Synchronous Digital Hierarchy) and the wavelength division multiplexing, WDM (Wavelength Division Multiplexing).

The main reason for the bandwidth needs increase has been the exponential growth of users and Internet applications. Nowadays, the data traffic overcomes the traditional voice traffic, and this fact has launched the Internet Protocol (IP) to support a wide variety of services. Thus, the convergence between the IP layer and the optical layer is totally understandable.

These days there exist several under-study mechanisms designed to transport the IP data traffic over WDM or DWDM (Dense WDM) with the aim of reducing the overflow of the intermediate adaptive layers, generally form by ATM (Asynchronous Transfer Mode) and SONET/SDH. Notwithstanding, as it is not possible the substitution of the whole infrastructure in a short period of time, it is much more important to achieve a mechanism that integrates the control of all the layers in a heterogeneous network architecture. The current developments in terms of standardization revolve around the GMPLS protocols, so they will be the focal point of this Thesis.

3.1.1. MPLS

In the last years of the 90's, concretely in 1997, the IETF (Internet Engineering Task Force) set up the MPLS working group with the aim of standardising and unifying the level 2 (link) switching solutions. The first and main result was the definition of the well-known protocol MPLS in 1998 [13].

MPLS provides traffic engineering benefits in the IP model over ATM, but moreover, other advantages such as simpler design and operation and better scalability of the network. Furthermore, unlike the other level 2 switching solutions, MPLS has been designed to operate over different technologies and not only ATM. Therefore, this fact eases the migration process to the last generation optical networks.

The initial objective of MPLS was to offer some features of the connection oriented networks to the non-oriented connection networks, thus enabling any type of service over the same IP network.

In the non-oriented connection traditional IP routing, the destination address, together with other parameters of the header, is examined each time the packet traverses a router, which supposes a variable processing time in function of the routing table size. Moreover, due to the path cannot be predicted it is difficult to reserve resources that guarantee the quality of service demanded by the different applications. Seeing that, MPLS combines the advantages of the routing in level 3 (network) and the fast switching in level 2 (link). To achieve that, it uses a small and fixed-length label that provides a better performance in the IP packets transport. Such a label is assigned to the packet on the basis of the destination address, the type of services parameters, the corresponding virtual network or other criteria.

To sum up, MPLS has been designed to work with the layer 2 and layer 3 protocols, so it is also named as “the technology of the layer 2.5”. Thus, MPLS itself does not represent a new layer, but an association of the control plane (within layer 3) with the data plane (within layer 2) [14]. This is the key factor of MPLS since the separation of the control and data layer permits an independent development.

As far as the information transmission is concerned, MPLS uses the LSP (Label Switched Path) establishment in layer 2, which support the routing and signalling protocols of layer 3. The standard MPLS permits the utilization of IPv4 and IPv6 over the main packet-switching technologies of layer 2, such as Gigabit Ethernet, ATM and Frame Relay.

3.1.1.1. MPLS Architecture and Network Elements

The following points reflect the main components of the MPLS architecture. Although some of them have been already introduced they are presented again in this glossary.

- **Label**

  The label is a 20 bits field that establishes the correspondence between the traffic and a specific FEC (Forwarding Equivalence Class). The label is transported in the MPLS header, and in turn, it identifies the path to be followed.

- **Label Switched Router (LSR)**
  This device integrates the main routing and switching functions. MPLS uses the technique known as the label switching to send the information through the network.
  In a MPLS network there are two types of LSR:

- *Label Edge Router (LER)*: These routers are located at the end points of the MPLS network and they link with the traditional networks (Ethernet, Frame Relay, ATM…).
- *Intermediate LSR*: These routers are located within the MPLS network, and receive and transmit the labelled packets by the corresponding links.

The LSRs, in the same way as the traditional routers, exchange information concerning the network topology by means of standard routing protocols, through which they build the routing tables mainly based on the reachability of the destination nodes. With the information stored in these tables, the LSPs are established, although these can be also established without following the results of the routing protocols.

Regarding the LSR operation, this component undertakes three basic operations in relation to the MPLS labels: push, pop and swap. Each label is inserted in the packet by the first LSR of the MPLS network. Then, each node routes this packet according to the label and the input interface by replacing (mapping) the label according its routing information. Thus, the packet leaves the node with a different label, verifying in this way, the local use of the MPLS labels. Once established the LSP, the information can be routed without being analysed in each hop.

When a packet ingresses in a MPLS network it is examined to determine the LSP through which it will be routed. This decision is totally local and based on factors such as the destination address, the QoS requirements and the current status of the network. There is another alternative that supports the forwarding of the packets and the LSP creation, the Forwarding Equivalence Class (FEC).

- **Forwarding Equivalence Class (FEC)**

  This element defines the set of packets that are sent through the same LSP. They are treated in the same way in terms of routing even if the destinations are different.

- **MPLS Header**

  The generic MPLS header is a 32 bits field that is added to the packet between the level 2 and 3 headers, and it defines a number of characteristics and requirements for the transmission in the MPLS network (Figure 3.1). It consists of the different sub-fields:

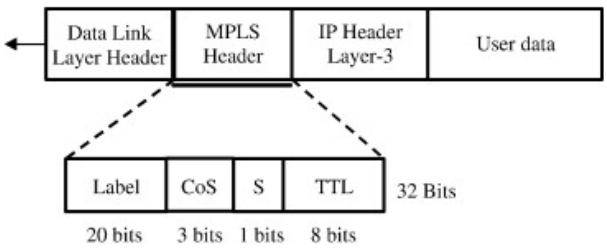

Figure 3.1 Generic MPLS header [14]

- *Label*: MPLS label value.
- *Exp/CoS*: It is sometimes named Class of Service (CoS). It marks the different types of traffic to achieve a better performance. Depending on the type of traffic to be transmitted, different characteristics and requirements will be taken into consideration.
- *Stack (S)*: It is also called Bottom of Stack (BoS). With this bit it is supported the labels hierarchy (label stacking). A positive value indicates that there is more than one MPLS header in the same packet. The MPLS headers behave as if they were stacking one over another, so that a router first processes the one positioned above.
- *Time To Live (TTL):* it indicates the lifetime of a packet. The objective is to avoid possible loops within the network by decreasing the initial value each time a hop takes place. When it reaches a null value the router in question will eliminate this packet.

- **Label Switched Path (LSP)**

  The Label Switched Path (LSP) is a specific traffic path through a MPLS network. The initial end point is the Ingress LSR, which is in charge of classifying and labelling the packets. At the final end point the Egress LSR is located, which eliminates the corresponding label and forwards the packet outside the MPLS network (Figure 3.2).

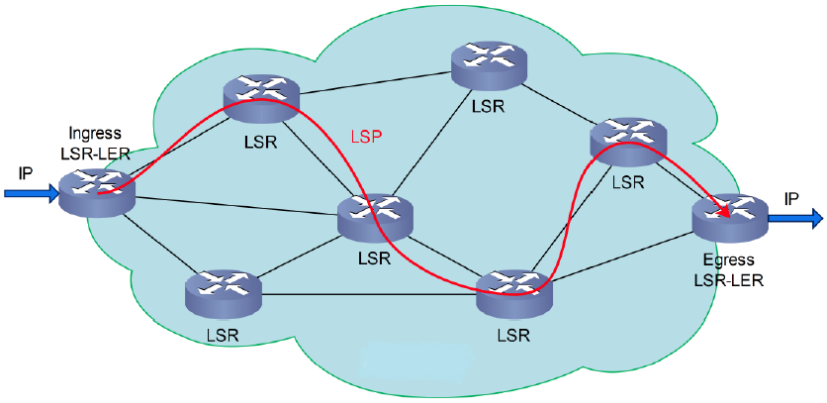

Figure 3.2 Example of MPLS network

- **Label Information Base (LIB)**

  This database is the routing table of one LSR. Each input in this table has a pair input interface and input label, which matches with another pair formed by the output label and the output interface.

3.1.1.2. Basic Transmission Operation

At the moment in which a packet arrives at the MPLS network, after being examined by the Ingress LSR, it is assigned with a FEC according to the destination IP address and the requested QoS. Moreover, the MPLS header is added to identify the transmission LSP. Thus, the packet will not be examined by any of the intermediate LSRs. Only, the intermediate nodes will identify the input label, consult the LIB and replace such a label by the associated output label. Finally, the packet will reach the Egress LSR, which

will eliminate the MPLS header and forward it by conventional routing while taking into account the destination IP address.

3.1.1.3. Advantages over the Conventional Routing

In the traditional IP routing, each time a packet arrives at a router the destination address and other parameters in the header are examined. The criterion followed when making this routing decision is to choose the route defined by the Longest Prefix Match technique, which chooses first the highest subnet mask. This action results both complex and time demanding, and it is up to the routing table size of each router. However, in the MPLS scenario the choice of paths is based on a simple label that is added to every packet that arrives at the network. Thus, only the LERs that connect a traditional network with the MPLS network examine the IP header. The treatment of the packets by the LSRs results much simpler and less time demanding than the Longest Prefix Match approach, which supposes a better performance in respect to the traditional IP routing.

In a word, the main advantages offered by MPLS are exposed hereinafter [15]:

- MPLS permits specifying mechanisms for the administration of different types of traffic flows.
- This protocol covers the link and network layer protocols. In addition, it offers an interface to the routing and signalling protocols.
- MPLS has at its disposal different mechanisms to translate the IP addresses into simple and fixed-length labels, which are used in forwarding and switching technologies.
- MPLS supports the layer 2 protocols traditionally used in IP. Moreover, it correctly operates over ATM and Frame Relay scenarios, which present similar transport and switching mechanisms.
- It allows to create Virtual Private Networks (VPN) both at level 2 (link) and level 3 (network).
- The MPLS protocol allows Traffic Engineering (TE).

3.1.2. MPλS

The first attempt to extend MPLS to operate over an optical network was MPλS (Multi-Protocol Lambda Switching). In this protocol the wavelength is used as a label. Although MPλS does not have all the features of MPLS, there is a remarkable correspondence between the LSRs and the wavelength switches, and between an LSP and an optical path (ligthpath). Thus, a wavelength switch is responsible for switching the wavelengths from the input to the output. In this optical scenario, the photonic switches also need information from the routing protocols, such as OSPF or IS-IS, through which they exchange information regarding the network topology links status and the optical resources availability. Likewise, the photonic switches need signalling protocols, such as RSVP or LDP (Label Distribution Protocol) to undertake the establishment of the optical path.

MPλS is an important advance in the control plane technology applied to the optical layer by the use of Optical Cross Connects (OXCs). The main purpose of this protocol is

to provide real time provisioning in optical networks and use uniform semantics for the control and management in hybrid networks formed by OXCs and LSRs. In conclusion, MPλS is a combination of the MPLS-TE control plane and the OXC technology.

In spite of the information above, MPλS only provides a solution for the wavelength switching scenario, so the other switching types are out of its reach. In this way, a new solution to cover all the switching types in a generalized manner arose; this is GMPLS (Generalized Multi-Protocol Label Switching).

3.1.3. GMPLS Fundamentals

The introduction of this protocol in optical and IP networks has produced a fast and simple spread of Internet services [16, 17, 18]. Not only does GMPLS cover packets switching but also wavelength and optical fiber switching. Therefore, besides IP routers and ATM switches, GMPLS may be present in other switching devices such as DXC (Digital Cross Connect), OXC (Optical Cross Connect) and PXC (Photonic Cross Connect). With this aim, GMPLS extends the basic functions of the traditional MPLS and in some cases it adds new functionalities. These adaptive procedures have supposed several extensions in the creation of generalized labels, the routing and signalling protocols, the traffic engineering characteristics, the protection and the restoration of the links, etc.

As for the data flow, the flow through an optical fiber can be split into different wavelengths, each one can transport signals previously multiplexed in time and with different bandwidths. Hence, it is necessary that the LSR source specifies the coding type in regard to the data flow, the switching type to be applied to the corresponding LSP and the total bandwidth required by such a LSP.

The concept of generalized label presented before consists of an extension of the fixed length label in MPLS. This label informs of the time slot number, the wavelength identifier or the fiber number within a set of fibers in the same wire. Contrary to MPLS, in the case of GMPLS the labels, besides representing the type of traffic, represent the network resources.

The initial MPLS architecture has been extended to include a new set of interfaces in the LSRs:

- *PSC (Packet Switch Capable) interfaces*: These interfaces recognize the packets limits and they can send data by paying attention to the packet header.
- *L2SC (Layer-2 Switch Capable) interfaces*: These interfaces recognize the frames/cells limits and they can send data according to the header content of these information units.
- *TDM interfaces*: They route the data in function of the time slot within a repetition cycle.
- *LSC (Lambda Switch Capable) interfaces*: These interfaces route the data on the basis of the wavelength in which the data are received.
- *FSC (Fiber Switch Capable) interfaces:* The routing is undertaken in accordance with the fiber identifier through which the data pass.

A circuit can be only established among interfaces of the same type. These circuits, regardless of the type of the interfaces, are named LSPs as it occurs in MPLS. All the components explained in the MPLS section are also valid in GMPLS and the main novelties are presented below:

- **Separation between data plane and control plane**

  Like in the MPLS case, there is also a separation between the data plane and the control plane, but while in MPLS this is a logical separation, in GMPLS besides logical it may be physical by even transmitting the control traffic through a different network.

- **Generalized label**

  As it has been mentioned before, GMPLS is able to support a wide variety of switching devices, so that is the reason why the generalized label has been created. This new concept may be summarized by stating that a unique label can represent a packet, a cell/frame, a time slot, a wavelength or a fiber. In this way, the generalized label makes reference to the network resources and its format and content depend on the switching type, in other words, in GMPLS the header length is variable and depends on the interfaces encoding.

- **Bidirectional LSP**

  In MPLS all the LSPs are unidirectional, but in GMPLS the concept of bidirectional LSPs has been introduced. Thus, GMPLS offers the possibility of establishing LSPs in which both the Ingress LSR and the Egress LSR may be source or destination in the communication process. In this situation both communication flows must have the same characteristics and TE parameters. The bidirectional LSPs are established through a simultaneous label distribution in both directions by saving time in the signalling process.

- **Forwarding Adjacency (FA)**

  This feature enables to establish LSPs of a superior level by undertaking intermediate switching changes. In this case, a concrete LSP is considered as a virtual link with its own traffic engineering characteristics. In the first place, this virtual link, called FA-LSP, is created and all the nodes are notified of this action. Then, each LSR may use this FA-LSP as though it was an additional link to establish other LSPs of a superior level, without any management concern.

  In the superior level of this hierarchy are the FSC interfaces, followed by LSC, TDM and PSC. Thus, a LSP that starts and ends in a PSC interface may be included in another LSP that uses the TDM interface. This latter, may be in turn included in another LSP among LSC interfaces, and finally, this last one may be included in a LSP of FSC interfaces (Figure 3.3).

- **Link Bundling**

  In a complex optical network, the LSRs are connected by several parallel optical fibers, so any change in a concrete wavelength within a fiber has to be advertised in a separated form such as for the LSP establishment process (Figure 3.3). This process increases the signalling traffic and a possible solution is the link bundling. When two LSRs are connected through multiple links it is

possible to announce several of these links as one only link by means of the routing protocol. The purpose of the Link Bundling approach is to improve the routing scalability and to reduce the information exchanged.

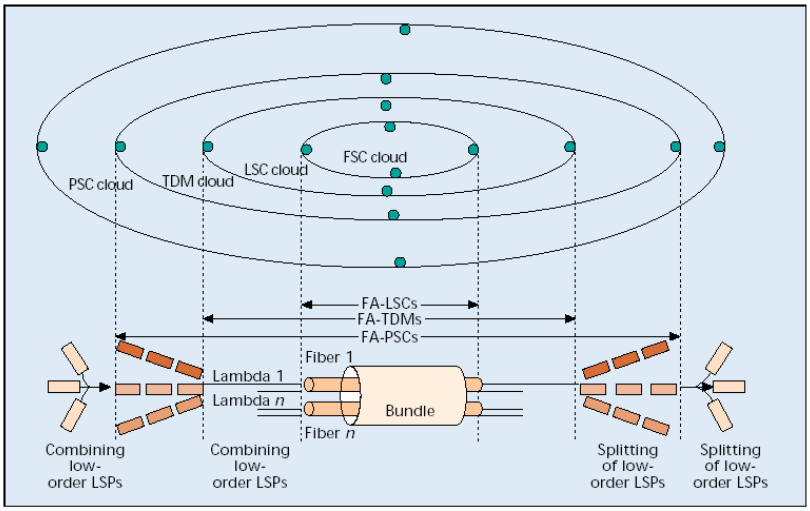

Figure 3.3 Forwarding Adjacency and Link Bundling

The bundling process can be realized when meeting the following conditions:

- All the links must have the same source and destination LSRs.
- All the links must have the same switching type.
- All the links must have the same TE characteristics.

- **Inter-area LSP**

  The traffic engineering mechanisms over MPLS/GMPLS networks were originally defined for intra-domain networks, that is, limited to one only area or domain. Nowadays, the alliances among NSPs (Network Service Providers) have produced an extension of these mechanisms to offer QoS guarantees between nodes belonging to different domains connected by the ABRs (Area Border Router). To achieve further information in this respect, since this Master of Research Thesis focuses in intra-domain networks, the IETF has defined a number of extensions for MPLS and GMPLS that enables to establish inter-area LPSs, which traverse several domains with strict QoS and TE parameters [18].

- **Network administration**

  The service providers monitor, configure and control the resources of their networks that generally are disseminated over extensive geographic domains. With the aim of facilitating the administration tasks in a network, protocols such as SNMP are used for undertaking such administrative procedures. This protocol, strongly related to GMPLS, will be studied later in detail by emphasizing on the operation, administration, maintenance and provisioning functionalities.

- **Security**

  By and large, once the LSP is established in a GMPLS environment, it can carry a great deal of data and consume important network resources. Thus, the security mechanisms are necessary to prevent the network from undesired

attacks and inappropriate use of the resources. The GMPLS control plane includes mechanisms that avoid these risks by providing authentication and confidentiality.

3.1.4. GMPLS Protocols

In this sub-section both the routing and signalling protocols that form the MPLS/GMPLS control plane are presented like continuation of the information presented in the previous sub-sections.

3.1.4.1. Routing protocols

The routing protocols are in charge of disseminating the network topology and the TE characteristics. Moreover, they are responsible for working out the routes between two points in the network under specific conditions for the transmission of concrete data traffic. To realize this computation, it is necessary to know the topology and the available resources in the network at a moment. There are two types of protocols in function of the knowledge of the network on the LSRs side:

- *Distance vector protocols:* In this type of routing protocols, each router communicates with its neighbours to exchange the routing tables. This process is repeated until the tables have stable information. The path is chosen according the number of hops towards the destination. These protocols have the setback of being rather slow, although they are easy to use and quite suitable for networks with a few nodes [19].
- *Link state protocols:* In this case, each router owns up to date information regarding the network topology and status as a whole. To achieve this, each node has stored a complete map of the network, and when facing with any change in the topology, the routing protocol must act quickly to disseminate the new information throughout the entire network. Due to the total information, Link State Database (LSD), each router is able to compute the following hop to each possible destination [19].

The GMPLS protocol is formed by link state protocols for the routing purposes, such as OSPF-TE (Open Shortest Path First-Traffic Engineering) [20] and IS-IS-TE (Intermediate System to Intermediate System-Traffic Engineering), which have been standardised by the IETF [21]. This work emphasizes on OSPF-TE since this is the routing protocol used in the DRAGON emulator environment.

OSPF-TE runs directly over the IP network layer and uses the LSAs (Link State Advertisement) as communication units to get/exchange the information about the network and build the LSD. The LSAs are messages that the LSRs send one another by flooding mechanism concerning the shortest path to each node in its area.

The OSPF-TE protocol counts on three mechanisms for carrying out the communication process [20]:

- *Hello:* This sub-protocol consists in a message sent periodically to check that links are on.
- *Exchange:* By means of this sub-protocol a router exchanges its database content with its neighbours.

- *Flooding:* This mechanism is used for advertising any change to the other routers. This flooding sub-protocol is the responsible for the distribution and synchronization of the database status in each router. For instance, if a link failure occurs, the router that realizes this fact will send a broadcast message for all the nodes to update their databases.

Regarding the communication units previously presented, LSAs, there are different types defined in OSPF-TE [20]:

- *Router LSA:* It contains the status and the cost of all the point-to-point links that ends in a router.
- *Network LSA:* It consists of information about the network as a whole.
- *Summary LSA:* Its content is basically the reachable destinations within a domain.
- *External LSA:* It contains the routes learnt by other domains in a multi-area scenario.
- *Opaque LSA:* This LSA was standardised to include the TE characteristics in OSPF [22]. This LSA consists of a header of 24 bytes followed by the payload field that is formed by one or more TLVs (Type-Length-Value) nested to facilitate scalability. Each TLV consists of three fields, as its own name indicates, the type and length are 2 bytes fields and the value is a field that contains the TE information with a size indicated in the length field. There are different types of Opaque LSAs:
    - *Type 9:* Opaque LSA with only local links reach.
    - *Type 10:* Opaque LSA with only local area reach.
    - *Type 11:* Opaque LSA with AS (Autonomous System) reach.

As for the TLV communication units that form the LSAs, one of the types used in GMPLS is the *Link TLV* which in turn is composed of different sub-TLVs (Table 3.1).

Sub-TLV Type	Length (octets)	Content
1	1	Type of link
2	4	Link ID
3	4	Local interface IP address
4	4	Remote interface IP address
5	4	Traffic Engineering Metric
6	4	Maximum bandwidth
7	4	Maximum bandwidth to be reserved
8	32	Free bandwidth
9	4	Administrative group

Table 3.1 Sub-TLVs of Link TLV

To achieve the GMPLS functionalities some extensions have been done by creating new sub-TLVs (Table 3.2).

Sub-TLV Type	Length (octets)	Content
11	8	Local/remote link identifier
14	4	Link protection type
15	variable	Interface switching capacity descriptor
16	variable	Shared risk link group

Table 3.2 News sub-TLVs of Link TLV

In summary, the improvements that GMPLS includes in terms of the routing protocols are the following:

- Forwarding Adjacency (FA)
- Link Bundling
- Unnumbered Links
- Link protection type
- Shared Risk Link Group
- Interface switching capability descriptor

The two former were presented when introducing GMPLS, so at this point only the rest of them are briefly presented:

- **Unnumbered Links**

  The unnumbered links do not have an IP address. So, each end point needs a type of local identifier in regards with the LSR to which it belongs. Thus, the two end LSRs of an unnumbered link exchange the identifier that they assign to the link. This information is in the Type 11 sub-TLV of the Link TLV, which in turn belongs to an Opaque LSA.

- **Link protection type**

  According to the OSPF-TE extensions, the sub-TLV Link Protection Type has been created to indicate the protection type of a link. The path computation algorithm will take into account this information when computing both the primary and backup LSPs.

  Currently, there are six types of indicator defined for protection issues:

  - *Extra Traffic:* That means that a link is protecting to one or more links. When not used for protection purposes, the protection path can either be left idle or be used to carry low priority extra traffic.
  - *Not protected:* A link is not protected by any protection link.
  - *Shared:* There is one or more links of type Extra Traffic that are protecting one or more links at the same time.
  - *Dedicated 1:1:* In this type of protection there is a dedicated Extra Traffic link that protects to one only link.
  - *Dedicated 1+1:* In this case there is one dedicated link that protects to one link. The traffic is transmitted on both primary and backup LSPs.
  - *Enhanced:* This type of protection makes reference to a more reliable protection scheme [20].

- **Shared Risk Link Group (SRLG)**

  A number of links can be defined as shared risk link group if they share any resource whose failure may affect to all the links. A link can belong to multiple SRLGs. Thus, this information describes the list of SRLGs to which a link belongs. The SRLG is a type 16 sub-TLV within the Link TLV.

- **Interface switching capability descriptor**

  This component describes the switching capability of an interface since in GMPLS the interface can be of different types. This information goes in a type 15 sub-TLV within the Link TLV.

  This sub-TLV has a field called Switching Capability that may contain one of the values of Table 3.3.

Value	Content
1	Packet Switching Capability 1 (PSC-1)
2	Packet Switching Capability 2 (PSC-2)
3	Packet Switching Capability 3 (PSC-3)
4	Packet Switching Capability 4 (PSC-4)
51	Layer 2 Switching Capability (L2SC)
100	TDM Switching Capability
150	LSC Switching Capability
200	FSC Switching Capability

Table 3.3 Interface switching capability descriptors

Now, with all the information presented so far, the LSRs can build the Traffic Engineering Database (TED) that is the table with all the traffic engineering information of the network.

3.1.4.2. Signalling protocols

As starting point, the signalling protocols are in charge of distributing the labels among the LSRs, reserving the physical resources of a LSP to undertake the transmission, maintaining the links connectivity, and detecting and notifying errors for the LSP restoration.

The IETF defined two signalling protocols for MPLS, which have been later extended to be implemented in GMPLS. These protocols are RSVP-TE (Resource reSerVation Protocol - Traffic Engineering) [17] and CR-LDP (Constrained-Based Routing Label Distribution Protocol) [23].

This Master of Research Thesis will focus on the RSVP-TE protocol as it is the one used in the DRAGON emulator. This signalling protocol reserves resources for a specific data flow throughout a path between the source and destination nodes, which enables to guarantee the Quality of Service (QoS). RSVP-TE operates on the superior level of IP layer, such as a transport protocol. However, RSPV-TE does not transport application data, but only it sends signalling messages to carry out the resources reservation [17].

RSVP-TE defines seven types of messages: *Path*, *Resv*, *PathErr* (Path Error), *ResvErr* (Reserve Error), *PathTear*, *ResvTear* and *ResvConf* (Reserve Confirm). The two former messages, Path and Resv, are used to request and reserve a signalling session. Then, the messages PathTear and ResvTear are used to finalize a signalling session. PathErr and ResvErr are error notification messages, and finally, ResvConf is sent to confirm a reservation.

The following epigraphs contain those concepts and most relevant activities that feature the RSVP-TE protocol.

- **LSP Establishment**

The process to establish a LSP starts when the Ingress LSR sends a Path message to the Egress LSR, where the former notifies the latter of the type of both the LSP and the traffic to be transmitted (Figure 3.4). In this message other specific parameters are included such as route record, explicit route, coloured links, label request, label suggest, label restriction, etc. The Path message is sent according to the pre-defined route stored in the routing table of each node, which was the result of the routing protocol operation.

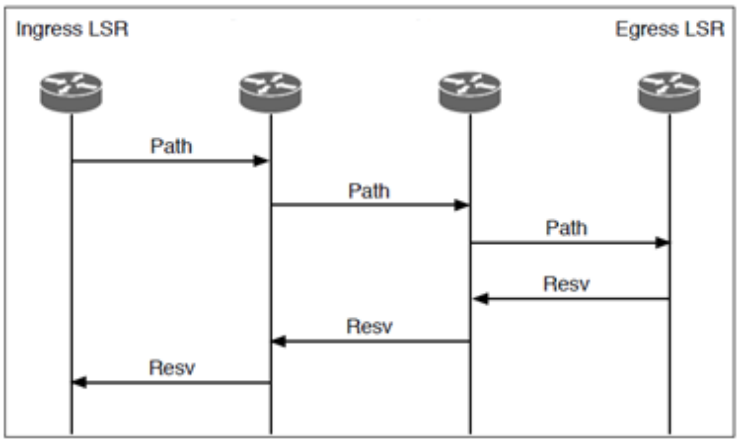

Figure 3.4 LSP establishment process in RSVP-TE

When the message arrives at the Egress LSR, it checks the resources needed for the transmission and sends back a Resv message with such resources information by following the same route. At the same time that this message traverses the intermediate nodes the indicated resources are reserved. Once the Resv message arrives at the Ingress LSR the LSP is considered as established.

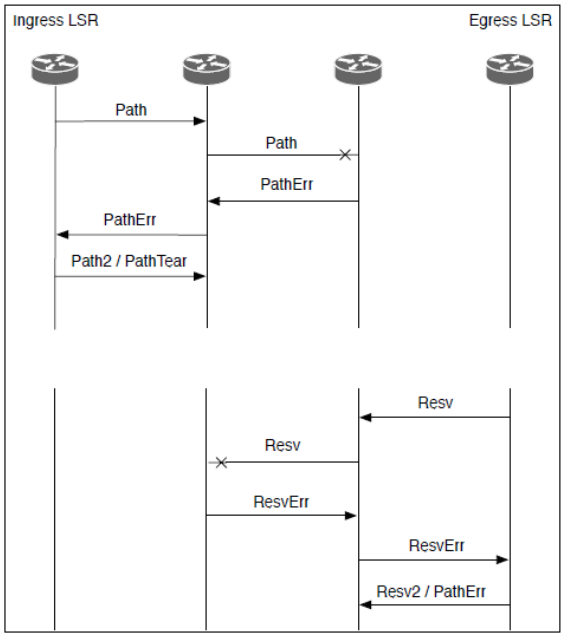

Figure 3.5 Errors management during LSP establishment in RSVP-TE

In case a node does not have available resources or cannot reserve the needed resources, it sends a PathErr message towards the Ingress LSR or a ResvErr message towards the Egress LSR indicating the failure reason. Thus, the LSP establishment process has to be rebooted from scratch by adopting the corresponding measures to avoid such a previous error (Figure 3.5).

- **New objects definition**

  In order to go ahead with the new RSVP-TE functionalities some new objects, which are integrated in the signalling messages, have been defined:

  - *Record Route Object (RRO):* A record of the nodes that a Path message traverses can be stored by including this object in such a message. This information arrives at the Ingress LSR in the Resv messages and can be used to detect loops and changes in the links during the establishment process or to create explicit routes later.
  - *Explicit Route Object (ERO):* It is possible to establish a LSP following a predetermined route. This is achieved by including this object in the Path message. The route may be either strict, where all the intermediate nodes are specified, or loose, where the intermediate nodes have its own choice capability.
  - *Session Attribute Object (SAO):* Another way to keep under control the route of a LSP is through the coloured links functionality. The links in a network may be of three different colours: exclude any, links that must not be used; include any, at least one link must be used; include all, all the links must be used. These restrictions act when the Session Attribute Object is included in the Path message.
  - *Generalized-Label request object (GL):* In the GMPLS architecture, the decision on assigning a label is made by the Egress LSR, as it has been explained before. When the GL object is included in the Path message the LSP to be established has to meet the requirements specified in this object (encoding type, switching type and data type) [17]. Below the structure and fields of this object are presented (Table **3.4**).

Encoding Type (8 bits)	Switching Type (8 bits)	G-PID (16 bits)

Table 3.4 GL object format

The Encoding Type field is the codification type of the LSP (Packet, Ethernet, SONET/SDH, Lambda, etc.). It represents LSP nature and not link nature through which the LSP passes.
The Switching Type indicates the switching that is being requested in a link (PSC, L2SC, TDM, LSC and FSC). This field is compulsory for links that have more than one switching capability.
The G-PID, Generalized PID, contains the type of traffic to be transmitted, such as Ethernet, Lambda, etc.

The objects explained hereinafter are used in those situations in which there is no label exchange among the nodes, in other words, a fixed label will be assigned throughout a specific route to transmit the data traffic.

- *Label Set Object (LS):* The presence of this object in the Path message permits to restrict the use of a concrete label for the LSP establishment. Each LSR can impose its restrictions to the following one and at the same time it has to respect the restrictions coming from the previous LSR. To do that, each LSR eliminates the labels that are not available when processing the Path message. When the message arrives at the Egress LSR it will assign to the LSP one of the labels contained in the LS object. If the LS object becomes idle during the Path message propagation the error notification mechanisms will enter in action (Figure 3.6).

  With the aim of putting the reader in the optical field context, the label restriction may be necessary in the following scenarios:

  - A node is only able to receive and transmit data in a small set of wavelengths.
  - A node does not have wavelength conversion capability and it is necessary to use the same wavelength in its links.
  - The wavelength conversion is limited to reduce the optical signal distortion.
  - The two end points of a link have only some wavelengths in common.
  - When the Wavelength Continuity Constraint (WCC) approach is active.

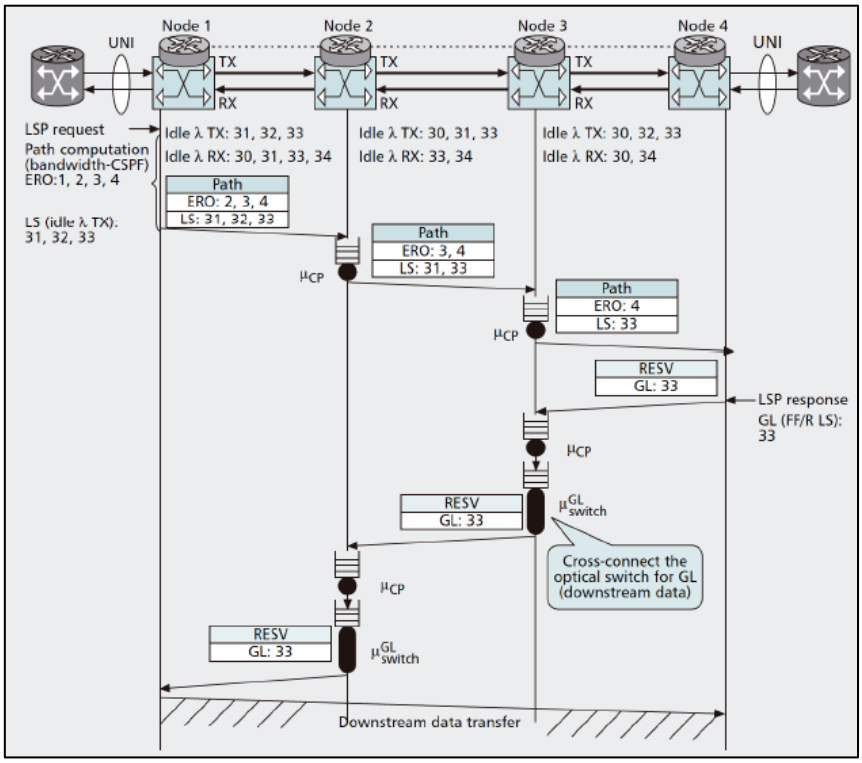

Figure 3.6 LSP establishment using the LS object [24]

- *Suggested Label object (SL):* This object suggests selecting a concrete label, although there is no guarantee of being successful in the LSP establishment process. If the suggested label is not available in a node the request

continues by taking into consideration the information in the previous explained LS object (Figure 3.7).

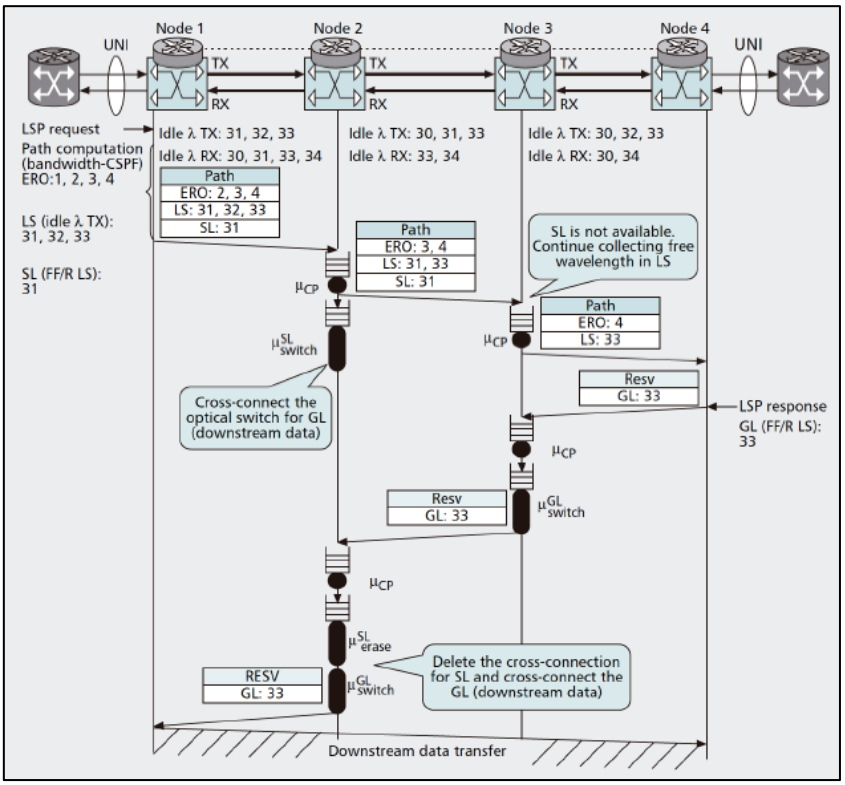

Figure 3.7 LSP establishment when using Suggested Label and Label Set objects [24]

- *Upstream Label Set (ULS):* This object introduces important improvements in the label selection process. It is included in the PathErr message, so it takes part when the request did not succeed. In this way, when the Path message is sent again, the SL object will contain the suggested label for the downstream data transfer and the ULS object will contain the suggested label for the upstream transference in case of dealing with bidirectional LSPs.
- *Upstream Label object (UL):* This object has been designed to reduce the signalling traffic. By making use of the UL object a bidirectional LSP can be established with one sole pair of messages, Path and Resv. As in the unidirectional LSP case, the Ingress LSR sends a Path message towards the Egress LSR, but now in this message there is a new object, the UL object, that suggests the destination node using a specific upstream label. If this label is available in all the nodes, the Resv message will confirm the establishment of both paths, downstream and upstream (Figure 3.8).

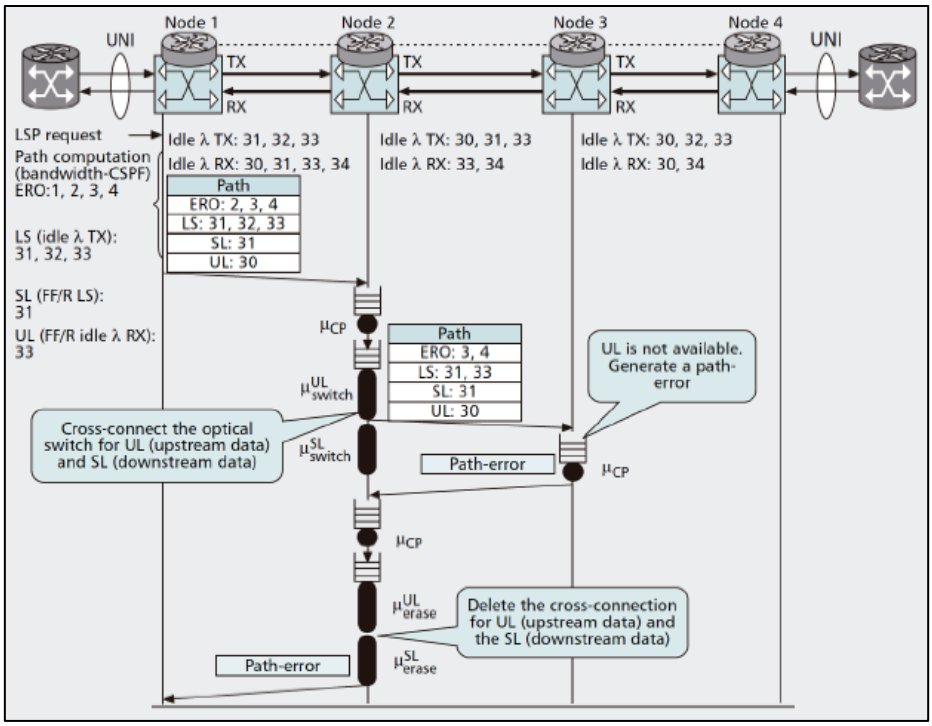

Figure 3.8 Bidirectional LSP establishment failure [24]

- **LSP Release**

  In the LSP release process there are two possible situations. On the one hand, the LSP can be released by the Ingress LSR, so it will send a PathTear message to the Egress LSR to free the assigned resources. On the other hand, if the decision of releasing a LSP is made by the Egress LSR, it will send a ResvTear message to the Ingress LSR, and the intermediates note will free the reserved resources at the same time that the message passes through them. Finally, the Ingress LSR can request again a LSP by sending a Path message or consider that the communication is over by sending a Path Tear or keeping passive (Figure 3.9).

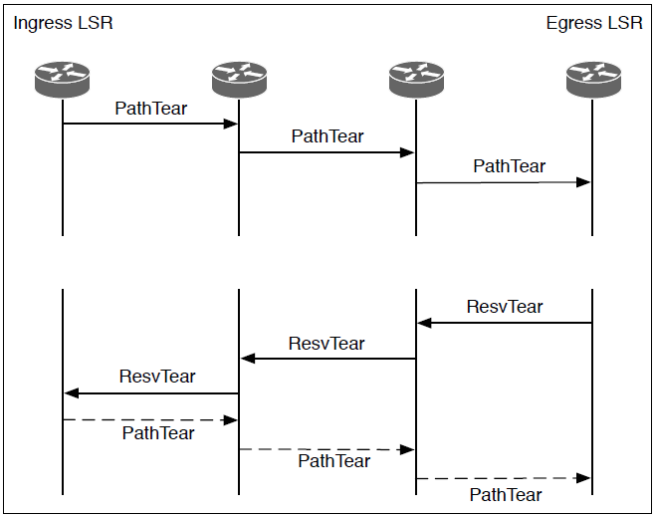

Figure 3.9 LSP release process in RSVP-TE

- **Errors Management**

  In an error occurs during the LSP establishment, the cause may be label restrictions or no available resources for the LSP requested. If the error takes place when transmitting the Path message, the next LSR will send a PathErr to the Ingress LSR, at which the message will arrive if any intermediate node can solve the problem. Thus, in case of reaching the Ingress LSR, this router can either try again sending a Path message or send a PathTear message to cancel the process (Figure 3.5).

  If the error appears during the Resv message transmission, the next LSR will send a ResvErr message to the Egress LSR. Like in the previous case, the intermediate nodes will try to solve the problem, but if the message arrives at the Egress LSR, this router can either modify the Resv message for a second attempt or send a PathErr message to the Ingress LSR to cancel the communication process (Figure 3.5).

  As for the errors occurred in the transmission stage, once started the transmission, each pair of contiguous nodes exchange periodically messages *Hello* and *HelloAck* to check the connectivity. If a timer expires, the LSRs involved in the failure will notify the Ingress LSR with a PathErr message and the Egress LSR with a PathTear message. Then the Ingress LSR will proceed to send a PathTear message (Figure 3.10).

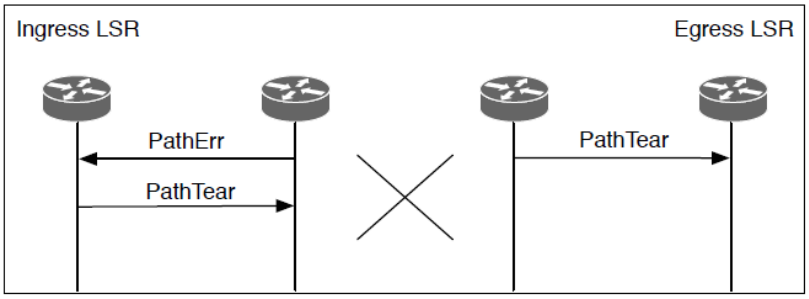

Figure 3.10 LSP release after a link failure

3.2 Management Approach based on SNMP

In this section an analysis of the SNMP (Simple Network Management Protocol) potential to undertake management tasks in the CHRON centralized architecture is presented, including the transmission of monitoring information from network elements to the cognitive decision system. First of all, a general overview of SNMP is presented. Then, a pair of use cases, dealing with two main scenarios within the CHRON project, is studied by paying attention to the SNMP messages structure and the information management.

The inclusion of two realistic use cases is aimed at easing the understanding of the protocol. The first case represents a scenario where a monitor issues an alarm to the cognitive decision system. As an example, it is considered that a monitor in the network senses the OSNR (Optical Signal to Noise Ratio) lower than a previously-fixed threshold and thus triggers an alarm. The second use case consists of a scenario where the cognitive decision system requests a monitor for up-to-date information, for example, the OSNR value.

3.2.1. General Overview

SNMP is considered as the *de facto* standard concerning network management protocols for the Internet. It includes the protocol itself, the definition of data structures and associate concepts [25]. SNMP was first standardised by the Internet Engineering Task Force (IETF) in 1990. The second version (SNMPv2) came to light in 1995, and finally, the third version (SNMPv3) was proposed in 1999.

There are three key aspects to take into consideration when using SNMP in general and, in particular, in the feedback system between the monitoring and the cognitive systems:

- *Management Information Base (MIB):* it is a virtual information store that contains a collection of the managed objects on an entity. Regarding monitoring, the objects that might be included in the MIB may be related to:
    - Information that changes continuously (e.g., current signal power)
    - Information that changes triggered by events (e.g., when facing failures)
    - Information configured by the manager (e.g., measurement rate, report rate)
- *Structure Management Information (SMI)*: it is a framework that describes the basic types of information that can be manipulated by SNMP, and thus provides a set of rules used in defining the managed objects in a MIB [26] [27]. It is a subset of ASN.1 (Abstract Syntax Notation One) [28].
- *The SNMP protocol itself*: it is responsible for the transmission and reception of messages among managers and network elements (agents). SNMP is an application layer protocol which belongs to the TCP/IP protocol suite and uses UDP (User Datagram Protocol) as transport protocol, with UDP ports 161 and 162 reserved for it.

As a whole, the SNMP architectural model is a collection of network management stations which monitor and control network elements, and agents located in these network elements (such as routers, optical cross-connects, amplifiers and the like), which perform the actions requested by the network management stations (Figure 3.11).

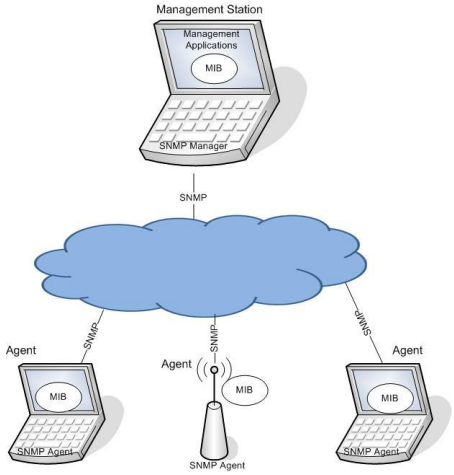

Figure 3.11 SNMP general architecture

Within the SNMP protocol, there are two approaches for a network management station to get information from the agents: the request/response approach, where the manager polls the agent for information, which is then provided by the agent, and the trap approach, where an agent sends unsolicited messages to the manager (Figure 3.12). Both of them will be explained in detail when presenting the different SNMP message types.

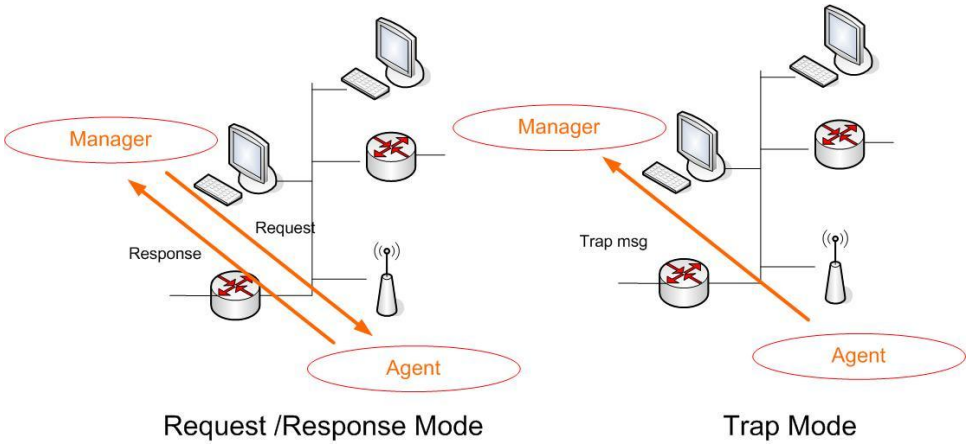

Figure 3.12 SNMP approaches for the manager to get information from an agent

3.2.2. SNMP Messages

Every SNMP entity is able to send and receive messages, which are identified through the PDU (Protocol Data Unit) type. The relationship between each SNMP message and PDU type is provided in Table 3.5.

SNMP Message	PDU Number
GetRequest	0
GetNextRequest	1
Response	2
SetRequest	3
Trap	4
GetBulkRequest	5
InformRequest	6

Table 3.5 Messages and PDUs relationship

In light of the functionality, the SNMP messages can be divided into several groups. For example, *GetRequest*, *GetNextRequest* and *GetBulkRequest* are sent from a manager to an agent to request data regarding an instance in the MIB, the following in a certain list, or some information stored in blocks, respectively. The *InformRequest* message is sent between two managers to share information. Then, the *SetRequest* message is sent from a manager to an agent in order to fix the value of an object in the MIB of the agent. As for the *Response* message, it is sent from an agent to a manager to reply to a previous request. Finally, the *Trap* message is sent from an agent to a manager to notify an exceptional event, such as a re-initialization, the recognition of a link failure or a link recovery. Each of these events is identified by a number, the Trap Type. It is worthy to note that one of the defined trap types (Trap Type 6) is enterprise-specific. When such a trap is used, a specific additional code identifies the particular enterprise-specific event that occurred, thus offering a degree of flexibility for trap implementations which could be exploited to meet the needs of CHRON [29].

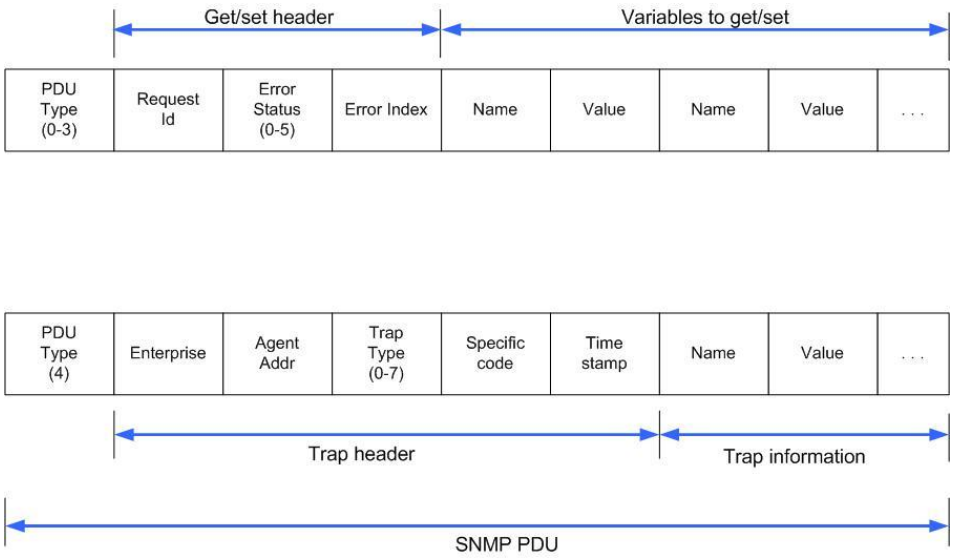

Figure 3.13 SNMP PDUs format

Figure 3.13 shows the format of the PDUs. The above part shows the PDU for types 0 to 3 (i.e., GetRequest, GetNextRequest, Response and SetRequest messages), while the part at the bottom shows the format for PDU type 4, i.e., the Trap-PDU, where the fields for specifying the Trap Type and the Specific Code can be seen.

3.2.3. SNMP Management Information

As previously mentioned, a MIB [30] is a virtual information store that contains a collection of the managed objects in an entity (such as parameters of the network elements and monitored parameters), organised into modules, and defined according to the SMI notation. Each object in the MIB is unambiguously identified by any management entity by means of its OID (Object Identifier). Hence, in the PDU formats shown in Figure 3.13, the "Name" field actually carries the OID of the object. These object identifiers are allocated in a tree fashion, and a portion of the OID tree is reserved to enterprises, thus offering vendors with flexibility for adding new objects. Related to this issue, it is important to remark that there is a great deal of standardized MIBs together with those developed by vendors.

Below there is an example from a vendor of how the monitoring information of optical parameters can be included within a MIB. The Figure 3.14 presents just a portion of a MIB and a drawing of the Object Identifier tree, including the specific part of the vendor, is presented in Figure 3.15 to describe how some objects are used to manage monitoring information by using the PDUs mentioned above.

```
-- ****************************************************************
-- CISCO-OPTICAL-MONITOR-MIB.my
-- May 2002, Sonal Maheshwari, Mickey Spiegel
-- Copyright (c) 2002,-2007 by Cisco Systems, Inc.
-- All rights reserved.
-- ****************************************************************

CISCO-OPTICAL-MONITOR-MIB DEFINITIONS ::= BEGIN

IMPORTS

    […]

ciscoOpticalMonitorMIB MODULE-IDENTITY
    LAST-UPDATED    "200701020000Z"
    ORGANIZATION    "Cisco Systems, Inc."
    CONTACT-INFO
            "Cisco Systems
            Customer Service
            Postal: 170 W Tasman Drive
            San Jose, CA 95134
            Tel: +1 800 553-NETS
            E-mail: cs-dwdm@cisco.com"
    DESCRIPTION
        "This MIB module defines objects to monitor optical
        characteristics and set corresponding thresholds on the
        optical interfaces in a network element.
        "
    REVISION        "200701020000Z"
    DESCRIPTION
        "Add cOpticalMonIfTimeGroup,
        cOpticalMIBEnableConfigGroup,
        cOpticalMIBIntervalConfigGroup,
        cOpticalMonThreshSourceGroup."
```

```
    REVISION        "200205100000Z"
    DESCRIPTION
        "The initial revision of this MIB."
          ::= { ciscoMgmt 264 }


-- Textual Conventions

OpticalParameterType ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "This value indicates the optical parameter that is
        being monitored. Valid values are -
        power (1)          : Optical Power (AC + DC) in 1/10ths of dBm
        […]
        ambientTemp (3)    : Ambient Temperature in 1/10ths of degrees
       […]
         "
    SYNTAX          INTEGER  {
                        power(1),
                        […]
                        ambientTemp(3),
                        […]
                    }

OpticalParameterValue ::= TEXTUAL-CONVENTION
    STATUS          current
    DESCRIPTION
        "The value of the optical parameter that is being monitored.
       […]
    SYNTAX          Integer32 (-1000000..1000000 )

[…]


-- MIB Object Definitions
cOpticalMonitorMIBObjects  OBJECT IDENTIFIER
    ::= { ciscoOpticalMonitorMIB 1 }

-- groups in this MIB module
cOpticalMonGroup  OBJECT IDENTIFIER
    ::= { cOpticalMonitorMIBObjects 1 }

[…]

-- cOpticalMonTable

cOpticalMonTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF COpticalMonEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This table provides objects to monitor optical

        parameters in a network element. […]"

    ::= { cOpticalMonGroup 1 }

cOpticalMonEntry OBJECT-TYPE
    SYNTAX          COpticalMonEntry
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "An entry in the cOpticalMonTable provides objects to
        monitor an optical parameter and set threshold levels
        on that parameter, at an optical interface.[…]"
    INDEX           {
                        ifIndex,
                        cOpticalMonDirection,
```

```
                        cOpticalMonLocation,
                        cOpticalMonParameterType
                    }
    ::= { cOpticalMonTable 1 }

COpticalMonEntry ::= SEQUENCE {

        […]
        cOpticalMonParameterType       OpticalParameterType,
        cOpticalParameterValue         OpticalParameterValue,

        […]
}


[…]

cOpticalMonParameterType OBJECT-TYPE
    SYNTAX          OpticalParameterType
    MAX-ACCESS      not-accessible
    STATUS          current
    DESCRIPTION
        "This object specifies the optical parameter that is being
        monitored in this entry."
    ::= { cOpticalMonEntry 3 }

cOpticalParameterValue OBJECT-TYPE
    SYNTAX          OpticalParameterValue
    MAX-ACCESS      read-only
    STATUS          current
    DESCRIPTION
        "This object gives the value measured for the particular
        optical parameter specified by the cOpticalMonParameterType
        object."
    ::= { cOpticalMonEntry 4 }

[…]
END
```

Figure 3.14 Example of MIB targeted towards optical monitoring [31, 32]

The MIB shown in Figure 3.14, `cisco-optical-monitor-mib`, defines a number of objects with the aim of monitoring optical characteristics as well as setting corresponding thresholds on optical interfaces. In particular, a table is defined (`cOpticalMonTable`), where each row or entry (`cOpticalMonEntry`) is related to the monitoring of one parameter. In this way, each entry in the table consists of a sequence of objects. Two of them are `cOpticalMonParameterType` and `cOpticalParameterValue`. The first of these objects, `cOpticalMonParameterType` (with OID 1.3.6.1.4.1.9.9.264.1.1.1.1.3, Figure 3.15), determines the parameter that it is being monitored. For instance, if its value is 1, the monitored parameter is the optical power; if it is 3, it is the ambient temperature. The second object, `cOpticalParameterValue` (with OID 1.3.6.1.4.1.9.9.264.1.1.1.1.4, Figure 3.15), provides the value of that parameter.

In this way, a network element can provide the manager with monitoring information by using the PDUs described, and according to a MIB definition similar to that described here.

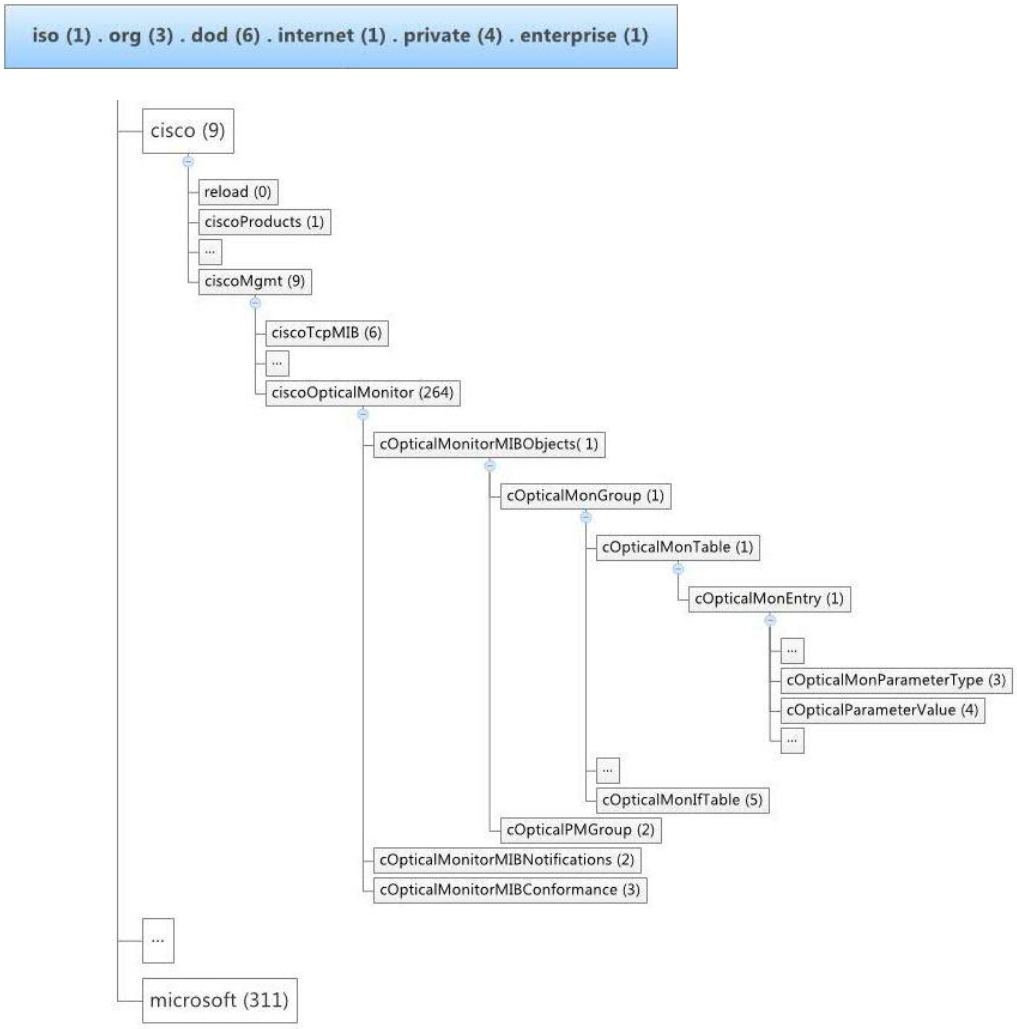

Figure 3.15 OSI Object Identifier Tree [33, 34]

3.2.4. SNMP Use Cases in CHRON

The presence of the Cognitive Decision System in the CHRON network architecture leads to the appearance of a new plane, the Knowledge Plane (KP). This plane interacts with other elements and in particular with the Network Monitoring System (NMonS) in order to get monitoring information. Thus, the connection between these elements is made through the MKI interface (Management-Knowledge planes Interface), Figure 3.16.

We can distinguish two different types of monitors: hardware and software monitors. A hardware monitor is a real physical box, which will be attached to a network element, such as an optical cross-connect (OXC), and even might be equipped with its own MIB and SNMP agent, so that it could directly communicate with the NMonS. On the other hand, a software monitor (such as DSP-based monitoring) would be directly located in the OCC and thus the OCC would handle management issues too.

According to Figure 3.16, in this scenario the OCC handles all the monitoring information gathered at an OXC, thus including an SNMP agent, an associated MIB, and also a Local Physical Parameter/Traffic Engineering Database (LPPD/LTED) where

the parameters described in the MIB are stored together with additional data for other purposes. Similarly, the NMonS includes an SNMP manager, which counts on MIB definitions and on a Global Physical Parameter/Traffic Engineering Database (GPPD/GTED).

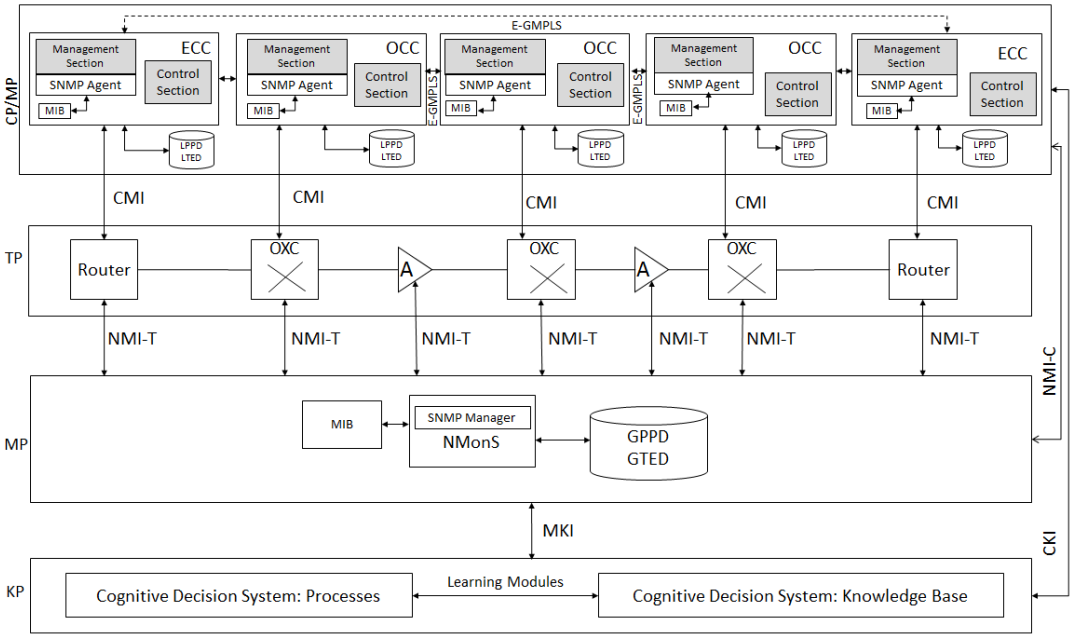

Figure 3.16 SNMP-based management architecture

Now, the rest of this section is devoted to defining two use cases to be considered in the CHRON SNMP management:

- **Use Case 1**

  Monitor located at OCC (A) senses that the OSNR level decreases significantly, so its value is lower than a pre-fixed value. Then, the cognitive system should be notified about this issue (Figure 3.17).

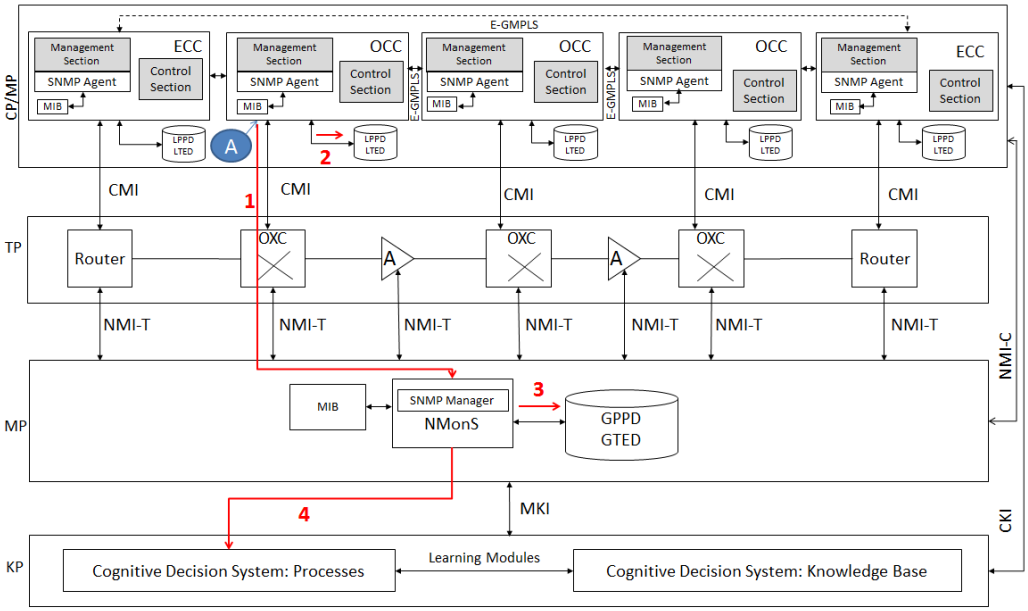

Figure 3.17 Steps of SNMP CHRON Use Case 1

Steps:

1. The optical monitor device in A identifies a significant decrease in the OSNR level (OSNR<THRESHOLD_OSNR), hence the SNMP agent, taking into account the MIB definition, and using the SNMP protocol, generates and sends a Trap message to the manager. The fields within the Trap message would be configured as follows (Figure 3.18):

Figure 3.18 Trap PDU

   - *PDU Type* is set to 4.
   - The *Enterprise* field indicates the agent that issues the trap.
   - The *Agent Address* makes reference to the agent IP address.
   - In this case the *Trap Type* would be set to 6 since this type includes the new traps included by vendors (a decrease in the OSNR level would be a new trap within the CHRON project).
   - The *Specific code* would be that associated with the trap type previously explained.
   - The *Timestamp* value refers to the time interval that lasts from the agent reboot to the trap generation.
   - Regarding the trap information, in this case the *Name* would be OSNR (actually, the OID associated to it) and the *Value* would be associated with the measurement of the monitor (e.g., 10.5 dB/0.1 nm properly coded according to the MIB definition).
2. Once submitted the trap, the local database will be updated.
3. The SNMP Manager will update the received information in the global database.
4. And it will pass the information to the Cognitive Decision System, which will decide which action to take.

- **Use Case 2**

  In this case the Cognitive Decision System requests (through the SNMP Manager) the Optical Monitor Device in OCC (A) for the OSNR value (Figure 3.19).

  Steps:

1. The cognitive system employs the MKI interface to order the NMonS to issue a request for the OSNR value to the device A.
2. The SNMP Manager sends a *GetRequest* message to the SNMP agent in OCC (A) to request the value of the OSNR object (actually by requesting

the value of the OID which corresponds to the OSNR, and which is available in the MIB).

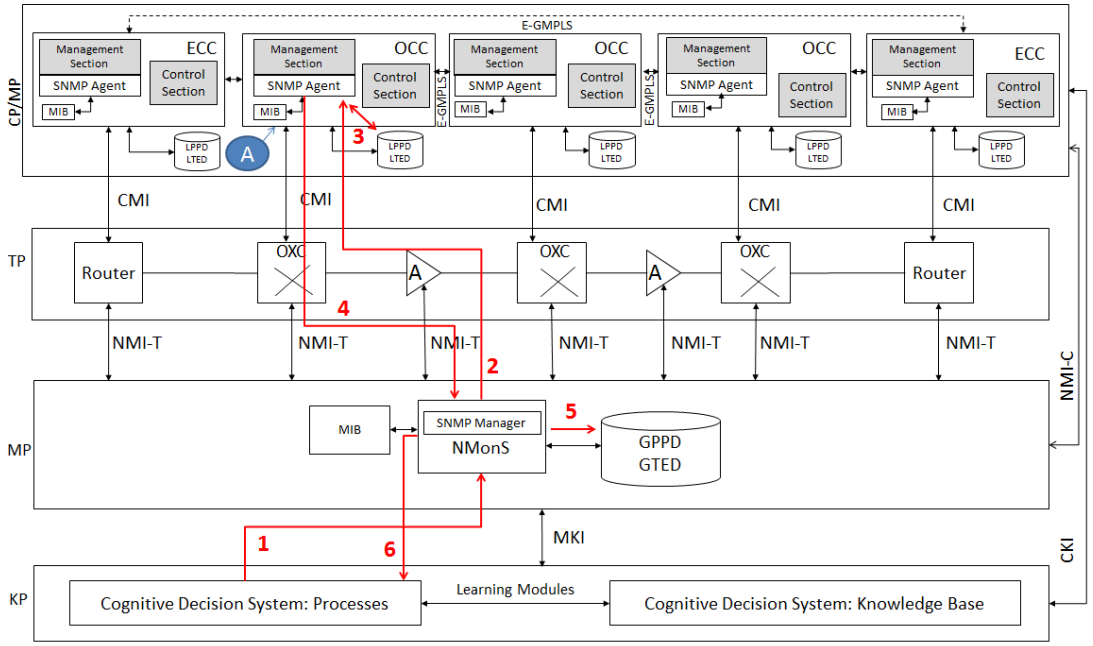

Figure 3.19 Steps of SNMP CHRON Use Case 2

3. The SNMP agent will consult this value in its local database according to the object identification in the MIB (the search on the local database could be omitted if the monitor has available the information in real time).

4. The SNMP agent will send the latest measurement to the SNMP manager by means of a *Response* message.

5. The SNMP manager will update the information in the global database.

6. Finally, the NMonS will inform the cognitive decision system about the value of the requested parameter.

Figure 3.20 shows the format of the *GetRequest* and *Response* PDUs exchanged in steps 2 and 4, respectively. The fields are set as follows:

- *PDU Type* is set to 0 for *GetRequest* and to 2 for *GetResponse*.
- The *Request Id* is used by the NMonS and the agent to send different requests and responses at the same time. In this situation both the request and the response will have the same id.
- The *Error Status* field is only used in the *Response* message. There are several error types that can be consulted in [29].
- As far as the information is concerned, in this case the Name would be the ONSR (actually, the OID related to it), and the Value would be associated with its latest value (e.g., the value of 32.5dB/0.1nm properly coded).

0	Request Id	Error Status (0-5)	Error Index	OSNR	?

2	Request Id	Error Status (0-5)	Error Index	OSNR	32.5 dB / 0.1 nm

Figure 3.20 GetRequest and Response PDUs

Just to mention that both previous use cases have been implemented in the DRAGON emulator such as the reader can find on the second section of Chapter IV.

3.3 Management Approach based on XML

SNMP is the network management solution that has been used by most of the industry since the early 1990. However, over the time, some drawbacks have turned up. Many of them, such as security issues, have been solved with SNMPv2 and SNMPv3. However, some others are still open [35]. To some extent, XML (eXtensible Markup Language) and its related technologies offer opportunities to solve these problems. Therefore, it is worthwhile to analyze in this Master of Research Thesis their applicability in the monitoring tasks of CHRON.

Firstly, the shortcomings of SNMP and the advantages of XML are described. Then, a full-XML management system based on the NETCONF protocol [36] is explained and some of the possible modeling languages used to encode the information are analyzed. Afterwards, the same real use cases considered in SNMP are studied for the XML management case.

3.3.1. SNMP versus XML-based management

As it was mentioned before, SNMP still presents some disadvantages. Some of them are detailed below [37], [38] and [39].

- SNMP has some weaknesses related to configuration management. The SNMP Structure of Management Information (SMI) is so simple that it is insufficient to represent large amount of network monitoring data.

- SNMP does not consider many of the physical layer parameters that come from the monitors that the CHRON project will use. Hence, extensions or even new OIDs definitions in MIBs may be required to support these new monitors. This represents a drawback, since proprietary MIB extensions that often supplement standard MIBs can lead to a problem for interoperability and developers.

- SNMP makes management applications very complicated, which implies a slow and expensive development.

- SNMP uses UDP as transport protocol. Therefore, the maximum packet size is equal to the MTU (Maximum Transmission Unit). Although fragmentation and re-assembling of SNMP messages is possible, it may leads to extra-processing. Moreover, the unreliability of UDP makes necessary the retransmission of important data.

- SNMP/MIBs are not human-readable.

On the other hand, XML supports, by its nature, a well-defined coding syntax. Therefore, using XML in network management presents some advantages. For example:

- The simple encoding of XML makes possible the representation of arbitrary hierarchies of data using tags.
- It defines data structure in an easy, flexible and powerful manner.
- It offers a higher abstraction level for application developers.

- XML representations are widely accepted as a good compromise between human and computer readability.
- It offers a better integration of management data from disparate sources, improving the interoperability among management applications from different vendors.
- It leads to more flexible communication between the managed object (agent) and the management application.
- The powerful tools and libraries associated with XML facilitate the construction of systems that are easy to update and extend and can make the development of management applications more efficient.

Based on the facts presented above, the use of XML in the network management is considered as a promising solution to overcome the shortcomings of SNMP. However, most network devices currently deployed are equipped only with SNMP agents. Therefore, in order to facilitate migration from current networks, hybrid alternatives, supporting the coexistence of SNMP and XML-based agents should be considered. This issue will be analysed later this section.

3.3.2. Fully XML-based management: NETCONF protocol

Being aware of the shortcomings of SNMP and the high potential of XML, the Internet Engineering Task Force (IETF) has standardized a new network configuration management protocol called NETCONF [36].

NETCONF defines a simple mechanism through which a network device can be configured, configuration data and status data can be retrieved, and new data can be issued to the device. This protocol uses XML for data encoding (which improves the interoperability among devices from different vendors), and employs a simple RPC (Remote Procedure Call)-based mechanism [40] to facilitate communication between a client and a server. The client is an application running as part of a network manager (and hence, as part of the cognitive decision system in CHRON). The server is typically a network device (e.g., a monitor). It is connection-oriented, so it requires a persistent connection between manager and agent.

This protocol can be conceptually partitioned into four layers (Figure 3.21) [38] [41]:

- *Content*: It represents the configuration data about the managed objects.
- *Operation*: It provides various operations. The basic ones, among others, are: `<get-config>` and `<get>` to retrieve information of the configuration and state data; `<edit-config>` to modify a configuration; `<lock>` to lock a configuration of a device; `<unlock>` to release it; etc. [36].
- *RPC*: It uses an RPC-based communication model to encapsulate XML messages generated in the operation layer. It uses the `<rpc>` and `<rpc-reply>` elements to indicate a request and a response message respectively, and `<notification>` when sending an alarm [48].
- *Transport*: This layer adopts a connection oriented transport protocol to ensure the security of the transmission. Possible transport protocols are Secure Shell

(SSH) [42], Simple Object Access Protocol (SOAP) [43] over HTTP, Block Extensible Exchange Protocol (BEEP) [44] and TLS (Transport Layer Security) [45]. Among them, SOAP over HTTP is a natural application protocol for NETCONF, essentially because it supports its own RPC interface [38]. Moreover, as SOAP is widely used in web services, its use in NETCONF offers some advantages such as better integration with deployed systems or the use of tools and libraries already developed for SOAP [46] [47].

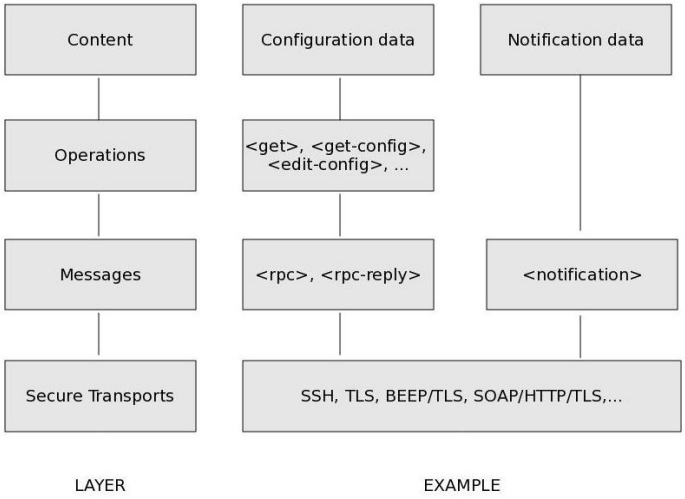

Figure 3.21 NETCONF protocol layer [38] [41]

A possible NETCONF architecture is shown in Figure 3.22. As it can be seen, the network manager has a NETCONF client, which creates the NETCONF messages using an XML parser. Then, the message is transmitted to the agent by means of SOAP over HTTP. It has also an XML database (XML DB), which contains the topology, and a summary of information and capabilities of the managed devices.

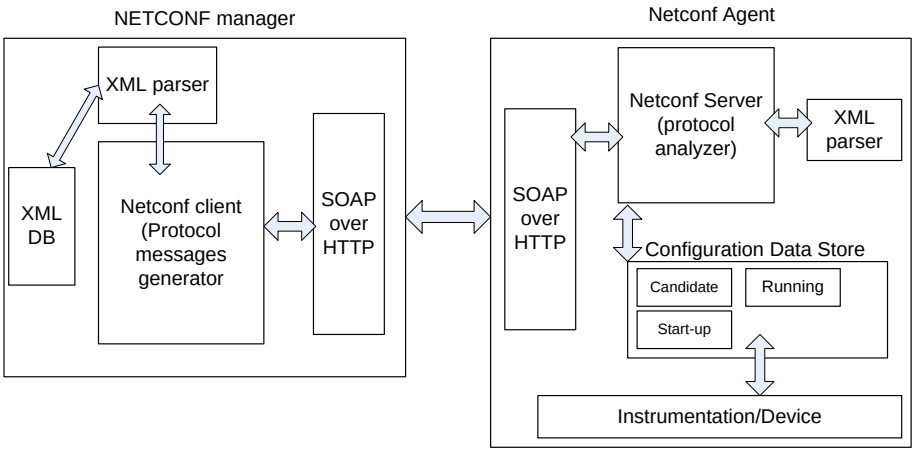

Figure 3.22 NETCONF Architecture

The agent contains the NETCONF server, which analyses the operations of the received messages and which also creates the NETCONF responses and notifications to the manager using an XML parser. Moreover, it has the configuration data store, where data of the device is stored. There are several options for the data store. The simplest model only requires the so called running configuration data store, where configuration changes are directly applied to the running configuration. However, an

agent can also have, additionally, a candidate data store (the changes in this one have no effect until they are committed to the running configuration) and a start-up configuration data store, which is loaded by a device when it reboots or reloads (Figure 3.22).

3.3.3. XML Management Information

NETCONF needs its own data modelling language, just like SNMP needs SMI [49]. Requests from the manager and replies from the server are encoded in XML. Thus, in order to allow both parts to recognize the syntax and the content exchanged, it is necessary to define the contents of requests and replies; that is, to establish the structure of the XML documents. To do this, different types of modelling languages can be used.

- Document Type Definition (DTD)

It describes the logical structure of the XML document. That is, it defines the elements that have to appear in the XML document, the constraints related to these elements, the associated attributes for each declared element, including the name and the type value. Thus, an XML document is valid only if it complies with the structure and constraints expressed in the .dtd file [50]. Examples of DTDs are provided in Annexes 1 and 2 together with XML files used for optical performance and traffic monitoring.

- XML Schema Definition (XSD)

It is the successor of DTD. It also defines the structure of an XML document but offers more possibilities than DTD, as it overcomes DTD weaknesses and implements new capabilities. One of its key features is the consideration of namespaces [51]. Namespaces play an important role in the identification process, as they are essentially a grouping of elements and attributes used for a particular purpose.

Another important feature is that XSD supports a variety of data types (integer, floating numbers, Booleans, etc.) and also supports user-defined types. Moreover, instead of treating all XML data as just plain text, the user can enforce formal syntax and semantics.

Moreover, it has to be taken into account that there are different translations of the SNMP MIBs into XSD files, such as the ones presented in [35] [52]. This can be very useful in the hybrid architectures, as it is shown in a sub-section later.

Finally, by using XSD, CHRON could take advantage of the Sensor Model Language Encoding Standard (Sensor ML), which offers XML schemas designed for encoding information of different types of sensors [53].

In order not to overload this section, other modelling languages are briefly presented below together with relevant references. For instance, the Relax New Generation (RelaxNG) was designed to be simple and easy to understand and it shares many features with XML Schema Definition that set both apart from

traditional DTD to understand [54]. Then, the YANG language has modules which define a hierarchy of data that can be used for NETCONF-based operations, including configuration, state data, RPCs and notifications. This makes possible a complete description of all data sent between the manager and the agent [55] [56].

3.3.4. XML Use Cases in CHRON

The two cases considered in the analysis of SNMP are also studied here in order to explain the interaction between the different modules. In these two use cases, it is assumed the utilization of SOAP over HTTP. Moreover, since the resulting XML messages of the previous modelling languages are very similar, any of the presented examples could give structure to the XML messages considered in this section.

When comparing the network architecture using XML-based management (Figure 3.23) to that presented previously for SNMP (Figure 3.16), the main differences are that the SNMP Manager has been replaced by a NETCONF Client, the SNMP Agents by NETCONF Servers, and MIBs have been replaced by XML language definition (i.e., either DTD, XSD, RelaxNG or YANG definition).

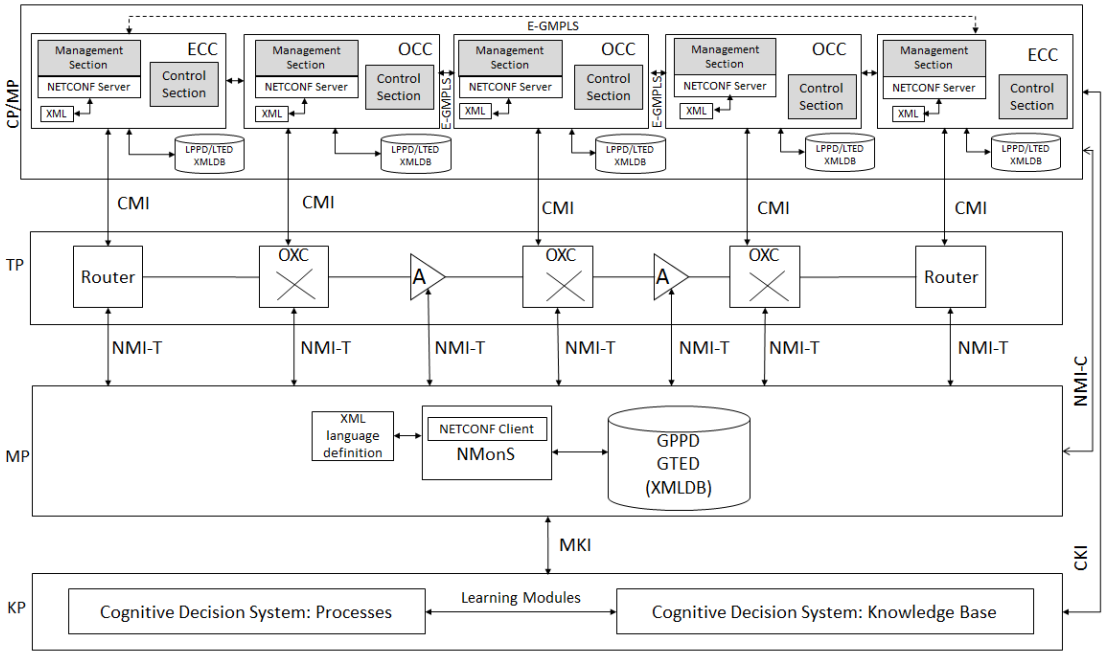

Figure 3.23 XML-based management approach

- **Use Case 1**

  A monitor A, located in an OCC, senses that the OSNR level has decreased below a pre-fixed threshold, and the cognitive decision system has to be notified (Figure 3.24).

  Preconditions:

  - The monitor located at OCC, A**,** is working correctly.

- A session has already been set up between the NETCONF client at the Network Monitoring System (NMonS) and the NETCONF server at A. To do this, the manager has previously sent a `<hello>` operation and has asked for the capabilities of the device. The capabilities (if exist) are additional operations (apart from the basic NETCONF operations) of the device. The agent replays with its capabilities and assigns an ID (identification number) for the connection [38].

- The manager can receive notifications of this agent as it has already created a subscription. That is, in order to receive asynchronous messages reporting event notifications, the manager first has to create a subscription with the agent of interest. The manager creates a subscription by means of the `<create-subscription>` operation and indicates the type of event notifications it wants to receive using the `<filter>` option. In order to know what type of events the agent can notify, the manager sends a `<get>` operation asking for the available event streams by means of the tag `<stream>`, which would then retrieve the set of event notifications offered by the agent. The subscription is set up when the agent reply with one `<ok>`. Once the subscription is made, the type of events selected cannot be modified. The event notification can be closed using `<close-session>` operation or specifying a stop time in the `<create-subscription>` operation [48].

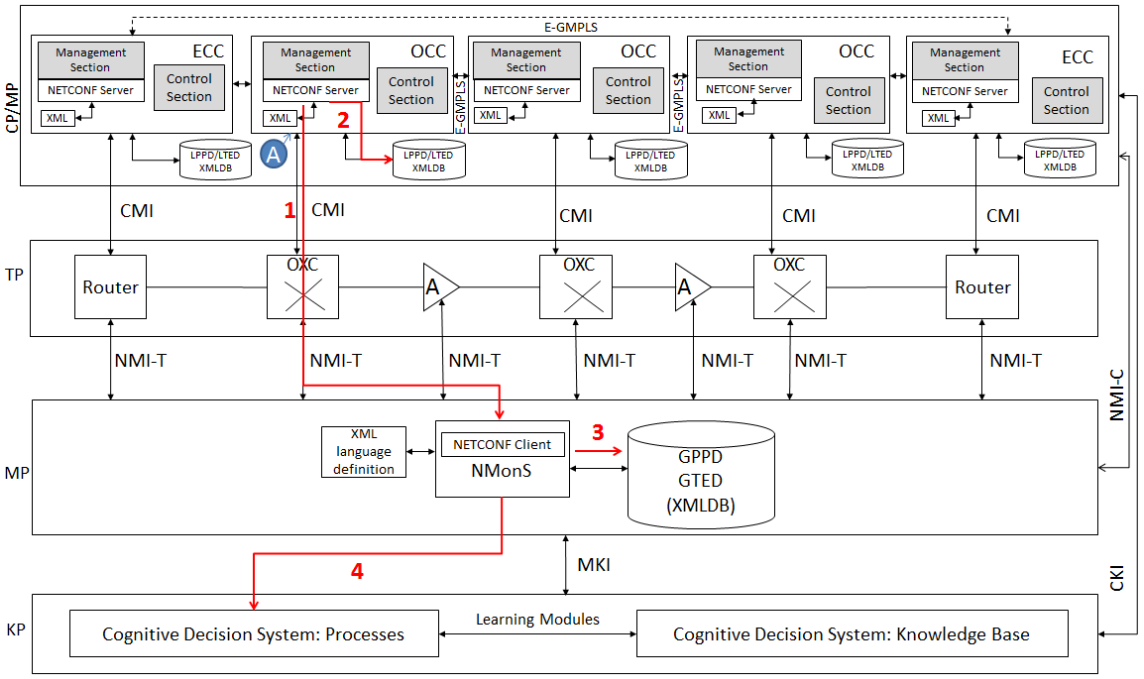

Figure 3.24 Steps of XML CHRON Use Case 1

Steps:

1. The monitor located in OCC, A, identifies a significant decrease in the OSNR. Thus, the NETCONF server sends a `<notification>` to the NMonS subscribed to that type of notifications using SOAP over HTTP. This is a complete and well-formed XML document following the structure defined in the XML language definition block by the encoding language selected. For example, [48] defines a base XML Schema for notifications where two elements are incorporated: the `eventTime` (time when the event was generated) and the `eventContent` (where the content is specified). A possible example of notification can be:

```
<notification

xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
      <eventTime>2010-07-08T00:01:00Z</eventTime>
         <event xmlns=" http://intranet.ict-
   chron.eu/event">
         <eventClass>alarm</eventClass>
         <eventContent> OSNR under threshold
</eventContent>
         <severity>minor</severity>
         <measurement>
            <value> 10.5 </value>
            <unit> dB/0.1nm </unit>
            <timestamp> 13:17:33 </timestamp>
            <validity_period> PT5M </validity_period>
         </measurement>
       </event>
   </notification>
```

2. The local database in A is updated with the new information.

3. The NETCONF client at the NMonS receives the notification and stores the information in the global database

4. Finally, the NMonS notifies the cognitive decision system of the event.

- **Use Case 2**

  The cognitive decision system requests (through the NMonS) the A optical monitor device for the OSNR value (Figure 3.25).

  Preconditions:

  - A session has already been set up between the NETCONF client at the network monitoring system (NMonS) and the NETCONF server in A.

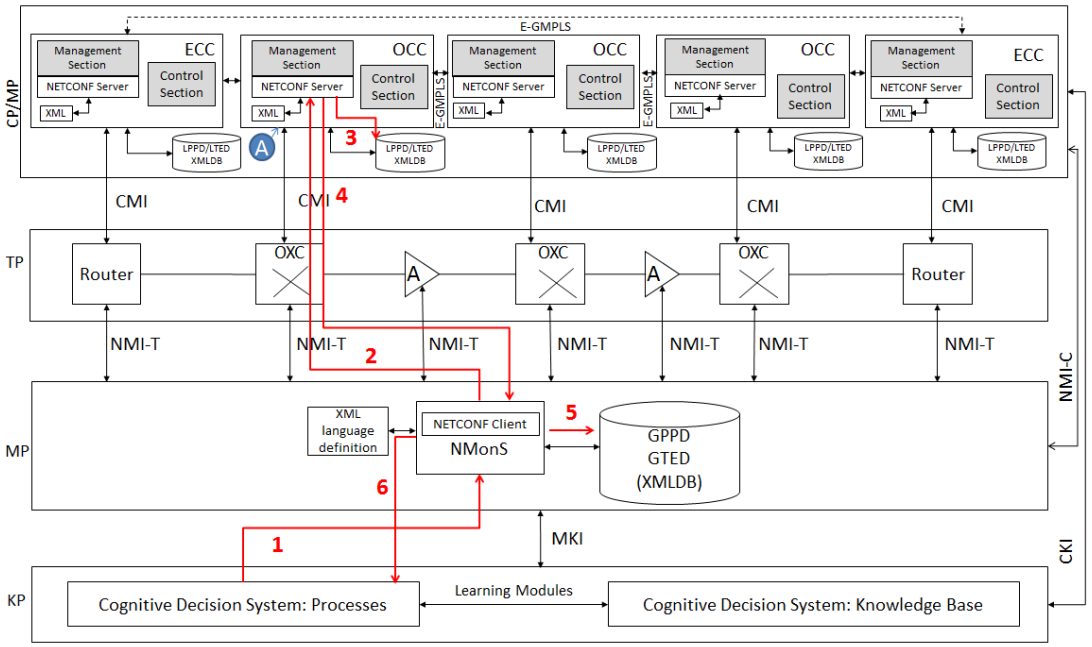

Figure 3.25 Steps of XML CHRON Use Case 2

Steps:

1. The Cognitive Decision System employs the MKI interface to order the NMonS to issue a request for the OSNR value to Monitor A.

2. The NETCONF client in the NMonS creates an XML document with the operation `<get>` inside an `<rpc>` tag. This document has to be compliant with the structure of codification language selected (DTD, XSD, RelaxNG or YANG stored in the XML language definition module of the figure). The XML document is sent to the NETCONF server in A. If the SOAP/HTTP protocol is selected, the document would include the SOAP envelope and the HTTP header shown below.

HTTP

```
POST /netconf HTTP/1.1
Host: netconfdevice
Content-Type: text/xml; charset=utf-8
Accept: application/soap+xml, text/*
Cache-Control: no-cache
Pragma: no-cache
Content-Length: 377
```

SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv=" http://intranet.ict-
chron.eu/XML/soap-envelope">
<soapenv:Body>
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
<source><running/></source>
<filter type="subtree">
<top xmlns="http://intranet.ict-chron.eu/Data">
<measurement/>
```

```
        </top>
        </filter>
        </get>
        </rpc>
SOAP    </soapenv:Body>
        </soapenv:Envelope>
```

3. The NETCONF server accesses to the LPPD database and checks the state value demanded (this step could be omitted if the monitor has available the information in real time).

4. The NETCONF server at the OCC generates an XML document with `<rpc-reply>` as to response for the request and sends it to the NMonS via SOAP/HTTP.

```
HTTP    HTTP/1.1 200 OK
        Content-Type: application/soap+xml; charset=utf-8
        Content-Length: 415

        <?xml version="1.0" encoding="UTF-8"?>
        <soapenv:Envelope
SOAP    xmlns:soapenv="http://intranet.ict-chron.eu/XML/soap-
        envelope">
        <soapenv:Body>
        <rpc-reply message-id="101"
        xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <data>
        <top xmlns="http:// intranet.ict-chron.eu/Data">
        <measurement>
        <value>10</value>
        <unit>db/0.1nm</unit>
        <timestamp>16:05</timestamp>
        <validity_period> PT5M </validity_period>
        </measurement>
        </top>
        </data>
        </rpc-reply>
SOAP    </soapenv:Body>
        </soapenv:Envelope>
```

5. The NMonS, by means of the NETCONF client, receives the document and processes it. Once the value is obtained, the GPPD database is updated with the value.

6. The cognitive decision system is given the obtained value and it will make the corresponding decisions if necessary.

3.3.5. Hybrid Management Approach

Three possible combinations between managers and agents can be considered when at least one of the elements relies on XML-based management in a hybrid architecture. Figure 3.26 (c) represents a fully XML-based architecture, where both manager and agents are based on XML. In contrast, Figure 3.26 (a) and Figure 3.26 (b) represent

hybrid scenarios as the manager and the agent follow different approaches, either SNMP or XML-based management. As it can be observed, in these cases it is necessary to introduce a gateway in order to translate and relay messages between the manager and the agents. In case (a), there is an SNMP manager and XML-based agents. This type of architecture is not very common, as if there are already XML-based agents the greatest benefits are achieved with an XML-based manager. Figure 3.26 (b) represents the integration of SNMP agents with an XML-based manager. This work focuses on this option, since this combination is the most realistic in migration scenarios, where an XML-based manager would manage network devices using legacy SNMP agents together as well as others based on XML [58].

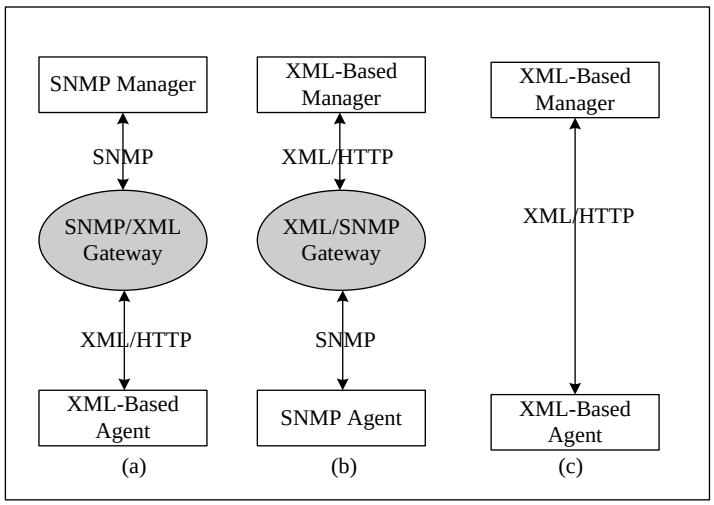

Figure 3.26 Combinations of manager and agent in hybrid architectures [57]

Figure 3.27 illustrates a basic example of the architecture of an XML/SNMP Gateway [58], which has been chosen to explain a possible implementation of the uses cases treated in this Thesis.

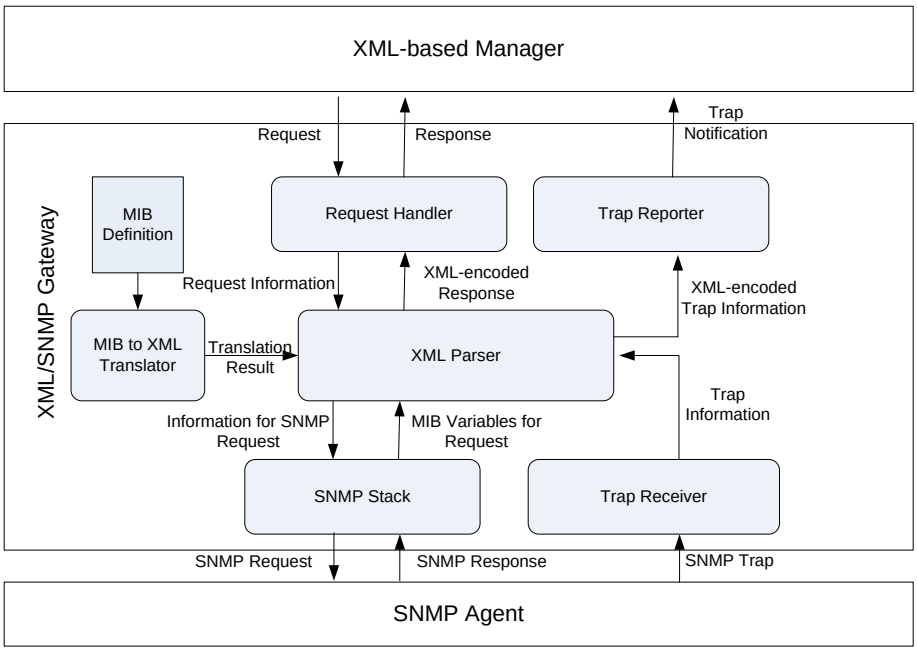

Figure 3.27 Architecture of the XML/SNMP Gateway [58]

- **Use Case 1:** The OSNR monitor (SNMP Agent) sends an alarm (a trap) to the cognitive decision system (via an XML-based manager).

  The monitor senses an OSNR value lower than a pre-fixed threshold and generates an SNMP trap, which is received by the *Trap Receiver* of the gateway. The trap information is then sent to the *XML Parser*, which is in charge of providing an XML-encoded response, i.e., in charge of creating an XML document with the trap information. For that aim, it relies on the *MIB to XML Translator* module, which translates the MIB associated to the monitor to an XML schema (or a DTD), and then combines that schema with the information provided by the monitor to produce the XML document. Finally, the XML document including the trap information is sent to the *Trap Reporter*, which will forward it to the manager in the NMonS and finally to the cognitive decision system.

- **Use Case 2:** The cognitive decision system (through an XML-based manager) requests updated information from an OSNR monitor (SNMP agent).

  In this case, the *Request Handler* analyzes the request from the XML-based manager (which comes from the cognitive decision system) and calls the corresponding method in the *XML Parser* application. Then, with the help of the *MIB to XML Translator* (whose task was explained in the previous use case), the *XML Parser* retrieves the OID associated to the monitor, the identifier of the target object (the OSNR parameter), and delivers them to the *SNMP Stack*. The *SNMP Stack* then sends an SNMP request message to the SNMP agent, i.e., to the monitor. When the monitor replies, the *SNMP Stack* receives the response and sends it to the *XML Parser*. The *XML Parser* then creates an XML document (as described in use case 1) and provides it to the *Request Handler*, which will forward it to the manager in the NMonS and finally to the cognitive decision system.

3.4 Conclusions

The main contribution of this Chapter III is to be familiar with the terminology and technical operation of the control and management approaches considered in CHRON.

In the first place, the description of the control approach based on GMPLS provides all the fundamentals to manage and understand the operation of the DRAGON emulator in the following chapter.

Secondly, the section that addresses the management approach based on SNMP constitutes an important reference guide to understand the operation of the Linux library AGENT++/SNMP++, which has been chosen to include the SNMP functionalities in the DRAGON emulator.

In the third place, the management approach based on XML although not being used in the CHRON emulation, it has been employed in the CHRON simulation environment to encode the messages exchanged between the cognitive decision system and the control and management plane as it is exposed in Annexes 1 and 2.

Last but not least, the consideration of the hybrid management approaches has given this chapter an added value in regards with the SNMP-equipped networks migration to the XML-based management approach.

The following chapter will serve to put in practice everything learnt here through an emulated environment quite similar to a real scenario.

Chapter IV – Emulation of Optical Transport Networks

4 Emulation of Optical Transport Networks

This fourth chapter analyses the DRAGON (Dynamic Resource Allocation Optical Networks) emulator in order to be familiar with this tool, which has been considered to support the emulation tasks in CHRON project. This emulator is associated with the project of the same name, DRAGON, which already finished, and was devoted to developing an open-source tool to emulate dynamic provisioning of network resources on an inter-domain basis across heterogeneous network technologies. Thus, DRAGON implements the GMPLS protocols to control optical transport networks [59].

In spite of the lack of documentation and support when working with this emulator, important effort has been done to meet the objectives of this Master of Research Thesis and put in practice all the aspects learnt in the theoretical stage.

In the first place, this chapter presents a DRAGON scenario based on the Deutsche Telekom (DT) network to emulate the GMPLS protocols functionality. Secondly, the same scenario will support the SNMP protocol emulation through the AGENT++/SNMP++ Linux library [60].

4.1 DRAGON: GMPLS Emulation in CHRON

DRAGON was created to be able to work on inter-domain scenarios, although in the framework of CHRON project only intra-domain networks have been considered. In addition, one of the main modules of this emulator, NARB (Network Aware Resource Broker), which will be explained in detail later, is going to be considered as the control node in terms of the CHRON centralized architecture presented at the beginning of this work. DRAGON allows the emulation of GMPLS in scenarios where devices are not geared for GMPLS, such as Ethernet switches, which are much more accessible for research groups due to the expensive cost of an optical testbed (Figure 4.1).

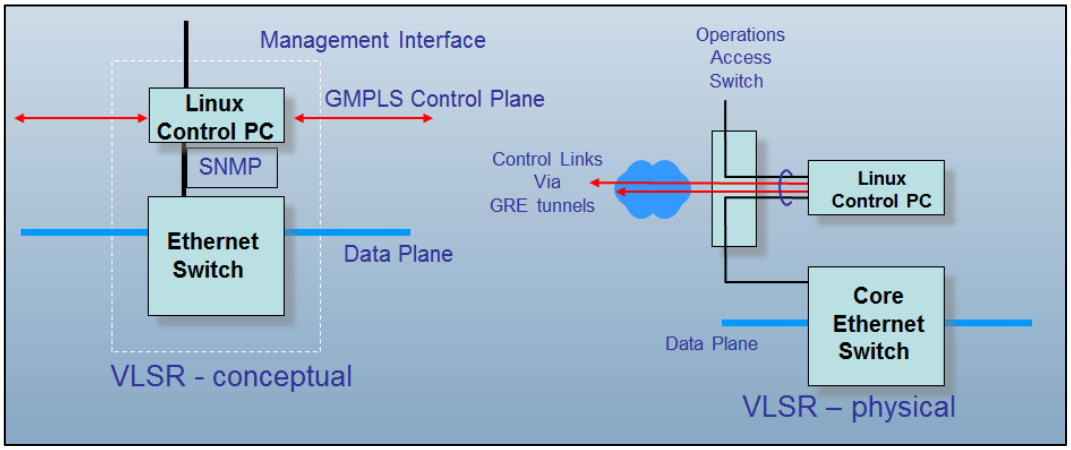

Figure 4.1 Conceptual and physical architecture of a network node in DRAGON [57]

Taking advantage of the previous functionality it is possible to implement an emulation environment in an Ethernet network, where end-to-end LSPs are established by VLANs (Virtual Local Area Network), which permit to undertake tests in logically-independent networks in a similar way to lightpaths in optical transport networks.

4.1.1. DRAGON Software Elements

The DRAGON software suite carries out the tasks of the GMPLS control plane in a network. The architecture of this control plane consists of four elements: Client System Agent (CSA), Network Aware Resource Broker (NARB), Virtual Label Switching Router (VLSR) and Application Specific Topology Builder (ASTB).

4.1.1.1. Client System Agent (CSA)/End System (ES)

CSA is software running in the end system that finalizes the data plane and it is connected to the network by the UNI (User to Network Interface) or Overlay Model. This element allows to request end-to-end provisioning services, from the CSA source to the CSA destination [58].

4.1.1.2. Network Aware Resource Broker (NARB)

This component represents a local Autonomous System (AS) and listens to the routing protocols within its domain. The main functions rest on topology abstraction, intra-domain path computation and routing, and inter-domain path computation by interacting with the counterpart elements in other domains (Figure 4.2). This work only focuses on intra-domain actions since the networks studied in CHRON present a single-domain topology.

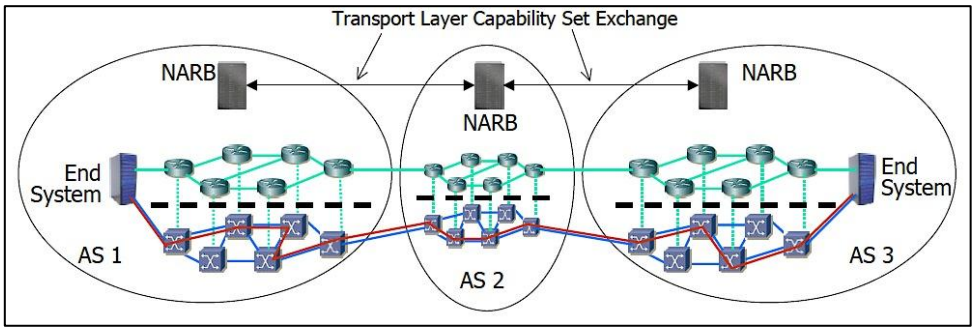

Figure 4.2 DRAGON Inter-domain Relationship [57]

A NARB subcomponent is the RCE (Resource Computation Element), which executes path computation tasks by advanced algorithms. The result of this computation process is the OSPF ERO object. Moreover, this module offers information concerning paths established between concrete source and destination nodes. Definitely, the NARB tasks are comparable with the ASBR tasks in the OSPF protocol [57].

- Intra-domain path computation in NARB

  When an LSP establishment request arrives at NARB, this node consults the provisioning module that is in charge of translating the attributes and the restrictions of the request to be understandable by the RCE. Then, the RCE computes the route by making use of the information provided from the OSPF daemon. Afterwards, the provisioning module passes the computed path to the

resources management module which requests the resources reservation for the LSP under establishment process.

Once, all the resources have been reserved and the RCE TEDB (Traffic Engineering Data Base) has been updated, the path computed by RCE will be used by NARB to create the ERO. The next step is to pass this object to the requester element (VLSR) or an error message in case of no having success in the computation (Figure 4.3).

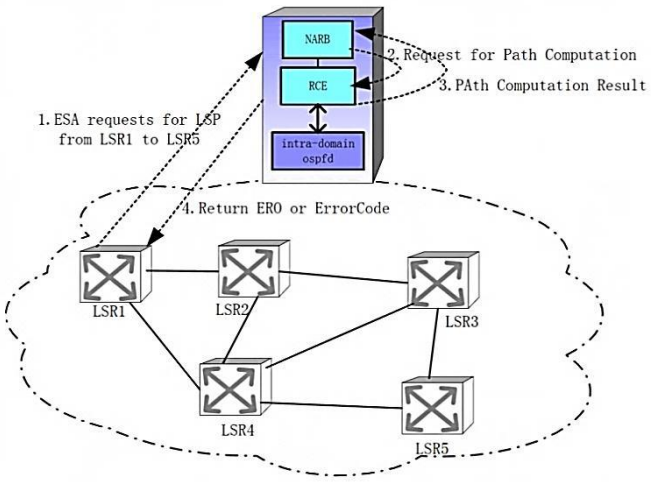

Figure 4.3 DRAGON Intra-domain Path Computation [57]

4.1.1.3. Virtual Label Switching Router (VLSR)

The DRAGON emulator offers the possibility of implementing GMPLS in commercial components that do not support the GMPLS capability. According to Figure 4.1, a VLSR is the combination of a PC with the DRAGON software suite and a switch without GMPLS capacity, for example, an Ethernet switch. Thus, the DRAGON software implements the control plane of a GMPLS node and configures the switch to activate/deactivate VLAN ports by specific commands such as SNMP, TL1, CLI or XML. The DRAGON emulator supports a wide variety of switch models and it can be also extended to support others different to those considered in [59].

As far as the protocols running in the VLSR, it uses the OSPF-TE routing protocol and the RSVP-TE signalling protocol for control purposes.

4.1.1.4. Application Specific Topology Builder (ASTB)

The DRAGON architecture permits to establish specific topology applications. The DRAGON suite has an API that offers the functionality of requesting a dynamic access route through the network [57]. This module is out of the scope of this Master of Research Thesis, so this epigraph is only included for documentation purposes.

4.1.2. DRAGON Processes

Apart from the previous elements, DRAGON consists of a set of daemon processes running in the background to undertake the GMPLS functionalities (Figure 4.4).

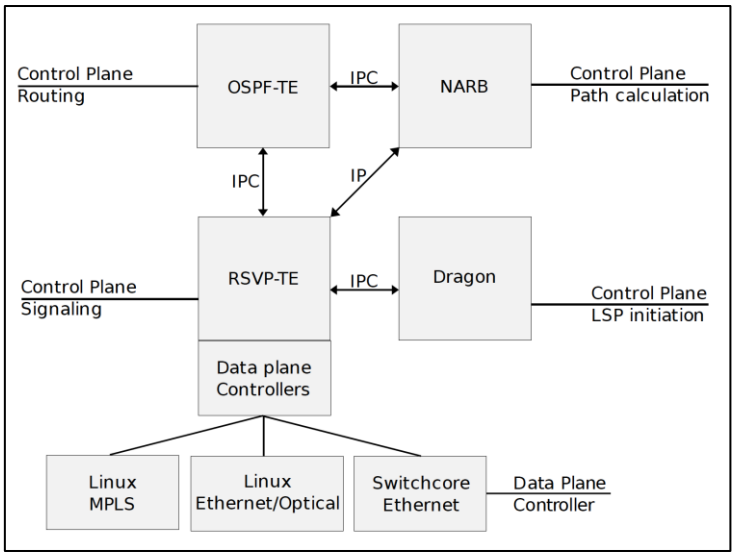

Figure 4.4 DRAGON daemons interaction [64]

- *Zebra*: This package is composed by a set of TCP/IP routing protocols. The DRAGON project has extended the open-source software distribution GNU Zebra to include GMPLS functionalities [61].
- *DRAGON*: This daemon provides the user with access by CLI to the GMPLS functionalities and control plane status monitoring.
- *RSVP-TE*: It is based on the RSVP-TE signalling protocol for the resources reservation prior to transmitting data with a specific QoS. The DRAGON project has extended the open-source package KOM RSVP-TE from the Technical University of Darmstadt [62] to include the GMPLS functionalities.
- *OSPF-TE*: This daemon has been extended from the GNU Zebra module to manage the information in DRAGON LSDs [63].

In order to access to the different modules and processes the CLI permits the configuration by the command `telnet localhost` *`port`*, where port may be 2604 (Zebra), 2611 (DRAGON), 2626 (NARB) and 2688 (RCE).

Figure 4.4 shows the communication among the different daemons/processes by IPC (Inter-Process Communication), which is a set of procedures for multiple threads in one or more processes to exchange information. It is important to remark that NARB and RSVP-TE communicate through IP [38].

4.1.3. DRAGON under User Mode Linux (UML)

The DRAGON software has been designed to be executed in several Unix PCs connected one another. The software suite uses sockets to communicate with the other network elements through GRE (Generic Routing Encapsulation) tunnels [65]. Thus, it is not possible to emulate a network topology with one only physical machine and the solution provided by DRAGON team is the use of User Mode Linux (UML).

UML is a modification of the Linux kernel that makes it operate over its own system calls interface. In this scenario, it is feasible to emulate a network topology in one only PC by running the UML as a user process in a physical machine. Then, each one of the

processes that belong to UML will be virtual machines that represent elements in the network topology (Figure 4.5).

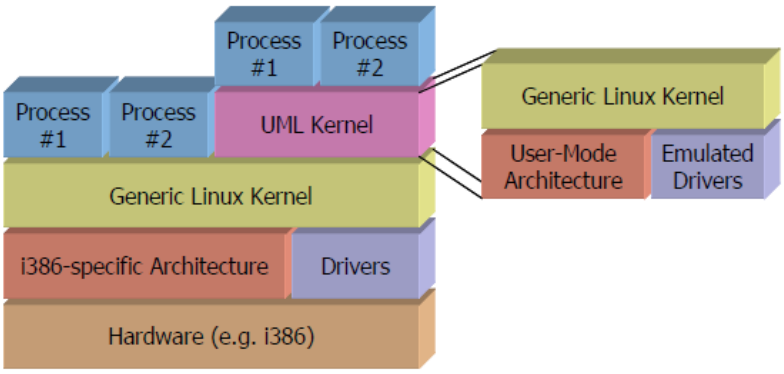

Figure 4.5 UML Architecture [3]

One of the UML applications is the UML Switch, which is a daemon running in the host for creating Ethernet switching virtual networks. This daemon connects the UML kernel with an Ethernet interface through a socket [66].

As for the GMPLS protocols, DRAGON creates a complete network in which the control plane is managed by GMPLS. The automation process provided by VNE (Virtual Network Experiment) as well as the XML syntax to configure the network topology leads DRAGON to a flexible and quickly editable tool. Thus, with the support of VNE the user can start and stop the different hosts that form the network topology and interact with them to undertake the LSPs establishment or the path computation processes.

4.1.4. Virtual Network Experiment (VNE)

VNE is an open-source software package that has been modified to run the DRAGON emulator under the UML environment. It offers an emulation environment in which from an XML file the network topology can be loaded to emulate the GMPLS operation. Thus, VNE initiates the UML instances associated with the different components of the network [66]. Figure 4.6 describes the operation of VNE, which first release was provided by the Networking Engineering group at the University of Amsterdam [67].

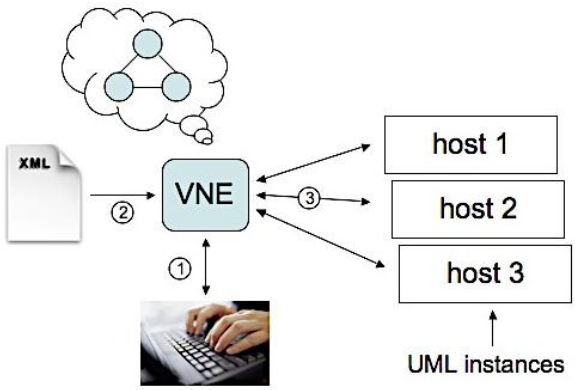

Figure 4.6 VNE execution diagram [67]

Once presented the different tools that have been used in the development of this work, Annex 3 contains a brief manual for a successful installation of the DRAGON UML environment.

4.1.5. DRAGON Emulator Structure

DRAGON emulates the control plane of a GMPLS optical network. Thus, this epigraph is aimed at presenting the process of designing, building, configuring and testing the GMPLS control plane through a virtual topology network. In order to facilitate the understanding of the DRAGON structure this point focuses on a simple topology provided by the original distribution and formed by 2 VLSRs, 2 ESs and a NARB server. Further information concerning this scenario and the XML-based network topology definition can be found in [68].

Each element in the topology has addresses associated with different purposes. The management addresses are used for the admin user to configure and monitor the nodes. Then, the loopback addresses are used when the host itself is the destination of a communication and are normally public addresses. Finally, the GMPLS control channels are established over GRE tunnels.

According to Figure 4.7, GRE tunnels are created over the *eth0* interface, which in turn supports the management plane. Then, the hypothetic data plane would be associated with the *eth1* interface.

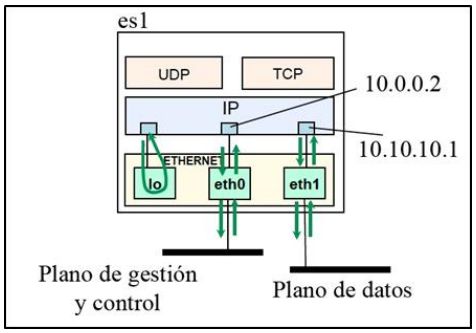

Figure 4.7 Network interfaces in a DRAGON node [3]

The Figure 4.8 presents the complete structure of the DRAGON emulator for the aforementioned scenario. The control PC (VLSRi-PC) is in charge of running the routing and signalling protocols as well as the configuration of the associated switch (VLSRi-SW). Each control channel is related to a data channel, which are identified with TE addresses. These TE addresses are not IP addresses but only reflect the relationship between the control plane and the data plane.

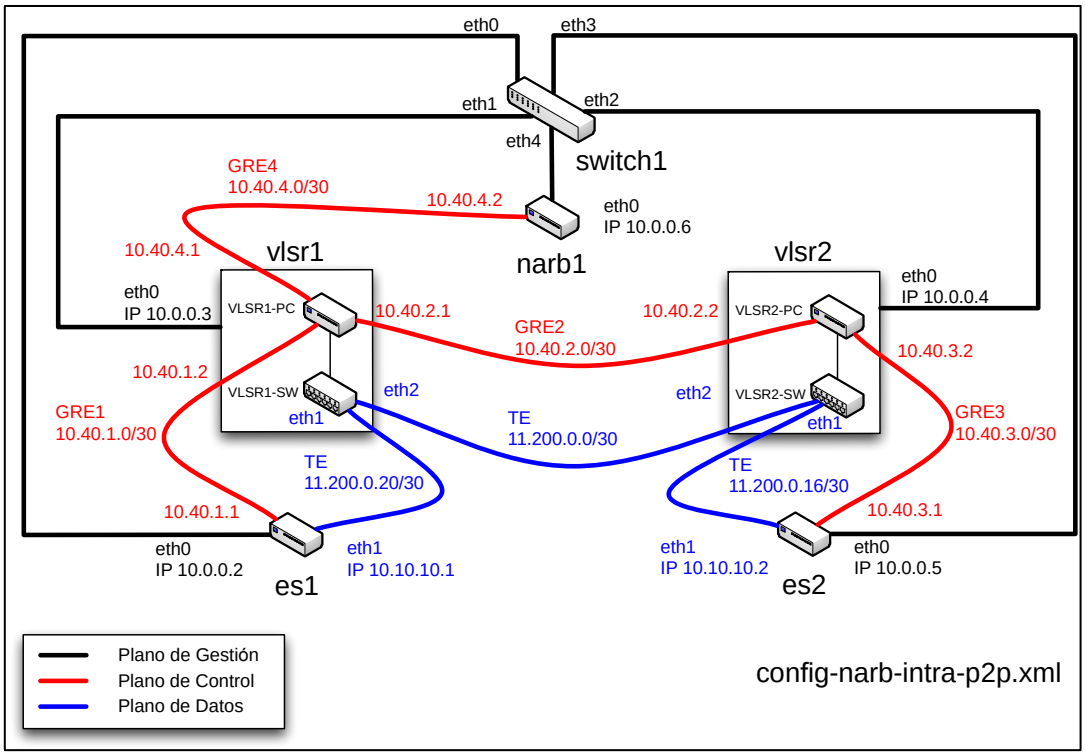

Figure 4.8 DRAGON topology for a simple scenario [68]

As far as the previous scenario in Figure 4.8 is concerned, several proves such as unidirectional LSPs creation and OSPF-TE topology information consults (`show ip ospf neighbor`, `show topology intra-domain`) were performed and the results are gathered in [68].

4.1.6. Emulation of the Deutsche Telekom (DT) Network

The Deutsche Telekom (DT) network (Figure 4.9) is one of the used in CHRON project to put in practice the research results [69].

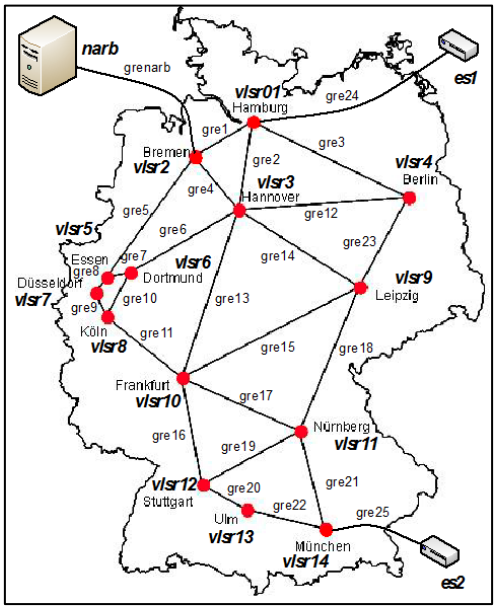

Figure 4.9 DT Network topology [69]

Hence, with the aim of preparing the emulation stage of the project this network has been designed in XML to be emulated with the DRAGON UML emulator (See Annex 4).

Prior to going deep in emulation issues, the characteristics of the PC in which the emulation has been undertaken are the presented in the next table (Table 4.1).

Manufacturer	Lenovo Z570
Model	Lenovo Win7 PC
Processor	Intel(R) Core (TM) i7-2670QM CPU @ 2.20GHz
Memory installed (RAM)	8,00 GB
Type of system	64 bits

Table 4.1 Characteristics of the PC used for emulation

According to Figure 4.9 the network consists of 14 nodes over the DRAGON components (2 ESs, NARB server and UML switch). The management addresses belong to the 192.168.1.0/16 network (Table 4.2). Then, for the data link identifiers, in other words, the TE addresses, the range of addresses belongs to 11.1.0.0./30 (See Annex 5). Lastly, the control plane uses addresses of the sub-network 10.1.0.0/30 (See Annex 6).

Node ID	Name	IP Address Management
vlsr01	Hamburg	192.168.1.1
vlsr2	Bremen	192.168.1.2
vlsr3	Hannover	192.168.1.3
vlsr4	Berlin	192.168.1.4
vlsr5	Essen	192.168.1.5
vlsr6	Dortmund	192.168.1.6
vlsr7	Düsseldorf	192.168.1.7
vlsr8	Köln	192.168.1.8
vlsr9	Leipzig	192.168.1.9
vlsr10	Frankfurt	192.168.1.10
vlsr11	Nürnberg	192.168.1.11
vlsr12	Stuttgart	192.168.1.12

vlsr13	Ulm	192.168.1.13
vlsr14	München	192.168.1.14
narb	-	192.168.1.15
es1	-	192.168.1.16
es2	-	192.168.1.17

Table 4.2 Management plane addresses in DT network

4.1.6.1. VNE Launching of the DT Network

After initiating the VNE by indicating the DT Network topology to be loaded, the command `start all` (Figure 4.10) starts the virtual instances corresponding to the network elements.

Figure 4.10 Launching the DT Network with VNE

Then, the emulation continues by launching a console for each one of the elements forming the network topology (`xterm all`) (Figure 4.11).

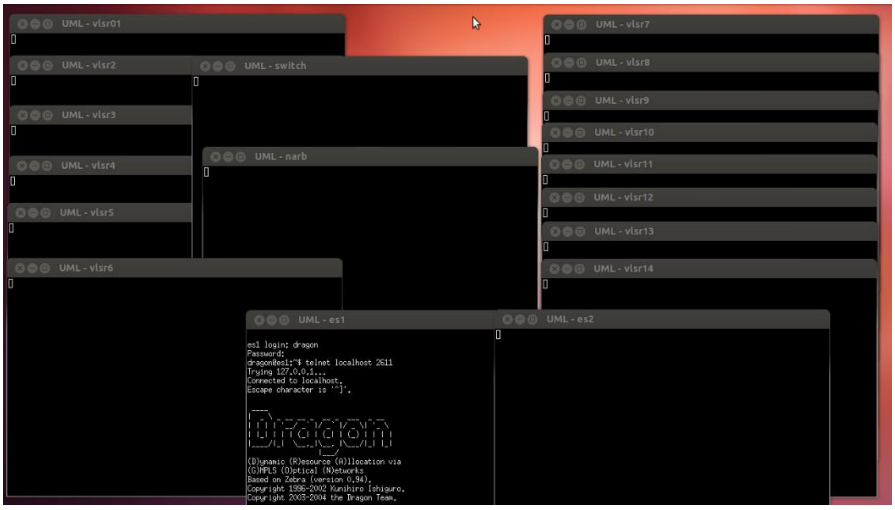

Figure 4.11 X-term environment emulating the DT Network

4.1.6.2. Tests

The tests carried out in this epigraph can be summarized as the creation of both unidirectional and bidirectional LSPs, the operation with the ERO object for the NARB server to route through a strict path, and finally the analysis of the RSVP-TE messages exchanged. Last but not least it is important to mention that the OSPF-TE messages cannot be analysis from the logs stored in DRAGON directories, although an important analysis of the OSPF-TE protocol is reflected in [68] with the use of NetGUI [70].

- **narb_test tool**

The NARB server provides the route through which the LSP will be established by means of the narb_test tool. For instance, considering the vlsr01 as source and the vlsr11 as destination, the ERO can be obtained with the following command:

```
/usr/local/dragon/sbin$ ./narb_test -P 2609 -S 192.168.1.1 -D
192.168.1.11
```

```
NARB@[2011/09/26 08:18:02] : Request successful! ERO returned...
NARB@[2011/09/26 08:18:02] : HOP-TYPE [strict]: 11.1.3.1
NARB@[2011/09/26 08:18:02] : HOP-TYPE [strict]: 11.1.3.2
NARB@[2011/09/26 08:18:02] : HOP-TYPE [strict]: 11.1.23.1
NARB@[2011/09/26 08:18:02] : HOP-TYPE [strict]: 11.1.23.2
NARB@[2011/09/26 08:18:02] : HOP-TYPE [strict]: 11.1.18.1
NARB@[2011/09/26 08:18:02] : HOP-TYPE [strict]: 11.1.18.2
dragon@narb:/usr/local/dragon/sbin$
```

Figure 4.12 Default ERO object in NARB server

Figure 4.12 shows the result achieved after running the previous command. The consult consists of a list of TE addresses that determines the route to be followed. Thus, the route will be determined by vlsr01, vlsr4, vlsr9 and vlsr11 (See information highlighted in blue in Table 8.1 of Annex 5). It is noteworthy to mention that by default the RCE computes the route taking into consideration the lower number of hops.

- **Unidirectional LSP establishment**

Figure 4.13 exposes the commands needed to create and put in service an unidirectional LSP, called '*lspuni'*, between vlsr01 (192.168.1.1) and vlsr11 (192.168.1.11).

In the first place, working from the vlsr01 console, a name is chosen for the new LSP, in this case 'lspuni'. The next command sets the source IP address, the source port, the destination IP address and the destination port. Then, the bandwidth is determined, in this case 100 Mbps, together with the switching capability (L2SC, level 2 switching capability), the encoding type (Ethernet due to emulating this type of topology) and the data type (Ethernet, by the same token of the previous parameter). Those parameters play an important role during the RSVP-TE signalling process. The command in the fourth place sets the identifier for the VLAN used to create the LSP. On this occasion the value chosen has been 125, although in DRAGON it is possible to choose any value from 100 to 200. Furthermore, unlike the GMPLS label, which presents a local significance at node level, this identifier presents a global significance throughout the LSP as a whole [68].

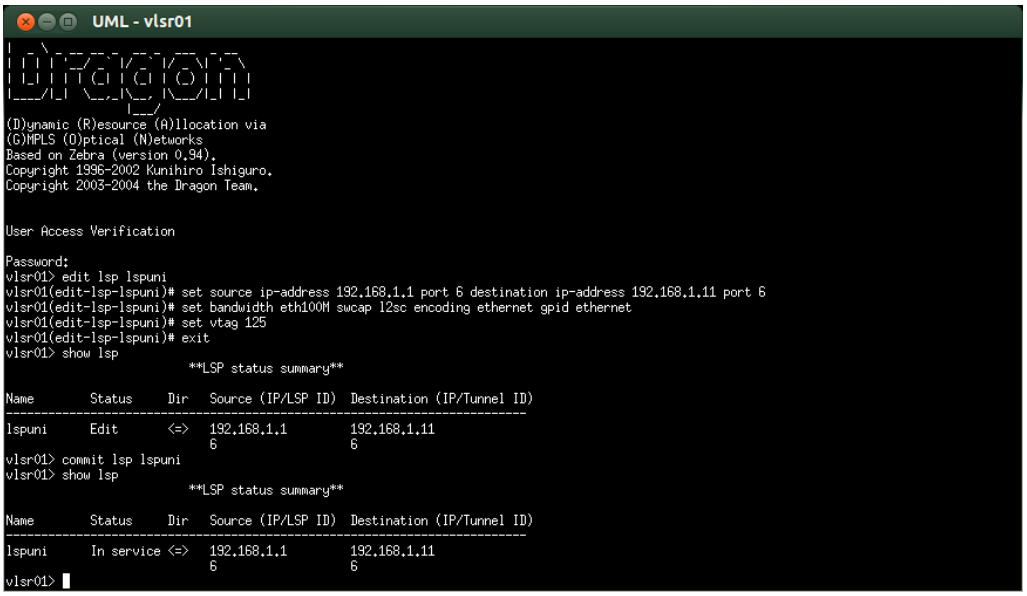

Figure 4.13 Unidirectional LSP establishment

Following with the commands to establish an LSP, the next step is to finalize the edition process (`exit`), establish the LSP (`commit lsp` *`name`*) and check that it is in service (`show lsp/show lsp` *`name`*).

Now, working from the vlsr11 console, it is possible to access to the information that this node has regarding the recently-created LSP. In Figure 4.14, the field '*Dir*' is fulfilled with the symbol '=>' to specify that the LSP 'lspuni' is unidirectional with destination the vlsr11 node.

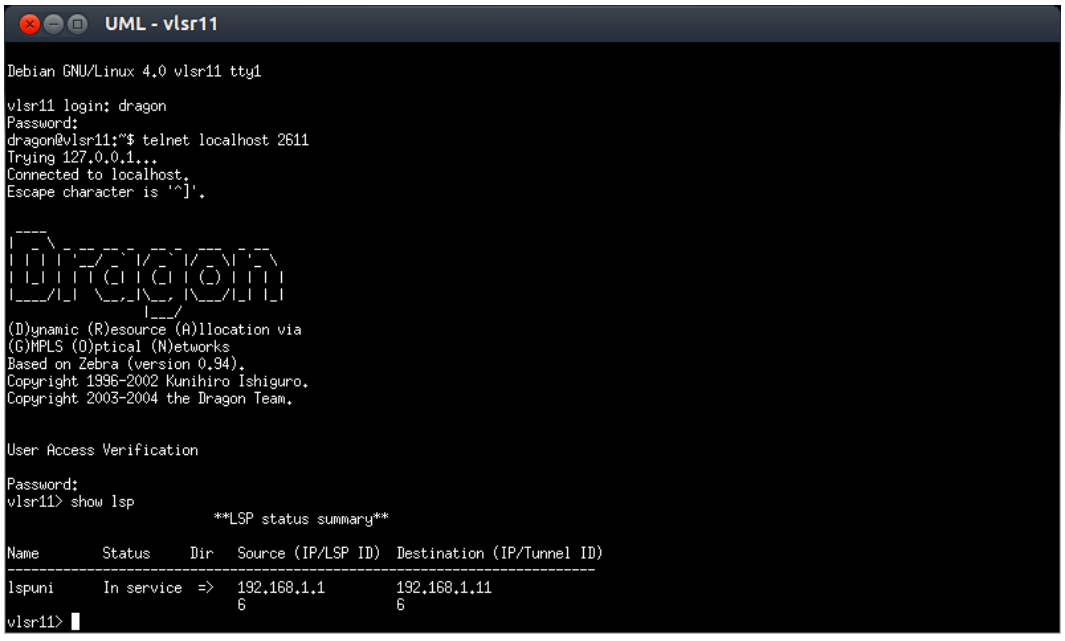

Figure 4.14 LSP status in destination node

In order to take a step forward, the NARB server has been consulted to check the status of the LSP 'lspuni'. Thus, once connected to NARB by CLI in port 2626, when running the command `show lsp` *`name`* it is possible to consult the information regarding all the LSPs established (Figure 4.15).

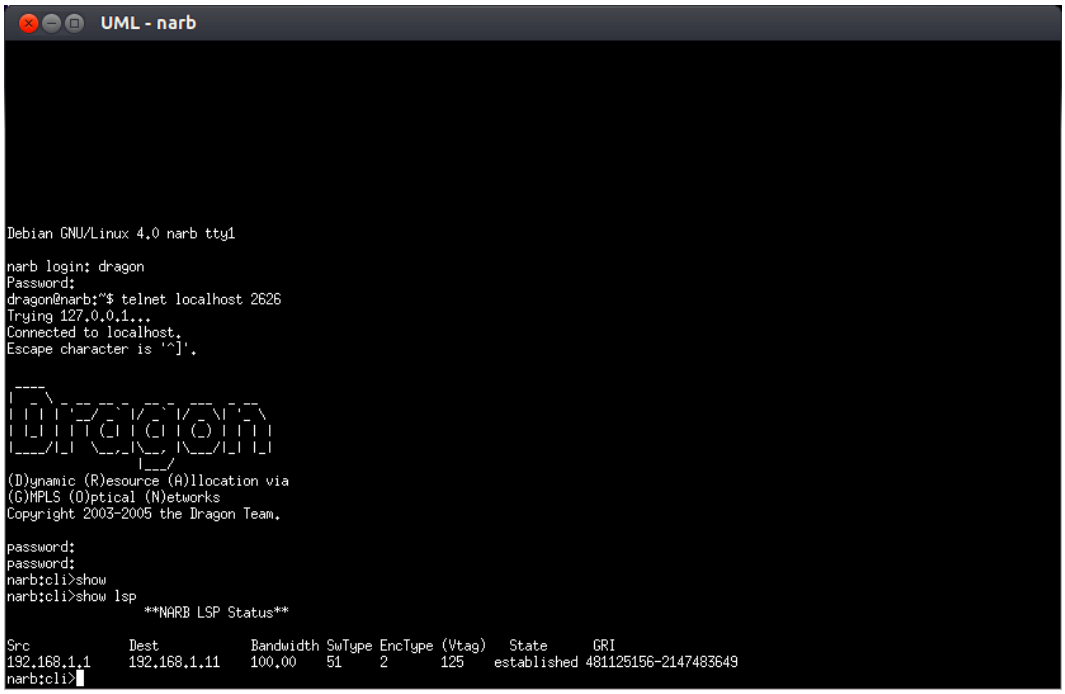

Figure 4.15 LSP status in NARB server

- **RSVP-TE messages exchanged**

Once created the new LSP, the next step consists of analysing the RSVP-TE information exchanged throughout the emulation. Such information presented at this point makes reference to the Path and Resv messages when establishing the LSP 'lspuni' between vlsr01 and vlsr11. In order to get this information one only has to run the command `pico /var/log/RSVPD.log` in the desired node.

The signalling process starts when the Ingress LSR (vlsr01) sends a Path message to the Egress LSR (vlsr11) to inform about the LSP required and the traffic to be transmitted. This Path message is propagated according to the default route defined in the routing table of each intermediate node (Figure 4.16).

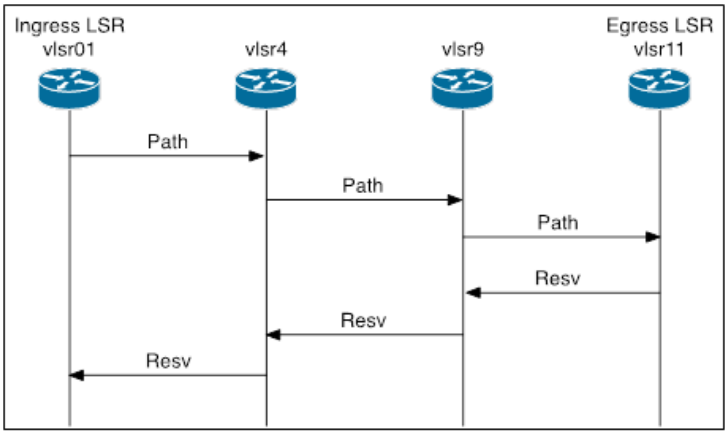

Figure 4.16 RSVP-TE messages during unidirectional LSP establishment

In this case, the route followed is the one provided by the NARB server. Due to the backward reservation process, when the Path message arrives at the Egress LSR (vlsr11), it checks the resources needed for the transmission and sends a Resv

message back following the same route (Figure 4.16). According as this message propagates the resources are reserved in the intermediate LSRs, and then, when the Resv message reaches the Ingress LSR, the LSP can be considered as established.

In case of a node cannot reserve the requested resources it will send a PathErr message to the Ingress LSR and a ResvErr message to the Egress LSR by indicating the failure. If this occurs, the signalling process would start again.

- *Path Message*

This message is created by the vlsr01 and sent to the vlsr4 by the control channel GRE3, then vlsr4 forwards it to vlsr9 through the control channel GRE23, and finally, vlsr9 sends the message to vlsr11 by GRE18 (See information highlighted in blue in Table 8.2 of Annex 6). Further information regarding this message content can be found in Chapter III. Figure 4.17 presents the Path message when it traverses the vlsr4.

Figure 4.17 RSVP-TE Path message traversing vlsr04

- *Resv Message*

If all the required parameters are available in the Egress LSR, this message is created by the vlsr11 and sent to the vlsr9 by GRE18, then vlsr9 sends the message to vlsr4 by GRE23, and finally, vlsr4 transmits it to the vlsr01 by GRE3. Figure 4.18 presents the Resv message when it traverses the vlsr4.


UML - vlsr4
07:46:13.297 ********** new message received **********
07:46:13.297 gre23 received MSG from 10.1.23.2 : RESV 1 1 ttl:63 length:168
MESSAGE_ID:flags:0 epoch:13332302 id:0
SESSION:192.168.1.11/6/16885952
RSVP_HOP:Hop address: 10.1.23.2LIH: 3 TLV: Type: 1 Length: 8 Value: IPv4: 1846
20801
TIME_VALUES:30000
STYLE:FF
FDESC:192.168.1.1/6 L:16 length:32 CLS r:100.000 b:100.000 p:100.000 m:100 M:15
00
DRAGON_EXT_INFO: [(1: Service Confirmation ID: ucid=481125156, seqnum=214748364
9)(5: MonNodeList: -192.168.1.1--192.168.1.4--192.168.1.9--192.168.1.11-)]
07:46:13.297 found Session: 192.168.1.11/6/16885952
07:46:13.297 creating Hop:10.1.23.2 via gre23
07:46:13.483 received RESV for 192.168.1.11/6,16885952
07:46:13.483 found Session: 192.168.1.11/6/16885952
07:46:13.483 processing flow descriptor 192.168.1.1/6 L:16 length:32 CLS r:100.0
00 b:100.000 p:100.000 m:100 M:1500
07:46:13.484 matched PSB:192.168.1.1/6 from PHopSB:10.1.3.1[5] via gre3
07:46:13.484 creating OutISB at gre23
07:46:13.484 creating RSB(nhop):10.1.23.2
07:46:13.484 stored RSB with ID 0 at recv hash value 0 from Hop:10.1.23.2 via gr
e23
07:46:13.484 OIatPSB:192.168.1.1/6 -> gre23 addRSB, count is 1
07:46:13.484 MPLS: allocated label 16
07:46:13.485 MPLS: storing output label 16 for OIatPSB:192.168.1.1/6 -> gre23
07:46:13.485 MPLS: binding outgoing label 16 to input label 16 from label space
0
07:46:13.485 new local eff. flowspec: length:32 CLS r:100.000 b:100.000 p:100.00
0 m:100 M:1500
07:46:13.485 Hop address: 10.1.3.1LIH: 5 marked for refresh: PSB:192.168.1.1/6 f
rom PHopSB:10.1.3.1[5] via gre3
07:46:13.485 calculating RESV for 10.1.3.1
07:46:13.486 PSB PSB:192.168.1.1/6 from PHopSB:10.1.3.1[5] via gre3 calculated f
orward flowspec length:32 CLS r:100.000 b:100.000 p:100.000 m:100 M:1500
07:46:13.486 creating RESV for PHOP 10.1.3.1
07:46:13.486 setting refresh timer to 30.000 sec at Hop:10.1.3.1 via gre3
07:46:13.486 chose ID 0 idSendCount is 1 at Hop:10.1.3.1 via gre3
07:46:13.486 stored PHopSB with ID 0 at send hash value 0 at Hop:10.1.3.1 via gr
e3
07:46:13.486 gre3 sends MSG to 10.1.3.1 : RESV 1 1 ttl:63 length:168
MESSAGE_ID:flags:0 epoch:3116949 id:0
SESSION:192.168.1.11/6/16885952
RSVP_HOP:Hop address: 10.1.3.2LIH: 5 TLV: Type: 1 Length: 8 Value: IPv4: 18461
5681
TIME_VALUES:30000

Figure 4.18 RSVP-TE Resv message traversing vlsr04

- **Bidirectional LSP establishment**

Figure 4.19 contains the commands to create and put in service a bidirectional LSP, called '*bidi*', between vlsr01 (192.168.1.1) and vlsr11 (192.168.1.11).

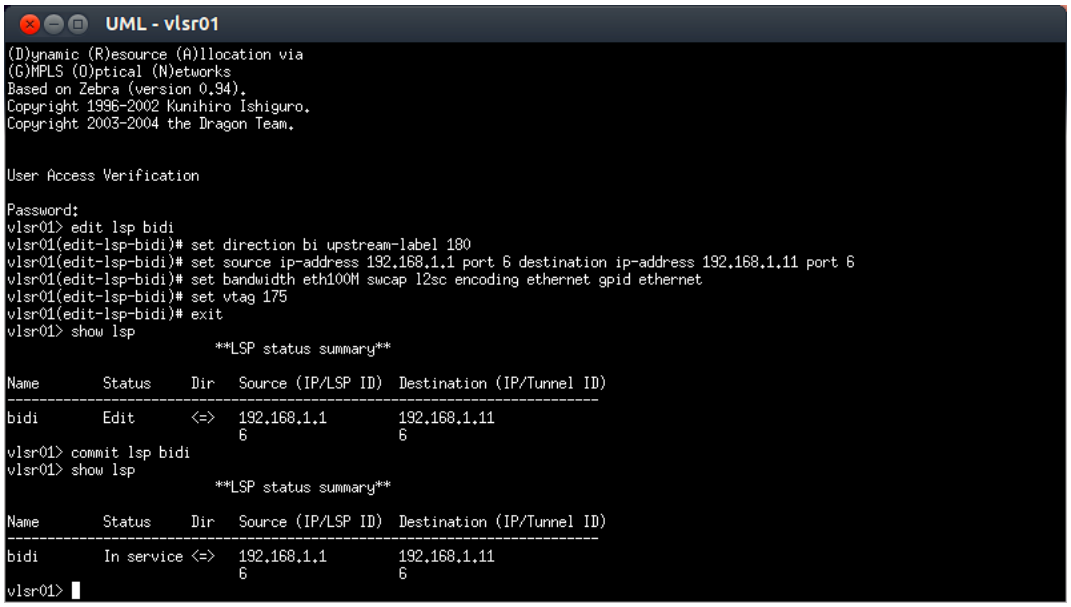

Figure 4.19 Bidirectional LSP Establishment

As it occurs for the unidirectional LSP establishment, the first command determines the name chosen for the new LSP, in this case 'bidi'. Then, within the edition process, the next command sets the upstream label, which is the label for the traffic from vlsr11 to vlsr01, in this case 180. This label also has local significance although it must be free in all the nodes involved in the route. The next steps consists in setting the source IP address, the source port, the destination IP address and the destination port. Then, the bandwidth is determined, in this case 100 Mbps, together with the switching capability (L2SC, level 2 switching capability),

the encoding type (Ethernet due to emulating this type of topology) and the data type (Ethernet, by the same token of the previous parameter). The command in the fifth place sets the identifier for the VLAN used to create the LSP. On this occasion the value chosen has been 175. Following with the commands to establish this bidirectional LSP, the next step is to finalize the edition process `(exit)`, establish the LSP `(commit lsp)` and check that it is in service `(show lsp/show lsp` *`name`*`)`. Just to check the recently-created LSP status in vlsr11, presents this information where now the field '*Dir*' has been fulfilled with the symbol '<=>'.

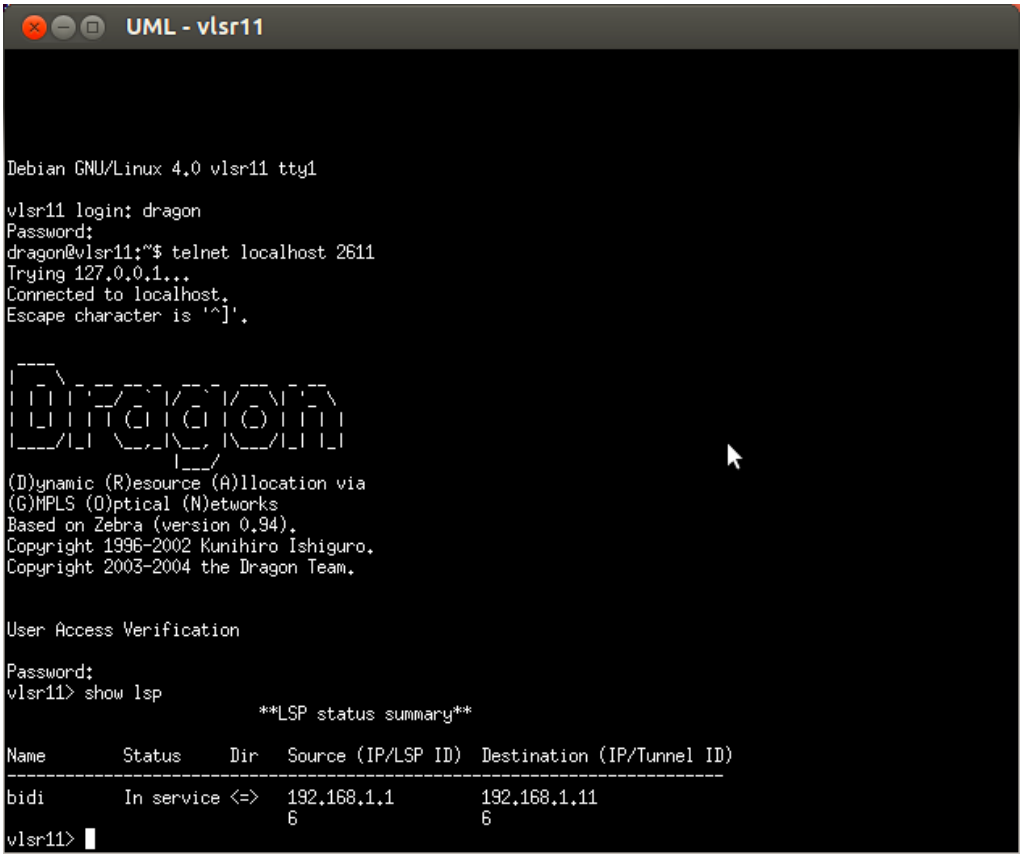

Figure 4.20 Bidirectional LSP status in destination node

- **Explicit Route Object (ERO)**

Another important functionality of DRAGON is to establish an LSP by forcing the route to be followed within the Path message. That is achieved by the ERO object such as it was seen in Chapter III. This explicit route may be either strict, all the intermediate LSR are specified, or loose, the intermediate nodes can decide the next hop.

In the network topology under study, DT Network, the links present different lengths so the transmission delay will vary in function of the route followed. The routes with lower delay from a source node to a destination node have been calculated offline and those routes constitute the ERO to be included in the Path message [68].

According to the aforementioned algorithm the route with the lowest delay between nodes vlsr01 and vlsr11 is *vlsr01 – vlsr3 – vlsr10 –vlsr11* [68]. Hence, the next step is to create a new LSP by making use of the ERO object, which will

contain the previous route. After accessing to the NARB console, the Figure 4.21 presents the commands to manage the LSP establishment under the ERO restriction.

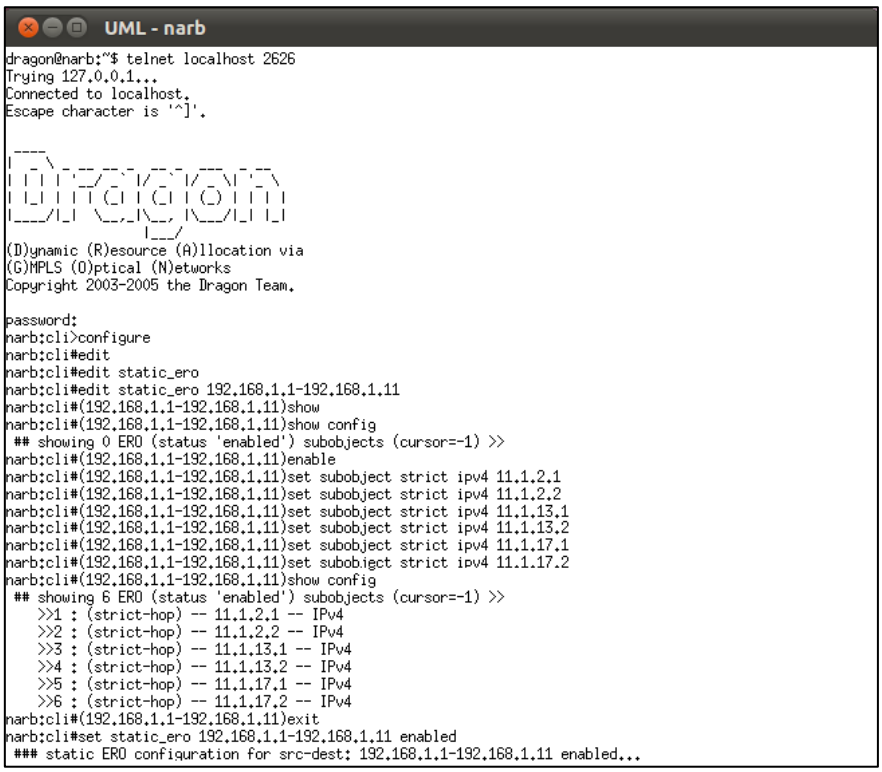

Figure 4.21 ERO object creation in NARB server

In order to configure the strict route, it is compulsory to specify the TE data links identifiers according to the information highlighted in green in Table 8.1 of Annex 5. In this way, the control messages will be sent from vlsr01 to vlsr3 by GRE2, from vlsr3 to vlsr10 by GRE13, and finally, from vlsr10 to vlsr11 by GRE17 (See information highlighted in green in Table 8.2 of Annex 6). Once generated the route in the NARB server the ERO can be consulted like in Figure 4.12. Lastly, the LSP will be established as it was explained previously [68].

4.2 DRAGON: SNMP Emulation in CHRON

After the theoretical presentation of SNMP in Section 3.2 of Chapter III, at this point an emulation environment of SNMP integrated in the DRAGON emulator is presented as management approach in an optical network.

The Linux library chosen to undertake these tests has been AGENT++/SNMP++, which is based on previous work developed in Hewlett-Packard (HP) and allows the addition of SNMP functionalities to a concrete network scenario [71]. AGENT++ is a set of C++ classes which provides a complete protocol engine for the development of SNMP agents. AGENT++ is a multilingual API which supports SNMPv1, SNMPv2c, and SNMPv3, although in this Master of Research Thesis SNMPv3 has been disabled not to overload the virtual emulation environment.

In order to get an emulation scenario where both the GMPLS and the SNMP protocols are implemented together, the file system of the DRAGON emulator has been modified prior to installing the aforementioned library (See Annex 7).

SNMP++ is able to retrieve any OID, but it does not support reading MIB files and use the human readable notation. This way, the OIDs chosen in this Master of Research Thesis have been those provided by CISCO for Optical Monitoring tasks [72]. Concretely, the OIDs used are the ones presented below (Table 4.3):

Monitored Parameter	OID
cOpticalMonDirection	1.3.6.1.4.1.9.9.264.1.1.1.1.1
cOpticalMonLocation	1.3.6.1.4.1.9.9.264.1.1.1.1.2
cOpticalMonParameterType	1.3.6.1.4.1.9.9.264.1.1.1.1.3
cOpticalParameterValue	1.3.6.1.4.1.9.9.264.1.1.1.1.4
cOpticalParamHighAlarmThresh	1.3.6.1.4.1.9.9.264.1.1.1.1.5
cOpticalParamLowAlarmThresh	1.3.6.1.4.1.9.9.264.1.1.1.1.9

Table 4.3 OIDs of optical monitored parameters

In order to take advantage of the examples provided by the AGENT++/SNMP++ distribution, an implementation for an ATM scenario has been modified by adding the previous OIDs for meeting the CHRON use cases purposes.

Although, both AGENT++ and SNMP++ packages are provided within the DRAGON emulator modified file system (Annex 7), the main changes are presented below to facilitate the tracking of this section. Thus, in function `init()` of *agent.cpp* file the code lines in Figure 4.22 have been added.

```
mib.add(new
SnmpDisplayString("1.3.6.1.4.1.9.9.264.1.1.1.1.1",READWRITE,new
OctetStr("162.168.1.5")));


mib.add(new
SnmpDisplayString("1.3.6.1.4.1.9.9.264.1.1.1.1.2",READWRITE,new
OctetStr("vlsr5")));


mib.add(new
SnmpDisplayString("1.3.6.1.4.1.9.9.264.1.1.1.1.3",READWRITE,new
OctetStr("Int_Units")));


mib.add(new
SnmpDisplayString("1.3.6.1.4.1.9.9.264.1.1.1.1.4",READWRITE,new
OctetStr("32.5dB/0.1nm")));


mib.add(new
SnmpDisplayString("1.3.6.1.4.1.9.9.264.1.1.1.1.5",READWRITE,new
OctetStr("35dB/0.1nm")));


mib.add(new
SnmpDisplayString("1.3.6.1.4.1.9.9.264.1.1.1.1.9",READWRITE,new
OctetStr("25dB/0.1nm")));
```

Figure 4.22 SNMP Agent modifications to cover additional OIDs

The remaining of this section is divided into two main parts. In the first place, a SNMP Trap scenario is presented as implementation in an emulation scenario of SNMP CHRON use case 1 described in Chapter III. Secondly, a polling (request/response) SNMP scenario is presented in order to understand better the SNMP CHRON use case 2 also described in the mentioned chapter.

Before delving into the use cases implementation it is noteworthy to mention that the topology used also corresponds to the DT Network (Figure 4.9). Moreover, the NARB server will play the role of SNMP manager and any of the VLSRs will act as SNMP agent.

4.2.1. Emulation of SNMP CHRON Use Case 1: Trap

In this use case the monitor located in vlsr5 (192.168.1.5:51433) senses that the OSNR level decreases significantly, so its value becomes lower than a pre-fixed value. Then, the CDS, which would be placed in the NARB server (192.168.1.15:10162) acting as SNMP Manager, is notified of this event.

As for the modifications to carry out the emulation of this use case, in the Agent side the trap to notify of the OSNR decrease has been defined (`#define PAYLOAD "OSNR_DECREASE", OID: 1.3.6.1.4.1.11.2.16.2`), and on the Manager side the port destination has been set to 10162.

Once launched the Manager in the NARB server, it will be listening to incoming notifications from the other nodes in the network. Figure 4.23 shows the receipt of the notification considered from the vlsr5 (Figure 4.24).

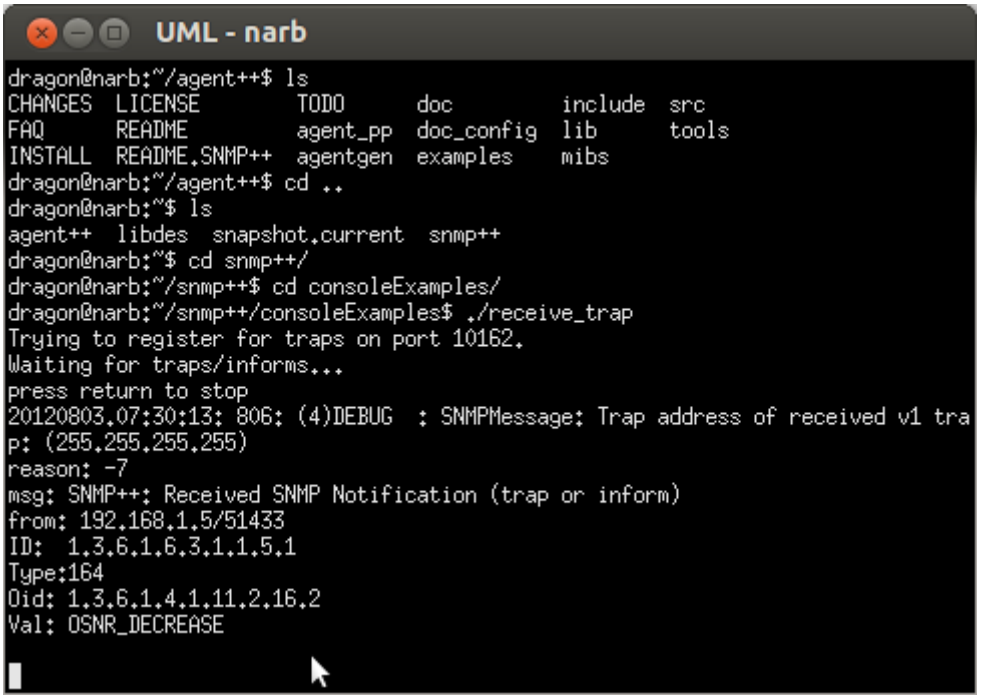

Figure 4.23 SNMP Manager when receiving a notification

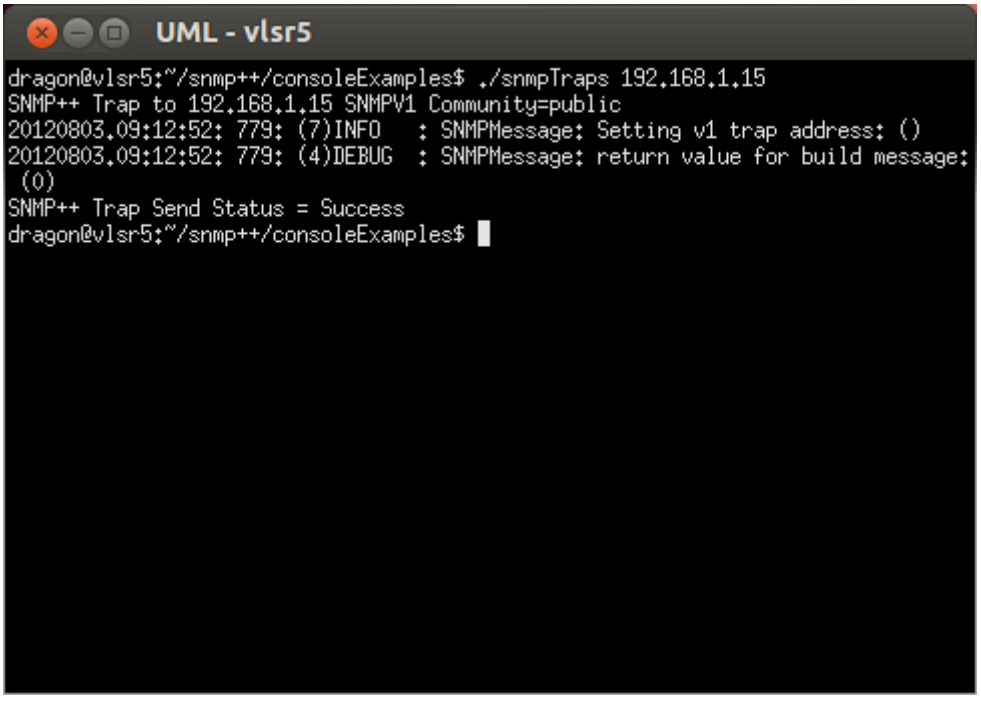

Figure 4.24 SNMP Agent when sending a notification

4.2.2. Emulation of SNMP CHRON Use Case 2: Polling

This second use case presents the procedure in which the SNMP Manager (192.168.1.15) requests for the OSNR value to the SNMP Agent located in vlsr5 (192.168.1.5).

In this case, the NARB server asks for the OSNR value (Figure 4.25) to the vlsr5 (Figure 4.26), which plays the role of monitored node.

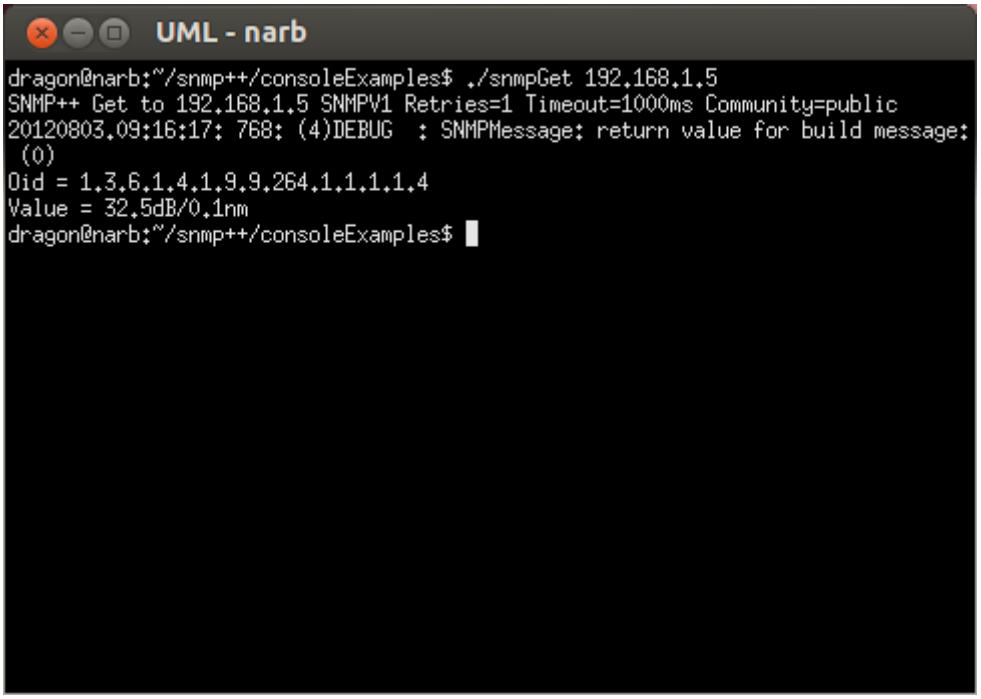

Figure 4.25 SNMP Manager when sending a request

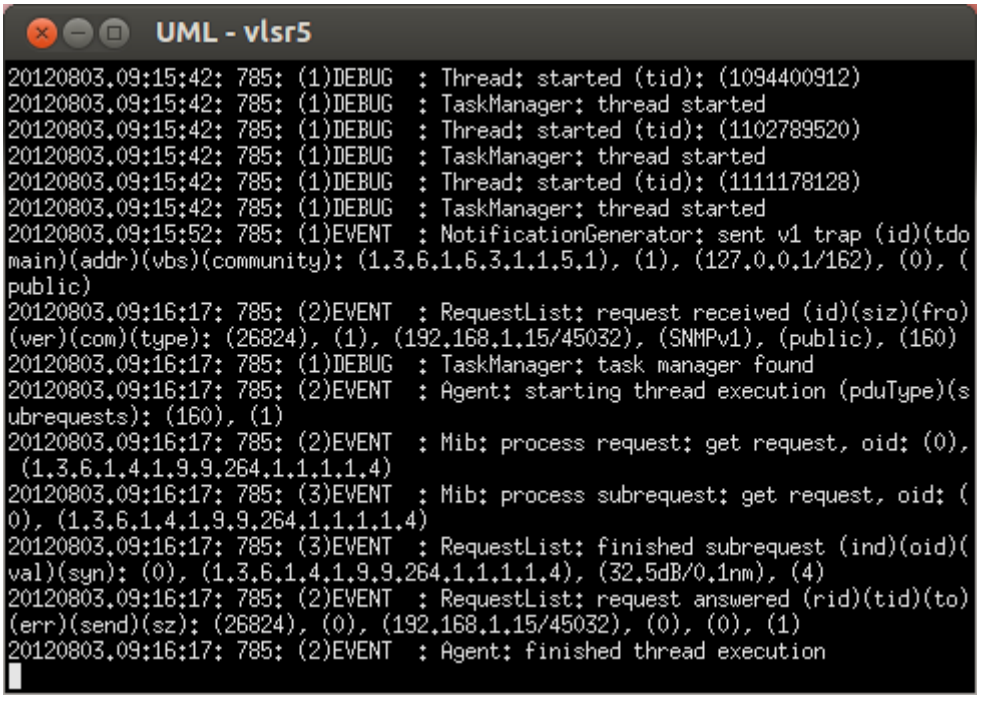

Figure 4.26 SNMP Agent when sending a response

Apart from the SNMP scenarios emulated here, there are a lot of other possible situations to be emulated with the AGENT++/SNMP++ package, but due to being coherent with the use cases definition in CHRON, only the research results in the framework of this project have been put in practice in this chapter.

4.3 Conclusions

Within the conclusions of this chapter, there is point in remarking that the main contribution has been the fact of putting in practice all the research results found in the different RFCs, article and documents consulted throughout the elaboration of this Master of Research Thesis.

The chance of manipulating an emulator provides us with important advantages and benefits, such as cost and deployment savings, at the moment of testing the results achieved during the research stage or even to check and better understand them.

Making reference to the CHRON project, this chapter provides an utmost input for the incoming emulation/testbed of the project since important know-how has been obtained, which together the contributions of the rest of partners in the consortium will lead to important findings to be implemented in real optical transport networks.

Chapter V – Conclusions and Future Research Lines

5 Conclusions and Future Research Lines

This fifth chapter gathers the conclusions, the future research lines and the related works to this Master of Research Thesis.

5.1 Conclusions

Although a sub-section devoted to conclusions has been added at the end of the majority of chapters that form this report, this section briefly presents a general conclusive point of view of this Master of Research Thesis.

In this work we have studied and emulated control and management strategies for heterogeneous optical networks within the European project CHRON framework. Thus, after introducing the CHRON project and its centralized network architecture, an exhaustive literature review has provided the grounds to analyse in an emulated environment both control and management approaches in optical transport networks.

In this emulation scenario, the pillar has been the DRAGON emulator through which the emulation of a real network (DT Network) controlled by GMPLS and managed by SNMP has been implemented. In this respect, the manipulation and the extension of this emulator has been a hardworking and demanding challenge.

Apart from the preparation of the emulation scenario, other tasks have been successfully completed with this contribution, such as an exhaustive state of the art of cognitive network architectures, and the definition of a cognitive specification language to encode the messages exchanged between the control and management plane and the cognitive decision system, core on the CHRON architecture.

Above all, the main contribution has been the achievement of an important know-how of the control and management approaches in optical networks in the DRAGON emulated scenario to successfully tackle the incoming CHRON testbed tasks. Definitely, this Master of Research Thesis together with the contributions of the rest of partners in the CHRON consortium will lead the current control and management approaches to be adapted to the emerging cognitive optical transport networks.

5.2 Future Research Lines

In this section some points that require more research are presented as possible future lines.

In the first place, extensions to the GMPLS protocols, such as OSPF-TE and RSVP-TE, should be studied in order to meet CHRON requirements when controlling and managing a cognitive heterogeneous reconfigurable optical network. Some initial steps have been taken in this respect as it is presented in [68] to define new TLVs and sub-TLVs for these protocols operation in optical transport networks. For this purpose, the source code of the DRAGON processes and daemons presented in Chapter IV should be modified.

Secondly, in current optical networks appears another control approach based on PCEP (Path Computation Element Protocol). Some tests have been undertaken with the Linux library *libpcep* [74] with the aim of integrating this approach in the DRAGON emulator, but at moment this is only a toy problem in terms of development and implementation.

Finally, another line to achieve important research results is the optical protection and restoration, which would be an important improvement for the CHRON emulation scenario when considering the establishment of both primary and back-up LPSs.

5.3 Related Works

In this section, the works related to the Master of Research Thesis are presented by distinguishing three types of contributions in which the author has taken part.

In the first place, according to the Grant Agreement of CHRON project, the results achieved during the project lifetime must be reflected in different deliverables. The following list presents by splitting into work-packages those **deliverables** in which the author has participated:

WP1 – Project Management

- Deliverable 1.1, "Internal Project Website", CHRON Project, FP7/2007-2013, Ref. 258644, September 2010.
- Deliverable 1.3, "First Annual Activity Report", CHRON Project, FP7/2007-2013, Ref. 258644, August 2011.
- Deliverable 1.4, "Second Annual Activity Report", CHRON Project, FP7/2007-2013, Ref. 258644, September 2012.

WP2 – Network and Service Definition

- Deliverable 2.2, "Initial Network and Service Control Architecture", CHRON Project, FP7/2007-2013, Ref. 258644, November 2011. Available in www.ict-chron.eu
- Deliverable 2.3, "Final network architecture and operation definition", CHRON Project, FP7/2007-2013, Ref. 258644, July 2012. Available in www.ict-chron.eu

WP3 – Cognitive Decision System

- Deliverable 3.1, "Specification of the Architecture and Methods of the Cognitive Decision System", CHRON Project, FP7/2007-2013, Ref. 258644, July 2011. Available in www.ict-chron.eu
- Deliverable 3.2, "Architecture of the Cognitive Decision System", CHRON Project, FP7/2007-2013, Ref. 258644, January 2012. Available in www.ict-chron.eu
- Deliverable 3.3, "The core of the cognitive system: The cognitive process and the knowledge engineering system", CHRON Project, FP7/2007-2013, Ref. 258644, July 2012. Available in www.ict-chron.eu
- Deliverable 3.4, "CHRON cognitive decision system simulator", CHRON Project, FP7/2007-2013, Ref. 258644, July 2012.

WP4 – Network Monitoring System and QoT Evaluation Techniques

- Deliverable 4.1, "Feedback system between the monitoring and the cognitive systems", CHRON Project, FP7/2007-2013, Ref. 258644, December 2011.

WP5 – Control and Management System

- Deliverable 5.1, "Interface Definition", CHRON Project, FP7/2007-2013, Ref. 258644, July 2011. Available in www.ict-chron.eu

- Deliverable 5.2, “Control plane architecture and protocol definition”, CHRON Project, FP7/2007-2013, Ref. 258644, May 2012. Available in www.ict-chron.eu

WP7 – Dissemination, Exploitation and Standardization

- Deliverable 7.1, “External project website”, CHRON Project, FP7/2007-2013, Ref. 258644, July 2010.
- Deliverable 7.2, “First report on dissemination and exploitation plan”, CHRON Project, FP7/2007-2013, Ref. 258644, July 2011. Available in www.ict-chron.eu
- Deliverable 7.3, “Initial business model plan”, CHRON Project, FP7/2007-2013, Ref. 258644, July 2012.
- Deliverable 7.4, “Second report on dissemination and exploitation plan”, CHRON Project, FP7/2007-2013, Ref. 258644, July 2012. Available in www.ict-chron.eu

In the second place, the next list consists of the chronological-ordered **publications** in which the author has participated:

- De Miguel, I.; Durán, R.J.; Abril, E. J.; Lorenzo, R.M.; Atallah, N.; Sánchez, D.; Guerrero, N.; Zibar, D.; Tafur, I.; Angelou, M.; Klonidis, D.; Tomkos, I.; Saradhi, C. V.; Salvadori, E.; Ye, Y.; Chen, M.;, "The European FP7 CHRON Project: Towards Cognitive Heterogeneous Reconfigurable Optical Networks", Broadband Communications, Networks, and Systems (BROADNETS), 2010 7th International Conference on, October 2010.
- Duran, R.J.; Fernandez, N.; de Miguel, I.; Angelou, M.; Sanchez, D.; Aguado, J.C.; Jimenez, T.; Fernandez, P.; Merayo, N.; Atallah, N.; Lorenzo, R.M.; Tomkos, I.; Abril, E.J.; , "Advantages of using cognition when solving impairment-aware virtual topology design problems," Transparent Optical Networks (ICTON), 2011 13th International Conference on , vol., no., pp.1-4, 26-30 June 2011.
- Jimenez, T.; de Miguel, I.; Aguado, J.C.; Duran, R.J.; Merayo, N.; Fernandez, N.; Sanchez, D.; Fernandez, P.; Atallah, N.; Abril, E.J.; Lorenzo, R.M.; , "Case-Based Reasoning (CBR) to estimate the Q-factor in optical networks: An initial approach," Networks and Optical Communications (NOC), 2011 16th European Conference on , vol., no., pp.181-184, 20-22 July 2011.
- N. Fernandez, R. Durán, I. de Miguel, J. Aguado, T. Jiménez, M. Angelou, D. Sánchez, P. Fernández, N. Merayo, N. Atallah, R. Lorenzo, I. Tomkos, and E. Abril, "Cognitive Genetic Algorithms to Design Impairment-Aware Virtual Topologies in Optical Networks," in Optical Fiber Communication Conference, OSA Technical Digest (Optical Society of America, 2012), paper OW3A.1.
- T. Jiménez, J. Aguado, I. de Miguel, R. Durán, N. Fernandez, M. Angelou, D. Sánchez, N. Merayo, P. Fernández, N. Atallah, R. Lorenzo, I. Tomkos, and E. Abril, "A Cognitive System for Fast Quality of Transmission Estimation in Core Optical Networks," in Optical Fiber Communication Conference, OSA Technical Digest (Optical Society of America, 2012), paper OW3A.5.
- Fernandez, N.; Duran, R.J.; de Miguel, I.; Aguado, J.C.; Jimenez, T.; Angelou, M.; Sanchez, D.; Fernandez, P.; Merayo, N.; Atallah, N.; Lorenzo, R.; Tomkos, I.; Abril, E.J.; , "Cognitive algorithm to solve the impairment-aware virtual topology design problem in reconfigurable optical networks," Cognitive

Methods in Situation Awareness and Decision Support (CogSIMA), 2012 IEEE International Multi-Disciplinary Conference on , vol., no., pp.170-173, 6-8 March 2012.

- Jimenez, T.; Aguado, J.C.; de Miguel, I.; Duran, R.J.; Fernandez, N.; Angelou, M.; Sanchez, D.; Merayo, N.; Fernandez, P.; Atallah, N.; Lorenzo, R.; Tomkos, I.; Abril, E.J.; , "Enhancing optical networks with cognition: Case-Based Reasoning to estimate the quality of transmission," Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2012 IEEE International Multi-Disciplinary Conference on , vol., no., pp.166-169, 6-8 March 2012.
- Fernandez, N.; Duran, R. J.; de Miguel, I.; Merayo, N.; Sanchez, D.; Angelou, M.; Aguado, J. C.; Fernandez, P.; Jimenez, T.; Lorenzo, R.M.; Tomkos, I.; Abril, E.J.; , "Cognition to design energetically efficient and impairment aware virtual topologies for optical networks," Optical Network Design and Modeling (ONDM), 2012 16th International Conference on , vol., no., pp.1-6, 17-20 April 2012.
- Jimenez, T.; Aguado, J. C.; de Miguel, I.; Duran, R. J.; Sanchez, D.; Angelou, M.; Merayo, N.; Fernandez, P.; Fernandez, N.; Lorenzo, R. M.; Tomkos, I.; Abril, E. J.;, "Optimization of the knowledge base of a cognitive quality of transmission estimator for core optical networks," Optical Network Design and Modeling (ONDM), 2012 16th International Conference on , vol., no., pp.1-6, 17-20 April 2012.
- Siracusa, D.; Salvadori, E.; Francescon A.; Zanardi, A.; Angelou, M.; Klonidis, D.; Tomkos, I.; Sánchez, D.; Durán, R.J.; de Miguel, I.; "A Control Plane Framework for Future Cognitive Heterogeneous Optical Networks", in International Conference on Transparent Optical Networks (ICTON), IEEE, July 2012, paper Tu.D1.3.
- Durán, R.J.; de Miguel, I.; Sánchez, D.; Fernandez, N.; Jimenez, T.; Aguado, J. C.; Yedugundla, V.K.; Angelou, M.; Merayo, N.; Fernandez, P.; Atallah, N.; Lorenzo, R. M.; Francescon, A.; Tomkos, I.; Abril, E. J.;, Future Network & MobileSummit 2012 Conference Proceedings, Paul Cunningham and Miriam Cunningham (Eds), IIMC International Information Management Corporation, 2012.
- N. Fernández, R. J. Durán, I. de Miguel, N. Merayo, J. C. Aguado, P. Fernández, T. Jiménez, I. Rodríguez, D. Sánchez, R. M. Lorenzo, E. J. Abril, M. Angelou and I. Tomkos, "Survivable and Impairment-Aware Virtual Topologies for Reconfigurable Optical Networks: a Cognitive Approach", 4th International Workshop on Reliable Networks Design and Modeling, October 2012.

Finally, another important related contribution has been the participation in the supervision of the following **graduation thesis**:

- Martín San José R., "Emulación con DRAGON del plano de control de redes ópticas intradominio", Graduation Thesis, Telecommunications Engineering School, University of Valladolid, March 2012.

6 Chapter VI – References

[1] M. A. L. Thathachar and P. S Sastry, Networks of Learning Automata: Techniques of Online Stochastic Optimization. Kluwer Academic Publishers, 2004.

[2] Deliverable 3.1, "Specification of the architecture and methods of the cognitive decision system", CHRON Project, FP7/2007-2013, Ref. 258644, July 2011. Available in: www.ict-chron.eu

[3] Open-source GMPLS emulator DRAGON, "Dynamic Resource Allocation via Gmpls Optical Network".
From: http://dragon.maxgigapop.net/twiki/bin/view/DRAGON/WebHome. Last access: September 2012.

[4] B. Mukherjee, Optical Communication Networks. McGraw-Hill Companies, 1997.

[5] R. Ramaswami, K. N. Sivarajan and G. H. Sasaki, Optical Networks: A Practical Perspective. Morgan Kaufmann Pub, 2009].

[6] Deliverable 2.2, "Initial network and service control architecture", CHRON Project, FP7/2007-2013, Ref. 258644, November 2011. Available in: www.ict-chron.eu

[7] Deliverable 2.3, "Final network architecture and operation definition", CHRON Project, FP7/2007-2013, Ref. 258644, July 2012. Available in: www.ict-chron.eu

[8] D. Siracusa, E. Salvadori, A. Francescon, A. Zanardi, M. Angelou, D. Klonidis, I. Tomkos, D. Sánchez, R.J. Durán and I. de Miguel, "A Control Plane Framework for Future Cognitive Heterogeneous Optical Networks", in International Conference on Transparent Optical Networks (ICTON), IEEE 2012, paper Tu.D1.3.

[9] Deliverable 5.1, "Interface Definition", CHRON Project, FP7/2007-2013, Ref. 258644, July 2011. Available in: www.ict-chron.eu

[10] D. Katz et al., "Traffic Engineering (TE) Extensions to OSPF Version 2", IETF RFC 3630, 2003.

[11] Deliverable 5.2, "Control plane architecture and protocol definition", CHRON Project, FP7/2007-2013, Ref. 258644, May 2012. Available in: www.ict-chron.eu

[12] D. Awduche et al., "RSVP-TE: Extensions to RSVP for LSP Tunnels", IETF RFC 3209, 2001.

[13] E. Rosen, A. Viswanathan & R. Callon, "[RFC3031] Multiprotocol Label Switching Architecture", IETF, 2001.

[14] Valentín, C. E. "Development of an SNMP agent for Ethernet switches", Thesis, UNIVERSITEIT GENT Faculty of Engineering, Department of Information Technology, 2007. Available in: http://upcommons.upc.edu/pfc/bitstream/2099.1/6743/2/FinalThesisReport.pdf Last Access: July 2012.

[15] A. C. Ordóñez, "GMPLS: Generalized MultiProtocol Label Switching. El futuro óptico de Internet", Universidad Politécnica de Madrid, Departamento de Ingeniería Telemática, Madrid.

[16] L. Berger, "[RFC3471] Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description", IETF, 2003.

[17] L. Berger, "[RFC3473] Generalized Multi-Protocol Label Switching (GMPLS) Signalling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", IETF, 2003.

[18] E. Mannie, "[RFC3945] Generalized Multi-Protocol Label Switching (GMPLS) Architecture", IETF, 2004.

[19] A. de Marcos García, "Estudio de la inclusión del sistema PCE en redes GMPLS", Proyecto, Universitat Politécnica de Catalunya, Barcelona. Obtained from: http://upcommons.upc.edu/pfc/bitstream/2099.1/8773/1/Proyecto PCE.pdf Last access: July 2012

[20] [RFC 2328] J. Moy, OSPF Version 2, RFC 2328, April 1998.

[21] [RFC 3784] H. Smit, Intermediate System to Intermediate System (IS-IS) Extensions for Traffic Engineering (TE), June 2004.

[22] R. Coltun, "[RFC2370] The OSPF Opaque LSA Option", IETF, 1998.

[23] P. Ashwood Smith, "[RFC3472] Generalized Multi-Protocol Label Switching (GMPLS) Signalling Constraint-based Routed Label Distribution Protocol (CR-LDP) Extensions", IETF, January 2003.

[24] R. Muñoz, R. Martínez & R. Casellas, "Challenges for GMPLS Lightpath Provisioning in Transparent Optical Networks: Wavelength Constraints in Routing and

Signaling", Centre Tecnològic de Telecomunicacions de Catalunya (CTTC), IEEE Communications Magazine, 2009.

[25] M. Kim, "Network Management Framework For Large Scale Heterogeneous Sensor Networks", Central R&D Laboratory. CEATEC Japan, October 2009.

[26] M.Rose, K.McCloghrie, "Structure and Identification of Management Information for TCP/IP based Internets", IETF RFC 1155, May 1990.

[27] K. McCloghrie, D. Perkins, J. Schoenwaelder, "Structure of Management Information Version 2 (SMIv2)", IETF RFC 2578, April 1999.

[28] O. Dubuisson, ASN.1. Communication Between Heterogeneous Systems, Morgan Kaufmann, 2001.

[29] J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol (SNMP)", IETF RFC 1157, May 1990.

[30] M. Rose, K. McCloghrie, "Concise MIB Definitions", IETF RFC 1212, March 1991.

[31] Simple Web. Information about MIBs (MIB validation, vendor MIBs). Available at: http://www.simpleweb.org/ietf/mibs/ Last access: July 2012.

[32] OID View MIB Browser, http://www.oidview.com/mibs/9/CISCO-OPTICAL-MONITORMIB.html Last access: July 2012.

[33] OSI Objects Identifier Tree. Available at: http://www.oid-info.com Last access: July 2012.

[34] Cisco SNMP Object Navigator. Available at: http://tools.cisco.com/Support/SNMP/do/BrowseOID.do?local=en Last access: July 2012.

[35] T. Klie, F. Strauss "Integrating SNMP Agents with XML-Based Management Systems," IEEE Communications Magazine, vol. 42 n.7 pp. 76-83. July 2004.

[36] R. Enss, "NETCONF Configuration Protocol", IETF RFC 4741, December 2006.

[37] T. Klie, F. Strauss "Integrating SNMP Agents with XML-Based Management Systems," IEEE Communications Magazine, vol. 42 n.7 pp. 76-83. July 2004.

[38] M.J. Choi, H.M. Choi, J.W. Hong, "XML-Based Configuration Management for IP Network Devices", IEEE Communications Magazine, vol. 42, n. 7, pp 84-91, July 2004.

[39] J. Bih, "Deploy XML-based network management approach," IEEE Potentials, vol.24 n.4 pp. 26-31 2005.

[40] Sun Microsystems, "RPC: Remote Procedure Call Protocol Specification", IETF RFC 1050, April 1988.

[41] J. Schönwälder, M. Björklund, P. Shafer, "Network Configuration Management Using NETCONF and YANG," IEEE Communications Magazine, vol.48 n.9 pp. 166-173, September 2010.

[42] T. Ylonen, C. Lonvick. "The Secure Shell (SSH) Protocol Architecture", IETF RFC 4251, January 2006.

[43] W3C, "SOAP Version 1.2 Part 0: Primer", W3C Recommendation, April 2007. Available at: http://www.w3.org/TR/soap12-part0/ Last access: July 2012.

[44] M. Rose, "The Blocks Extensible Exchange Protocol Core", IETF RFC 3080, March 2001.

[45] T. Dierks, C. Allen, "The TLS Protocol version 1.0", IETF RFC 2246, January 1999.

[46] T. Goddard, "Using NETCONF over the Simple Object Access Protocol (SOAP)", IETF RFC 4743, December 2006.

[47] G. Münz, A. Antony, F. Dressler, G. Carle, "Using NETCONF for Configuring Monitoring Probes," IEEE Network Operations and Management Symposium (NOMS), 2006.

[48] S. Chisholm, H. Trevino, "NETCONF Event Notifications", IETF RFC 5277, July 2008.

[49] H. Xu, D.Xiao, "Data Modeling for NETCONF-Based Network Management: XML Schema or YANG", Proceedings of the 11th IEEE International Conference on Communication Technology, pp. 561-564, 2008.

[50] T.Bray, J. Paoli, C.M. Sperberg, E. Maler, F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fourth Edition). W3C Recommendation, 2006. Available at: http://www.w3.org/TR/2006/REC-xml-20060816 Last access: July 2012.

[51] D.C. Fallside, P. Walmsley "XML Schema Part 0: Primer Second Edition", W3C Recommendation, 2004. Available at: http://www.w3.org/TR/xmlschema-0/ Last access: July 2012.

[52] M. Ellison, B. Natale, "Expressing SNMP SMI Datatypes in XML Schema Definition Language", IETF RFC 5935, August 2010.

[53] Sensor Model Language (SensorML). OGC. The OpenGIS Sensor Model Language Encoding Standard. Available at: http://www.opengeospatial.org/standards/sensorml Last access: July 2012.

[54] J. Clark and M. Makoto, "RELAX NG Specification," OASIS Committee Spec., Dec. 2001.

[55] M. Bjorklund, "YANG – A Data Modeling Language for the Network Configuration Protocol (NETCONF)", IETF RFC 6020, October 2010.

[56] H. Cui, B. Zhang, G. Li, "Contrast Analysis of NETCONF Modeling Languages: XML Schema, Relax NG and YANG", International Conference on Communication Software and Networks, 2009.

[57] M.J.Choi, J.W.Hong, H.T. Ju "XML-Based Network Management for IP Networks," ETRI Journal, vol. 25, n. 6, pp. 445-463. December 2003.

[58] Y.J. Oh, H.T. Ju, M.J. Choi, J.W. Hong "Interaction Translation Methods for XML/SNMP Gateway" in Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems. 2002.

[59] M. Meijerink, & R. Prickaerts, "Generalized MPLS. The DRAGON Project implementation at SARA", Universiteit van Amsterdam, 2006. Obtained from: https://noc.sara.nl/nrg/ Last access: July 2012.

[60] AGENT++ Library. Avaliable in: http://www.agentpp.com/agentpp3_5/agentpp3_5.html Last access: July 2012.

[61] University of Southern California (USC), University of Maryland (UMD), George Mason University (GMU), Information Science Institute (ISI) & Mid-Atlantic Crossroads (MAX), "Virtual Label Switching Router Implementation Guide. User Manual", 2008.

Obtained from: http://dragon.east.isi.edu/twiki/pub/DRAGON/VLSR/dragon-vlsr-implement v2.1b.pdf Last access: July 2012.

[62] "DRAGON Supported Switches", 2011s. Obtained from: https://wiki.internet2.edu/confluence/display/DCNSS/DRAGON+Supported+Switches Last access: July 2012.

[63] "GNU Zebra", 2011. Obtained from: http://www.zebra.org

[64] "KOM RSVP Engine", 2011. Obtained from: http://www.kom.tu-darmstadt.de/en/downloads/software/kom-rsvp-engine

[65] The DICONET Project Consortium, "Final report on control plane protocol extension prototype and studies". Obtained from: http://www.diconet.eu/documents/DICONET_D5.3_WP5_updated31July2010_CN_V4.0.pdf

[66] P. Sköldström, "Multi-region GMPLS control and data plane integration", KTH Information and Communication Technology, Stockholm, 2008.
Obtained from:
http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/080821-Pontus_Skoldstrom-with-cover.pdf Last access: July 2012.

[67] D. Farinacci, S. Hanks, D. Meyer & P. Traina, "[RFC2784] Generic Routing Encapsulation (GRE)", IETF, 2000.

[68] J. Van der Ham, & G. J. Verhoog. "Virtual environments for networking experiments", Analytical Network Project, University of Amsterdam, Masters System and Network Administration, Netherlands, 2004. Obtained from: https://alumni.os3.nl/~gjv/uml/anp-jvdh-gjv.pdf

[69] "Dynamic Resource Allocation via Gmpls Optical Network". Obtained from: http://dragon.maxgigapop.net/twiki/bin/view/DRAGON/WebHome Last access: July 2012.

[70] Martín San José R., "Emulación con DRAGON del plano de control de redes ópticas intradominio", Graduation Thesis, Telecommunications Engineering School, University of Valladolid, March 2012.

[71] Deliverable 2.1, “Specification of the network scenario and service requirements”, CHRON Project, FP7/2007-2013, Ref. 258644, November 2011. Available in: www.ict-chron.eu

[72] “NetGUI: Configuración de OSPF en Zebra”, Universidad Rey Juan Carlos, Departamento de Sistemas Telemáticos y Computación (GSyC), 2011. Obtenido de: http://docencia.etsit.urjc.es/moodle/file.php/53/Practicas/3-ospf/netgui-zebra-ospf.pdf Last Access: July 2012.

[73] AGENT++ Library. Avaliable in: http://www.agentpp.com/agentpp3_5/agentpp3_5.html Last access: July 2012.

[74] CISCO Optical Monitoring MIB. Available in: ftp://ftp.cisco.com/pub/mibs/oid/oid/tar.gz Last access: July 2012.

[75] Deliverable 3.2, “Architecture of the Cognitive Decision System”, CHRON Project, FP7/2007-2013, Ref. 258644, January 2012. Available in www.ict-chron.eu

[76] LibPCEP Library, Available in: http://www.acreo.se/en/Technology-Areas/Broadband-Technology/Software/LibPCEP/, Last access: July 2012.

7 Chapter VII –Acronyms and Abbreviations

A

A	Amplifier
ABR	Area Border Router
API	Application Programming Interface
AS	Autonomous System
ASN.1	Abstract Syntax Notation One
ASTB	Application Specific Topology Builder
ATM	Asynchronous Transfer Mode
AUR-ESS	Adaptive Unconstrained Routing – Exhaustive Spectrum Search

B

BEEP	Block Extensible Exchange Protocol
BoS	Bottom of Stack

C

CAC/RM	Control Access Channel/ Resource Manager
CDS	Cognitive Decision System
CHRON	Cognitive Heterogeneous Reconfigurable Optical Network
CKI	Control Knowledge Interface
CLI	Command Line Interface
CMI	Connection and Management Interface
CMP	Control and Management Plane
COMM	Communication
CoS	Class of Service
CP	Control Plane
CR-LDP	Constraint-based Routed Label Distribution Protocol
CSA	Client System Agent

D

DB	Database
DRAGON	Dynamic Resource Allocation via GMPLS Optical Networks
DT	Deutsche Telekom
DTD	Document Type Definition
DWDM	Dense Wavelength Division Multiplexing

E

E-GMPLS	Extended GMPLS
ERO	Explicit Route Object
ES	End System

F

FA	Forwarding Adjacency
FEC	Forwarding Equivalent Class
FSC	Fiber Switch Capable

G

GL	Generalized Label
GMPLS	Generalized Multi-Protocol Label Switching
GPPD	General Physical Parameter Database
GRE	Generic Routing Encapsulation
GTED	Global Traffic Engineering Database

H

HTTP	Hypertext Transfer Protocol

I

ICT	Information and Communication Technologies
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPC	Inter-Process Communication
IS-IS	Intermediate System-Intermediate System
ITU-T	International Telecommunication Union

K

KB	Knowledge Bases
KP	Knowledge Plane

L

L2SC	Layer-2 Switch Capable/Capability
LDP	Label Distribution Protocol
LER	Label Edge Router
LIB	Label Information Base
LPPD	Local Physical Parameter Database
LS	Label Set
LSC	Lambda Switch Capable/Capability
LSD	Link State Database
LSP	Label Switched Path
LSR	Label Switched Router
LTED	Local Traffic Engineering Database

M

MIB	Management Information Base
MKI	Management Knowledge planes Interface
ML	Model Language
MP	Management Plane
MPLS	Multi-Protocol Label Switching
MPλS	Multi-Protocol Lambda Switching

MUT	Maximum Transmission Unit
N	
NARB	Network Aware Resource Broker
NETCONF	Network Configuration Protocol
NMI-C	Network Management Interface – CP
NMI-T	Network Management Interface – TP
NMonS	Network Monitoring System
NMS	Network Management System
NPDM	Network Planner and Decision Maker
NSP	Network Service Provider
O	
OCC	Optical Connection Controller
OID	Object Identifier
OSNR	Optical Signal to Noise Ratio
OSPF-TE	Open Shortest Path First – Traffic Engineering
OXC	Optical Cross Connect
P	
PDU	Protocol Data Unit
PHY	PHYsical Plane
PSC	Packet Switch Capable/Capability
Q	
QoS	Quality of Service
QoT	Quality of Transmission
R	
RFC	Request for Comments
RMLSA	Routing Modulation Level and Spectrum Allocation
RPC	Remote Procedure Call
RRO	Record Route Object
RSVP-TE	Resource reSerVation Protocol – Traffic Engineering
RWA	Routing and Wavelength Assignment
S	
SAO	Session Attribute Object
SDH	Synchronous Digital Hierarchy
SL	Suggested Label
SLA	Service Level Agreements
SMI	Structure Management Information
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SOAP	Simple Object Access Protocol
SONET	Synchronous Optical NETwork
SRLG	Shared Risk Link Group

SSH	Secure Shell
T	
TC	Topology and Configuration
TCP/IP	Transport Control Protocol/Internet Protocol
TDM	Time Division Multiplexing
TE	Traffic Engineering
TED	Traffic Engineering Database
TG	Traffic Grooming
TL1	Transaction Language 1
TLS	Transport Layer Security
TLV	Type-Length-Value
TP	Transport Plane
U	
UDP	User Datagram Protocol
UL	Upstream Label
ULS	Upstream Label Set
UML	User Mode Linux
UNI	User to Network Interface
V	
VLSR	Virtual Label Switching Router
VNE	Virtual Network Experiment
VPN	Virtual Private Network
VTD	Virtual Topology Design
W	
WCC	Wavelength Continuity Constraint
WDM	Wavelength Division Multiplexing
X	
XML	eXtensible Markup Language
XSD	XML Schema Definition

8 Chapter VIII – Annexes

This chapter contains information that completes the content presented in the previous chapters throughout this Master of Research Thesis report.

8.1 Annex 1: CSL for encoding network monitoring requests and data – Optical performance monitoring

Messages for requesting and receiving optical performance monitoring data associated to a lightpath. The CDS sends the request message (`OPTICAL_PERFORMANCE_MONITORING_DATA_REQUEST`) and the CMP module answers by means of the `OPTICAL_PERFORMANCE_MONITORING_DATA` message. However, the CMP module can provide alarm-type information to the CDS without being asked first, by means of the `OPTICAL_PERFORMANCE_MONITORING_TRAP` message.

In these messages, the `RequesterElement` refers to the node that requested the establishment of the lightpath which is now monitored.

If the reader needs further information concerning the other uses of CSL such encoded information is presented in [72].

8.1.1. OPTICAL_PERFORMANCE_MONITORING_DATA_REQUEST message

```
<!-- DTD : OpticalPerformanceMonitoringRequest.dtd -->

<!ELEMENT Message (Type, LightpathID, RequesterElement)>
<!ELEMENT Type (#PCDATA)>
<!ELEMENT LightpathID (#PCDATA)>
<!ELEMENT RequesterElement (#PCDATA)>

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Message SYSTEM "OpticalPerformanceMonitoringRequest.dtd">

<Message>
    <Type> OPTICAL_PERFORMANCE_MONITORING_DATA_REQUEST </Type>
    <LightpathID> </LightpathID>
    <RequesterElement> </RequesterElement>
</Message>
```

8.1.2. OPTICAL_PERFORMANCE_MONITORING_DATA and OPTICAL_PERFORMANCE_MONITORING_TRAP messages

```
<!-- DTD : OpticalPerformanceMonitoringData/Trap.dtd -->

<!ELEMENT Message (Type, LightpathID, RequesterElement, Location*)>
<!ELEMENT Type (#PCDATA)>
```

```
<!ELEMENT LightpathID (#PCDATA)>
<!ELEMENT RequesterElement (#PCDATA)>
<!ELEMENT Location (Parameter*)>
<!ELEMENT Parameter (Field*>
<!ELEMENT Field (value)>
<!ELEMENT value (#PCDATA)>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Message SYSTEM "OpticalPerformanceMonitoringData /Trap.dtd">

<Message>
    <Type> OPTICAL_PERFORMANCE_MONITORING_DATA /
 OPTICAL_PERFORMANCE_MONITORING_TRAP </Type>
    <LightpathID> </LightpathID>
    <RequesterElement> </RequesterElement>
    <Location name = " ">
        <Parameter name = "OSNR" code = "dB/0.1 nm">
            <field name = "MeasuredValue">
                <value> </value>
            </field>
            <field name = "ObservationTime" code = "s">
                <value> </value>
            </field>
            <field name = "Periodicity" code = "s">
                <value> </value>
            </field>
            <field name = "SurveyPeriod" code = "s">
                <value> </value>
            </field>
        </Parameter>

        <Parameter name = "Power Level" code = "dBm">
            <field name = "MeasuredValue">
                <value> </value>
            </field>
            <field name = "ObservationTime" code = "s">
                <value> </value>
            </field>
            <field name = "Periodicity" code = "s">
                <value> </value>
            </field>
            <field name = "SurveyPeriod" code = "s">
                <value> </value>
            </field>
        </Parameter>

        <Parameter name = "Variations of power levels" code = "dB">
            <field name = "MeasuredValue">
                <value> </value>
            </field>
            <field name = "ObservationTime" code = "s">
                <value> </value>
            </field>
            <field name = "Periodicity" code = "s">
                <value> </value>
            </field>
            <field name = "SurveyPeriod" code = "s">
                <value> </value>
            </field>
        </Parameter>
```

```
<Parameter name = "Chromatic Dispersion" code = "ps">
    <field name = "MeasuredValue">
        <value> </value>
    </field>
    <field name = "ObservationTime" code = "s">
        <value> </value>
    </field>
    <field name = "Periodicity" code = "s">
        <value> </value>
    </field>
    <field name = "SurveyPeriod" code = "s">
        <value> </value>
    </field>
</Parameter>

<Parameter name = "PMD/DGD" code = "ps">
    <field name = "MeasuredValue">
        <value> </value>
    </field>
    <field name = "ObservationTime" code = "s">
        <value> </value>
    </field>
    <field name = "Periodicity" code = "s">
        <value> </value>
    </field>
    <field name = "SurveyPeriod" code = "s">
        <value> </value>
    </field>
</Parameter>

<Parameter name = "I&Q Imbalances" code = "%">
    <field name = "MeasuredValue">
        <value> </value>
    </field>
    <field name = "ObservationTime" code = "s">
        <value> </value>
    </field>
    <field name = "Periodicity" code = "s">
        <value> </value>
    </field>
    <field name = "SurveyPeriod" code = "s">
        <value> </value>
    </field>
</Parameter>

<Parameter name = "Source-LO offset" code = "MHz">
    <field name = "MeasuredValue">
        <value> </value>
    </field>
    <field name = "ObservationTime" code = "s">
        <value> </value>
    </field>
    <field name = "Periodicity" code = "s">
        <value> </value>
    </field>
    <field name = "SurveyPeriod" code = "s">
        <value> </value>
    </field>
</Parameter>
```

```
        <Parameter name = "BER">
            <field name = "MeasuredValue">
                <value> </value>
            </field>
            <field name = "ObservationTime" code = "s">
                <value> </value>
            </field>
            <field name = "Periodicity" code = "s">
                <value> </value>
            </field>
            <field name = "SurveyPeriod" code = "s">
                <value> </value>
            </field>
        </Parameter>

        <Parameter name = "Q-factor" code = "dB">
            <field name = "MeasuredValue">
                <value> </value>
            </field>
            <field name = "ObservationTime" code = "s">
                <value> </value>
            </field>
            <field name = "Periodicity" code = "s">
                <value> </value>
            </field>
            <field name = "SurveyPeriod" code = "s">
                <value> </value>
            </field>
        </Parameter>

        <Parameter name = "EVM" code = "%">
            <field name = "MeasuredValue">
                <value> </value>
            </field>
            <field name = "ObservationTime" code = "s">
                <value> </value>
            </field>
            <field name = "Periodicity" code = "s">
                <value> </value>
            </field>
            <field name = "SurveyPeriod" code = "s">
                <value> </value>
            </field>
        </Parameter>
    </Location>
</Message>
```

8.2 Annex 2: CSL for encoding network monitoring requests and data – Traffic monitoring

Messages for requesting and receiving traffic monitoring data associated to a non-optical LSP. The CDS sends the request message (`TRAFFIC_MONITORING_DATA_REQUEST`) and the CMP module answers by means of the `TRAFFIC_MONITORING_DATA` message. However, the CMP module can provide alarm-type information to the CDS without being asked first, by means of the `TRAFFIC_MONITORING_TRAP` message.

In these messages, the `RequesterElement` refers to the node that requested the establishment of the LSP which is now monitored.

If the reader needs further information concerning the other uses of CSL such encoded information is presented in [72].

8.2.1. TRAFFIC_MONITORING_DATA_REQUEST message

```
<!-- DTD : TrafficMonitoringRequest.dtd -->

<!ELEMENT Message (Type, LspID, RequesterElement)>
<!ELEMENT Type (#PCDATA)>
<!ELEMENT LspID (#PCDATA)>
<!ELEMENT RequesterElement (#PCDATA)>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Message SYSTEM "TrafficMonitoringRequest.dtd">

<Message>
    <Type> TRAFFIC_MONITORING_DATA_REQUEST </Type>
    <LspID> </LspID>
    <RequesterElement> </RequesterElement>
</Message>
```

8.2.2. TRAFFIC_MONITORING_DATA and TRAFFIC_MONITORING_TRAP messages

```
<!-- DTD : TrafficMonitoringData/Trap.dtd -->

<!ELEMENT Message (Type, LspID, RequesterElement, Location*)>
<!ELEMENT Type (#PCDATA)>
<!ELEMENT LspID (#PCDATA)>
<!ELEMENT RequesterElement (#PCDATA)>
<!ELEMENT Location (Parameter*)>
<!ELEMENT Parameter (Field*>
<!ELEMENT Field (value)>
<!ELEMENT value (#PCDATA)>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Message SYSTEM "TrafficMonitoringData/Trap.dtd">
```

```
<Message>
<Type> TRAFFIC_MONITORING_DATA /
 TRAFFIC_MONITORING_TRAP </Type>
    <LightpathID> </LightpathID>
    <RequesterElement> </RequesterElement>
    <Location name = " ">
<Parameter name = "PacketLoss code = "%">
            <field name = "MeasuredValue">
                <value> </value>
            </field>
            <field name = "ObservationTime" code = "s">
                <value> </value>
            </field>
            <field name = "Periodicity" code = "s">
                <value> </value>
            </field>
            <field name = "SurveyPeriod" code = "s">
                <value> </value>
            </field>
        </Parameter>

<Parameter name = "Jitter" code = "ms">
            <field name = "MeasuredValue">
                <value> </value>
            </field>
            <field name = "ObservationTime" code = "s">
                <value> </value>
            </field>
            <field name = "Periodicity" code = "s">
                <value> </value>
            </field>
            <field name = "SurveyPeriod" code = "s">
                <value> </value>
            </field>
        </Parameter>

<Parameter name = "Delay" code = "ms">
            <field name = "MeasuredValue">
                <value> </value>
            </field>
            <field name = "ObservationTime" code = "s">
                <value> </value>
            </field>
            <field name = "Periodicity" code = "s">
                <value> </value>
            </field>
            <field name = "SurveyPeriod" code = "s">
                <value> </value>
            </field>
        </Parameter>
</Location>
</Message>
```

8.3 Annex 3: DRAGON UML environment installation steps

This annex contains the main steps for the installation, configuration and emulation of a network under the DRAGON UML environment. It is compulsory to undertake these steps once logged in the system as user 'root'.

```
user@host:~$ su – root
```

Install the following packages:

```
root@host:~# apt-get install python
root@host:~# apt-get install uml-utilities
root@host:~# apt-get install xterm
root@host:~# apt-get install xauth
root@host:~# apt-get install xorg
root@host:~# apt-get install screen
root@host:~# apt-get install vde2
```

A new user 'dragon' has to be created. For simplicity, the password chosen is also 'dragon':

```
root@host:~# useradd -m dragon
root@host:~# passwd dragon
```

Modify the command interpreter to become /bin/bash for the dragon user:

```
root@host:~# gedit /etc/passwd
dragon:x:1001:1001::/home/dragon:/bin/bash
```

The /opt folder will be the destination of the different components:

```
root@host:~# cd /opt/
root@host:/opt# mkdir uml
root@host:/opt# cd uml
root@host:/opt/uml#wget
http://dragon.maxgigapop.net/twiki/pub/DRAGON/UserModeLinux/uml_
linux-2.6.27.8.bz2
root@host:/opt/uml#wget
http://dragon.maxgigapop.net/twiki/pub/DRAGON/UserModeLinux/Debi
an-4.0-x86-root_fs-2008Dec10.bz2
```

Unzip both the UML Linux Kernel and the DRAGON file system:

```
root@host:/opt/uml# bzip2 -d uml_linux-2.6.27.8.bz2
root@host:/opt/uml# bzip2 -d Debian-4.0-x86-root_fs-
2008Dec10.bz2
```

VNE installation:

```
root@host:/opt/uml# cd /home/
root@host:/home# cd dragon/
```

```
root@host:/home/dragon#wget
http://dragon.maxgigapop.net/twiki/pub/DRAGON/UserModeLinux/VNEs
napshot-2009Jan12.tar.bz2
```

Unzip the file downloaded:

```
root@host:/home/dragon# bzip2 -d VNE-snapshot-2009Jan12.tar.bz2
| tar –xvf
```

UML Kernel running permissions:

```
root@host:/home/dragon# cd /opt/uml/
root@host:/opt/uml# chmod 755 uml_linux-2.6.27.8
```

Phyton package installation:

```
root@host:/opt/uml# apt-get install python-setuptools
root@host:/opt/uml# cd /home/dragon
root@host:/home/dragon# cd VNE-snapshot-2009Jan12/
root@host:/home/dragon/VNE-snapshot-2009Jan12# cd src
root@host:/home/dragon/VNE-snapshot-2009Jan12/src#python
setup.py install
```

Create a file named *'.vnerc'* to include the different paths. Prior to do that, you should create the directory /home/dragon/uml and give it both read and write permissions:

```
root@host:/home/dragon# mkdir uml
root@host:/home/dragon# chmod 777 uml
root@host:/home/dragon# gedit .vnerc (See below its content)
```

```
 [DEFAULT]
kernel = /opt/uml/uml_linux-2.6.27.8
filesystem = /opt/uml/Debian-4.0-x86-root_fs-2008Dec10
fstype = cow
xmllint = /usr/bin/xmllint
screen = /usr/bin/screen
xterm = /usr/bin/xterm
uml_switch = /usr/bin/vde_switch
memorysize = 64
logdir = /home/dragon/uml
removecow = True
tap_device = tap0
```

Mount the DRAGON file system and install DRAGON:

```
root@host:/home/dragon# cd /opt/uml/
root@host:/opt/uml# mkdir mnt
root@host:/opt/uml# mount -o loop Debian-4.0-x86-root_fs-
2008Dec10 mnt
root@host:/opt/uml# chroot mnt/ /bin/bash
host:/# mount -t proc proc /proc
host:/# mount
host:/# cd home/dragon/snapshot.current/dragon-sw/
```

```
host:/home/dragon/snapshot.current/dragon-sw#sh do_build.sh
vlsr-linux
```

When building the CLI prompt introduce the following: type: shell; username: dragon; password: dragon.

```
host:/home/dragon/snapshot.current/dragon-sw# sh do_install.sh
host:/home/dragon/snapshot.current/dragon-sw# cd ..
host:/home/dragon/snapshot.current# cd narb-sw/
host:/home/dragon/snapshot.current/narb-sw# sh do_build.sh
host:/home/dragon/snapshot.current/narb-sw# sh do_install.sh
```

Edit the '*.bashr*' file…

```
host:/home/dragon/snapshot.current/narb-sw# cd /home/dragon/
host:/home/dragon# pico .bashrc
```

… and check that its content is the following:

```
export CVS_RSH=/usr/bin/ssh
export
CVSROOT=:ext:isiehpn@hpn.east.isi.edu:/home/isiehpn/cvsroot
export LC_CTYPE=POSIX
export PATH=/usr/local/dragon/bin:/usr/local/dragon/sbin:$PATH
tset –s
```

Edit the *'genDragonConfig.pl'* file and add the following information:

```
host:/home/dragon# cd /usr/local/dragon/bin
host:/usr/local/dragon/bin# pico genDragonConfig.pl

set local-id port 2
set local-id port 3
set local-id port 4
set local-id port 5
set local-id port 6
set local-id port 7
set local-id port 8
set local-id port 11
set local-id port 12
```

Un-mount the DRAGON file system:

```
root@host:/usr/local/dragon/bin# umount /proc/
host:/usr/local/dragon/bin# exit
root@host:/opt/uml# umount /opt/uml/mnt
```

IMPORTANT: Before launching the DRAGON UML environment with VNE it is necessary to run the following command logged as the host user:

```
user@host:~$ sudo xhost +
```

Launching DRAGON UML:

```
user@host:~$ su – dragon
dragon@host:~$ cd VNE-snapshot-2009Jan12/configs/
dragon@host:~/VNE-snapshot-2009Jan12/configs$ VNE
network_topology.xml (See Annex 4)
```

Then, the **VNE environment** management can be summarized with the following commands:

>**start all**: UML instances initialization.

>**xterm all**: console launching for each UML instance. After typing 'Enter' key in each console and introducing 'dragon' as user and password, you will be able to manage each network element.

>**stop all**: UML instances stopping.

>**quit**: exit VNE[1].

[1] If you quit VNE before stopping all the instances, you will have to remove the content of the hidden file /home/dragon/.uml to be successful in the next VNE running.

8.4 Annex 4: DT Network XML Definition

This XML file specifies the Deutsche Telekom network topology defined to emulate both the control and management approaches under the DRAGON UML environment.

```
##XML file: DTNetwork.xml##
##Autor: ROBERTO MARTÍN SAN JOSÉ & DAVID SÁNCHEZ CARABIAS##
##Version: 1.0##
##Fecha: 4/12/2011##


<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE config SYSTEM "dtd-VNEControlCfg.dtd">
<config>
    <devices>
        <switch name="switch">
            <interface name="eth0"  bridge="br0"/> <!-- narb -->
            <interface name="eth1"  bridge="br0"/> <!-- vlsr01 -->
            <interface name="eth2"  bridge="br0"/> <!-- vlsr2 -->
            <interface name="eth3"  bridge="br0"/> <!-- vlsr3 -->
            <interface name="eth4"  bridge="br0"/> <!-- vlsr4 -->
            <interface name="eth5"  bridge="br0"/> <!-- vlsr5 -->
            <interface name="eth6"  bridge="br0"/> <!-- vlsr6 -->
            <interface name="eth7"  bridge="br0"/> <!-- vlsr7 -->
            <interface name="eth8"  bridge="br0"/> <!-- vlsr8 -->
            <interface name="eth9"  bridge="br0"/> <!-- vlsr9 -->
            <interface name="eth10"  bridge="br0"/> <!-- vlsr10 -->
            <interface name="eth11"  bridge="br0"/> <!-- vlsr11 -->
            <interface name="eth12"  bridge="br0"/> <!-- vlsr12 -->
            <interface name="eth13"  bridge="br0"/> <!-- vlsr13 -->
            <interface name="eth14"  bridge="br0"/> <!-- vlsr14 -->
            <interface name="eth15"  bridge="br0"/> <!-- es1 -->
            <interface name="eth16"  bridge="br0"/> <!-- es2 -->
        </switch>
        <!-- ELEMENTOS DE RED -->
        <host name="vlsr01">
            <dragon role="vlsr" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.1"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
            <interface name="eth3"/>
        <interface name="eth4"/>
        </host>
        <host name="vlsr2">
            <dragon role="vlsr" narb="192.168.1.15"
narbintra="grenarb"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.2"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
            <interface name="eth3"/>
        </host>
        <host name="vlsr3">
            <dragon role="vlsr" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
```

```
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.3"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
           <interface name="eth3"/>
        <interface name="eth4"/>
            <interface name="eth5"/>
            <interface name="eth6"/>
        </host>
    <host name="vlsr4">
            <dragon role="vlsr" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.4"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
            <interface name="eth3"/>
        </host>
    <host name="vlsr5">
            <dragon role="vlsr" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.5"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
            <interface name="eth3"/>
        </host>
    <host name="vlsr6">
            <dragon role="vlsr" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.6"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
            <interface name="eth3"/>
        </host>
    <host name="vlsr7">
            <dragon role="vlsr" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.7"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
        </host>
    <host name="vlsr8">
            <dragon role="vlsr" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.8"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
            <interface name="eth3"/>
        </host>
    <host name="vlsr9">
            <dragon role="vlsr" narb="192.168.1.15"/>
```

```
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.9"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
            <interface name="eth3"/>
        <interface name="eth4"/>
        </host>
    <host name="vlsr10">
            <dragon role="vlsr" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.10"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
            <interface name="eth3"/>
        <interface name="eth4"/>
            <interface name="eth5"/>
        </host>
    <host name="vlsr11">
            <dragon role="vlsr" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.11"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
            <interface name="eth3"/>
        <interface name="eth4"/>
        </host>
    <host name="vlsr12">
            <dragon role="vlsr" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.12"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
            <interface name="eth3"/>
        </host>
    <host name="vlsr13">
            <dragon role="vlsr" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.13"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
        </host>
    <host name="vlsr14">
            <dragon role="vlsr" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.14"/>
            </interface>
        <interface name="eth1"/>
        <interface name="eth2"/>
        <interface name="eth3"/>
        </host>
```

```
        <host name="narb">
            <dragon role="narb" narbdomain="192.168.1.0"
narbintra="grenarb"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.15"/>
            </interface>
        </host>
    <host name="es1">
            <dragon role="p2p-csa" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.16"/>
            </interface>
        <interface name="eth1"/>
        </host>
    <host name="es2">
            <dragon role="p2p-csa" narb="192.168.1.15"/>
            <interface name="eth0" loopback="True">
                <ipv4 netmask="255.255.0.0"
broadcast="192.168.255.255" addr="192.168.1.17"/>
            </interface>
        <interface name="eth1"/>
        </host>
  </devices>
    <connections>
    <!-- PLANO DE DATOS -->
    <connection>
            <device name="vlsr01" interface="eth1"/>
            <device name="vlsr2" interface="eth1"/>
    </connection>
    <connection>
            <device name="vlsr01" interface="eth2"/>
            <device name="vlsr3" interface="eth1"/>
    </connection>
    <connection>
            <device name="vlsr01" interface="eth3"/>
            <device name="vlsr4" interface="eth1"/>
    </connection>
    <connection>
            <device name="vlsr2" interface="eth2"/>
            <device name="vlsr3" interface="eth2"/>
    </connection>
    <connection>
            <device name="vlsr2" interface="eth3"/>
            <device name="vlsr5" interface="eth1"/>
    </connection>
    <connection>
            <device name="vlsr3" interface="eth3"/>
            <device name="vlsr6" interface="eth1"/>
    </connection>
    <connection>
            <device name="vlsr5" interface="eth2"/>
            <device name="vlsr6" interface="eth2"/>
    </connection>
    <connection>
            <device name="vlsr5" interface="eth3"/>
            <device name="vlsr7" interface="eth1"/>
    </connection>
    <connection>
            <device name="vlsr7" interface="eth2"/>
```

```
        <device name="vlsr8" interface="eth1"/>
</connection>
<connection>
        <device name="vlsr6" interface="eth3"/>
        <device name="vlsr8" interface="eth2"/>
</connection>
<connection>
        <device name="vlsr8" interface="eth3"/>
        <device name="vlsr10" interface="eth1"/>
</connection>
<connection>
        <device name="vlsr3" interface="eth4"/>
        <device name="vlsr4" interface="eth2"/>
</connection>
<connection>
        <device name="vlsr3" interface="eth5"/>
        <device name="vlsr10" interface="eth2"/>
</connection>
<connection>
        <device name="vlsr3" interface="eth6"/>
        <device name="vlsr9" interface="eth1"/>
</connection>
<connection>
        <device name="vlsr9" interface="eth2"/>
        <device name="vlsr10" interface="eth3"/>
</connection>
<connection>
        <device name="vlsr10" interface="eth4"/>
        <device name="vlsr12" interface="eth1"/>
</connection>
<connection>
        <device name="vlsr10" interface="eth5"/>
        <device name="vlsr11" interface="eth1"/>
</connection>
<connection>
        <device name="vlsr9" interface="eth3"/>
        <device name="vlsr11" interface="eth2"/>
</connection>
<connection>
        <device name="vlsr11" interface="eth3"/>
        <device name="vlsr12" interface="eth2"/>
</connection>
<connection>
        <device name="vlsr12" interface="eth3"/>
        <device name="vlsr13" interface="eth1"/>
</connection>
<connection>
        <device name="vlsr11" interface="eth4"/>
        <device name="vlsr14" interface="eth1"/>
</connection>
<connection>
        <device name="vlsr13" interface="eth2"/>
        <device name="vlsr14" interface="eth2"/>
</connection>
<connection>
        <device name="vlsr4" interface="eth3"/>
        <device name="vlsr9" interface="eth4"/>
</connection>
<connection>
        <device name="es1" interface="eth1"/>
        <device name="vlsr01" interface="eth4"/>
```

```
</connection>
<connection>
        <device name="es2" interface="eth1"/>
        <device name="vlsr14" interface="eth3"/>
</connection>
<!-- PLANO DE CONTROL -->
    <connection>
        <device name="switch" interface="eth0"/>
        <device name="narb" interface="eth0"/>
    </connection>
    <connection>
        <device name="switch" interface="eth1"/>
        <device name="vlsr01" interface="eth0"/>
    </connection>
    <connection>
        <device name="switch" interface="eth2"/>
        <device name="vlsr2" interface="eth0"/>
    </connection>
    <connection>
        <device name="switch" interface="eth3"/>
        <device name="vlsr3" interface="eth0"/>
    </connection>
    <connection>
        <device name="switch" interface="eth4"/>
        <device name="vlsr4" interface="eth0"/>
    </connection>
    <connection>
        <device name="switch" interface="eth5"/>
        <device name="vlsr5" interface="eth0"/>
    </connection>
    <connection>
        <device name="switch" interface="eth6"/>
        <device name="vlsr6" interface="eth0"/>
    </connection>
    <connection>
        <device name="switch" interface="eth7"/>
        <device name="vlsr7" interface="eth0"/>
    </connection>
    <connection>
        <device name="switch" interface="eth8"/>
        <device name="vlsr8" interface="eth0"/>
    </connection>
    <connection>
        <device name="switch" interface="eth9"/>
        <device name="vlsr9" interface="eth0"/>
    </connection>
    <connection>
        <device name="switch" interface="eth10"/>
        <device name="vlsr10" interface="eth0"/>
    </connection>
    <connection>
        <device name="switch" interface="eth11"/>
        <device name="vlsr11" interface="eth0"/>
    </connection>
    <connection>
        <device name="switch" interface="eth12"/>
        <device name="vlsr12" interface="eth0"/>
    </connection>
    <connection>
        <device name="switch" interface="eth13"/>
        <device name="vlsr13" interface="eth0"/>
```

```
        </connection>
        <connection>
            <device name="switch" interface="eth14"/>
            <device name="vlsr14" interface="eth0"/>
        </connection>
        <connection>
            <device name="switch" interface="eth15"/>
            <device name="es1" interface="eth0"/>
        </connection>
        <connection>
            <device name="switch" interface="eth16"/>
            <device name="es2" interface="eth0"/>
        </connection>
        <!-- data plane circuit ID: D1 -->
        <gretunnel name="gre1" mask="10.1.1.0/30"
temask="11.1.1.0/30">
            <device name="vlsr01" interface="eth0" switchport="1"/>
            <device name="vlsr2" interface="eth0" switchport="1"/>
        </gretunnel>
        <!-- data plane circuit ID: D2 -->
        <gretunnel name="gre2" mask="10.1.2.0/30"
temask="11.1.2.0/30">
            <device name="vlsr01" interface="eth0" switchport="2"/>
            <device name="vlsr3" interface="eth0" switchport="1"/>
        </gretunnel>
        <!-- data plane circuit ID: D3 -->
        <gretunnel name="gre3" mask="10.1.3.0/30"
temask="11.1.3.0/30">
            <device name="vlsr01" interface="eth0" switchport="3"/>
            <device name="vlsr4" interface="eth0" switchport="1"/>
        </gretunnel>
        <!-- data plane circuit ID: D4 -->
        <gretunnel name="gre4" mask="10.1.4.0/30"
temask="11.1.4.0/30">
            <device name="vlsr2" interface="eth0" switchport="2"/>
            <device name="vlsr3" interface="eth0" switchport="2"/>
        </gretunnel>
        <!-- data plane circuit ID: D5 -->
        <gretunnel name="gre5" mask="10.1.5.0/30"
temask="11.1.5.0/30">
            <device name="vlsr2" interface="eth0" switchport="3"/>
            <device name="vlsr5" interface="eth0" switchport="1"/>
        </gretunnel>
    <!-- data plane circuit ID: D6 -->
        <gretunnel name="gre6" mask="10.1.6.0/30"
temask="11.1.6.0/30">
            <device name="vlsr3" interface="eth0" switchport="3"/>
            <device name="vlsr6" interface="eth0" switchport="1"/>
        </gretunnel>
    <!-- data plane circuit ID: D7 -->
        <gretunnel name="gre7" mask="10.1.7.0/30"
temask="11.1.7.0/30">
            <device name="vlsr5" interface="eth0" switchport="2"/>
            <device name="vlsr6" interface="eth0" switchport="2"/>
        </gretunnel>
    <!-- data plane circuit ID: D8 -->
        <gretunnel name="gre8" mask="10.1.8.0/30"
temask="11.1.8.0/30">
            <device name="vlsr5" interface="eth0" switchport="3"/>
            <device name="vlsr7" interface="eth0" switchport="1"/>
        </gretunnel>
```

```
    <!-- data plane circuit ID: D9 -->
        <gretunnel name="gre9" mask="10.1.9.0/30"
temask="11.1.9.0/30">
            <device name="vlsr7" interface="eth0" switchport="2"/>
            <device name="vlsr8" interface="eth0" switchport="1"/>
        </gretunnel>
    <!-- data plane circuit ID: D10 -->
        <gretunnel name="gre10" mask="10.1.10.0/30"
temask="11.1.10.0/30">
            <device name="vlsr6" interface="eth0" switchport="3"/>
            <device name="vlsr8" interface="eth0" switchport="2"/>
        </gretunnel>
    <!-- data plane circuit ID: D11 -->
        <gretunnel name="gre11" mask="10.1.11.0/30"
temask="11.1.11.0/30">
            <device name="vlsr8" interface="eth0" switchport="3"/>
            <device name="vlsr10" interface="eth0" switchport="1"/>
        </gretunnel>
    <!-- data plane circuit ID: D12 -->
        <gretunnel name="gre12" mask="10.1.12.0/30"
temask="11.1.12.0/30">
            <device name="vlsr3" interface="eth0" switchport="4"/>
            <device name="vlsr4" interface="eth0" switchport="2"/>
        </gretunnel>
    <!-- data plane circuit ID: D13 -->
        <gretunnel name="gre13" mask="10.1.13.0/30"
temask="11.1.13.0/30">
            <device name="vlsr3" interface="eth0" switchport="5"/>
            <device name="vlsr10" interface="eth0" switchport="2"/>
        </gretunnel>
    <!-- data plane circuit ID: D14 -->
        <gretunnel name="gre14" mask="10.1.14.0/30"
temask="11.1.14.0/30">
            <device name="vlsr3" interface="eth0" switchport="6"/>
            <device name="vlsr9" interface="eth0" switchport="1"/>
        </gretunnel>
    <!-- data plane circuit ID: D15 -->
        <gretunnel name="gre15" mask="10.1.15.0/30"
temask="11.1.15.0/30">
            <device name="vlsr9" interface="eth0" switchport="2"/>
            <device name="vlsr10" interface="eth0" switchport="3"/>
        </gretunnel>
    <!-- data plane circuit ID: D16 -->
        <gretunnel name="gre16" mask="10.1.16.0/30"
temask="11.1.16.0/30">
            <device name="vlsr10" interface="eth0" switchport="4"/>
            <device name="vlsr12" interface="eth0" switchport="1"/>
        </gretunnel>
    <!-- data plane circuit ID: D17 -->
        <gretunnel name="gre17" mask="10.1.17.0/30"
temask="11.1.17.0/30">
            <device name="vlsr10" interface="eth0" switchport="5"/>
            <device name="vlsr11" interface="eth0" switchport="1"/>
        </gretunnel>
    <!-- data plane circuit ID: D18 -->
        <gretunnel name="gre18" mask="10.1.18.0/30"
temask="11.1.18.0/30">
            <device name="vlsr9" interface="eth0" switchport="3"/>
            <device name="vlsr11" interface="eth0" switchport="2"/>
        </gretunnel>
    <!-- data plane circuit ID: D19 -->
```

```
        <gretunnel name="gre19" mask="10.1.19.0/30"
temask="11.1.19.0/30">
            <device name="vlsr11" interface="eth0" switchport="3"/>
            <device name="vlsr12" interface="eth0" switchport="2"/>
        </gretunnel>
    <!-- data plane circuit ID: D20 -->
        <gretunnel name="gre20" mask="10.1.20.0/30"
temask="11.1.20.0/30">
            <device name="vlsr12" interface="eth0" switchport="3"/>
            <device name="vlsr13" interface="eth0" switchport="1"/>
        </gretunnel>
    <!-- data plane circuit ID: D21 -->
        <gretunnel name="gre21" mask="10.1.21.0/30"
temask="11.1.21.0/30">
            <device name="vlsr11" interface="eth0" switchport="4"/>
            <device name="vlsr14" interface="eth0" switchport="1"/>
        </gretunnel>
    <!-- data plane circuit ID: D22 -->
        <gretunnel name="gre22" mask="10.1.22.0/30"
temask="11.1.22.0/30">
            <device name="vlsr13" interface="eth0" switchport="2"/>
            <device name="vlsr14" interface="eth0" switchport="2"/>
        </gretunnel>
    <!-- data plane circuit ID: D23 -->
        <gretunnel name="gre23" mask="10.1.23.0/30"
temask="11.1.23.0/30">
            <device name="vlsr4" interface="eth0" switchport="3"/>
            <device name="vlsr9" interface="eth0" switchport="4"/>
        </gretunnel>
    <!-- data plane circuit ID: D24 -->
        <gretunnel name="gre24" mask="10.1.24.0/30"
temask="11.1.24.0/30">
            <device name="es1" interface="eth0" switchport="1"/>
            <device name="vlsr01" interface="eth0" switchport="4"/>
        </gretunnel>
    <!-- data plane circuit ID: D25 -->
        <gretunnel name="gre25" mask="10.1.25.0/30"
temask="11.1.25.0/30">
            <device name="es2" interface="eth0" switchport="1"/>
            <device name="vlsr14" interface="eth0" switchport="3"/>
        </gretunnel>
        <!-- GRE tunnel between VLSR2 and NARB -->
        <gretunnel name="grenarb" mask="10.1.26.0/30">
            <device name="vlsr2" interface="eth0"/>
            <device name="narb" interface="eth0"/>
        </gretunnel>
    </connections>
</config>
```

8.5 Annex 5: Data link/TE identifiers in the DT Network

The following table presents the information related to the data plane of the emulated DT network (Table 8.1).

ID	Network Element	Management IP Address	TE Address	Interface/Port	Network Element	Management IP Address	TE Address	Interface/Port
D1	vlsr01	192.168.1.1	11.1.1.1	eth1/Port 1	vlsr2	192.168.1.2	11.1.1.2	eth1/Port 1
D2	vlsr01	192.168.1.1	11.1.2.1	eth2/Port 2	vlsr3	192.168.1.3	11.1.2.2	eth1/Port 1
D3	vlsr01	192.168.1.1	11.1.3.1	eth3/Port 3	vlsr4	192.168.1.4	11.1.3.2	eth1/Port 1
D4	vlsr2	192.168.1.2	11.1.4.1	eth2/Port 2	vlsr3	192.168.1.3	11.1.4.2	eth2/Port 2
D5	vlsr2	192.168.1.2	11.1.5.1	eth3/Port 3	vlsr5	192.168.1.5	11.1.5.2	eth1/Port 1
D6	vlsr3	192.168.1.3	11.1.6.1	eth3/Port 3	vlsr6	192.168.1.6	11.1.6.2	eth1/Port 1
D7	vlsr5	192.168.1.5	11.1.7.1	eth2/Port 2	vlsr6	192.168.1.6	11.1.7.2	eth2/Port 2
D8	vlsr5	192.168.1.5	11.1.8.1	eth3/Port 3	vlsr7	192.168.1.7	11.1.8.2	eth1/Port 1
D9	vlsr7	192.168.1.7	11.1.9.1	eth2/Port 2	vlsr8	192.168.1.8	11.1.9.2	eth1/Port 1
D10	vlsr6	192.168.1.6	11.1.10.1	eth3/Port 3	vlsr8	192.168.1.8	11.1.10.2	eth2/Port 2
D11	vlsr8	192.168.1.8	11.1.11.1	eth3/Port 3	vlsr10	192.168.1.10	11.1.11.2	eth1/Port 1
D12	vlsr3	192.168.1.3	11.1.12.1	eth4/Port 4	vlsr4	192.168.1.4	11.1.12.2	eth2/Port 2

D13	vlsr3	192.168.1.3	11.1.13.1	eth5/Port 5	vlsr10	192.168.1.10	11.1.13.2	eth2/Port 2
D14	vlsr3	192.168.1.3	11.1.14.1	eth6/Port 6	vlsr9	192.168.1.9	11.1.14.2	eth1/Port 1
D15	vlsr9	192.168.1.9	11.1.15.1	eth2/Port 2	vlsr10	192.168.1.10	11.1.15.2	eth3/Port 3
D16	vlsr10	192.168.1.10	11.1.16.1	eth4/Port 4	vlsr12	192.168.1.12	11.1.16.2	eth1/Port 1
D17	vlsr10	192.168.1.10	11.1.17.1	eth5/Port 5	vlsr11	192.168.1.11	11.1.17.2	eth1/Port 1
D18	vlsr9	192.168.1.9	11.1.18.1	eth3/Port 3	vlsr11	192.168.1.11	11.1.18.2	eth2/Port 2
D19	vlsr11	192.168.1.11	11.1.19.1	eth3/Port 3	vlsr12	192.168.1.12	11.1.19.2	eth2/Port 2
D20	vlsr12	192.168.1.12	11.1.20.1	eth3/Port 3	vlsr13	192.168.1.13	11.1.20.2	eth1/Port 1
D21	vlsr11	192.168.1.11	11.1.21.1	eth4/Port 4	vlsr14	192.168.1.14	11.1.21.2	eth1/Port 1
D22	vlsr13	192.168.1.13	11.1.22.1	eth2/Port 2	vlsr14	192.168.1.14	11.1.22.2	eth2/Port 2
D23	vlsr4	192.168.1.4	11.1.23.1	eth3/Port 3	vlsr9	192.168.1.9	11.1.23.2	eth4/Port 4
D24	es1	192.168.1.16	11.1.24.1	eth1/Port 1	vlsr01	192.168.1.1	11.1.24.2	eth4/Port 4
D25	es2	192.168.1.17	11.1.25.1	eth1/Port 1	vlsr14	192.168.1.14	11.1.25.2	eth3/Port 3

Table 8.1 Data plane information in the DT Network

8.6 Annex 6: Control Addresses in the DT Network

The following table presents the information related to the control plane of the emulated DT network (Table 8.2).

GRE Control Channel	Network Element	Management IP Address	Control Address	Interface	Network Element	Management IP Address	Control Address	Interface
gre1	vlsr01	192.168.1.1	10.1.1.1	eth0	vlsr2	192.168.1.2	10.1.1.2	eth0
gre2	vlsr01	192.168.1.1	10.1.2.1	eth0	vlsr3	192.168.1.3	10.1.2.2	eth0
gre3	vlsr01	192.168.1.1	10.1.3.1	eth0	vlsr4	192.168.1.4	10.1.3.2	eth0
gre4	vlsr2	192.168.1.2	10.1.4.1	eth0	vlsr3	192.168.1.3	10.1.4.2	eth0
gre5	vlsr2	192.168.1.2	10.1.5.1	eth0	vlsr5	192.168.1.5	10.1.5.2	eth0
gre6	vlsr3	192.168.1.3	10.1.6.1	eth0	vlsr6	192.168.1.6	10.1.6.2	eth0
gre7	vlsr5	192.168.1.5	10.1.7.1	eth0	vlsr6	192.168.1.6	10.1.7.2	eth0
gre8	vlsr5	192.168.1.5	10.1.8.1	eth0	vlsr7	192.168.1.7	10.1.8.2	eth0
gre9	vlsr7	192.168.1.7	10.1.9.1	eth0	vlsr8	192.168.1.8	10.1.9.2	eth0
gre10	vlsr6	192.168.1.6	10.1.10.1	eth0	vlsr8	192.168.1.8	10.1.10.2	eth0
gre11	vlsr8	192.168.1.8	10.1.11.1	eth0	vlsr10	192.168.1.10	10.1.11.2	eth0
gre12	vlsr3	192.168.1.3	10.1.12.1	eth0	vlsr4	192.168.1.4	10.1.12.2	eth0

gre13	vlsr3	192.168.1.3	10.1.13.1	eth0	vlsr10	192.168.1.10	10.1.13.2	eth0
gre14	vlsr3	192.168.1.3	10.1.14.1	eth0	vlsr9	192.168.1.9	10.1.14.2	eth0
gre15	vlsr9	192.168.1.9	10.1.15.1	eth0	vlsr10	192.168.1.10	10.1.15.2	eth0
gre16	vlsr10	192.168.1.10	10.1.16.1	eth0	vlsr12	192.168.1.12	10.1.16.2	eth0
gre17	vlsr10	192.168.1.10	10.1.17.1	eth0	vlsr11	192.168.1.11	10.1.17.2	eth0
gre18	vlsr9	192.168.1.9	10.1.18.1	eth0	vlsr11	192.168.1.11	10.1.18.2	eth0
gre19	vlsr11	192.168.1.11	10.1.19.1	eth0	vlsr12	192.168.1.12	10.1.19.2	eth0
gre20	vlsr12	192.168.1.12	10.1.20.1	eth0	vlsr13	192.168.1.13	10.1.20.2	eth0
gre21	vlsr11	192.168.1.11	10.1.21.1	eth0	vlsr14	192.168.1.14	10.1.21.2	eth0
gre22	vlsr13	192.168.1.13	10.1.22.1	eth0	vlsr14	192.168.1.14	10.1.22.2	eth0
gre23	vlsr4	192.168.1.4	10.1.23.1	eth0	vlsr9	192.168.1.9	10.1.23.2	eth0
gre24	es1	192.168.1.16	10.1.24.1	eth0	vlsr01	192.168.1.1	10.1.24.2	eth0
gre25	es2	192.168.1.17	10.1.25.1	eth0	vlsr14	192.168.1.14	10.1.25.2	eth0
grenarb	narb	192.168.1.15	10.1.26.1	eth0	vlsr2	192.168.1.2	10.1.26.2	eth0

Table 8.2 Control plane information in the DT Network

8.7 Annex 7: Adding SNMP functionalities to the DRAGON file system

To modify the original DRAGON file system (installed according to Annex 3) to cover SNMP functionalities the next steps must be followed:

```
user@host:~$ su – root
root@host:/home/dragon# cd /opt/uml/
root@host:/opt/uml# mount -o loop Debian-4.0-x86-root_fs-
2008Dec10 mnt
root@host:/opt/uml# chroot mnt/ /bin/bash
host:/# mount -t proc proc /proc
host:/# mount
```

In the root directory, download and unzip the following packages:

```
wget http://www.agentpp.com/libdes-l-4.01a.tar.gz
wget http://www.agentpp.com/snmp++v3.2.25.tar.gz
wget http://www.agentpp.com/agent++v3.5.31.tar.gz
tar -xvzf libdes-l-4.01a.tar.gz
tar -xvzf snmp++v3.2.25.tar.gz
tar -xvzf agent++v3.5.31.tar.gz
sudo apt-get install snmpd
```

Compile the downloaded libraries (libdes[2], snmp++, agent++):

```
cd libdes
make
cd ~
```

```
cd snmp++
cd include/snmp_pp/
pico config_snmp_pp.h
```

Change `// #define _NO_SNMPv3` **by** `#define _NO_SNMPv3`

```
cd ../../../snmp++/src/
make -f Makefile.linux
sudo make -f Makefile.linux install
cd ../consoleExamples/
```

After the modifications required…

```
make -f Makefile.linux
```

```
cd ~
cd agent++
cd agent++/src
make -f Makefile.linux
```

[2] This library does not take part in the SNMP emulation of this work due to disabling SNMPv3.

To build one of the example agents, for instance, the ATM scenario used as reference in this work:

```
cd agent++/examples/atm_mib/src
```

After the modifications required…

```
make -f Makefile.linux
```

In case of problems with dependencies packages the following commands may result useful for you:

```
dpkg --force-all -i file_name.deb
apt-get update
apt-get dist-upgrade
```

The DRAGON file system with both GMPLS and SNMP functionalities can be downloaded from:

https://dl.dropbox.com/u/11913620/DRAGON%20FS%20%28GMPLS%20%26%20SNMP%29/Debian-4.0-x86-root_fs-2008Dec10

Contact: dscarabias@gmail.com