



# **“Initial Report On Control Plane Protocol Extension Prototype And Design Issues”**

*D5.2*

**‘DICONET\_D5.2\_WP5\_CN\_15July09\_V1.0\_R’**

Version: 1.0

Last Update: 20/7/2009 5:09:00 PM

Distribution Level: PU



**The DICONET Project Consortium groups the following Organizations:**

Partner Name	Short name	Country
JCP-Consult	JCP	FR
Research and Education Laboratory in Information Technologies	AIT	GR
Center of REsearch And Telecommunication Experimentations for NETworked communities	Create-NET	IT
Groupe des Ecoles des Telecommunications	ENST	FR
Huawei Technologies Deutschland GmbH	Huawei	DE
Interdisciplinair Instituut voor Breedband Technologie, VZW	IBBT	BE
Research Academic Computer Technology Institute	CTI	GR
University of Essex	UEssex	UK
Universitat Politècnica de Catalunya	UPC	SP
ADVA AG Optical Networking	ADVA	DE
Deutsche Telekom AG	DTAG	DE
Alcatel-Lucent France	ALF	FR
ECI Telecom	ECI	IL

**Abstract:** Standard GMPLS protocols which are used for establishment of lightpaths in optical networks, suffers from 1) lack of physical layer impairments (PLIs), etc., and 2) lack of good techniques to disseminate and utilize PLI details. Hence, whenever there is a change in network status, it may need to be communicated to all the nodes. The availability of this up-to-date information is essential to evaluate the effects of PLIs and to decide whether a proposed lightpath is feasible in the optical domain. Hence, there is strong need for development of efficient techniques to address these issues, without which it would be impossible to automatically initiate a lightpath from higher layers. We extend GMPLS protocols to carry PLIs. We present initial design of two selected control plane architectures, namely, hybrid and PCE-based. As both architectures are developed in DRAGON emulator, we briefly discuss DRAGON and related extensions.

*The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 216338”*



### Document Identity

Title:	Initial Report On Control Plane Protocol Extension Prototype And Design Issues
Subject:	Deliverable
Number:	D5.2
File name:	DICONET_D5.2_WP5_CN_15July09_V1.0_R
Registration Date:	19/05/2009 11:32:00 AM
Last Update:	7/20/2009 5:09:00 PM

### Revision History

No.	Version	Edition	Author(s)	Date
1	0	1	Chava	19/05/2009
	Comments:	ToC Release		
2	0	2	Antonio	21/05/2009
	Comments:	ToC Rewriting		
3	0	3	Chava	29/06/2009
	Comments:	Review of first part of the document		
4	0	4	Antonio	30/06/2009
	Comments:	Merging of modifications to the first part		
5	0	5	Chava	
	Comments:	Merging with University of UEssex contributions		
6		6	Chava	
	Comments:	Writing Executive summary, introduction, and conclusions		
7		7	Chava	15/07/2009
	Comments:	Final modifications and review		
8		8		
	Comments:			
9		9		
	Comments:			
10				
	Comments:			
11				
	Comments:			
12				
	Comments:			
13				
	Comments:			
14				
	Comments:			
15				
	Comments:			

## Table of Contents

<b>LIST OF ACRONYMS</b>	<b>6</b>
<b>LIST OF FIGURES</b>	<b>7</b>
<b>LIST OF TABLES</b>	<b>8</b>
<b>EXECUTIVE SUMMARY</b>	<b>9</b>
<b>1. INTRODUCTION</b>	<b>10</b>
<b>2. HIGH LEVEL SYSTEM DESIGN: HYBRID ARCHITECTURE</b>	<b>13</b>
<b>2.1. Modules description</b>	<b>14</b>
2.1.1. TE-Agent Module	15
2.1.2. Signalling Module (RSVP-TE)	15
2.1.3. Feasibility Control Module	15
2.1.4. Resource Manager Module	16
2.1.5. Routing Module (OSPF-TE)	16
2.1.6. CSPF Module	16
2.1.7. Q-Tool Module	17
<b>2.2. Interfaces description</b>	<b>17</b>
2.2.1. Interfaces classification	17
2.2.2. Basic <i>on demand</i> and <i>notification</i> service protocol	17
2.2.3. Interfaces Description	19
<b>2.3. Protocol extensions</b>	<b>21</b>
2.3.1. New/Extended Mechanisms	21
2.3.1.1. Resource Availability-Aware Routing	21
2.3.1.2. Resource Availability Status/Change Notification	22
2.3.1.3. Resource Availability Aware LSP Creation	23
2.3.1.4. Impairment-Aware LSP Creation	24
2.3.1.5. User Request Management	25
2.3.1.6. K-Path Routing	26
2.3.1.7. Resource Locking	27
2.3.1.8. Wavelength Assignment	28
2.3.2. New/ Extended Messages	28
2.3.2.1. RSVP/RSVP: PATH message modification	30
2.3.2.2. RSVP/RSVP: RESV message modification	34
2.3.2.3. RSVP/RSVP: Error messages modification	34
2.3.2.4. Routing Module/Routing Module: OSPF-TE messages extension	34
2.3.2.5. Subscription, Query, and Notification template messages	35
2.3.2.6. User/TE-Agent: Supervising and Operational Requests	36
2.3.2.7. TE-Agent/Signalling Module: Signalling Requests	37
2.3.2.8. TE-Agent/Routing Module: K Routes Evaluation and Network Status Requests	38
2.3.2.9. Routing Module/Resource Manager: Resources Availability Change Notification	38
2.3.2.10. Routing Module/CSPF: CSPF Evaluation Request	39
2.3.2.11. Signalling Module /FCM: Resources Availability & Impairments Feasibility Check	39
2.3.2.12. Signal/Resource Manager: Command Execution Request	39
2.3.2.13. FCM/Resource Manager: Resource Availability Request	40
2.3.2.14. FCM/Q-Tool: Remote and Local Q-Factor Evaluation Request	40
2.3.2.15. FCM/FCM: Protocol definition	40
2.3.2.16. FCM/FCM: Remote Impairments Feasibility Check	41
2.3.3. New/Extended Objects and Data Structures	41

<b>3. PCE-BASED ARCHITECTURE: HIGH LEVEL SYSTEM DESIGN</b>	<b>43</b>
<b>3.1. Modules description</b>	<b>43</b>
3.1.1. NPOT	44
3.1.1.1. IA-RWA	44
3.1.1.2. Q-Tool	45
3.1.1.3. Failure Localization	45
3.1.2. PCE	45
3.1.3. OCC	46
3.1.3.1. PCC	46
3.1.3.2. Routing Module (OSPF-TE)	47
3.1.3.3. Signaling Module (RSVP-TE)	47
3.1.4. TED/PPD	47
<b>3.2. Interfaces description</b>	<b>48</b>
3.2.1. User to NMS requests and response	48
3.2.2. NMS to OCC request	48
3.2.3. OCC (PCC) to PCE request and response	48
3.2.4. PCE to NPOT request and response	48
3.2.5. NPOT to TED/PPD request and response	48
3.2.6. OCC (OSPF-TE) to TED/PPD	49
3.2.7. OCC to NPOT/Failure Localization Module notification	49
3.2.8. NPOT to NMS	49
<b>3.3. Protocol extensions --- OSPF-TE extensions</b>	<b>49</b>
3.3.1.1. Considerations	50
3.3.1.2. Update to the RI LSA	51
3.3.1.3. LSA Flooding Scope	51
3.3.1.4. Motivation for a New LSA	52
3.3.1.5. New PLI LSA	52
3.3.1.6. The PLI TLV	53
3.3.1.7. The Sub-TLVs	54
3.3.1.8. The Sub-sub-TLVs	55
<b>4. IMPLEMENTATION IN A EMULATION ENVIRONMENT: DRAGON</b>	<b>56</b>
<b>4.1. DRAGON overview</b>	<b>56</b>
4.1.1. Overall DRAGON Architecture	57
4.1.1.1. Network Aware Resource Broker (NARB)	57
4.1.1.2. Client System Agent (CSA)	58
4.1.1.3. Application-Specific Topology Builder (ASTB)	58
4.1.1.4. Virtual Label Switch Router (VLSR)	59
4.1.2. Why DRAGON?	59
<b>4.2. Original DRAGON software</b>	<b>59</b>
4.2.1. Restricting the Action Field	59
4.2.2. Dragon Processes and their Interactions	59
<b>4.3. DRAGON Modifications/Extensions</b>	<b>62</b>
<b>5. CONCLUSIONS</b>	<b>65</b>
<b>REFERENCES</b>	<b>66</b>
<b>APPENDIX-A</b>	<b>67</b>
<b>APPENDIX-B</b>	<b>69</b>

## List of Acronyms

AS	Autonomous System
CSPF	Constrained Shortest Path First
D-OCF	Distributed Optical Control Plane
ERO	Explicit Route Object
FA	Forwarding Adjacency
FCM	Feasibility Control Module
GMPLS	Generalized Multiprotocol Label Switching
IA-RWA	Impairment-Aware Routing and Wavelength Assignment
IETF	Internet Engineering Task Force
LI	Linear Impairments
LMP	Link Management Protocol
LSA	Link State Advertisement
LSP	Label Switch Path
LSR	Label Switch Router
NLI	Non-Linear Impairments
OCP	Optical Control Plane
ONIC	Optical Network Interface Card
OSPF-TE	Open Shortest Path First with TE extensions
PCE	Path Computation Element
PCEP	Path Computation Element Protocol
PLI	Physical Layer Impairments
PLIAR	Physical Layer Impairment Aware Routing
PLIA-RWA	Physical Layer Impairment-Aware Routing and Wavelength Assignment
PPD	Physical Parameter Database
PPD	Physical Parameters Database
PSB	Path State Block
QoS	Quality of Service
QoT	Quality of Transmission
RM	Routing Module
RMM	Resource Manager Module
RSB	Reserve State Block
RSVP-TE	Resource reSerVation Protocol with TE extensions
RWA	Routing and Wavelength Assignment
SM	Signalling Module
TE	Traffic Engineering
TE-AM	Traffic Engineering Agent Module
TED	Traffic Engineering Database
TLV	Type/Length/Value
VTY	Virtual TeletYpe

## List of Figures

Figure 1. Hybrid integration approach .....	10
Figure 2. PCE based approach .....	11
Figure 3: Node's Local Modules (Numbers in figure are identifiers for modules and interfaces: X.Y stands for interface #Y exposed by module #X).....	13
Figure 4: Subscription, Query & Notification.....	19
Figure 5: Resources Availability Check.....	23
Figure 6: Local and Remote Impairment Check .....	24
Figure 7: Supervising and Operational Requests .....	25
Figure 8: Control Plane Function Blocks .....	44
Figure 9: Combined GMPLS/General PCE Architecture without Considering the PLI.....	46
Figure 10: Combined GMPLS/PCE Architecture with Considering PLI Information .....	50
Figure 11: Path-Scope Sub-TLV Format .....	51
Figure 12: New Bit in the PCE-CAP-FLAGS sub-TLV .....	51
Figure 13: Proposed PLI LSA ID.....	53
Figure 14: Standard Opaque LSA Header Format .....	53
Figure 15: PLI TLV Format .....	53
Figure 16: Link ID Sub-TLV Format.....	54
Figure 17: Waveband Sub-TLV Format .....	54
Figure 18: Wavelength Sub-TLV Format .....	54
Figure 19: Impairment Parameter Sub-TLV Format.....	54
Figure 20: Sub-Sub-TLV Format .....	55
Figure 21: IEEE Floating Point Format .....	56
Figure 22: DRAGON Control Plane Architecture .....	57
Figure 23: Node's Original Daemons .....	61
Figure 24: Node's Revised Daemons .....	63

## List of Tables

Table 1. Comparison of various control plane architectures.....	11
Table 2: System's Services & Related Modules.....	14
Table 3: Interfaces Description .....	20
Table 4: Mechanisms Description.....	21
Table 5: Services and Messages.....	28
Table 6: User's Supervising Requests.....	36
Table 7: User's Operational Requests.....	36
Table 8: Services Data Requirements .....	41

## **Executive Summary**

A recent approach to network control and management using the GMPLS framework developed by Internet engineering task force (IETF) seems to be emerging as the winning control plane solution for the next-generation optical networks. One of the main applications of GMPLS in the context of optical networks is the dynamic establishment and tear-down of lightpaths. However, it suffers from a lack of physical layer details such as PLIs, transponder characteristics and availability, etc., as discussed in several DICONET milestone/deliverable reports. Hence, whenever there is a change in network status (either due to lightpath setup or teardown), it may need to be communicated to all the nodes in the network. The availability of this up-to-date information is essential for a GMPLS-capable node to evaluate the effects of PLIs and to decide whether a proposed lightpath is feasible in the optical domain. In addition, GMPLS also suffers from the lack of good techniques to disseminate and utilize physical layer details. Hence, there is strong need for development of efficient techniques to address these issues, without which it would be impossible to automatically initiate a lightpath from client layers, for example, a router. In this document we extend GMPLS protocols to carry impairment information. We present initial design of two selected control plane architectures, namely, hybrid control plane architecture in which both RSVP-TE and OSPF-TE are extended and centralized PCE based control plane architecture. As both control plane architectures are developed based on DRAGON emulator environment, we also present a brief discussion of DRAGON emulator. Furthermore, we present the protocol extensions required in each of the above cases and their corresponding message encoding used.



The PCE based architecture is a centralized architecture (shown in Figure 2) with extensions to OSPF-TE to provide PLI and other required information to the PCE manager. The PCE client runs on all nodes and contacts centralized PCE server for IA-RWA and feasibility check of LSP setup. Brief working of PCE-based architecture is as follows: The source node contacts the PCE server asking for a lightpath establishment to destination node. The IA-RWA and Q-Tool are implemented as part of NPOT will be used by PCE feasible route computation and wavelength assignment. All the required information (i.e. network topology, currently established connections and physical layer impairment information) are already collected in the PCE central database through OSPF-TE extensions. The IA-RWA computes a route and wavelength (i.e. lightpath) for the requested demand and will be provided to RSVP-TE. The standard signaling component of the control plane (i.e. GMPLS/RSVP-TE) is used for lightpath provisioning. Note that extended RSVP-TE can also be used to carry PLI information to double check the feasibility of lightpath to avoid possible use of outdated information. In this case the destination will use the tools developed to for feasibility check before actually provisioning the resources during RESV phase.

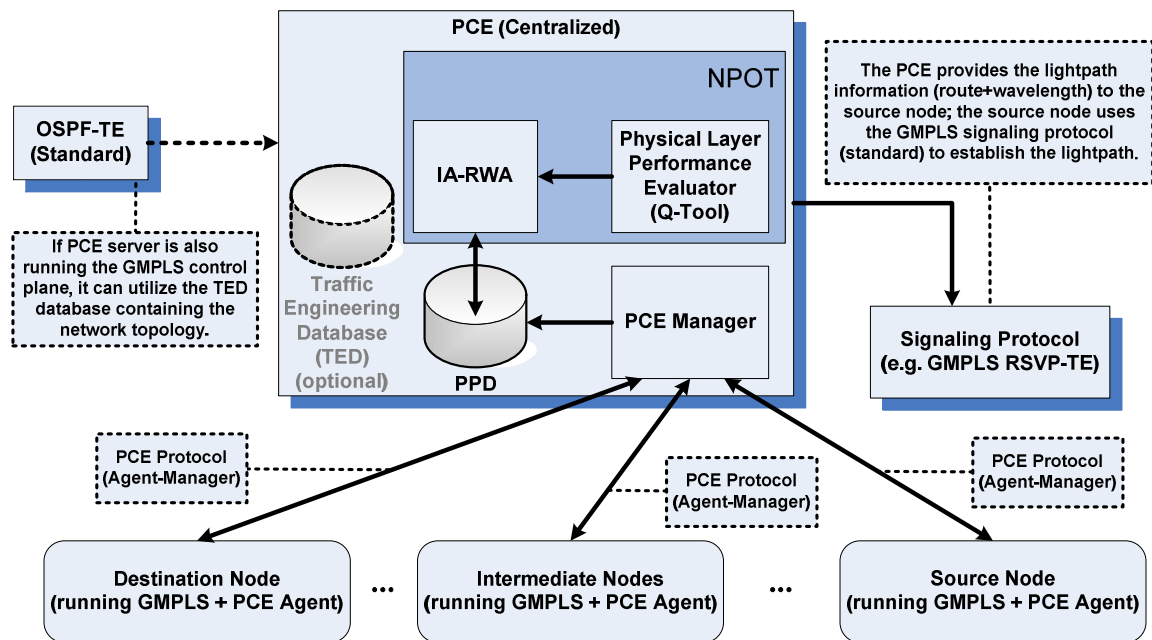


Figure 2. PCE based approach

Based on D2.3 and several discussions, Table .1 summarizes the various performance metrics and engineering metrics that led to the choice of these architectures.

Table 1. Comparison of various control plane architectures

Metrics		Signaling based approach	Routing based approach	Hybrid approach	PCE based approach
Performance	Blocking probability	High	Medium	Low	Low
	Average number of attempts	High	Medium	Low	Low
	LSP set-up time	High	Medium	Low	Low
	Path computational complexity	Low	High	Very high	Very high
Engineering	Control Plane Load	Low	Medium	High	Medium
	Robustness to TED inconsistency	High	Low	Medium	Low

<i>Protocols to modify</i>	RSVP-TE and implementation of new protocol for handling possible active lightpath disruption	OSPF-TE	Both RSVP-TE & OSPF-TE	PCEP or OSPF-TE (in DICONET we consider only OSPF-TE extensions)
<i>Distribution of PLI information</i>	Local	Globally flooded	Globally flooded	Communicating PLIs information to PCE server
<i>Impact on control overhead</i>	Low	High	Very high	Medium
<i>Impact on existing protocol modules</i>	High	Low	Very high	Medium
<i>Standardization Efforts</i>	Initial efforts started in IETF in recent years	Some existing efforts inside IETF	No	Initial stages

The objective of this document is to present the detailed prototype design of both selected architectures. Section 2 presents a detailed design of hybrid architecture. It identifies the extensions that need to be done in DRAGON modules and the new modules that need to be introduced. Furthermore, it also identifies and defines several interfaces. The full implementation of various messages across different modules is identified and described in Appendix-B. Section. 3 presents the high-level architecture of the PCE-based architecture. It also provides extensions required by OSPF-TE with encoding details. Section. 4 discusses the DRAGON emulator and briefly describes various modules involved. Finally, Section. 5 provides a summary of the document.

## 2. High Level System Design: Hybrid Architecture

The hybrid control plane architecture is distributed GMPLS control plane architecture with modifications to RSVP-TE to carry PLI information. The standard OSPF-TE is extended to carry wavelength availability information. The high level system design of hybrid control plane architecture with various modules and interfaces in shown in Figure. 3. As hybrid control plane architecture is distributed architecture, all the nodes in the network implement the various modules/services. The various modules and the interfaces are discussed briefly in this section.

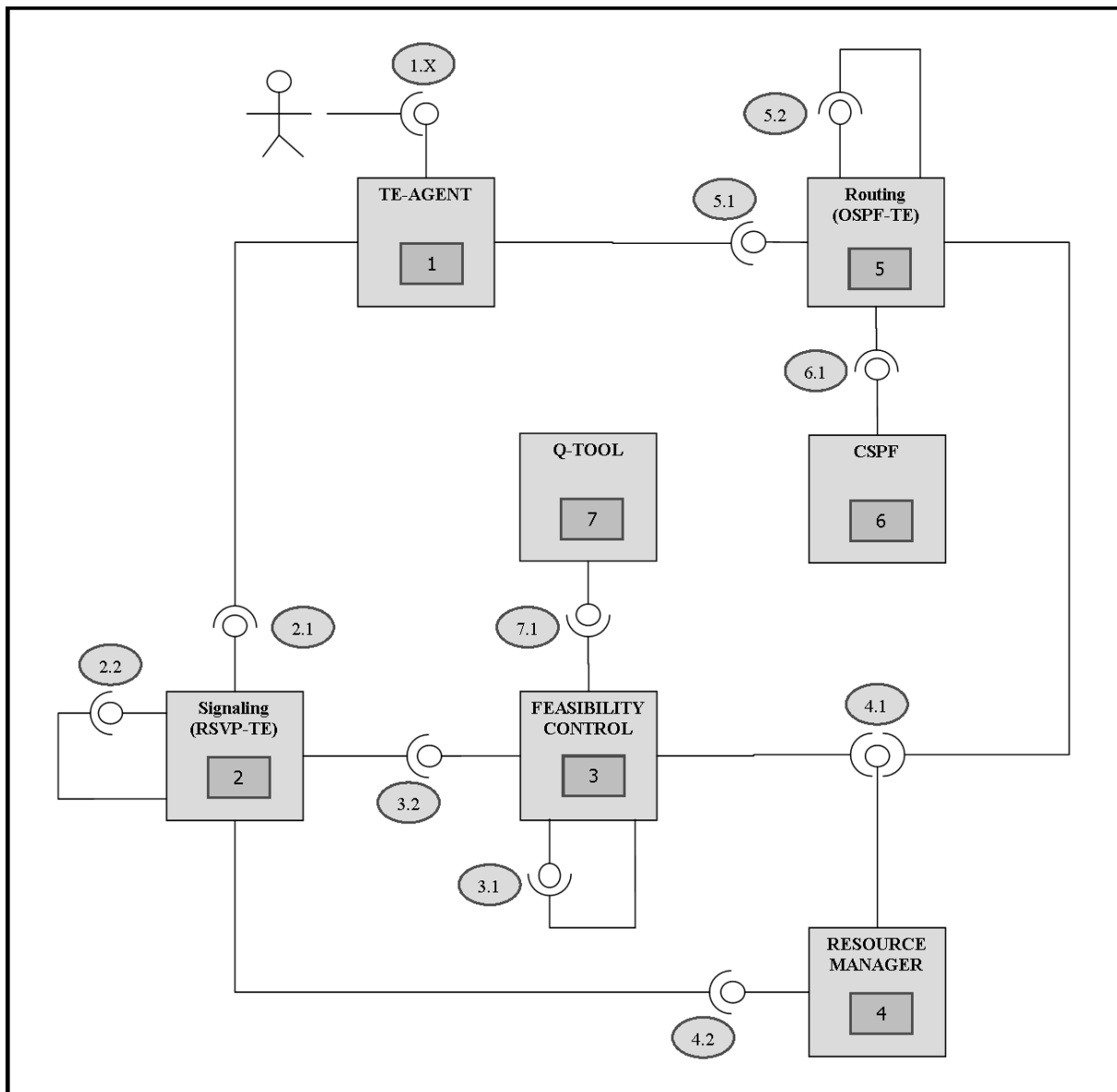


Figure 3: Node's Local Modules (Numbers in figure are identifiers for modules and interfaces: X.Y stands for interface #Y exposed by module #X)

## 2.1. Modules description

In our design, the hybrid architecture is going to be implemented in a distributed fashion: no centralized controllers are placed in the network, but every node has total control over its resources and performs locally all the feasibility evaluation needed by incoming requests. Clearly, each node needs to retrieve information required for performing reliable operations (e.g., knowledge of the existent optical links and of resources available on other nodes), but such information is retrieved communicating directly with other nodes, without passing through a unique centralized management unit. This implies that on every node all operations are locally implemented and supported by specific pieces of software, called “modules” further in the document. Every module is supposed to implement a specific service and to eventually interact with modules on the same or different node. For example, OSPF-TE is extended to disseminate wavelength availability information. RSVP-TE is extended to carry PLI information required to evaluate the feasibility of new lightpath and all active lightpaths.

Six main services have been identified and described in Table. 2.

**Table 2: System's Services & Related Modules**

<b>Service</b>	<b>Description</b>	<b>Module(s) that provide the service</b>
<i>User's Requests Handling</i>	Reception of user's requests for LSP setup / modification / teardown, their translation in appropriate messages for signalling layer and communication of operation's response back to the user	TE-Agent
<i>Routing</i>	Selection (inside the set of all possible paths in the network going from a specific source node to a specific destination node) of those paths which seem to offer the best performances under specific conditions	Routing (OSPF-TE) + CSPF
<i>Signalling</i>	The entire process which coordinates nodes activity to allow the achievement of user's requested operations and the maintaining of the current data layer configuration	Signalling (RSVP-TE)
<i>Resource Management</i>	Management of those resources every node puts at system's disposal for GMPLS operation, i.e., link and node resources such as wavelengths, etc	Resource Manager
<i>Feasibility Checks</i>	Check of user's requests feasibility (Actual resource availability, checking whether new LSP is feasible in optical domain, checking whether the new LSP disrupts any active LSPs by introducing excessive crosstalk)	Feasibility Control + Q-Tool
<i>Supervision</i>	Supervision over the current status of data layer (current active LSPs, inactive LSPs, etc...), available wavelengths, and network configuration	TE-Agent

Starting from these services, seven different modules have been defined: all of them are implemented and run on every node and their description is given in the following paragraphs.

### 2.1.1. TE-Agent Module

TE-Agent Module (TE-AM) has been introduced to implement the User's Request Handling. It works as an interface between the provisioning system and the end-user, initiating every provisioning action requested by the latter and notifying eventual errors occurred during the execution of such request, specifying the nature of the failure and keeping track of it.

TE-AM implements the supervision feature, too: it offers monitoring capabilities over data link layer status, answering to some simple status requests coming from user (e.g., the list of the currently active or inactive LSPs, LSP description, etc.), and may query the routing module in order to retrieve information about current resource allocation and network topology.

The decision to implement monitoring on TE-Agent derives from the fact that all the information about data-link current status and modifications already pass through this module, as consequence of being the "interface" module between user and system. Appropriate data structures are required to store the information such service need to be supported.

### 2.1.2. Signalling Module (RSVP-TE)

It is the module in charge of the signalling protocol, i.e. the process of exchanging messages within the control plane to set-up, maintain, modify and terminate data-paths (LSPs) in the data plane.

Consequently, its messages carry all the information needed for LSP's creation, maintenance and deletion, including some information "transparent" to signalling layer and addressed to those local modules which actually update these messages as these traverse from one node to other along the route and perform the feasibility evaluation (Feasibility Control) of the required operation at designated nodes. The results obtained by such evaluation are then used by signalling module. Such interactions between signalling module and local evaluation modules require defining appropriate interfaces and protocols for data exchange.

Specific objects and data structures are required by signalling module to store locally the information related to current status and LSPs management and information which could be later used in order to process further incoming signalling messages. Moreover signalling module requires during LSP setup/teardown process to send instructions to the resource manager module, in order to reserve/release resources.

In this document, signalling module is seen as an implementation of RSVP-TE within GMPLS framework. RSVP-TE is a GMPLS oriented protocol derived from RSVP (Resource reSerVation Protocol), a transport layer protocol designed to reserve resources across a network for an integrated services Internet and described in [2] and [3].

### 2.1.3. Feasibility Control Module

Feasibility control module (FCM) is the module where all the decisions regarding the feasibility of LSPs creation/modification/deletion are taken. The LSPs are checked for optical feasibility considering the PLIs which are carried in RSVP-TE messages. Requests for feasibility may come from:

- local RSVP module: check if node available resources (e.g., wavelengths) match the requirements specified for the creating LSP and the received signal quality is above certain threshold for the lightpath under consideration

- other FCMs running on different nodes: cross-check the impact of creating a new LSP on the already existing LSPs, i.e., make sure that setup of new LSP does not disrupt any active LSPs

FCM is also interfaced to resource manager module to retrieve all the local configuration parameters needed in its evaluations (actual availability of local resources) and to the Q-Tool module, which performs the overlapping LSPs impact evaluation.

#### **2.1.4. Resource Manager Module**

Resource manager module (RMM) is the module interfacing control plane with the managed devices, allowing the RMM to query the devices about their current configuration, resource availability, and even to ask the devices to actually perform the lightpath creation (cross-connections) or release.

RMM is also responsible for resource availability changes notification to routing module: every time there is a resource status change (e.g., lambda availability, bandwidth, etc.), a notification has to be sent to the routing module.

#### **2.1.5. Routing Module (OSPF-TE)**

Routing module is in charge of the route provisioning system for the control/data plane. It requires an updated knowledge of the overall network (in order to perform reliable path computation) and to define path ranking metrics tailored to end user needs (in order to handle the availability of multiple paths). Depending on the chosen metric, some information regarding nodes configuration may be required (e.g., lambda's or bandwidth availability) by evaluation process, so such information must be stored in routing module and efficiently updated according to status change. Every resource availability change need to be notified to routing module and disseminated in the network as soon as it occurs, in order to grant an efficient functioning of the routing algorithm and to avoid misalignments in the routing tables.

In this document routing module is implemented by extending OSPF-TE (Open Shortest Path First) protocol, a link-state routing protocol which routes Internet Protocol (IP) packets solely within a single routing domain. In OSPF-TE, the network description is translated into a graph in which network nodes become nodes in the graph and links becoming edges in the graph. Depending on links characteristics, a weight is assigned to every edge in the graph, and then shortest path first algorithm is applied. The OSPF-TE is deeply bound to the CSPF module, which performs a more restrictive path search by filtering the links following some defined rules.

#### **2.1.6. CSPF Module**

Constrained shortest path first (CSPF) performs an extension to the path search strategy adopted by OSPF-TE. The path computed using CSPF is the shortest path fulfilling a set of constraints, i.e., shortest path algorithm is run after pruning those links which violate a given set of constraints.

CSPF module is strictly related to OSPF-TE as it accesses the graph representation of the network build by OSPF-TE and then applies its filters on it. As for OSPF-TE running on routing module, different metrics could be defined for link weight assignment before evaluating shortest path algorithm. From the above discussion, CSPF could be seen more as a plug-in for OSPF-TE than as a stand alone module, but we kept it separate to emphasize its fundamental role in driving the route selection process.

### 2.1.7. Q-Tool Module

Q-Tool is the main module where optical feasibility is evaluated. It uses the PLI information carried in RSVP-TE messages to evaluate the feasibility. Q-Tool main function is to evaluate Q-factor value, i.e., to give an estimation of the impact that a set of partially or fully overlapping LSPs have on new LSP and vice versa. Such value (calculated for new LSP and for all active LSPs which share at least a link with the new LSP) is used by feasibility control module in the overall feasibility evaluation.

## 2.2. Interfaces description

In order to allow the various modules described earlier to communicate several interfaces are defined. Before presenting detailed interfaces description (Section. 2.2.3), some general clarifications regarding use of interfaces is presented in Sections. 2.2.1 and 2.2.2.

### 2.2.1. Interfaces classification

Interfaces may be divided into two groups, depending on the existence of a priori defined hierarchy between the two connected devices:

- Client/server interface  
It connects two modules, one (server) characterized by providing a service and the other (client) characterized by requesting such service.  
A service requires performing an action or a set of actions: at this level we can make a distinction between *on demand* service in which client explicitly asks the server for a particular service by passing some parameters in the request message, and *notification* service, where client just subscribes to the service at the beginning and then it is servers responsibility to send to all clients (subscribers) a notification message every time a specific event occurs (in the subscribe request clients may specify the event it is interested and the way it wants to be notified).
- Peer-to-peer interface  
This kind of interfaces do not have specific distinct roles in the message exchange: none of them (a priori) just provides or receives a service. It can be seen as a bidirectional client/server interface, where both nodes may ask or being asked for service.

### 2.2.2. Basic on demand and notification service protocol

As described in previous chapter, modules may be classified in two sets (partially or totally overlapping): service providing modules and service requesting modules. Generalizing, for a service requesting module the basic needs are: having granted the access to the service and service realization. Regarding service realization, we have to make a further distinction between the on demand and the notification service realization procedures. Applying such considerations to client/server Interfaces, three kinds of basic message exchange patterns may be identified and shown in flow chart diagrams in Figure 4:

- Subscription: is the procedure which allows a candidate node (client) to request to another node (server) about its interest in one or more of the services the server offers/provides. It consists merely the ordered exchange of 2 kinds of messages: a *subscription request (subReq)*, sent by subscribing module to service providing

module, and a subsequent *subscription response* (*subRsp*), sent by service providing module to subscribing module after having received and processed *subReq* message. Different behavior may be implemented in subscribing module in case of missing *subRsp* from service providing module: if required service is a mandatory service, subscribing module may wait for a certain time interval and then resend the *subReq* or *subReq* may be considered refused and the subscription process quitted (or postponed). Similarly, different policies (depending on the nature of requested service) may be implemented in service providing module for *subReq* acceptance or refusal. Also depending on the nature of requested service, *subReq* message may specify some additional parameters in order to realize its purpose: for example, a *subReq* to a notification service may add some additional parameters describing the way client wants notification to be sent (e.g. periodical or event driven).

- Query: is the message exchange between a client module and server module in which client module asks a server module for a subscribed *on demand* service (via *service request* message, *serReq*). Server answers with a *service response* message (*serRsp*), containing the result or response for the required operation.
- Notification: is a *notification* service to notify occurrence of a specific event as per subscriber needs. It just consists of an event notification (*evtNtf*) message, which server sends to all subscribing clients when given specific event occurs. Event triggering notifications could be for example, changes in value of monitored parameters or having a parameter exceeding or going below a given threshold.

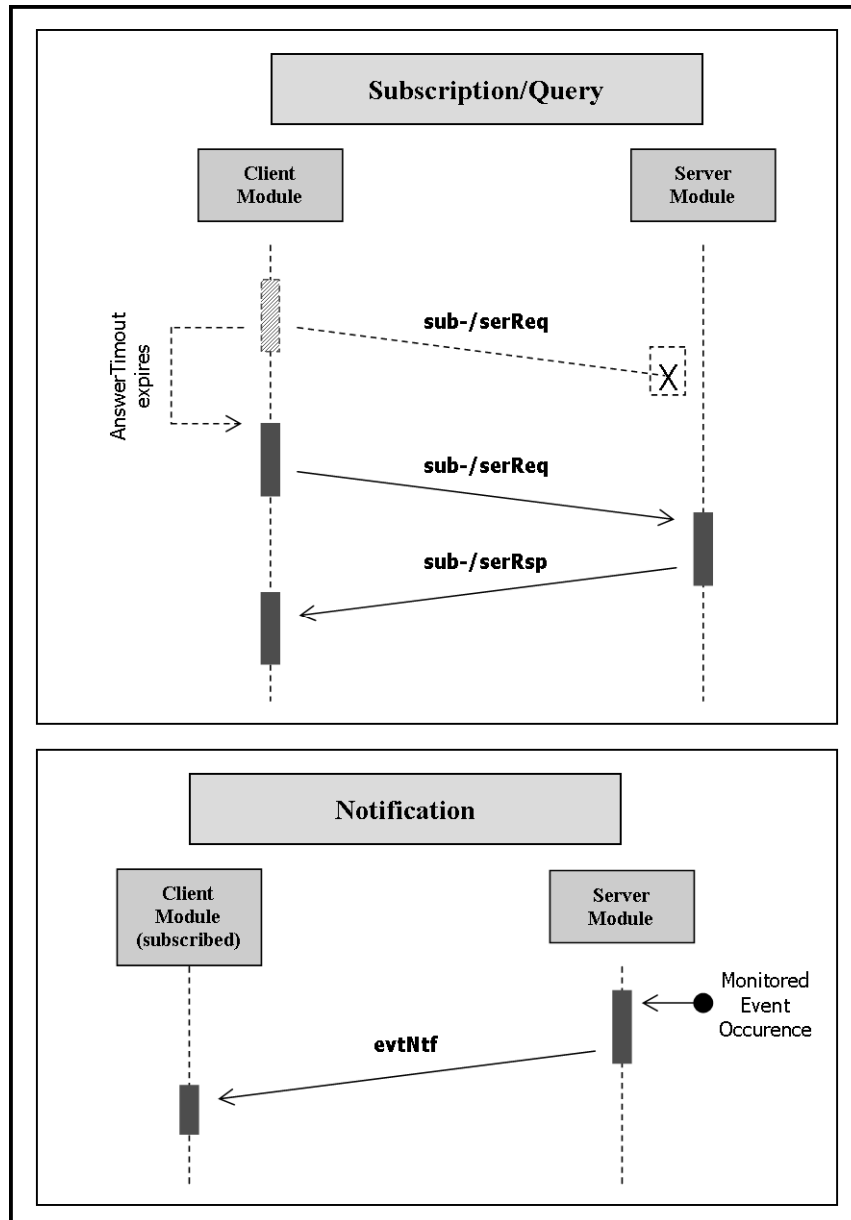


Figure 4: Subscription, Query & Notification

To perform a query, a client may not necessarily need to subscribe to service, in which case the server decides how to answer to such “irregular” request. For a notification service instead subscription is a mandatory step, in order to let the server be aware of who is actually interested in the service and how to respond. Finally, both on demand and notification services may be provided concurrently over the same interface. The description is also valid for peer-to-peer interfaces as described above. The actual difference between the two interfaces is a priori definition of a hierarchy in service provisioning. Note that subscription and query messages flow is similar, but, the contents of the message is not same and depends on the type of message.

### 2.2.3. Interfaces Description

Figure 3 shows the planned interfaces in order to have the system compliant with the requirements of the DICONET project. A description of all interfaces is given in Table 3.

Table 3: Interfaces Description

Code/No.	Involved Modules	Type of Interface	Description of Interface
1.X	User → TE-AM	C/S	Allows users (being them human or software) to send their requests into the system and to receive back the related responses.
2.1	TE-AM → Signalling	C/S	Allows TE-AM to inject user requests (formatted in a proper way) in the signalling layer and to get back related responses.
2.2	Signalling - Signalling	P2P	Allow communication between signalling modules on different nodes, as expected by RSVP-TE protocol, which is supposed to be implemented by signalling. RSVP-TE messages (extended in order to carry the additional information required for impairment management) are exchanged through such interface.
3.1	Signalling → FCM	C/S	In order to carry on/abort the LSP creation/teardown process signalling module requires querying the local FCM for actual resources availability and/or for the impairment feasibility evaluation. Related responses may contain additional information later used in the signalling process.
3.2	FCM – FCM	P2P	When a LSP setup request arrives to destination node, local FCM is also asked to perform a remote Q-factor evaluation, i.e., to request all other FCMs on destination nodes of LSPs potentially affected by the creating new LSP to evaluate such disrupting effect and to send back response. This done using Q-CHK-Protocol defined in D2.3.
4.1	Signalling → RMM	C/S	Allows signalling to request resources actual allocation/release to RMM and to get back related responses whether such operations are successfully performed.
4.2	FCM → RMM Routing → RMM	C/S	Such an interface is used for two different services and subscribed by two different local modules. FCM asks RMM for all the local available resources, which it uses in its feasibility computations. Routing instead subscribes a notification service which allows it to get updates on resource's availability status change as soon as change occurs, in order to perform more reliable route's evaluations.
5.1	TE-AM → Routing	C/S	Allows TE-AM to request route computation to Routing Module and to get related response. Moreover TE-AM may retrieve network topology and resource availability information.
5.2	Routing – Routing	P2P	OSPF-TE protocol implementation in routing module requires all routing modules to share information: such an interface allows local routing modules to communicate with remote routing modules on other nodes. OSPF-TE messages (extended in order to carry the additional information required for resources availability awareness) are exchanged through such interface.

6.1	Routing → CSPF	C/S	Allows routing module to access route evaluation service provided by CSPF (specifying the desired LSP characteristics) and to receive the resulting list of k candidate paths.
7.1	FCM → Q-Tool	C/S	Allows other local FCM to access Q-factor evaluation service (impairments estimation and feasibility check) provided by Q-Tool module.

## 2.3. Protocol extensions

In this section we provide a description of all the new services meant to be introduced in DICONET project. A part of them just requires some little adjustments to the original behavior in order to be integrated; others instead require a full definition, as they are not simply extensions, but completely new features. A general description of the introduced mechanisms and their translations in protocols and messages exchanges is given in the following subsections.

### 2.3.1. New/Extended Mechanisms

All the to-be-introduced mechanisms are listed in Table 4

**Table 4: Mechanisms Description**

Code	Mechanism	Kind	Description
1	<i>Resources Availability Aware Routing</i>	Ext.	During route evaluation the actual resource availability is considered in the computation
2	<i>Resource Availability Status/Change Notification</i>	New	A system for real time monitoring of the actual availability of resources at each node
3	<i>Resource Availability Aware LSP Creation</i>	Ext	An extension to the LSP creation process, checking LSP feasibility from resources availability's point of view
4	<i>Impairment Aware LSP Creation</i>	Ext	An extension to the LSP creation process, checking LSP feasibility from impairment's point of view
5	<i>User Requests Management</i>	Ext	A system for handling incoming request and responses from/to user(s)
6	<i>K Path Routing</i>	Ext	An extension to the way candidate route for a setting-up LSP is provided
7	<i>Resource Locking</i>	New	A new feature aiming to serialize access to resources in case of parallel LSP setting-up processes
8	<i>Wavelength Assignment</i>	Ext	Optimization of the wavelength assignment

Most of the above listed mechanisms are involved in LSP's setting up process: some of them (2 and 7) have been introduced just to support the correct working of other mechanisms. A more in depth explained in the following dedicated subsections.

#### 2.3.1.1. Resource Availability-Aware Routing

LSP creation process requires setting-up LSP's main characteristics defined *a priori*, e.g., required Q-factor, QoS, type of protection. So a certain number of constraints to the route selection is known and it is up to the routing module take advantage from such knowledge in

order to perform some optimizations in the route computation process. As described in Section. 2.1.5, we expect routing module (RM, running OSPF-TE) to collect at least the most significant configuration parameters from all interacting devices and to continuously update them, in order to have up-to-date view of the current status of the network. CSPF is the module in charge of the actual selection of the potential paths (see k-path routing description in Section 2.3.1.6): applying the constraints defined in the request message as a filter over network description available through OSPF-TE TED. CSPF can prune all those devices/links that does not satisfy the constraints requested in LSP request. Performing the route calculation over such modified graph leads to a faster and successful solution.

Two issues that need to be considered in using such approach (especially for large size networks) are the amount of resources required to store the detailed network description and the time required to perform pruning and eventual weight calculations. A possible solution could be to reduce the number of monitored parameters to most significant parameters and adopting some alternative LSP setup procedures (k-path routing being one of them) to increase the probability of acceptance. To perform correctly all the above described routing calculations, RM needs up-to-date knowledge of the network status from a resource availability point of view. Hence, it is necessary to introduce a service which provides up-to-date information to RM in a fast and reliable way.

#### **2.3.1.2. Resource Availability Status/Change Notification**

With reference to the notification service description given in Section. 2.2.2, an instance of such a service is provided by resource manager module (RMM) to allow all the local modules to get updates regarding changes in resources allocation status. In fact RMM is the module which monitors and accesses all the local physical devices and their configurations and for these reasons it is the perfect place to implement such services.

Routing module (RM) which implements resource availability-aware routing protocol as described in the previous paragraph, in turn gets updated information regarding resource availability and all parameters related to resource availability which are used by route computation algorithm. This makes RM an ideal candidate client for such notification service. As discussed earlier, RM implements the OSPF-TE protocol which needs to know exact network configuration and status in order to perform route computation correctly. Therefore it needs both to subscribe to the service and to retrieve configuration's details during its initialization phase; otherwise it will not have enough information to perform the route computation.

After completing registration and (the first) query for local resources allocation, RMs initialization phase is considered completed and OSPF-TE can start its activity. Incoming notifications have to be handled by RM in order to update the changes in network resources and local parameters locally and to advertise to other nodes in the network. Following notification, RM may detect some misalignments between stored network configuration and the parameters carried in the notification message (due to resource release/usage). It also may happen in case of loss of previous notification message or node/link failures. In such a case, RM sends a new query to RMM for local resources allocation and updates the overall configuration to resolve the detected misalignments or inconsistencies. With reference to Figure 3, such protocol is implemented using interface (4.1), which is exposed by RMM module and subscribed by RM. As another service is also implemented over the same interface and queried by feasibility control module, so the exchanged messages have to be designed to avoid overlapping errors between the two services.

### 2.3.1.3. Resource Availability Aware LSP Creation

During the LSP creation process, signaling protocol (i.e., RSVP-TE) which is implemented in signalling module (SM) sends downstream (i.e., from source node to destination node) a PATH message which collects resources availability (i.e., wavelength availability) information along the path. Collected information is used to perform feasibility checks in order to verify that the candidate route actually satisfies the LSP requirements. It mainly checks that there exist at least one common wavelength/lambda available on all the links along the path. Such a check is performed by the information regarding common available wavelengths in the PATH message (see PATH message extension) and the local resource availability information at each node along the path. It prunes on every hop those wavelengths which are not available in the current node. If available lambda set (i.e., label-set) is null, no route can be established and hence, LSP creation for selected path is aborted (by sending PATH\_ERR message in the upstream direction to the source node). As described in 2.3.1.1, a similar check is also performed by RM during route calculation, but it is repeated locally during setting-up phase to be sure that the resources are still available during actual PATH message. So it is preferable from a performance point of view to make a more generic estimation at RM level and then proceed with a more reliable check during the actual LSP setup. The locking system described later further improves the chances of finding the available resources for successful LSP setup.

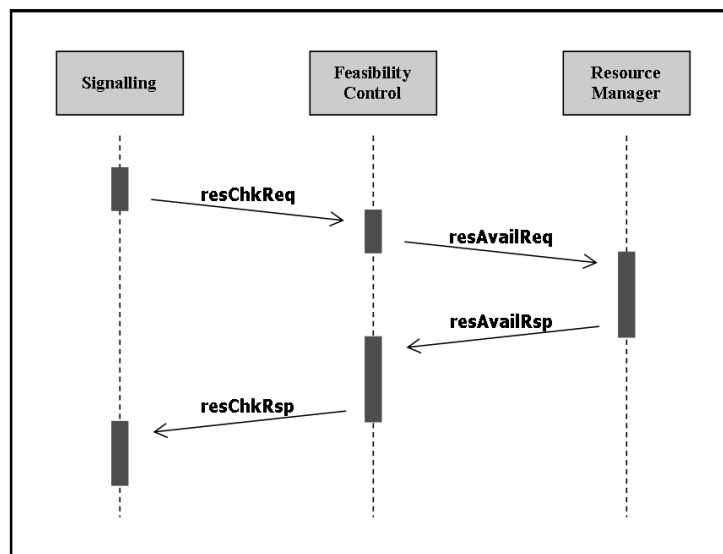


Figure 5: Resources Availability Check

In our design all feasibility checks are performed by feasibility control module (FCM). It receives a check request (*resChkReq* in Figure 5, containing the set of previous nodes available resources) from SM, which in turn queries the resource manager module (RMM) for local available resources (*resAvailReq*). As RMM answers providing the set of available resources (*resAvailRsp*), FCM can compare them with those received from RMM request, looking for common/compliant elements and checking parameters suitability to LSP requirements. If both checks are positive then a positive answer (*resChkRsp*, containing the set of selected local resources) is sent to SM carrying on LSP creation process, otherwise a negative response is sent back and LSP creation process is aborted by SM.

### 2.3.1.4. Impairment-Aware LSP Creation

Along the path chosen for a new LSP, several impairments accumulate and increase the noise level which may exceed a certain threshold leading to unfeasible LSP in optical domain. Moreover, the new LSP may potentially disrupt the active LSPs by introducing excessive crosstalk on the links that are common for both new and active LSPs. Note that setting-up of an unfeasible LSP is a waste of time and resources. Furthermore, disrupting active LSPs is even worse (considering the huge amount of information which may be lost before problem being detected and solved). Hence, it is essential to introduce an extra check to make sure that new LSP is feasible in optical domain and will not disrupt any active connections during LSP setup.

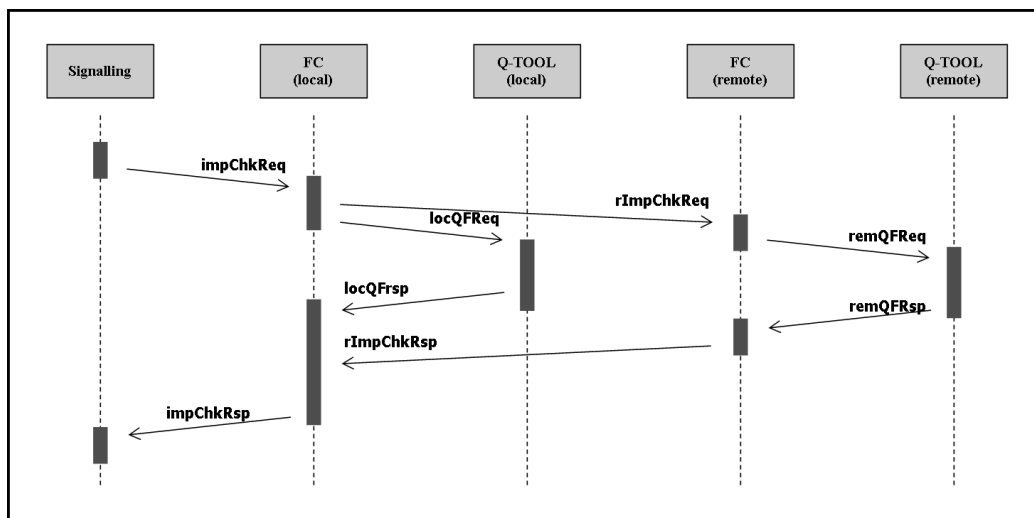


Figure 6: Local and Remote Impairment Check

Q-Tool module has been developed to evaluate the effects of impairments on new and active lightpaths. When a RSVP-TE PATH message which collects all PLI information along the path reaches the destination node of new LSP, the signalling module sends request to FCM to check the resources availability and the impact of impairments (*impChkReq*) on new LSP and also active LSPs, as shown in Figure 6. Then FCM sends a request (*locQFReq*) to local Q-Tool module to evaluate the Q-factor value, which is an estimation of the impact of the impairments on new LSP. Moreover, FCM also sends a remote impairment evaluation request (*rImpChkReq*) to all other remote FCMs running on destination nodes of active LSPs sharing some common links with the new LSP, in order to check the effect of new LSP on potentially affected active LSPs. All the remote FCMs receiving such request immediately query their local Q-Tool (*remQFReq*) and send back the response (*remQFRsp*) to the awaiting FCM (*rImpChkRsp*). If the FCM module on destination node of new LSP receives at least one response that carries a Q-factor value less than the required threshold, then LSP setup process is aborted and a RSVP-TE PATH\_ERR message is sent upstream. Otherwise, RSVP-TE sends upstream a RESV message in order to complete the setup of new LSP. Clearly, the first negative response that arrives at FCM module on destination node of new LSP is enough to kill the LSP setup process. In order to avoid having Q-checks on remote destination nodes pending for indefinite time, affected FCMs response should arrive to the requesting FCM within a limited amount of time. If this timeout expires before having all the positive responses, remote Q-check is considered negative and a negative response is sent back to SM. The factors that affect or need to be considered in deciding the time out value: the Q-factor computation time and req/resp times.

### 2.3.1.5. User Request Management

User requests are all those messages end user sends through “TE-Agent to End User” interface (marked 1.X in Figure 3). Incoming requests may be divided in two classes:

- Supervising requests
- Operational requests

Supervising requests are those related to retrieving some information about network status. In order to answer such requests, TE-Agent avoids interacting with RSVP-TE module, instead it directly collects the desired information from its internal data structures (for LSPs information) or queries routing module (RM) if necessary. Whereas, operational requests are those which require an intervention, implying a potential change in the network configuration/topology. All requests related to LSP’s creation/modification/deletion are clearly operational requests. Operational requests needs to be translated by the TE-Agent in to a list of operations (e.g., querying of routing module etc.,) targeting the preparation of the appropriate message to send to the signalling module in order to satisfy user’s request.

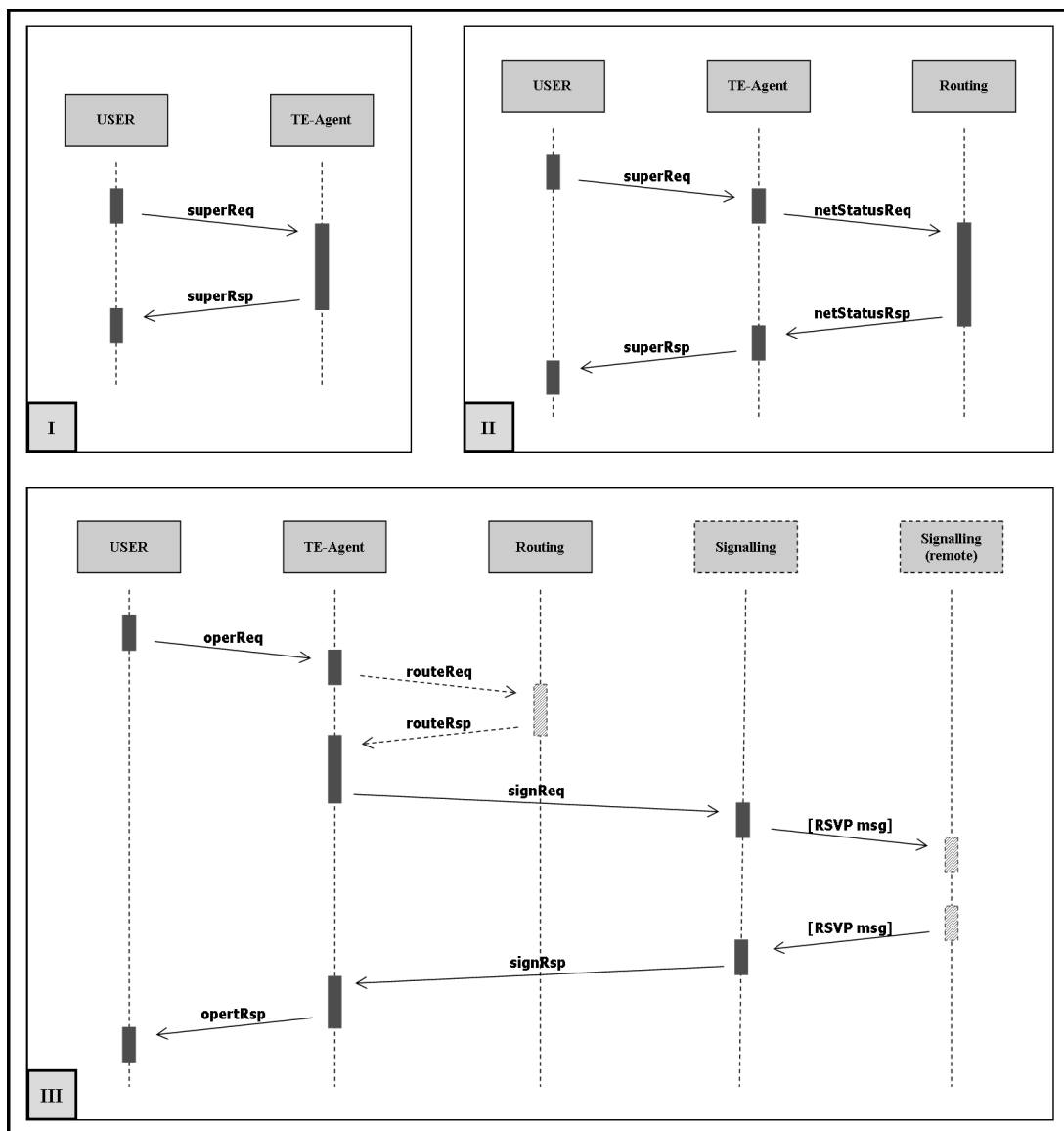


Figure 7: Supervising and Operational Requests

Sequence diagrams shown in Figure 7 describe graphically how supervising (I and II) and operational (III) requests behave:

- I, *LSP Information Request*: User asks TE-Agent for information regarding LSPs status via *superReq* message and TE-Agent answers with a *superRsp* message retrieving the information directly from its internal data structures.
- II, *Network Status Request*: User asks TE-Agent (TE-AM) for information regarding the network status (available nodes, links, resources etc..) via *superReq* message. The TE-AM in turn queries routing module (RM, via *netStatuReq* message) for information regarding network status, then after getting response (*netStatuRsp*) from RM an answer (*superRsp*) is sent to the awaiting user. Note that, TE-AM will not directly query the RMM for local resources.
- III, *Operational Request*: with an *operReq* message user may ask TE-AM to perform some operation via SM. In order to specify correctly the requested action, TE-AM may query RM for some additional information (e.g., if user wants a new LSP to be created, TE-AM asks RM for candidate routes). After obtaining all the required information, TE-AM sends a *signReq* to SM specifying the requested operation and required parameters. The request is translated by SM in to an appropriate RSVP-TE message, which is sent to other SMs according to the required operation. When a *signRsp* containing the response about requested operation arrives to TE-AM; such response may trigger some other operation on TE-AM (e.g., a failure in LSP setup may start another try on next candidate route, see 2.3.1.6) or an *operRsp* message be sent to the awaiting user, communicating the final result.

Independently from the class of the request, TE-Agent always answers with a response message. However, the content of such response message depends on the kind of request (e.g., it could be a simple list of LSPs, just a simple success notification, a failure notification with an error code or description etc.). In order to avoid having pending requests for an indefinite time it will be introduced on TE-Agent's side a timer for every incoming request. If timer expires before getting an answer, a response message containing a failure notification is sent to the user. The choice regarding the timer value is particularly relevant in case of an operational message, whose target action may require a long sequence of messages exchanges and computations between modules (often belonging to other nodes). Link locking feature (2.3.1.7), which is helpful in avoiding conflicts during LSP setup, may cause an extra delay in request processing and consequently need to be considered in selecting time value.

#### 2.3.1.6. K-Path Routing

Due to impairment checks performed during LSP setup process and assumptions made during route computations, there is some probability that the selected candidate route is unfeasible. Therefore it is worth to try  $k$ -paths than a single shortest path, for LSP setup. So for every LSP request  $k$  potentially feasible paths are evaluated, and are tried in that order till a feasible path is found. The ordered path list is processed sequentially, trying to setup the LSP along the selected route, as soon as it finds feasible path, LSP setup process is complete and a success notification is returned. If none of the candidate paths is feasible, LSP setup request is considered failed, and a failure notification is returned.

Optimization based on a given metric is the main target of routing algorithm, hence candidate paths returned in the form of ordered list based on the given metric. However, the route computation made by RM may not fully consider impairments effects and/or affected by misaligned data due to RMs not up-to-date view of the network resources. Hence, to reduce the many failure attempts and waste of time the following solutions may be helpful:

- Route reordering: reorder the routes list taking in account both optimality and potential feedback from occurrence of errors.
- List pruning: let TE-Agent aware of failing link and let it prune all those remaining candidate paths which contain it.
- Links pruning at RM level: let TE-Agent query RM only for a single route and send it to signalling, but eventually let it repeat (max for  $k-1$  times) the path request to RM specifying eventual critical links to prune and send the LSP request over the new proposed path

but even them are not totally free from errors (e.g., a node may lead to failure as effect of its and previous links impairments, but not being intrinsically critical) and may require a considerably larger amount of computations (and time) to be performed.

### 2.3.1.7. Resource Locking

One of the potential issues that may occur during LSP creation is having another concurrent LSP creation process in progress which is sharing one or more links. Such an issue is potentially harmful due to the fact that the first LSP does not know of the existence of the second LSP and therefore does not consider it in feasibility evaluation and/or vice versa. In order to prevent such a situation and handle concurrent requests, resource locking mechanism over shared resources has been implemented. The idea is to lock all the resources which could be used by the first one and block the second LSP request till the first LSP request is finished. Two approaches may be followed:

- Restriction unawareness: the second LSP just sees a reduced set of potentially available resources, but it is unaware of the reason for reduced set, so proceeds with its normal process on a reduced set of resources.
- Restriction awareness: the second LSP creation process knows that some of the resources which it could use are blocked by an ongoing LSP request, so it waits for the actual resources used to be notified or until the lock expires.

The unawareness grants a faster but far from optimality route feasibility evaluation, whilst the awareness grants a smarter usage of the available resources at the cost of a (potentially) higher setup time. The main drawbacks of restriction unawareness is that it leads to an increased chance of LSP creation failures due to lack of resources due to temporary reservation of ongoing request and most of which (if not all) are going to be freed when the ongoing LSP creation process ends. Such misleading evaluations should clearly be avoided in our model, hence the resource availability awareness approach has been adopted in our implementation. Restriction awareness implements a serialized access to resources and requires every node to be able to “freeze” any incoming LSP request which tries to access locked resources. So incoming request are queued and processed in a FIFO fashion when their required resources are unlocked. Such an approach introduces the risk of deadlocks, i.e., the risk of having two or more processes blocking each other: e.g., having 2 LSP requests and the second LSP waiting on a shared node for the first to be answered, if the first LSP during the route creation needs to access another node’s resources which are already locked by the second, then the system enters a stall situation where no one of the requests can be answered. Deadlocks may involve more than two LSP requests and be very hard to be detected. The detection system itself may imply a heavy overhead over control plane.

To overcome problem with deadlocks, we will introduce a timeout for locked resources. When timeout expires, the LSP request which originally locked the resources is considered failed, so a PATH\_ERR message is sent upstream, locked resources are released and eventually the next pending LSP request can be processed. The module in charge of resources

locking is the RMM. After receiving a PATH message, it is processed and checked for resource availability and then RMM locks resources. They will be freed or reserved:

- after receiving RESV message
- after receiving an error notification (PATH\_ERR/RESV\_ERR message)
- after related timer expiring.

Locked resources are not notified to RM through resource availability status/change notification, in order to avoid having notification system engulfed by bursts of locking notification messages and using the temporarily unavailable resources in the routes evaluation. Resources locking practically translates into links locking: every link listed in the (Explicit Route Object) ERO section of a PATH message is locked as soon as it is reached by the PATH message traversing from source to destination node and unlocked only after a PATH\_ERR/RESV/RESV\_ERR message comes back (or locking timeout expires) and local resources availability are consequently updated. After releasing the resources, next LSP request is processed, as described above.

#### 2.3.1.8. Wavelength Assignment

When the PATH message arrives at the destination node, local FCM checks the resource availability and the impairment evaluation. But before the latter check, a wavelength among the feasible wavelengths has to be selected and assigned to the setting-up LSP. As discussed earlier, wavelength availability is one of the main information collected by PATH message during its traversal from source to destination node.

At every node, available wavelengths set (label-set) is updated by taking intersection of label-set and free wavelengths on outgoing link. Assuming that the label-set at destination node is not null (otherwise no LSP could be setup), a wavelength has to be chosen from available wavelength set. Many different criteria may be used, e.g., the first available wavelength in the list or the farther wavelength from those already in use (to reduce the impact over already existent LSPs), etc. We're going to implement lambda choice criterion as "the first available in the list", but in general the wavelength selection policy may affect the feasibility of the LSP itself and the usage of the link for future connections. An intelligent wavelength selection policy may lead to an optimized usage of the network resources and improved performance, so the usage of alternative lambda selection policies should be considered in future developments/implementations.

#### 2.3.2. New/ Extended Messages

Table 5 shows all the services we propose to implement in our architecture. As you can see, the services are more than those described above. This is due to the fact that some of the mechanisms described in Section. 2.3.1 require the interaction between more than two modules, hence we decided to have a specific service for every message exchange between two modules. A service could be required as part of another service: in this case we say that the first is triggered by the latter (see the "Trigger" column in Table 5). Thus, through the composition of these "atomic" services, the mechanisms we have introduced could be implemented and univocally be tracked.

**Table 5: Services and Messages**

Services & Messages					
Code	Service	Trigger (By Code)	Involved Modules	Interface (see Figure 3)	Messages [section]
1	<i>Supervising Request</i>	[User]	<b>C:</b> User <b>S:</b> TE-Agent	1.X	→ superReq ← superRsp [2.3.2.5 2.3.2.6]
2	<i>Operational Request</i>	[User]	<b>C:</b> User <b>S:</b> TE-Agent	1.X	→ operReq ← operRsp [2.3.2.5 2.3.2.6]
3	<i>Network Status Request</i>	1	<b>C:</b> TE-Agent <b>S:</b> Routing Module	5.1	→ netStatusReq ← netStatusRsp [2.3.2.5 2.3.2.8]
4	<i>Signalling Request</i>	2	<b>C:</b> TE-Agent <b>S:</b> Signalling	2.1	→ signReq ← signRsp [2.3.2.5 2.3.2.7]
5	<i>K Routes Evaluation Request</i>	2	<b>C:</b> TE-Agent <b>S:</b> Routing Module	5.1	→ routeReq ← routeRsp [2.3.2.5 2.3.2.8]
6	<i>CSPF Evaluation Request</i>	5	<b>C:</b> Routing Module <b>S:</b> CSPF	6.1	Function call [2.3.2.10]
7	<i>RAN Subscription</i>	[Init]	<b>C:</b> Routing Module <b>S:</b> Resource Manager Mod.	4.1	→ subRANReq ← subRANrsp [2.3.2.5 2.3.2.9]
8	<i>Resources Availability Change Notification</i>	(6)	<b>C:</b> Routing Module <b>S:</b> Resource Manager Mod.	4.1	← evtRANntf [2.3.2.5 2.3.2.9]
9	<i>OSPF-TE</i>	(8) + OSPF-TE protocol	<b>P2P:</b> Routing Modules	5.2	OSPF-TE msgs [2.3.2.4]
10	<i>RSVP-TE</i>	(2) + RSVP-TE protocol	<b>P2P:</b> Signalling Modules	2.2	RSVP-TE msgs [PATH =2.3.2.1 RESV =2.3.2.2 ERR =2.3.2.3]
11	<i>Impairments Feasibility Check</i>	10	<i>Local</i> <b>C:</b> Signalling Module <b>S:</b> FCM	3.2	→ impChkReq ← impChkRsp [2.3.2.11]
		11	<i>Remote</i> <b>C:</b> FCM <b>S:</b> affected LSPs FCM	3.1	→ rImpChkReq [2.3.2.14 2.3.2.15 2.3.2.16] ← rImpChkRsp [2.3.2.14]
12	<i>Q-Factor Evaluation Request</i>	11	<i>Local</i> <b>C:</b> FCM <b>S:</b> Q-Tool	7.1	→ locQFReq ← locQFRsp [2.3.2.14]
		11	<i>Remote</i> <b>C:</b> FCM <b>S:</b> Q-Tool	7.1	→ remQFReq ← remQFRsp [2.3.2.14]

13	<i>Resources Availability Check</i>	10	<b>C:</b> Signalling Module <b>S:</b> FCM	3.2	→ resChkReq ← resChkRsp [2.3.2.5 2.3.2.11]
14	<i>Resources Availability Request</i>	13	<b>C:</b> FCM <b>S:</b> Resource Manager Mod.	4.1	→ resAvailReq ← resAvailRsp [2.3.2.5 2.3.2.13]
15	<i>Command Execution Request</i>	10	<b>C:</b> Signalling <b>S:</b> Resource Manager Mod.	4.2	→ cmdExeReq → cmdExeRsp [2.3.2.5 2.3.2.12]

For all messages we define some general message templates covering the subscription, query, and notification feature. Then we explain in general how to fill and handle these messages in order for every service to work correctly. This because such protocols are not extended from standard protocols, but are introduced on ad-hoc basis for these new services, so there is more freedom for the developer in choosing how implement them. In the following subsections we discuss detailed description of all the extensions made to standard protocols used in the architecture (RSVP-TE and OSPF-TE).

### 2.3.2.1. RSVP/RSVP: PATH message modification

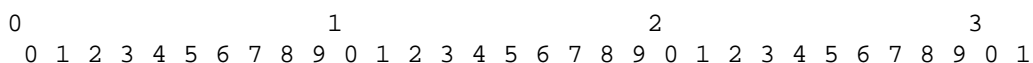
In order to carry all the impairment related information, RSVP-TE PATH message has to be extended. The information required for impairment evaluation and feasibility check is the following:

- The list of sections composing the proposed route
- For every section
  - Optical parameters
  - Active LSPs list, specifying for every LSP
    - wavelength
    - input power
    - LSP id
  - Available wavelengths list, specifying for every wavelength
    - wavelength id
    - input power (on the section)

Due to the specific usage of the collected parameters, we decided to add a new object inside the PATH available object: the Optical Path Description object (OPD object), which contains all the additional information required for impairment evaluation and feasibility check. As input power is one of the important parameter for Q-factor evaluation it is worth to be signaled.

Finally, OPD object may contain three TLVs:

- **AFFECTED\_LSPS\_TLV**  
 Contains an ordered sequence of affected LSP\_ID fields, which is later used by SECTION\_DESCRIPTION\_TLV to identify the affected LSP, using identifier as the index of the position in the sequence. Index value of 0 is considered related to the new/current LSP, so first element of the sequence has index 1.



```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |                                     |
|          Type                       |          Length                       |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
/      Lsp_Id #1                      /
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
/      Lsp_Id ...                      /
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
/      Lsp_Id #M                      /
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
  
```

Type, Length: TLV Header  
 Lsp\_Id[]: sequence of LSP\_ID (which is source node address + serial number)

- SECTION\_DESCRIPTION\_TLV

It is the TLV describing a section of the proposed route. It contains the full description of a section in terms of identifier, optical components and available wavelengths. Its structure is the following:

```

0                                     1                                     2                                     3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |                                     |
|          Type                       |          Length                       |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|  SectClass   |                               | SEQ_Number   |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
/      NodeId   /
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
/      SectInPower
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
/      Optical_Element #1
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
/      Optical_Element ...
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
/      Optical_Element #N
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
/      Channels
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
  
```

Type, Length: TLV Header  
 SectClass: IN, FIBER, OUT, TRANSPONDER, RECEIVER  
 SEQ\_Number: ID sequence number  
 NodeId: ADDR\_SUBTLV containing node id  
 SectInPower: section's default input power (which is required for Q-factor evaluation)  
 Optical\_Element[]: ordered sequence of OPTICAL\_ELEMENT\_SUBTLV  
 Channels: CHANNEL\_LIST\_SUBTLV

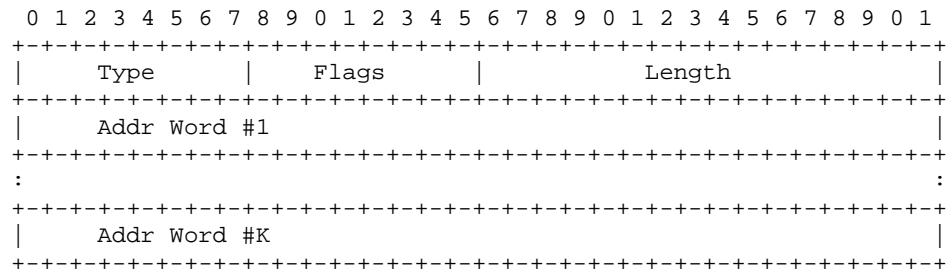
“SectInPower” parameter is the default input power value for the section, which can be used for any wavelength whose input power is not specified in Channels field. ”Optical\_Element” is an ordered sequence of optical elements along the section. This TLV requires the definition of three SUB-TLVs:

- o ADDR\_SUBTLV

It contains the ID of a node, in an IPv4 or an IPv6 address format.

```

0                                     1                                     2                                     3
  
```

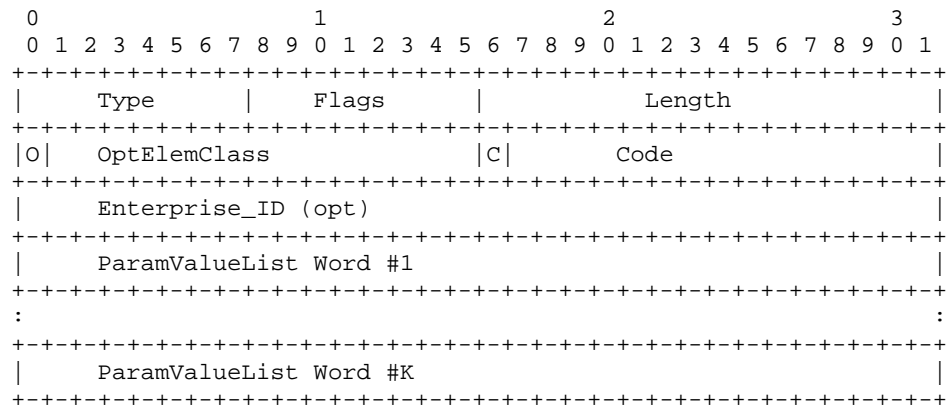


Type, Flags, Length: SUB-TLV Header  
 Addr: ID creating node

Addr, depending on the type value, may be an IPv4 address (4 bytes, K=1) or an IPv6 address (16 bytes, K=4).

o **OPTICAL\_ELEMENT**

Describes any optical component of the current section.



Type, Flags, Length: SUB-TLV Header  
 O: Custom/Standard optical element class  
 OptElemClass: DCU, VOA, FIBER, AMP  
 C: Custom/Standard code  
 Code: Code referring ElemClass Dictionary definition  
 Enterprise\_ID: ENTERPRISE ID (optional field)  
 ParamValueList: List of K word containing instance parameters for the optical element

The “o” and “c” fields refers to their following fields “OptElemClass” and “Code”: a value of 0 states that the following class/code is a STANDARD code, otherwise it is a CUSTOM code. If a code is standard, an extra field (“Enterprise\_ID”) has to be specified in order to have a univocal reference to the right component. In case of custom code, no “Enterprise\_ID” is given.

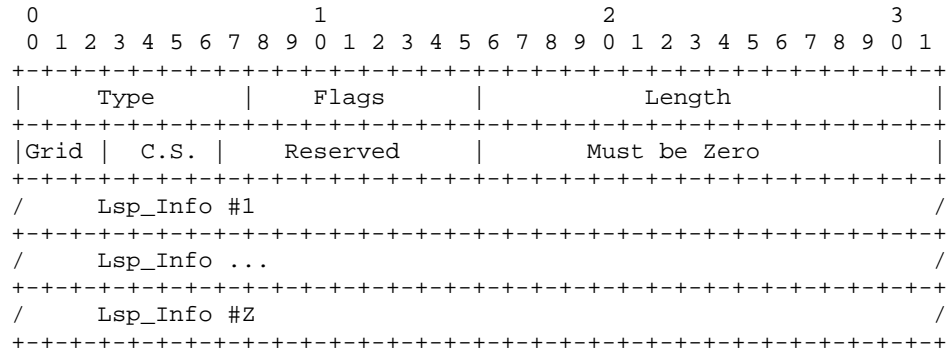
Standard classes and codes are supposed to be stored in a Dictionary (“ElemClass Dictionary”) and the generic parameters (e.g. attenuation) of referred elements are supposed to be retrievable just querying for their code. Instance parameters (e.g., length) are instead given in the “ParamValueList” section of the SUBTLV: for every standard element, it is defined a set of instance parameters which need to be specified.

Conversely, custom classes/codes may not have their description stored, so all of their parameters may have to be specified in the “ParamValueList” section. From what said above, “ParamValueList” is specific for every kind of optical element, and the kind and order of specified parameters may vary from

element to element.

o CHANNEL\_LIST

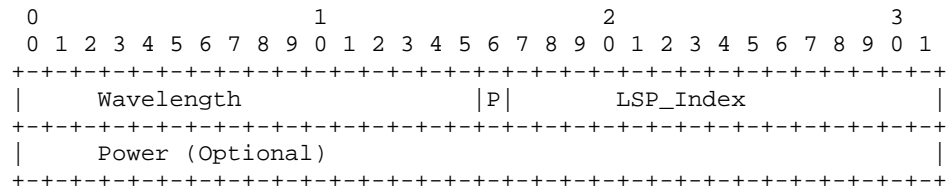
It provides a list of all the supported channels (used and available wavelengths) specifying for each of them the input power on the section (if such parameter is not specified, the default section's input power specified in the SECTION\_DESCRIPTION\_TLV has to be used).



Type, Flags, Length: SUB-TLV Header  
 Grid: ITU-T grid specification  
 C.S.: Channel spacing used in a DWDM system  
 Lsp\_Info[]: LSP\_INFO ordered sequence

“Grid” and “c.s.” are parameters defined in section 3.3 of [5].

LSP\_INFO is a data structure containing the wavelength and (optionally) the input power of a (active or new) LSP:



Wavelength: Wavelength number (193.1THz +/- w\*(channel spacing)  
 P: Input power bit mask  
 LSP\_Index: Position of LSP in AFFECTED\_LSPS TLV  
 Power: Input Power (optional)

“Wavelength” is a signed integer specifying a given wavelength in function of its “cell spacing distance” from the given pivot frequency of 193.1THz. “P” just tells the parser if “Power” field is present ( p = 1) or not (p = 0).

“LSP\_Index” is the index value of current channel with reference to AFFECTED\_LSPS\_TLV’s LSPs list given in the same OPD object. As explained earlier, index 0 is referred to available channels, so LSP are indexed in AFFECTED\_LSPS\_TLV starting from 1.

Power is an optional parameter referring to the input power of the current channel on current section. If not specified, SECTION\_DESCRIPTION\_TLV’s SectInPower parameter is used.

### 2.3.2.2. RSVP/RSVP: RESV message modification

RESV message does not require any particular modification in order to implement described behaviors: the information transported is the same as the original RESV messages.

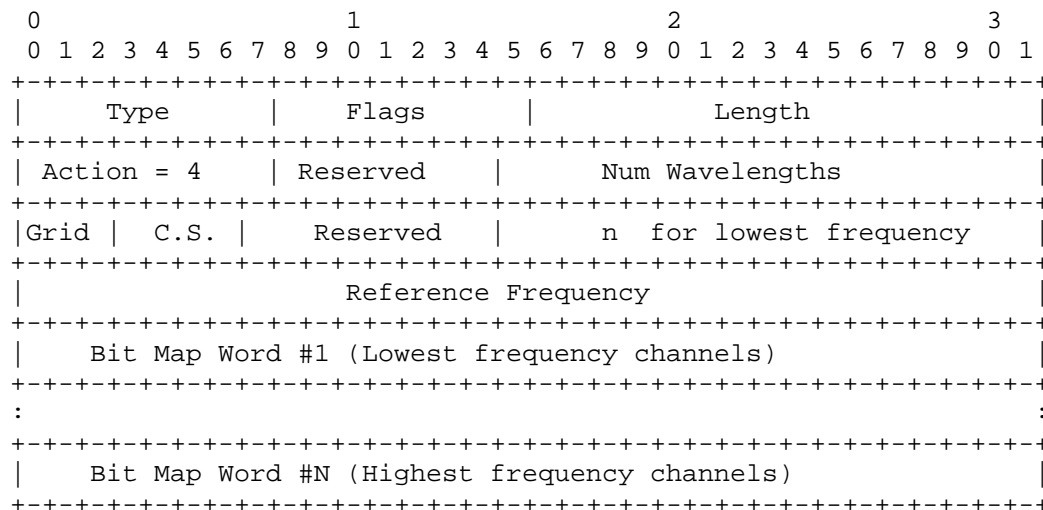
### 2.3.2.3. RSVP/RSVP: Error messages modification

Error messages do not need *per se* any change. New error classes (due to impairment, resources unavailability, etc.) can be translated as new error types, extending the set of the existing ones already defined by standard protocol. However, there are small changes in the way they are sent/handled. All errors have to be translated by SM into messages for the TE-Agent module, in order to allow it to update its information correctly and to deliver a response to users for their required actions.

### 2.3.2.4. Routing Module/Routing Module: OSPF-TE messages extension

As discussed in Section. 2.1.5, routing module implements OSPF-TE protocol. The changes we need to introduce in this protocol is just to allow Link-State Advertisement messages to carry additional information regarding resources availability at node, i.e., wavelength availability information. For our purpose the only additional resources monitored are the available wavelengths, thus the simplest solution is to add a new sub-TLV for wavelengths (defining a new sub-type for it) to the Link TLV structure described at section 2.4.2 of [4].

A valid encoding for wavelengths is the wavelength set sub-TLV depicted at section 2.4 of draft-ietf-ccamp-rwa-wson-encode: between the 5 encoding solution proposed, the one which looks more suitable for our purposes (mainly, packet's size optimization) is the one which stores the available wavelengths in a bit map (draft-ietf-ccamp-rwa-wson-encode [4], section 2.4.3 ).



Type, Flags, Length:	SUB-TLV Header
Action:	Specify encoding type (4 = bit mask)
Num Wavelengths:	Number of wavelengths represented by the bit-map
Grid:	ITU-T grid specification
C.S.:	Channel spacing used in a DWDM system
n:	Lowest frequency (bit 0, bit map word 1)
Reference Frequency	Reference frequency used for wavelength's frequency retrieval (see below)
Bit Map Word:	Bit map's entries (N = $\lceil (\text{Num Wavelengths} / 32) \rceil$ )

“n” value is a signed integer defined as that number such that, being  $w$  wavelength’s frequency value and channel spacing derived from “C.S.” and “Grid” fields,

$$w = \text{reference\_frequency} + n * (\text{channel spacing})$$

### 2.3.2.5. Subscription, Query, and Notification template messages

In Figure 4 we have described the protocol for service access and realization over client/Server interfaces. Here we give a general description of involved messages structure and provide generic structure for each class of messages. The description is not given in a detailed fashion because it refers to messages exchanges which don’t extend existent protocols, but instead implement their own. So only the main fields of messages are introduced (explicating the required parameters in order to have the service performing correctly), whilst the exact structure of the message is not given, depending on developer’s implementation choices. This structure could be modified depending on the kind of service (as described in the next sections)

The subscription message *subReq* must contain all the information to allow the client to specify the kind of service required and the server to identify and contact the client. So it could be seen as message with at least 4 fields:

- *Subscriber ID*: identification code for subscriber (client) module
- *Subscriber Address*: address which server has to send response and eventually other service related messages.
- *Subscription Request ID*: identification code for the subscription request. It is used by client to understand which service request it has received response.
- *Service Configuration Parameters*: all the needed parameters to configure the service to client needs

The related *subRsp* just replies to the above message notifying if the subscription was successful. So 2 fields are at least required

- *Subscription Request ID*: tracking of the subscription request
- *Subscription Response*: accepted/refused (eventually we can add other codes explaining the reasons for rejection)

The query for service (*serReq*) message must contain an identification for the client and its address (this field is optional if client is subscribed), the specification of the required service and a field with all the parameters needed by server in order to perform the required operation in correct and efficient manner. So following fields must be defined in a generic *serReq* message:

- *Client ID*: identification code for the client
- *Client Address*: address which server has to send response (optional if client is subscribed)
- *Service Request ID*: identification code of the current instance of service request
- *Service Type*: the kind of service required by the client
- *Service Required Parameters*: all the additional parameters required by server

The response message *serRsp* must refer to the previous *serReq* message and report if any error occurred during service evaluation and the eventual result of such service. Following fields need to be present in *serRsp*:

- *Service Request ID*: tracking of the original service request
- *Error Occurred*: a code describing eventual error occurred during evaluation, 0 if no error occurs
- *Results*: the eventual results of the service, in a format readable by client.

Finally, the notification message (*evtNtf*) must refer to the subscribed service that has generated the notification, a reference to the time of occurrence of the event, and all the additional related information required by client.

- *Subscription Request ID*: the same *subReq ID* of the related approved subscription request
- *Occurrence Time*: a timestamp or simply a sequence number allowing to locate event occurrence in time
- *Additional Information*: the additional information provided by the service, in a format readable by the client.

### 2.3.2.6. User/TE-Agent: Supervising and Operational Requests

Users access TE-Agent's services using client/server interface. The user queries TE-Agent and waits (not in a blocking way) TE-Agent response. TE-Agent's main services accessed by Users are shown in Table 6 and Table 7. These requests are from User for monitoring network status or for setting-up/releasing LSP: how to implement is left to the developer.

Table 6: User's Supervising Requests

Supervising Requests (superReq message)		
Kind	Required Parameters	TE-Agent Response
<i>Active LSPs</i>	---	Active LSPs (with their main parameters) List
<i>Network Topology</i>	---	Nodes & Links List
<i>Resources Availability</i>	---	List of Available Resources for every link on every node

Table 7: User's Operational Requests

Operational Requests (operReq message)		
Kind	Required Parameters	TE-Agent Response
<i>LSP Setup</i>	Setting-up LSP's parameters (Label Request + Sender TSpec + Explicit Route)	Success Failure (with code error)
<i>LSP Release</i>	Target LSP ID	Success Failure (with code error)

Supervising requests listed in Table 6 can be answered by TE-Agent by retrieving the information from its internal data structures (*Active LSPs*) or by querying the RM (*Network Topology* and *Resources Availability*). Depending on the size of the network, all the supervising requests may imply a response message of huge dimension: data retrieving and message creation may slow down TE-Agent's and (if involved) RM's activities. Extensions to such services will be implemented, letting User ask for a specific LSP (with TE-Agent answering with required LSP's data or notifying that specified LSP does not exist) or for

*Resources Availability* on a specific node (and TE-Agent answering with related data or with a “node not existent” notification).

Supervising requests in our design are carried by *superReq* message (see Figure 7): *superReq* is an instance of *serReq* message described earlier. In their general form *superReq* messages have *Service Required Parameters* field left void, but in case of an *Active LSPs* request or *Resources Availability*, LSP ID or the NODE ID respectively needs to be specified in the *superReq* message. Similarly, *superRsp* is an instance of *serRsp* message: depending on the kind of service required, *Results* field is filled with the information as shown in Table 7.

Operational requests are much more complex for the TE-Agent, because these are not limited to data retrieval but try to modify data layer configuration. In comparison with the supervising requests, these requests require user to provide a wide set of parameters which are used by TE-Agent to initialize and send appropriate messages through SM, to start the process (PATH/PATHTEAR messages, depending on required operation).

The two *Operational Requests* are related to LSPs creation and deletion. The third request could be regarding LSPs modification which is trickier than the previous two requests, due to the fact that only a limited set of changes are allowed to be performed *on the fly* over an active LSP without disrupting LSP itself. Thus as a trade-off solution, we consider the LSP modification as a sequence of new LSP setup and old LSP release. However, this solution may not be optimal due to the fact that the old LSP is still active during new LSP setup, reducing the available resources set to the new one and still having impact on all active and/or new LSP(s).

*Operational Request* message (*operReq*, see Figure 7) is an instance of *serReq* message, even if it uses a more articulated *Service Required Parameters* field containing request specific parameters (see Table 7).

*Operational Response* message (*operRsp*, see Figure 7) is also an instance of *serRsp* message: due to the fact it just signals the success or failure (specifying the kind of error occurred) of required operation, its *Results* field may be left void.

#### 2.3.2.7. TE-Agent/Signalling Module: Signalling Requests

The arrival of an *Operational Request* message to TE-Agent is always related to SM operation. Essentially, user may ask SM (via TE-Agent) for two possible operations: LSP setup or LSP release.

In case of *LSP Release Request*, all the required parameters have been already specified in the original *operReq* message, so the easiest solution for TE-Agent is just forward it to SM, after having checked if target LSP is indeed active.

In case of *LSP Setup Request*, the same solution can be adopted, as long as *all* parameters required to set up LSP are specified. But, typically *operReq* does not specify explicit route object (ERO) related parameters (usually, only source and destination nodes are provided, while route computation is done by RM). Hence, TE-Agent needs to retrieve ERO information querying RM (see following section), which returns a list of k candidate routes (as explained in Section. 2.3.1.6).

So, after having received such list, TE-Agent can integrate the ERO in the original incoming message with the first candidate route and forward it to SM. If the candidate route fails for some reason, the next candidate route will be used “recycling” the same request, but changing the invalid route with the next in the list, as described in 2.3.1.6. So, at the end, the *signReq* message (see Figure 7) TE-Agent sends to SM is the same (with integration of ERO) *operReq* message arrived to TE-Agent.

The signalling response message (*signRsp*) is an instance of *serRsp* message. In this message, SM can simply specify the *Service Request ID* which it has retrieved from *signReq* message and the *Error Occurred* which explains if operation was successful or not (information which could easily be retrieved from RESV or PATH\_ERR / RESV\_ERR error code). When *signRsp* arrives to TE-Agent, it is processed by TE-Agent in order to decide the next step to perform: if it is a LSP Setup failure and another candidate path is available, then TE-Agent sends the *signReq* with an updated ERO; otherwise TE-Agent updates its active LSPs database and then forwards it to User.

#### 2.3.2.8. TE-Agent/Routing Module: K Routes Evaluation and Network Status Requests

TE-Agent queries RM for two services:

- *K Routes Evaluation Request* using *routeReq* message (see Figure 7)
- *Network Status Request* (Network Topology or Resources Availability) using *netStatusReq* message (see Figure 7)

The *routeReq* message from TE-Agent can use the related *operReq* and simply forwards it to RM. Note that, all the information contained in it may not be necessary for path computation, but it is left to RM for the selection of the useful fields. The *routeRsp* is a *serRsp* message instance, having the list of the *k* candidate routes stored in its *Results* field. The routes are stored in the same format used by RSVP-TE PATH messages in order to avoid converting the route format.

Similarly, *netStatusReq* which is an instance of *serReq*, is just the original incoming *superReq* forwarded to the RM. The *netStatusRsp*, being a *serRsp* instance having all the network/resource availability information stored in the *Results* field, has the same structure of a *superRsp* message. Hence, when TE-Agent receives the message, it could be forwarded directly to User (as TE-Agent is not interested in its content).

#### 2.3.2.9. Routing Module/Resource Manager: Resources Availability Change Notification

In order to perform tasks correctly, RM needs an always updated view of the network. To achieve such result, it subscribes the *Resources Availability Notification Service* provided by RMM. Subscription to the service is performed exactly as described in the generic subscription procedure shown in Figure 5. *subRANReq* (instance of *subReq* message) is sent to RMM by RM (specifying constraints to the service using the *Service Configuration Parameters* field) and then RMM answers with *subRANRsp* message.

After subscription, whenever a change occurs in monitored resources, RMM creates and sends a *evtRANNtf* notification message (containing a reference to the affected resource, its actual value and/or other values, as specified in the subscription request) to all the subscribed modules. Sending the notification information adopting the same format used by RM in its internal structures would speed up notification processing, reducing the load on RM.

### 2.3.2.10. Routing Module/CSPF: CSPF Evaluation Request

RM queries CSPF in order to retrieve the list of  $k$ -candidate routes to set-up LSP when it receives a *routeReq* from TE-Agent. As described in Section 2.1.6, CSPF module is integrated with Rm, sharing the graph representation of the network. This is because, keeping CSPF separate from RM would lead to the need for replication of the network status databases which implies a useless redundancy and the risk of misalignment between RM and CSPF modules network representation. Hence, in our implementation CSPF could be seen as an internal function of RM and the query to CSPF could be translated into a function call for the computation of the  $k$ -paths applying CSPF filters. The parameters needed are the source, destination, and the new LSPs configuration parameters, then is left to CSPF to prune and compute the routes. The returned list of routes is then reordered following optimization rules described in Section. 2.3.1.6, then inserted in the *Results* field of the *routeRsp* message to send to TE-Agent.

### 2.3.2.11. Signalling Module /FCM: Resources Availability & Impairments Feasibility Check

As discussed earlier, SM is a client for the following two services provided by feasibility check module (FCM):

- *Resources Availability Check Request* using *resChkReq* message (see Figure 5), for checking the actual available local resources
- *Impairments Check Request* using *impChkReq* message (Figure 6), for checking the impact of impairment over new and active LSPs

*resChkReq* message is a *serReq* message which needs to contain all the common available resources found by PATH message (such information could be stored in the *Service Required Parameters* field). Similarly, the related response message, *resCheckRsp* (an instance of *serRsp* message), need not only to transport request's reference and response (feasible or not), but the list of all the available resources common to local available resources and to the list of resources given in the *Service Required Parameters* field of the request. This information is later used to update the information stored in the PATH message (or to decide which resources to be used, if PATH message has reached the final node).

*impChkReq* is a more complex *serReq* message (it is sent only by destination node's SM): it contains all the information related to new and affected active LSPs and is sent to FCM. Such information is already available in the OPD object in the PATH message, so the simplest solution is to include such object in *impChkReq*'s *Service Required Parameters* field and to let the FCM to decide which parameters to use. The related *impChkRsp* message is a *serReq* message containing in the Results field the local feasibility response (i.e., the feasibility response obtained comparing Q-factor value with local threshold) and the Q-factor value itself.

### 2.3.2.12. Signal/Resource Manager: Command Execution Request

SM queries resource manager module (RMM) to have some physical operation to be performed on available wavelength at OXC level:

- *Perform Cross Connection*, which is usually requested during processing of a RESV message to cross-connect the input and output ports of given wavelength
- *Release Wavelength*, which is usually requested during processing of a PATHTEAR message to release the requested wavelength

The *cmdExeReq* message is a *serReq* message and contains the wavelength on which the operation is requested in *Service Required Parameters*. The response message, *cmdExeRsp*, just contains operation's response, leaving *Results* field void.

#### 2.3.2.13. FCM/Resource Manager: Resource Availability Request

When *resChkReq* message arrives to FCM, it triggers *resAvailReq* from FCM to RMM. The purpose of such a request is to retrieve all the available resources at node, in order to allow FCM to perform a cross check between resources available locally and common resources available on all previous nodes along the route (collected by *PATH* message). *resAvailReq* is a *serReq* message which does not require any additional request parameters to be specified, so *Service Required Parameters* field can be left void. The related response is *resAvailRsp* message, an instance of *serRsp*, having all the available resources at node, stored in the *Results* field.

#### 2.3.2.14. FCM/Q-Tool: Remote and Local Q-Factor Evaluation Request

FCM queries the Q-Tool module when interested in an estimation of the impact of the impairments on new and all affected active LSPs. Such request may trigger two different kind of messages, an *impChkReq* (if FCM is on the final node of new LSP) or a *rImpChkReq* (if FCM is on the final node of an affected LSP). As we have seen, *impChkReq* and *rImpChkReq* have the same structure and content, in particular both messages contain entire OPD object carried in *PATH* message and is stored in *Service Required Parameter*.

Q-Tool has a well defined interface to other modules, so FCM have to implement client side of such interface and follow the appropriate protocol to deliver its request to Q-Tool. Details regarding Q-Tool interface, protocol and messages are given in Q-Tool documentation in D3.2. Clearly, OPDs information has to be extracted and rearranged in order to satisfy Q-Tool format requirements. Queries to Q-Tool have been distinct between local and remote queries. The local request (*locQFReq*) contains all the information about new and affected LSPs, whereas the remote request (*remQFReq*) requires only the new LSP information, being the residual information known by previous *locQFReq* (and *remQFReq*) computations.

The Q-Tool responses (both *locQFRsp* and *remQFRsp*) are formatted in the same way and carries the Q-factor value. In case of *rImpChkReq*, a *rImpChkRsp* message containing Q-factor is sent back to the querying FCM. Here, it is compared with the given threshold, obtaining a partial feasibility response. The final feasibility response is calculated by evaluating the logical conjunction of all partial feasibility responses. After having obtained the final feasibility response, an *impChkRsp* message is created with the response value inserted in its *Results* field and sent back to SM.

#### 2.3.2.15. FCM/FCM: Protocol definition

Communication between two different FCMs is a inter-node communication, so a well defined protocol is needed in order implement it properly. For the sake of simplicity and commodity, SOAP has been chosen to such a purpose.

### 2.3.2.16. FCM/FCM: Remote Impairments Feasibility Check

As shown in Table. 4, Impairments Feasibility Check is split in two different procedures: one, called *local*, which involves local Signalling Module and FCM and has been described at section Section. 2.3.2.11, the other, called *remote*, which is triggered by the above local procedure and involves the local FCM and all the other FCM running on destination nodes for LSPs affected by the setting-up one.

Thus such remote Impairments Feasibility Check request is sent after *impChkReq* arrival by the local FCM to other FCMs, then after some time a response will be sent back by every queried FCM to the local FCM, which performs its final evaluation about LSP's feasibility. The content of the *rImpChkReq* message is similar to the one carried by *impChkReq*, but for message's size sake, just only the information regarding the setting-up LSP is sent, because all the information about other existent LSPs is supposed to be already known by destination FCMs. In fact, every FCM receiving a *rImpChkReq* is supposed to be a destination node for an already existing LSP, so by construction at least it has received at the beginning an *impChkReq* message, carrying all the information about then setting-up LSP and then affected LSPs. After then LSP setup, information regarding eventual new link-sharing LSPs is brought with the arrival of *rImpChkReq* messages, keeping the FCM knowledge of affected LSPs updated. Is also left to FCM fill the *remQFReq* in a proper way (essentially, rebuilding from its stored information an appropriate OPD object), allowing Q-Tool to perform correctly its Q-factor evaluation. Having chosen to implement it in SOAP, a description of such service is given in WSDL 1.1 format in Appendix-A.

### 2.3.3. New/Extended Objects and Data Structures

In order to support all the above described services, already existing data structures may need to be extended and new ones need to be added. In the following table (Table 8) a general description of the data required by every service is given

Table 8: Services Data Requirements

Services Data Requirements		
Service	Involved Modules	Data to be stored locally
<i>Supervising Request</i>	User, TE-Agent	TE-Agent needed data: <ul style="list-style-type: none"> <li>LSP's source and destination (eventually full route)</li> <li>LSP related data (LSP's wavelength, LSP's creation parameters as specified by user, LSP's status)</li> </ul>
<i>Operational Request</i>	User, TE-Agent	TE-Agent needed data: <ul style="list-style-type: none"> <li>list of current pending requests</li> </ul>
<i>Network Status Request</i>	TE-Agent, Routing Module	TE-Agent needed data: <ul style="list-style-type: none"> <li>list of current pending requests</li> </ul> Routing Module needed data: <ul style="list-style-type: none"> <li>information about resources availability and network topology (extension to OSPF-TE db)</li> </ul>

<i>Signalling Request</i>	TE-Agent, Signalling	TE-Agent needed data: <ul style="list-style-type: none"> <li>list of current pending requests</li> </ul>
<i>K Routes Evaluation Request</i>	TE-Agent, Routing Module	TE-Agent needed data: <ul style="list-style-type: none"> <li>list of current pending requests</li> <li>list of all not yet probed candidate routes for every LSP request</li> </ul>
<i>CSPF Evaluation Request</i>	Routing Module, CSPF	Routing module and CSPF needed data: <ul style="list-style-type: none"> <li>all information required by CSPF computation (extension to OSPF-TE db)</li> </ul>
<i>RAN Subscription</i>	Routing Module, Resource Manager Mod.	Resource Manager Mod needed data: <ul style="list-style-type: none"> <li>list of service's subscribers</li> </ul>
<i>Resources Availability Change Notification</i>	Routing Module, Resource Manager Mod.	Resource Manager Mod needed data: <ul style="list-style-type: none"> <li>list of service's subscribers</li> <li>actual status/value of every monitored resource</li> </ul>
<i>OSPF-TE</i>	Routing Modules	Routing module needed data: <ul style="list-style-type: none"> <li>all information required by routing computation (extension to OSPF-TE db)</li> </ul>
<i>RSVP-TE</i>	Signalling Modules	---
<i>Impairments Feasibility Check</i>	<i>Local</i> Signalling Module, FCM	Signalling needed data: <ul style="list-style-type: none"> <li>list of current pending requests</li> <li>All information regarding setting-up LSP required for Q-factor evaluation</li> <li>All information regarding LSPs affected by the setting-up LSP, required for Q-factor evaluation</li> </ul> Every FCM needed data: <ul style="list-style-type: none"> <li>All information regarding other LSPs having in this node their destination required for Q-factor evaluation</li> <li>All information regarding LSPs affected by previous setting-up LSPs, required for Q-factor evaluation</li> </ul>
	<i>Remote</i> FCM, affected LSPs FCM	
<i>Q-Factor Evaluation Request</i>	<i>Local</i> FCM, Q-Tool	Every FCM needed data: <ul style="list-style-type: none"> <li>list of current pending requests</li> <li>All information regarding setting-up LSP required for Q-factor evaluation</li> <li>All information regarding LSPs affected by the setting-up LSP, required for Q-factor evaluation</li> <li>All information regarding other LSPs having in this node their destination required for Q-factor evaluation</li> <li>All information regarding LSPs affected by previous setting-up LSPs, required for Q-factor evaluation</li> </ul>
	<i>Remote</i> FCM, Q-Tool	
<i>Resources Availability Check</i>	Signalling Module, FCM	Signalling needed data: <ul style="list-style-type: none"> <li>list of current pending requests</li> </ul>
<i>Resources Availability Request</i>	FCM, Resource Manager Mod.	FCM needed data: <ul style="list-style-type: none"> <li>list of current pending requests</li> </ul>

<i>Command Execution Request</i>	Signalling, Resource Manager Mod.	Signalling needed data: <ul style="list-style-type: none"> <li>• list of current pending requests</li> </ul>
----------------------------------	-----------------------------------	--

The “list of current pending requests” (a list of all the not yet answered requests which appears in almost every service) is required in order to match correctly the response to its request due to the fact that some services may be called more than one time in a short time interval, responses could arrive differently than forecasted.

The TED/PPD should be extended including information about resource availability (i.e., available wavelengths). The way such data is stored in module is not relevant for this document’s purposes, depending on the original data structure implementing them and so left to developer’s choice.

### 3. PCE-based Architecture: High Level System Design

This section addresses the control plane software design deploying a centralized PCE approach. It defines modules and describes its functionality including the interfaces between modules. The protocol extensions for OSPF-TE to support physical layer impairment information flooding are at the end of this section. The standard RSCP-TE is deployed to establish, maintain, and tear down connections.

#### 3.1. Modules description

This subsection depicts the functional blocks of control plane as shown in Figure 8. It is a PCE based centralised approach in which the PCE receives the path calculation request and it is forwarded to the NPOT which is responsible for the actual path computation and failure handling. In this model, OSPF-TE is extended to disseminate the PLI information through the network. The PCE communication protocol (PCEP) is a protocol used to communicate between PCE and path computation client (PCC) and between PCEs for inter-domain path calculation. If there is an available path, the information will be transferred back to OCC via the PCEP, and then the standard RSVP-TE messages will be triggered to setup the lightpath. If the path reservation fails or NPOT is unable to find an available path, the NMS will be notified accordingly. The detailed module definition is addressed as follows.

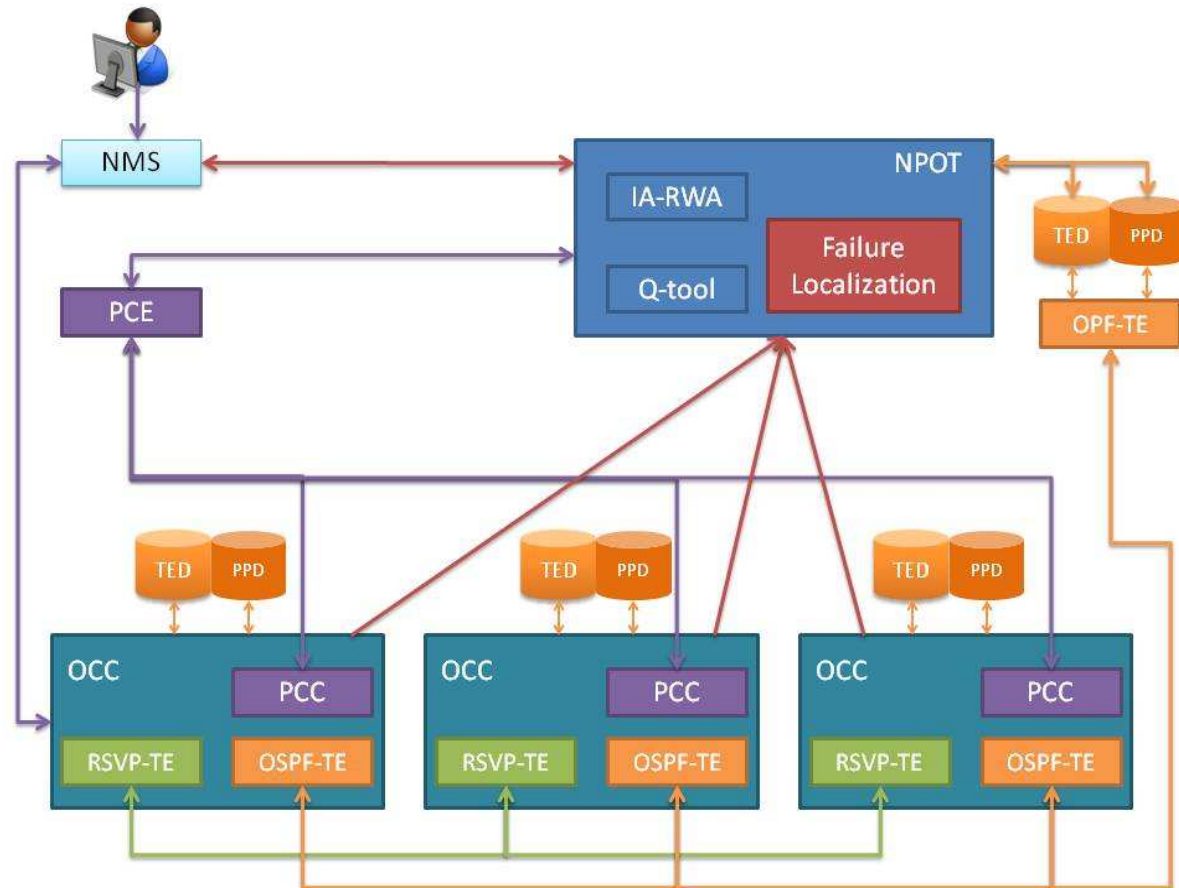


Figure 8: Control Plane Function Blocks

### 3.1.1. NPOT

Network planning and operation tool (NPOT) is a key concept and innovation of DICONET. It acts as an agent orchestrating different modules to perform the routing and wavelength assignment (RWA) with real-time physical layer impairment information taking into account responses for failure localization. From a structural perspective, the NPOT also contains three major independent functions, *i.e.* IA-RWA, Q-Tool and failure localization. Apart from these, the NPOT also provides APIs to communicate with other modules, *i.e.* NMS, PCE, OCC, and TED/PPD.

#### 3.1.1.1. IA-RWA

In order to avoid O/E/O conversion in the intermediate nodes of a WDM network when transferring data from one edge node to another, an optical connection needs to be established in advance. This can be achieved by locating a path in the network between the two edge nodes, and allocating a free wavelength on all of the links on the path. With lack of wavelength conversion, a common wavelength must be assigned throughout the whole lightpath traversing each link. This restriction is known as the wavelength continuity constraint.

Being a fundamental entity of WDM networks, lightpath has to be employed efficiently to assure an economic optical network operation cost. It is thus important to design efficient algorithms to select the routes for the connection requests and to assign wavelengths on each of the links along these routes, among the possible choices, in order to have ideal wavelength

utilization, and other performance metrics. This is known as the RWA problem. Providing a good solution to the RWA problem is important to enable more customers (connections) to be accommodated by a given network, and fewer customers need to be rejected during periods of congestion.

The DICONET IA-RWA module implements algorithms considering PLI information in order to deliver more feasible routing and wavelength results. The objective of IA-RWA is to assign routes and wavelengths to the requested connections in order to satisfy impairment constraints, while minimizing the number of different wavelengths used. By incorporating physical impairments in the routing algorithms, the IA-RWA offers cross-layer optimization to both network and physical layer. This improves the signal quality of each lightpath, decreases the physical-layer blocking probability and increases the probability of satisfying future connection requests.

#### **3.1.1.2. Q-Tool**

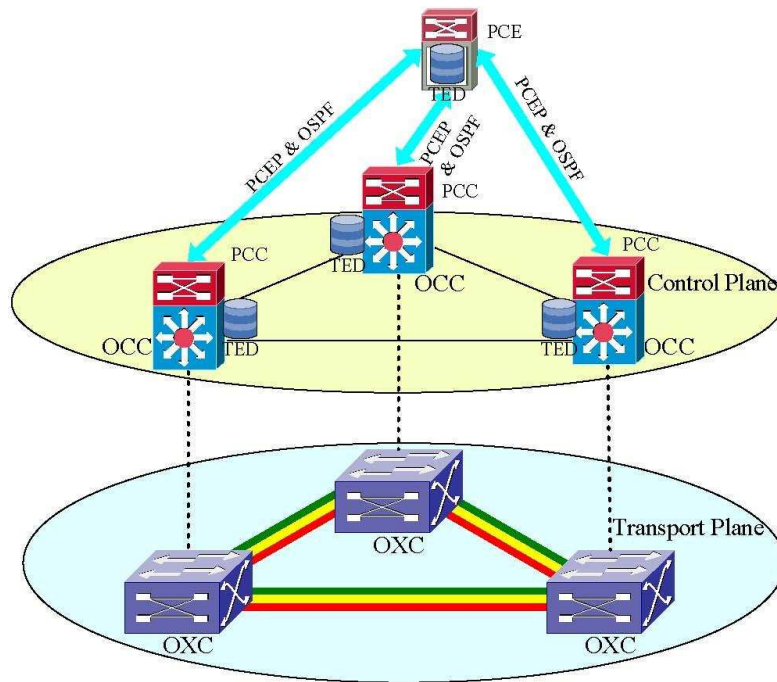
Q-Tool is a Q-factor estimation tool developed in DICONET. It takes the result of a certain IA-RWA process as the input and uses the embodied physical layer models to estimate the signal quality and validate or reject a connection if no fulfil the physical layer constraints. It gives an estimation of the impact that a set of partially or fully overlapping LSPs have on a chosen LSP. The lower is Q-factor value, the higher is the disrupting effect of overlapping LSPs over the monitored one. The Q-Tool implements analytical models for each of the dominant physical impairments and integrates these physical layer models into a single figure of merit. After evaluating the dominant effects that affect the quality of transmission (QoT) merely accepts or rejects the setup of a given lightpath. If a lightpath fails to achieve the minimum level of acceptable quality then the IA-RWA process will have to take over again and calculate an alternative route. It is worth pointing out that the Q-Tool is better to be used as less as possible, in order to reduce computation and network overhead. The detailed Q-tool usage coupled with inputs/outputs parameters description is presented in D3.2.

#### **3.1.1.3. Failure Localization**

Fault detection and localization are critical issues; they are essential for providing continuous and reliable services in transparent optical networks. Failure localization is one of the NPOT modules acting after the failure detection. It performs the analysis based on the up-to-date TED/PPD information to find out the source of failure. Then the NPOT will be responsible for the following failure handling. When OCC forwards the failure report generating from the monitors to NPOT, the TED/PPD are updated at the same time. The NPOT retrieves detailed information from the database and invokes failure localization module to perform the analysis. Final failure localization algorithm will be reported in D4.4.

### **3.1.2. PCE**

A PCE is a path computation element (*e.g.* server) that specialises in complex path computation on behalf of its PCC. The PCE can be located in a Label Switching Router (LSR), Network Management System (NMS), or a dedicated server. The path computation client (PCC) can be located at any of the Optical Connection Controllers (OCC), LSRs or the NMS. A generic PCE based architecture combined with GMPLS is shown in Figure 9.



**Figure 9: Combined GMPLS/General PCE Architecture without Considering the PLI**

A PCC sends requests to the PCE to compute a path from A to B with TE constraints, *i.e.* bandwidth requirement, cost limits, etc. The PCEP protocol is used with this purpose. Typically, the PCE computes the path based on the network state information and responds with a path including some details about the computed path such as cost or bandwidth; In DICONET, the NPOT module is the responsible for the path computation, so PCE forwards the requests received from the PCC to the NPOT waiting for a route or a blocked connection response.

### 3.1.3. OCC

The Optical connection controller (OCC) is mainly responsible for establishing connections across a domain using standard RSVP-TE after receiving a calculated path from NPOT. It also controls the extended OSPF-TE to flood PLI information across the network. When receiving connection request from the NMS, the OCC will order PCC to send this request to NPOT via PCE.

#### 3.1.3.1. PCC

Each OCC has one PCC to request a path computation from PCE using the PCEP protocol. The request includes information such as source and destination addresses bandwidth, or type and value of constraints [5]. The request can be an LSR requesting a path for a LSP for which it is the head-end, or a PCE requesting a path computation of another PCE for inter-PCE communication. However the request is limited by the PCE supported objective functions. Thus the PCC has to be able to discover the set of objective functions supported by the PCE [6].

A PCC is required to be aware of the location of one or more PCEs in its domain so that PCC can select the appropriate PCE based on its capabilities and perform efficient request load balancing. It does not have to be aware PCEs in other domains for the inter-domain path

computation; this is left for the specific PCE which is capable for inter-domain path computation and is out of scope of DICONET.

### 3.1.3.2. Routing Module (OSPF-TE)

The Routing Module in DICONET is in charge of the distribution of network state information throughout the network. In order to perform a reliable and accurate path computation, it is required an updated knowledge of the overall network. Depending on the chosen metric, some information regarding node's configuration may be required (*e.g.*, lambda's or band availability) by evaluation process, so such information must be gathered by the Routing Module and efficiently updated accordingly to changes in the resources availability. Every resource availability changes have to be notified to the Routing Module and distribute in the network as soon as it occurs, in order to grant an efficient and updated functioning of the routing algorithm.

For the sake of simplicity in this document the Routing Module is supposed to implement OSPF-TE (Open Shortest Path First) protocol, a link-state routing protocol used to distribute Links State information. This protocol is extended as described in Section 2.3 to flood also PLI information

### 3.1.3.3. Signaling Module (RSVP-TE)

In our GMPLS architecture design, the signalling module implements the signalling protocol, which is the process in charge of exchanging messages within the control plane to set-up, maintain, modify and terminate data-paths (LSPs) in the data plane. Consequently, its messages carry all the information needed for LSP's creation, maintenance and deletion. Specific objects and data structures are required by the Signalling Module to forward connection information and LSPs management.

In this implementation model, the Signalling Module implements the RSVP-TE protocol, first proposed for MPLS, and then GMPLS, derived from RSVP (Resource reSerVation Protocol). This transport layer protocol is designed to reserve resources across the network upon connection requests, using Traffic Engineering extensions. An RSVP module is running on every node in the network.

### 3.1.4. TED/PPD

Path computation requires knowledge of the available network resources such as network topology, nodes and links status, connectivity, available bandwidth, link costs, etc. All this information is stored in the TED, which in our model is built from information distributed by the routing protocol (OSPF-TE). DICONET also implements the PPD, a database that stores PLI information, also distributed by OSPF-TE thanks to our proposed extensions. These two databases are located locally at each node and centralized in the NPOT location, to be used by the path computation entity, in this case, the IA-RWA module.

## 3.2. Interfaces description

### 3.2.1. User to NMS requests and response

A user launches NMS software to request to establish a lightpath between certain nodes. This request includes not only a pair of source and destination, but also the number of lightpaths to be established and associated lightpath protection methods, *i.e.* 1+1 protected or restorable. In case the requested connection establishment fails, the NMS software GUI will update the display to inform the user.

### 3.2.2. NMS to OCC request

The NMS gathers the user requirements and initiates the first step of the lightpath establishment by sending the soft-permanent request to the OCC at the source end via the NMI-A interface. The OCC can be a computer or any network device which runs the GMPLS protocol stack. This request must include all the required information that the OCC needs to trigger the path establishment such as source and destination address, bandwidth or other constraints. This interface is not standard and can be implemented using any proprietary protocol. When the path has been established or blocked, this interface is also used to escalate the response back to the NMS.

### 3.2.3. OCC (PCC) to PCE request and response

After receiving a lightpath request from the NMS, the OCC commands the PCC to contact the PCE for requesting the path computation via PCEP. The computation result is transferred back to PCC from PCE. The result can be either positive for successfully finding a qualified path, or negative for the computation failure. When sending the lightpath calculation response back to the PCC, the PCE will use the dedicated PCC IP address and TCP port number to locate the destination. Apart from the path computation request and response, PCEP is also used to maintain a PCEP session.

### 3.2.4. PCE to NPOT request and response

The PCE is not an ordinary path computation element in this scenario. It is actually a PCE agent to forward the actual path computation request to the NPOT. The reason to keep PCE is to retain the compatibility and scalability. This interface includes a pair source-destination and constraints to be used by the NPOT to calculate an appropriate path. Then, the NPOT sends back an ACK response with the computed path or a NACK informing about a failure in the computation.

### 3.2.5. NPOT to TED/PPD request and response

After receiving the request from PCE, the NPOT needs to get the latest network status (both the network and the physical layer) from the TED/PPD in order to precisely compute the lightpath against the user's request and the network and PLI information.

Alternatively, if the NPOT is informed about the optical component failure, it will invoke the failure localization module to locate the failure source. After the failure source is found, the NPOT will update the TED/PPD accordingly using this interface.

### 3.2.6. OCC (OSPF-TE) to TED/PPD

No matter whether the requested lightpath is established or failed due to impairments, the OCC will update the TED/PPD connected. In detail, if establishment succeeds, the OCC should store the connection inventory into the database and use OSPF-TE to distribute the information; if it fails, the latest impairment from the monitor should be stored in the database. After OSPF-TE floods the updated information, all distributed and centralized TED/PPD are updated accordingly and synchronised with the rest of the databases. It is important to have a consistent network status for all the global databases, *i.e.* TED. The local database, *i.e.* PPD, does not have to be synchronised so as to reduce the network overhead. It is worth mentioning that not all the PPD is local database, the one connected to NPOT has an entire view of the network whole PLI information.

### 3.2.7. OCC to NPOT/Failure Localization Module notification

When the OCC is informed about the failure of optical nodes via the monitor, the OCC will send an alarm to NPOT/Failure Localization using this interface. Because it is possible that there are more than one OCC sending the alarm, the NPOT FL module will analyse alarm information and find out the original failure source. After NPOT finishes the FL function, it should update its TED/PPD. Then the NPOT requests the OCC in the source end of the lightpath to restore to start the re-routing and provides a new route for the requested connection.

### 3.2.8. NPOT to NMS

After failure localization process, the NPOT informs the NMS about the failure and other accordance information. Then the user can identify the failure from the software GUI.

## 3.3. Protocol extensions --- OSPF-TE extensions

As addressed in [7], the RWA problem can be computationally intensive. A dedicated centralised PCE is a feasible and economic way to perform the RWA computing, being one of the main motivations for the PCE architecture. This document will concentrate on defining the extensions to the PCE based architecture to support dissemination of physical layer impairments (PLI) as the main objective of the DICONET project. Various GMPLS protocols are extended to include the PLI information. Several models have been proposed: a signalling based architecture (extended RSVP-TE and standard OSPF-TE), routing based architecture (extended OSPF-TE and standard RSVP-TE), hybrid based (extended OSPF-TE and extended RSVP-TE), and the PCE based. The comparison of the models has been given in [8, page 66]. Particularly, the PCE architecture can be implemented by two approaches depending on the method for disseminating the PLI information, *i.e.* the PCEP extension and the OSPF-TE extension. Both approaches use the PPD to store all the PLI information. The PCEP extension approach uses the standard OSPF-TE to flood the TED information, the standard RSVP-TE to signal the lightpath setup and the extended PCEP to update the PPD information. The OSPF-TE approach uses the standard RSVP-TE to signal the lightpath setup, the standard PCEP to the PCE-PCC and PCE-PCE communication and the extended OSPF-TE to flood the PPD information between nodes. The latter approach is the focus of UESSEX, while the former was investigated in WP2 by IBBT.

As shown in Figure 10, additional database PPD is added to the PCE architecture compared with Figure 9. PLI information such as bit error rate (BER), channel power, or polarisation

mode dispersion (PMD) gathered from the optical performance monitors (OPM) and optical impairment monitors (OIM) [9] is stored in the PPD. The PCE utilises the information from both PPD and TED and deploys IA-RWA to compute a path based on PCCs request. However, the IA-RWA cannot directly access the PPD or the TED; a proper interface to these two databases is provided through the extended OSPF-TE component which is addressed in the following subsections.

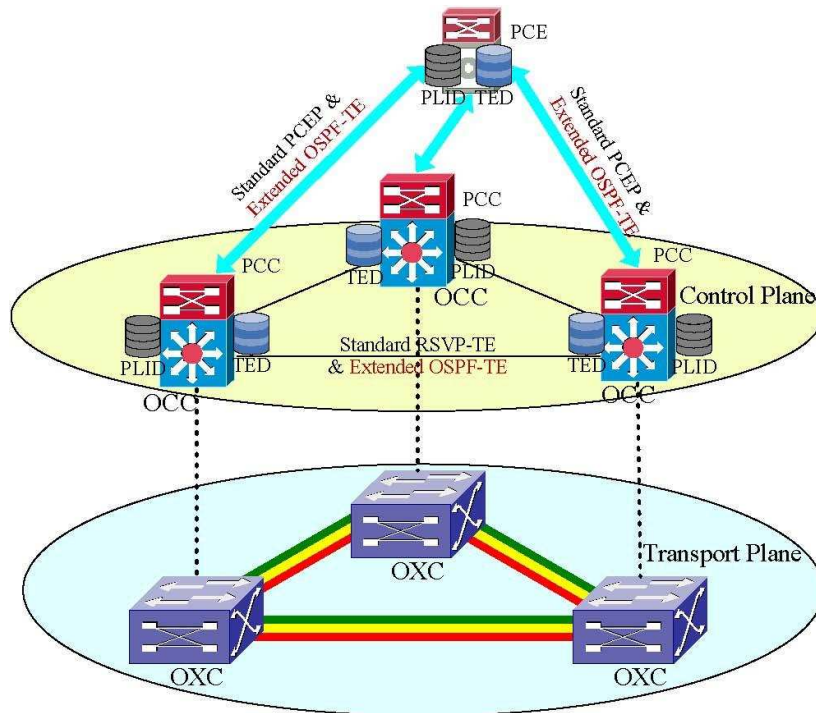


Figure 10: Combined GMPLS/PCE Architecture with Considering PLI Information

Path computation for light-paths has multiple constraints:

1. Routing constraints: Reachability, cost, path diversity, protection level, etc.
2. RWA constraints: Wavelength continuity, cross-connect capabilities.
3. Optical impairment constraints: Power, OSNR, PMD, and many others [8, page 25]:

The conventional routing and RWA constraints are stored in the TED while the new defined optical impairment constraints are stored in the additional PPD. The OSPF-TE is extended to retrieve impairment information from both TED and PPD and also to synchronise databases. The OSPF-TE extension must be able to distinguish the originating data-base of impairments.

The following subsections define the OSPF-TE extensions to distribute PLI information based on the work which has been presented in [10] and describe the corresponding extension usage occurring in the PPD, PCE, and PCC.

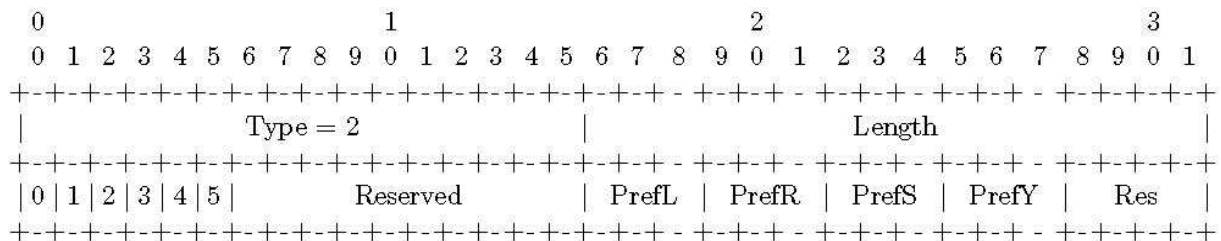
### 3.3.1.1. Considerations

This work proposes the OSPF-TE extensions to distribute PPD information using opaque link state advertisements (LSAs) [15]. This is fulfilled by simply adding more attributes to links in OSPF LSAs. However, there are three aspects that need to be considered when defining the extensions as follows:

1. The PCE path computation scope, *i.e.* intra-area, inter-area, inter anonymous system (AS), or inter-layer.
2. The PCE capacity, say, supported PLI in different PCEs, different scope, and the prioritisation.
3. The communication traffic amount.

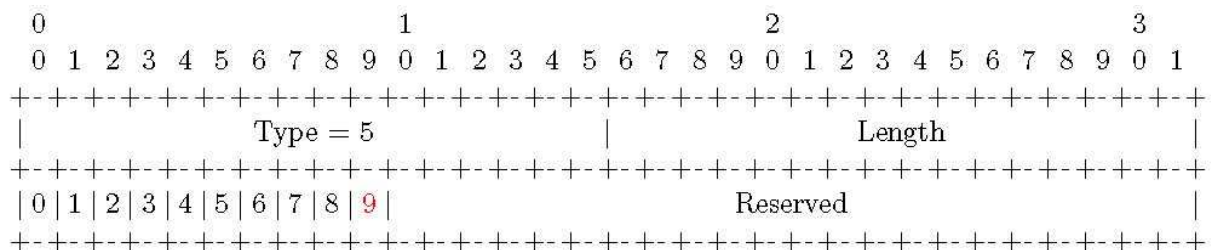
### 3.3.1.2. Update to the RI LSA

JL. Le Roux et al. proposed the OSPF extension for PCE discovery in [11], where the PCE path computation scope and capacity issue are addressed. When applying the new OSPF extension proposed in this document, the PCE discovery (PCED) TLV within the OSPF routing information (RI) LSA needs to be updated to reflect the latest supported constraints. In the PCED TLV, the path-scope sub-TLV indicates the PCE path computation scope. This represents the PCE’s ability to compute or take part in the computation of paths or intra-area, inter-area, inter-AS, or inter-layer TE label switching paths LSP. The path-scope sub-TLV format is shown in Figure 11:



**Figure 11: Path-Scope Sub-TLV Format**

The bit 0 – bit 5 indicates the PCE path computation scope, the PrefL field to Res field indicates the PCE path computation scope preference (refer to [11] for details). In DICONET, new PLI constraints have different functional scopes which are addressed in [9, page 54]. In turn, if the PCEs are being configured to update the path computation scope, the path-scope sub-TLV needs to be updated to reflect the changes. To advertise the new PLI processing capacity, the PCE-CAP-FLAGS sub-TLV is the most appropriate candidate. Its format is shown below the new bit 9 used to advertise the new PLI processing capability as shown in Figure 12.



**Figure 12: New Bit in the PCE-CAP-FLAGS sub-TLV**

### 3.3.1.3. LSA Flooding Scope

In OSPFv2, the flooding scope is controlled by the opaque LSA type [15] and in OSPFv3, by

the S1/S2 bits [12]. If the flooding scope is area local, then the PCED TLV must be carried within an OSPFv2 type 10 RI LSA or an OSPFv3 RI LSA with the S1 bit set and the S2 bit clear. If the flooding scope is the entire IGP domain, then the PCED TLV MUST be carried within an OSPFv2 type 11 RI LSA or OSPFv3 RI LSA with the S1 bit clear and the S2 bit set. When only the L bit of the PATH-SCOPE sub-TLV is set, the flooding scope must be area local. A router not supporting the PCED TLV will just silently ignore the TLV as specified in [14].

#### 3.3.1.4. Motivation for a New LSA

In this work, the proposed OSPF extension is based on the work presented in [10], but with modification. Before address the modification, it is necessary to present the approach proposed by Y. Ye et al. in [10].

In [10], the extension is based on [13] which defined the OSPF-TE LSA. So it only uses type 10 LSA which has an area flooding scope. However, as mentioned above, not all the PLI information has the same flooding scope, for example, the PMD is not suitable for relatively small area network, *i.e.* metro network, it only makes sense in regional or long haul network [9, page 55]. From the networking point of view, taking the flooding scope into account when distributing PLI information will reduce the generated traffic. From the network resource provider point of view, it gives the flexibility and capability to the choice of flooding scope, in order to apply local policy. From the software implementation point of view, it reduces the need of big memory to buffer more PLI information. For the same purpose, a different mechanism is proposed in [10]. It uses hierarchical PLI, *i.e.* link, waveband, and wavelength PLI. A combinational mechanism is proposed in this document.

A RI LSA like LSA needs to be defined. The flooding scope for a RI LSA is determined by the LSA type. For OSPFv2, type 9 (link-scoped), type 10 (area-scoped), or a type 11 (AS-scoped) opaque LSA may be flooded. For OSPFv3, the S1 and S2 bits in the LSA type determine the flooding scope. If AS-wide flooding scope is chosen, the originating router should also advertise area-scoped LSA(s) into any attached Not-So-Stubby Area (NSSA) area(s). An OSPF router may advertise different capabilities when both NSSA area scoped LSA(s) and an AS-scoped LSA are advertised. This allows functional capabilities to be limited in scope. For example, a router may be an area border router but only support PLI in a subset of its attached areas. The reason behind not using just RI LSA is that the informational capabilities advertised have no impact on the OSPF protocol's operation – they are advertised purely for informational purposes [14]. Meanwhile, since the TE LSA uses the fixed type 10 LSA, a new LSA which supports different flooding scopes for PLI information has to be defined.

#### 3.3.1.5. New PLI LSA

The new LSA is defined as PLI LSA. The OSPFv2 PLI LSA is an opaque LSA of type 9, 10, or 11 as defined in [15]. The PLI LSA ID is defined as having one octet of type data and 3 octets of type-specific data. The PLI LSA uses type 5. The remaining 3 octets are the instance field, as shown in Figure 13: The instance field is an arbitrary value used to maintain multiple PLI LSAs. A maximum of 16777216 PLI LSA may support in a single system.

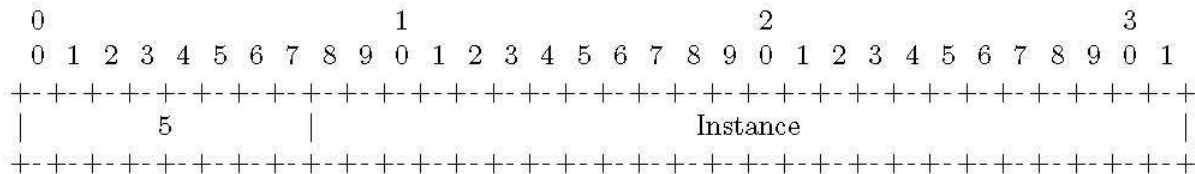


Figure 13: Proposed PLI LSA ID

The PLI LSA starts with the standard opaque LSA header as shown in Figure 14:

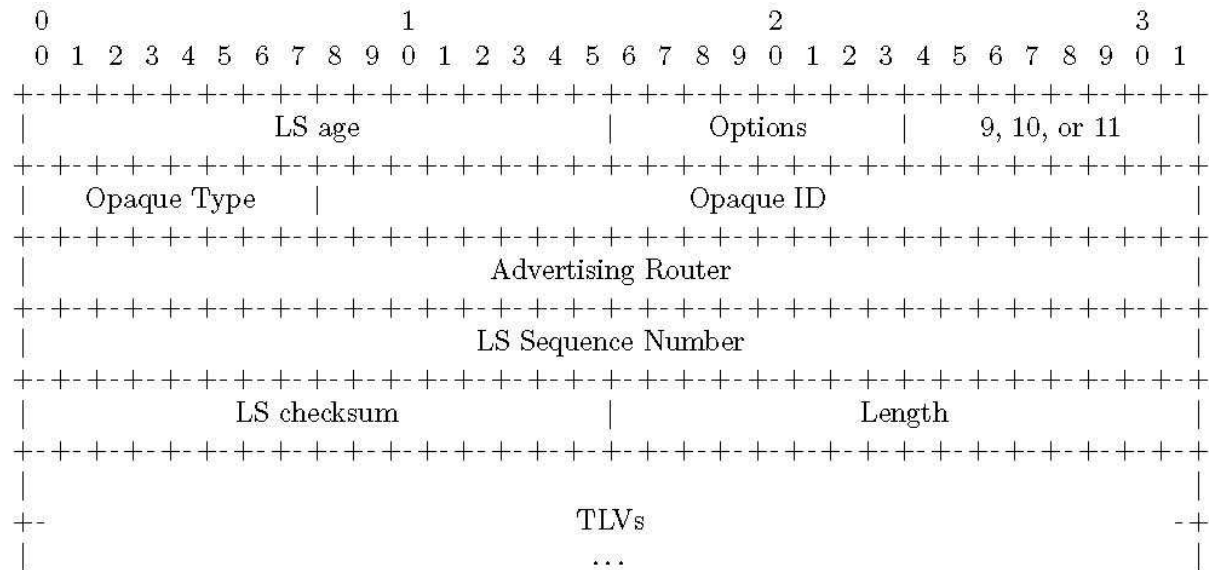


Figure 14: Standard Opaque LSA Header Format

### 3.3.1.6. The PLI TLV

The PLI LSA payload consists of one or more nested type/length/value (TLV) for extensibility. The format of each TLV is shown in Figure 15.

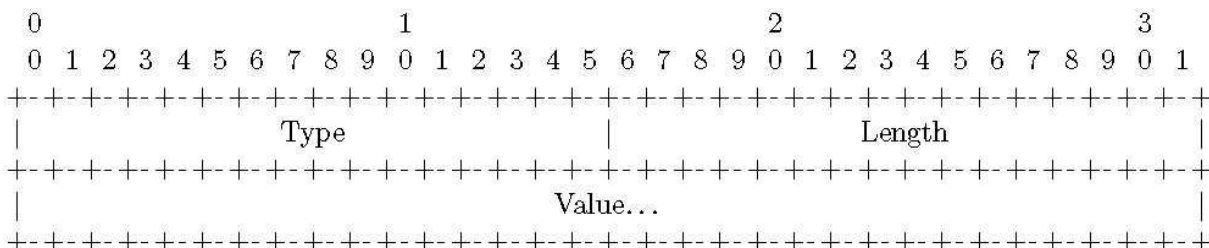


Figure 15: PLI TLV Format

The Length field defines the length of the value portion in octets (thus a TLV with no value portion would have a length of zero). The TLV is padded to four-octet alignment; padding is not included in the length field (so a three octet value would have a length of three, but the total size of the TLV would be eight octets). Nested TLVs are also 32-bit aligned. Unrecognized types are ignored. This document defines type 1 — PLI.

In PLI LSA, there is only one top level TVL — PLI TLV to describe the physical layer

impairments and the length is variable.

### 3.3.1.7. The Sub-TLVs

The four sub-TLVs are adopted from [10], *i.e.* link ID sub-TLV as shown in Figure 16, waveband sub-TLV as shown in Figure 17, wavelength sub-TLV as shown in Figure 18, impairment parameter sub-TLV as shown in Figure 19.

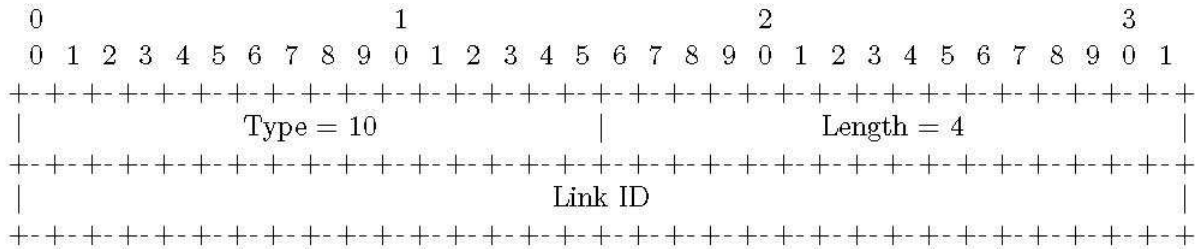


Figure 16: Link ID Sub-TLV Format

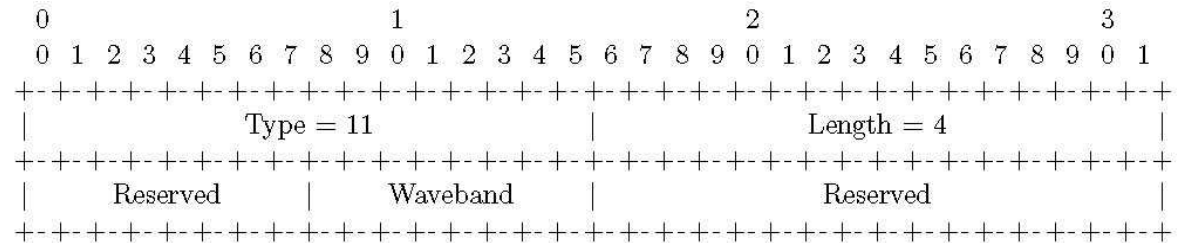


Figure 17: Waveband Sub-TLV Format

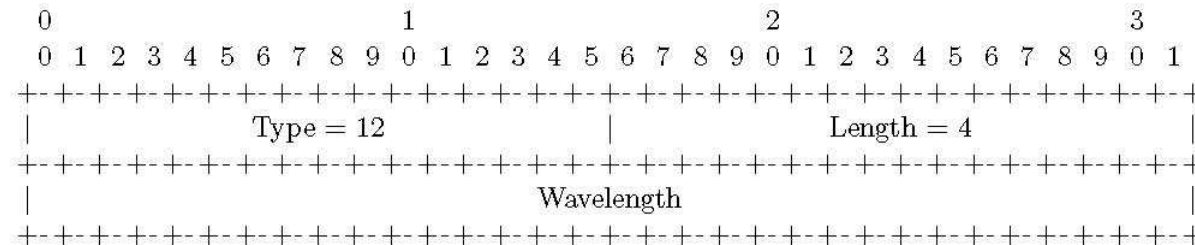


Figure 18: Wavelength Sub-TLV Format

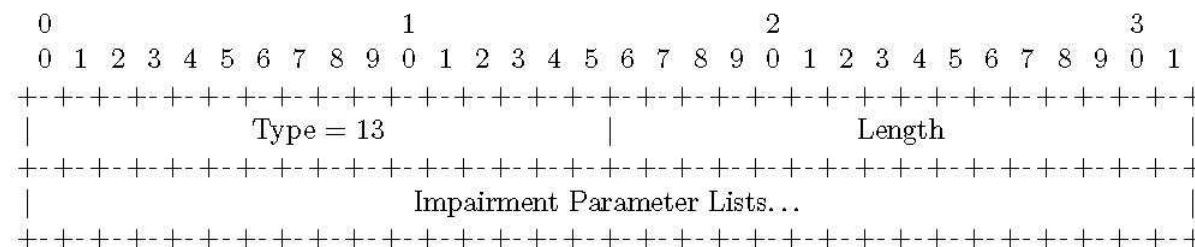


Figure 19: Impairment Parameter Sub-TLV Format

### 3.3.1.8. The Sub-sub-TLVs

Being different with [10] which does not use nested TLVs in the impairment parameter lists, 11 sub-sub-TLVs are defined. This approach uses the standard en/decapsulation method, in turn simplifies the implementation and increases the code reusability. One or more sub-sub-TLVs can be carried in the impairment parameter sub-TLV side by side. The ordering of each sub-sub-TLV also represents the priority. The format of these sub-sub-TLVs is shown

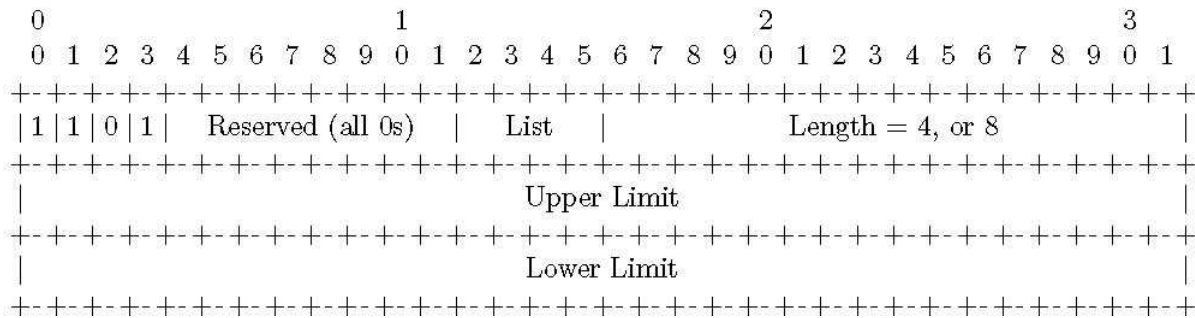


Figure 20. The list in the sub-sub-TLVs is actually the supported PLI, can be one of the following [8, page 25]:

- OPM Q-factor
- OPM BER
- OIM total power
- OIM channel power
- OIM channel wavelength
- OIM OSNR
- OIM PMD
- OIM residual CD
- OIM routing state
- OIM amplifier noise
- OIM amplifier transients

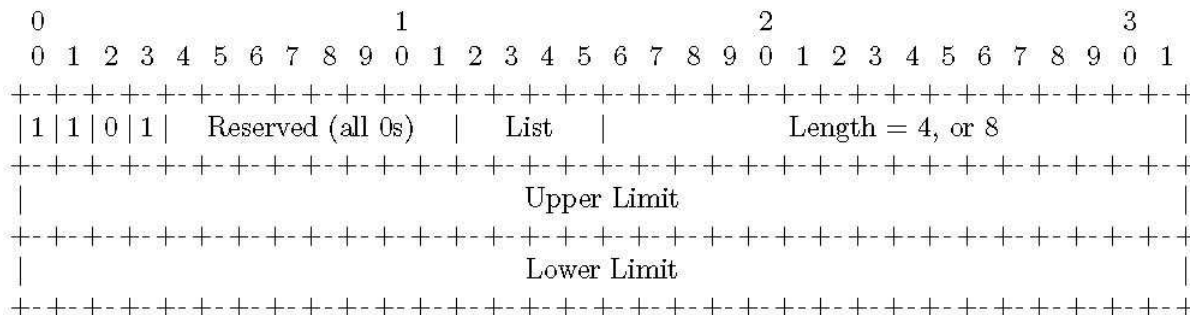
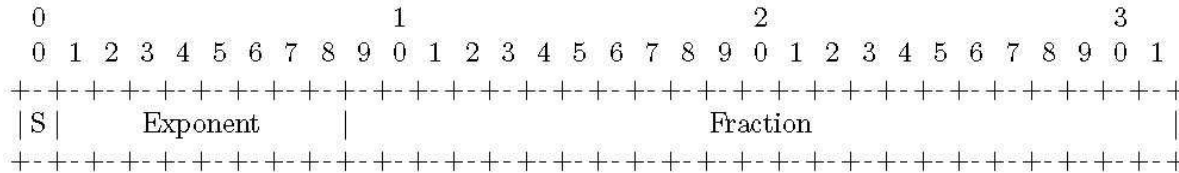


Figure 20: Sub-Sub-TLV Format



**Figure 21: IEEE Floating Point Format**

All the sub-sub-TLVs are optional. Unrecognised sub-TLVs are ignored. If Length equals to 4, then means the impairment’s value is a unique number, *i.e.* there is no Lower Limit field in this element. If Length equals 8, then means the impairment’s value is a range, *i.e.* there is Lower Limit field in this element. The upper limit and lower limit values use the (32 bit) IEEE Floating Point format. For quick reference, this format is as Figure 21: S is the sign, Exponent is the exponent base 2 in “excess 127” notation, and Fraction is the mantissa -1, with an implied binary point in front of it. Thus, the above represents the value:

$$(-1)^S \times 2^{Exponent-127} \times (1 + Fraction)$$

For more details, refer to [17]. For the usage details of the TLVs, refer to [11-16].

## 4. Implementation in a Emulation Environment: DRAGON

DRAGON (Dynamic Resource Allocation via GMPLS Optical Networks) is a GMPLS-based control plane which includes advanced inter-domain service routing techniques and detailed application formalizations.

As explained in Section 4.1.2, being it working and open source makes it a suitable starting point to introduce and emulate DICONET hybrid extensions previously described (Section 2).

### 4.1. DRAGON overview

As described in [6], the Dynamic Resource Allocation in GMPLS Optical Networks (DRAGON) project is developing technology and deploying network infrastructure that allows advanced e-science applications to dynamically acquire dedicated and deterministic network resources to link computational clusters, storage arrays, visualization facilities, remote sensors, and other instruments into globally distributed and application specific topologies. These advanced network services are motivated by e-science applications which use expensive resources such as radio telescopes, powerful computational clusters, and large data repositories that allow data sharing among researchers regardless of location. The DRAGON project is addressing this by employing all-optical network technologies, Generalized Multi-Protocol Label Switching (GMPLS) routing and signaling protocols, advanced interdomain-service routing techniques, and detailed application formalizations to deliver these advanced services. The objective is to enable the dynamic configuration of these large scientific resources and network infrastructures into application-specific network topologies in an automated response to domain scientists’ requests.

### 4.1.1. Overall DRAGON Architecture

Network infrastructures are highly diverse, and this diversity will increase in the future. The diversity referenced here includes the type of network technologies, internal provisioning mechanisms, administrative ownership, use policies, and capabilities. As a result of this diversity, different networks will have different internal provisioning mechanisms. Though DRAGON control plane architecture utilizes GMPLS as a basic building block, it does not assume that GMPLS protocols should be used within a domain for actual provisioning. The issues that must be resolved for an inter-domain control plane are the protocols and messages exchanged across a domain boundary. The inter-domain architecture and provisioning mechanisms used are GMPLS based in the sense that they rely on a link-state protocol for inter-domain topology exchanges, engage in multi-constraint path computation to determine appropriate network routes, and utilize RSVP-TE for signaling. Should a domain decide to use a provisioning mechanism other than GMPLS internally, interoperability will require translation of those internal representations to external representations that are compatible with these control plane definitions. The overall DRAGON architecture is depicted in Figure 22. A description of this architecture is provided in the following subsections.

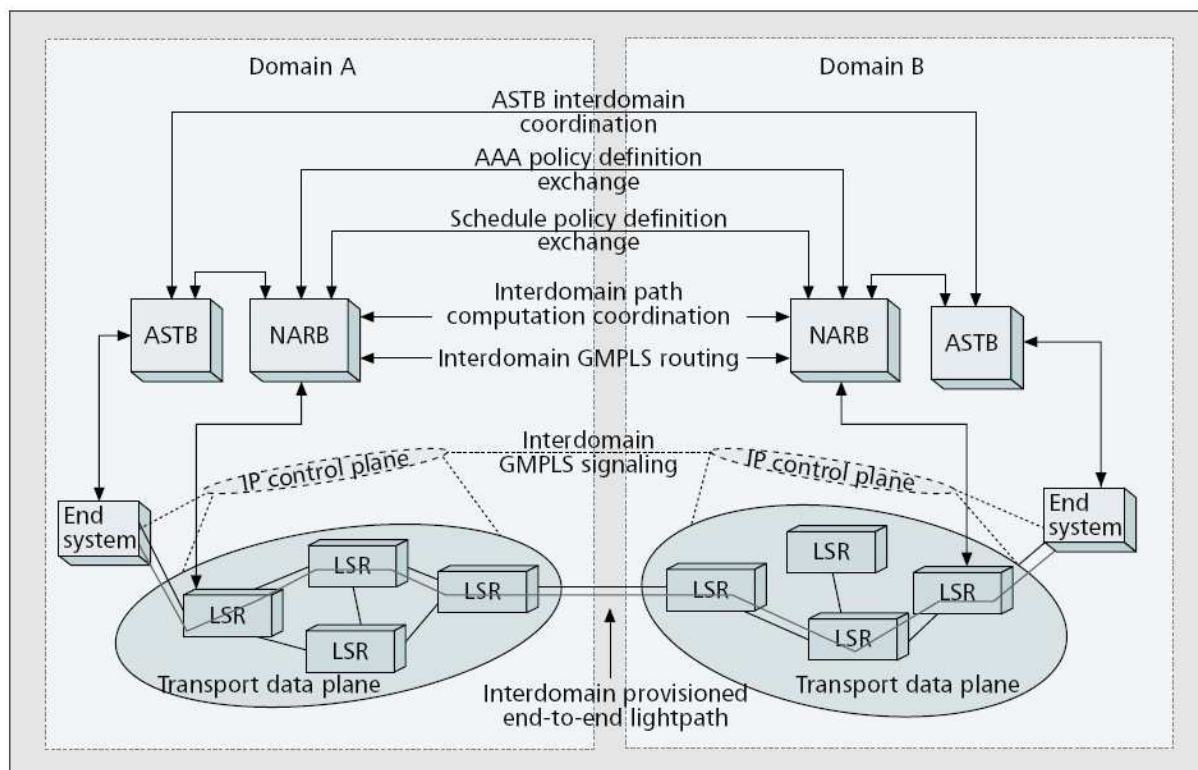


Figure 22: DRAGON Control Plane Architecture

#### 4.1.1.1. Network Aware Resource Broker (NARB)

NARB is an entity that represents the local Autonomous System (AS) or domain. The NARB serves as path computation engine which can be queried by end-systems or other devices to request the availability of traffic engineered paths between specified source and destination pairs. A stand-alone subcomponent of the NARB called the Resource Computation Element (RCE) performs the path computation tasks. The NARB is also responsible for inter-domain routing. NARBs peer across domains and exchange topology information to enable inter-domain path computation and Label Switched Path (LSP) provisioning. This inter-domain

topology exchange can be based on the actual topology obtained from local OSPF-TE, or optionally based on an "abstracted" view of the domain topology (generated by configuration file or automatic synthesis of the OSPF-TE link state database). Domain abstraction provides mechanisms for an administrative domain to advertise to the outside world a highly simplified view of its topology. This allows domains to hide their real topologies as well as minimize the amount of external updates required. The trade-off is reduced accuracy for path computations. Each administrative domain can utilize configuration parameters to tailor its domain abstraction to the desired level.

Because the RCE functionality is a coherent part of the overall NARB functionality, RCE is often considered a subcomponent of NARB. In the DRAGON control plane, NARB/RCE provides path computation, resource management, and LSP provisioning services to other control plane components. NARB/RCE has the following virtual label switch router (VLSR) related features:

- Dynamic resource state collection (via both intra- and inter-domain OSPF-TE) and resource management
- Domain topology summarization, abstraction and advertisement via OSPF-TE
- Multi-constraint based path computation
- Inter-domain routing

#### **4.1.1.2. Client System Agent (CSA)**

The CSA is software that runs on (or on behalf of) any system which terminates the data plane (traffic engineering) link of the provisioned service. This is the software that participates in the GMPLS protocols to allow for on-demand end-to-end provisioning from client system to client system. In this context, Client System (CS) is a very broad term. It generally means any device which is on the edge of a DRAGON enabled dynamically provisioned network. This could include a host, a computational cluster, a router, a radio telescope, and various other networked devices. A "client" is any system which is requesting network services. The CSA may also interact with the Application-Specific Topology Builder (ASTB) if a more complicated topology is to be built.

#### **4.1.1.3. Application-Specific Topology Builder (ASTB)**

The DRAGON architecture includes the notion of establishing application-specific topologies (ASTs). These are requested by an end user and are generally a set of LSPs, which an application domain desires to be set up as a group. The Application-Specific Topology Builder (ASTB) accepts requests from users or end systems for multiple network connections, and utilizes the services of the NARB to determine if the requested network paths are available with appropriate Authentication, Authorization, and Accounting (AAA) and schedule constraints applied. The NARB views these requests as individual LSPs and the ASTB is responsible for the assembly of multiple LSPs in to a specific topology group. The ASTB utilizes the capabilities of the other DRAGON control plane elements (e.g., NARB, VLSR, CSA, etc.) to request instantiation of the individual LSPs, maintain the mapping of individual LSP groupings to specific topologies, and interact with user requests.

#### 4.1.1.4. Virtual Label Switch Router (VLSR)

In order to provide end-to-end automated provisioning, it was necessary to provide the GMPLS protocols to cover switching components that did not have their own native GMPLS protocols. For this reason the virtual label switch router (VLSR) was developed. A non-GMPLS capable network device is converted to a VLSR by the addition of a small UNIX-based PC which runs a GMPLS control plane consisting of OSPF-TE [5] and RSVP-TE [4]. The VLSR PC acts as a GMPLS proxy agent for a device and translates protocol events into commands that the local switching element understands, such as SNMP, TL1, or even scripted CLI commands. This allows non-GMPLS devices to be included in end-to-end path instantiations. The primary use for VLSR in the DRAGON project is to control Ethernet switches via the GMPLS control plane. However, the VLSR has also been adapted to control TDM and optical switches. While a VLSR is not identified directly in the architectural diagram, any of the LSRs identified could in fact be a VLSR.

#### 4.1.2. Why DRAGON?

DRAGON is an open-source working GMPLS software platform, which, even if not fully implemented, allows reliable GMPLS network emulation and allows developers to improve the original software with extensions/modifications. Though the related documentation is not fully available and mostly consists of high level descriptions, DRAGON software is the best solution, as it satisfies our requirements in terms of functionality and accessibility to code and allows us to extend it to DICONET project without any serious problem.

## 4.2. Original DRAGON software

### 4.2.1. Restricting the Action Field

The overall DRAGON architecture is described in Figure 22. However, we are interested only in the single transport data layer, due to the fact that the solution we are developing in DICONET project just considers intra-domain paths set-up/release/monitoring operations. Moreover, as anticipated in the module description, the idea is to provide a distributed system, so centralized modules like ASTB and NARB are not considered and eventually replaced by local modules on every node with similar functionality. However, several modules need to be introduced into the overall architecture to account for PLIs, in addition to modifications to existing modules.

### 4.2.2. Dragon Processes and their Interactions

Particularly we are going to use VLSR's and (partially) CSA's functionalities (even modifying or integrating them) by running four specific daemon processes:

- Zebra daemon: A free TCP/IP routing protocol providing inter-node communication. The DRAGON project has extended the open source GNU Zebra routing software package to include required GMPLS functionality. The GNU Zebra distribution is a routing protocol suite and includes multiple network protocols such as RIP, OSPF-TE, BGP and others.
- DRAGON daemon: It provides the user access to GMPLS functionalities and monitoring of data-layer status. It implements DRAGON Command Line Interface (CLI) commands specific for configuring VLSR-specific parameters on RSVP-TE daemon

- **RSVP daemon:** RSVP-TE is a signaling protocol that allows the set up of reserved highway for data transmission with a specified Quality of Service (QoS). Applications running on IP end systems can use RSVP-TE to indicate to other nodes, the nature (e.g. bandwidth) of the packet streams they want to receive. The DRAGON project has extended the open source KOM RSVP-TE Engine from the Technical University of Darmstadt to include required GMPLS-TE functionality.
- **OSPF-TE daemon:** OSPF-TE is a link-state routing protocol developed for the IP networks. OSPF-TE sends Link State Advertisements (LSAs) to all other routers within the same area. Each OSPF-TE router maintains an identical database describing the network topology. From this database, a routing table is calculated by constructing a shortest path tree. In other words, the OSPF-TE routers accumulate Link State information and use the SPF algorithm to calculate the shortest path to each node. OSPF-TE recalculates routes quickly in the face of topological changes, utilizing a minimum overload. OSPF-TE also carries the link state information for the GMPLS. It includes the OSPF-TE. The Constraint Shortest Path First (CSPF) module is provided as a separate module which includes an Application Programming Interface (API) in the form of function calls and provides the ability to compute the traffic engineered paths based on the OSPF-TE derived Link State Database (LSDB).  
The DRAGON OSPF-TE software is extended from the GNU Zebra OSPF daemon module and understands how to interpret those LSDB data structures. The current CSPF implementation is limited to a single network region (or layer) and considers the standard GMPLS-TE constraints of bandwidth availability and interface switching capability.

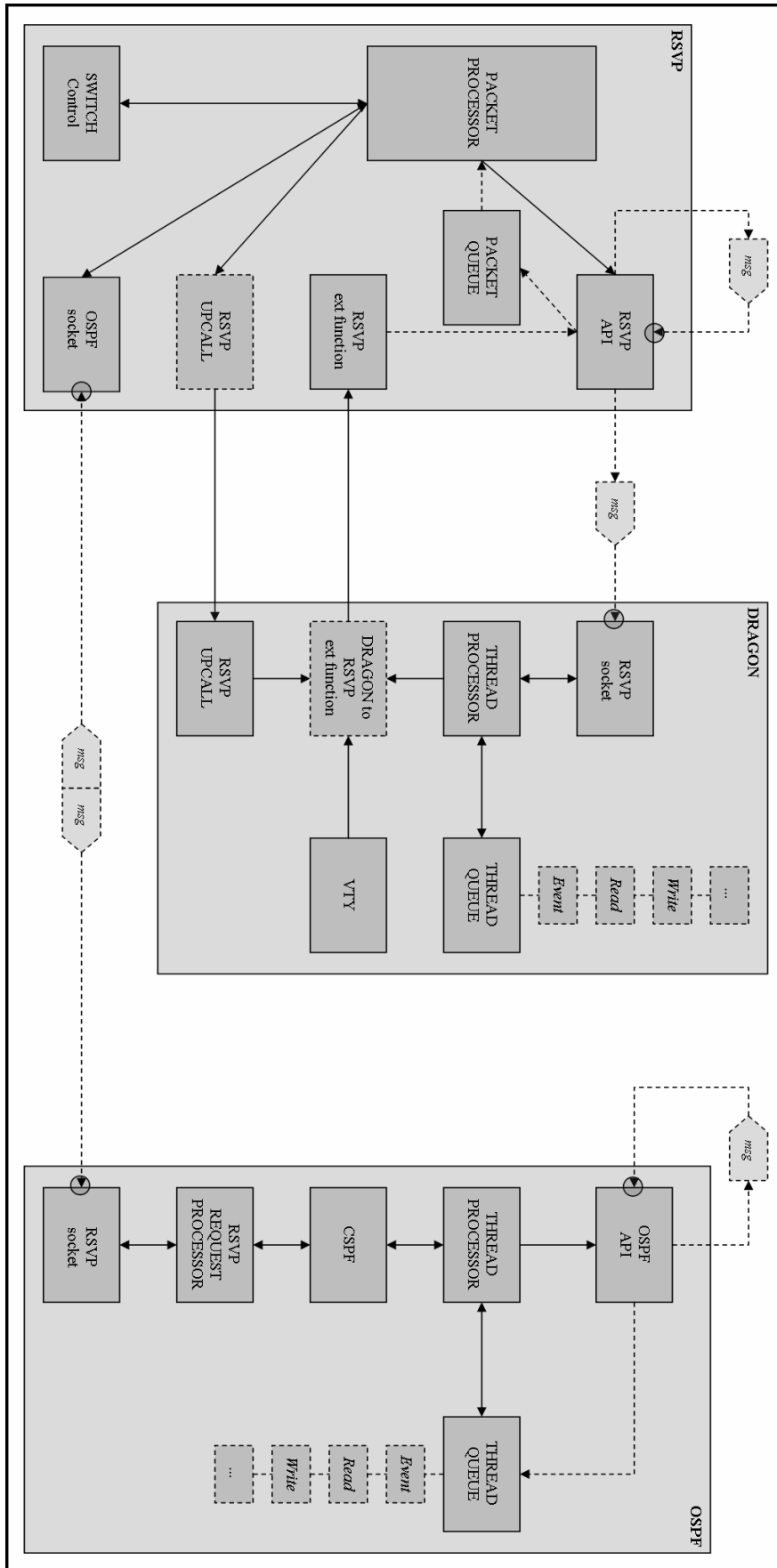


Figure 23: Node's Original Daemons

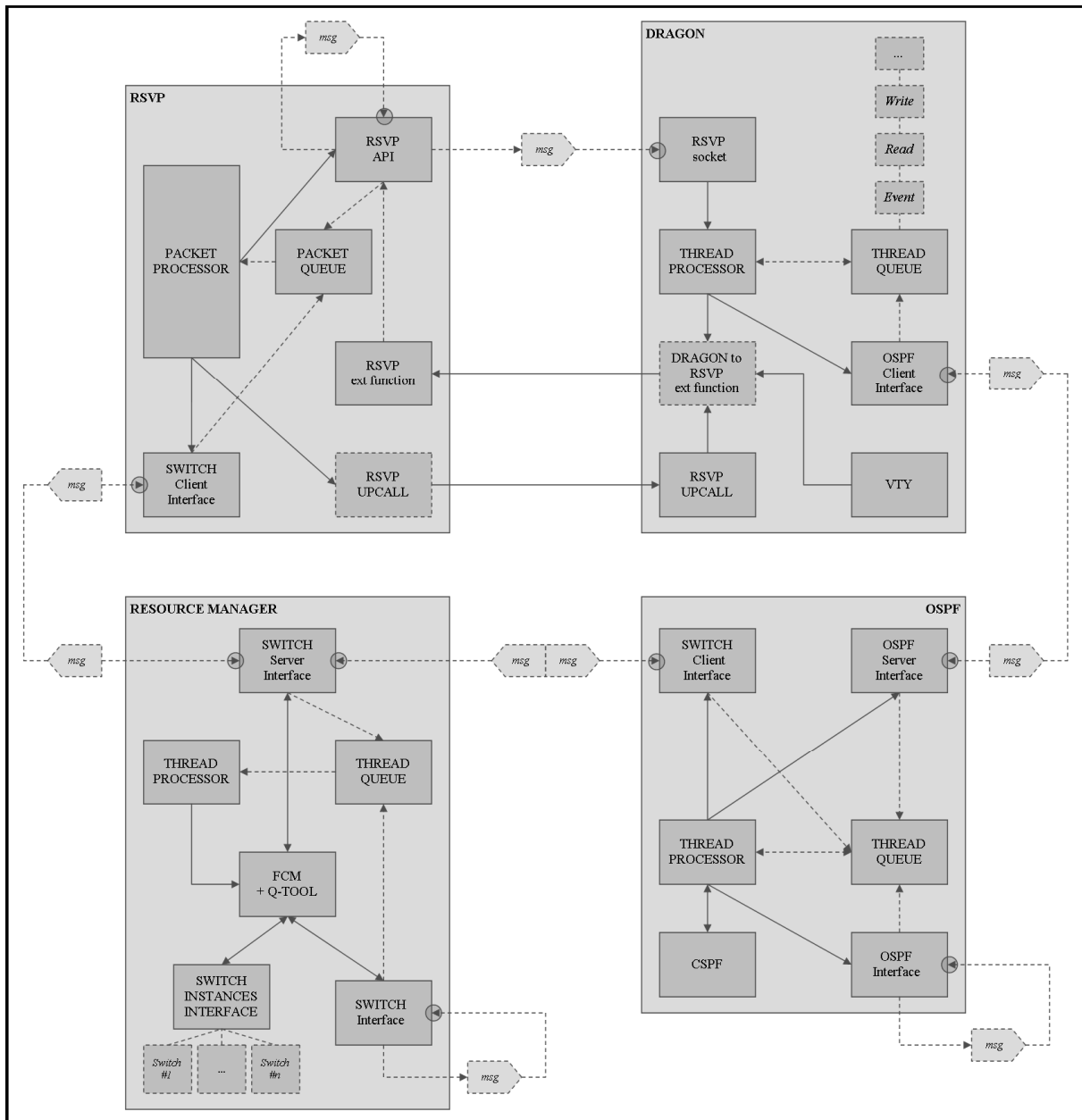
Figure 23 shows various daemons involved in DRAGON. Note that Zebra is not shown in the Figure. As described, it allows inter nodes communication and it's practically the ground process which runs other daemons. DRAGON, RSVP and OSPF are Zebra's additional modules: DRAGON and OSPF are written in C and follows the standards for Zebra modules (they have similar internal units/data structures), whilst RSVP is a C++ application modified in order to interact with DRAGON and RSVP-TE daemons.

In both OSPF and DRAGON daemons a "thread processor" unit is present. Threads are supposed to be atomic operations, which daemons should perform in order to accomplish their tasks. Threads are essentially event notifications or read/write requests over specific socket, which may then trigger other operations, like messages processing, other daemons querying, query answering and so on. Threads can be generated by other threads or during the execution of internal node operations. They are stored in a thread queue structure and sequentially retrieved and processed by "thread processor" following well defined priority rules and timing.

OSPF-TE is then equipped with a CSPF unit in order to perform the CSPF algorithm for RSVP-TE packet routing and provided with OSPF-TE API unit in charge of handling message exchanges according to OSPF-TE protocol. An additional socket for interaction with RSVP-TE daemon is available, allowing RSVP-TE process to query OSPF-TE for a route between a source and a destination node (request which occurs when a PATH message arrives without explicit route object specified).

Dragon has a VTY unit which allows user to interface with the system and to inject his/her requests. A set of *ad-hoc* defined external functions is given in order to allow interaction with RSVP-TE module: such functions access RSVP-TE API allowing DRAGON to initialize user required operations inside RSVP-TE process. RSVP-TE daemon is a packet driven application in which incoming packets are stored in a queue, where they are accessed in a FIFO fashion by packet processor, producing an answer message whose kind and destination are related to the kind and content of the incoming message. RSVP-TE contains a "switch control" unit which is supposed to interface/emulate physical devices. RSVP-TE daemon may interact with RSVP-TE daemons on other nodes (as required by RSVP-TE protocol) and with local OSPF-TE and DRAGON daemons.

### **4.3. DRAGON Modifications/Extensions**



**Figure 24: Node's Revised Daemons**

In order to introduce the new features described in Section 2, the overall DRAGON architecture shown in Figure 23 has to be modified and integrated. For the sake of simplicity we decided to introduce one new process, in order to reduce the complexity related to the interactions between modules. Thus, in Figure 24 (new daemons/units in red), we add a new daemon, “Resource Manager”, which intuitively is in charge of resource management, meaning:

- Access to physical devices
- Monitor of resources availability changes
- Feasibility checks

Practically, Resource Manager will implement the functionalities of Resource Manager Module (Section. 2.1.4) and Feasibility Control Module (section. 2.1.3), the interface to Q-Tool Module (Section. 2.1.7) and all the related services. With such a solution, RSVP-TE

module is free from all the resources availability and feasibility calculations and works simply as a transport layer protocol, as it was designed for, leaving most of the decision making activities to other modules.

On the other hand, Resource Manager must be equipped with a unit (Switch Instances Interface) which allows it to access, monitor and send commands to all the active physical devices available at node. *Ad hoc* plug-ins should be developed for such unit in order to let it correctly interface every device available at nodes. Switch Instances Interface is controlled by FCM unit (in charge of all feasibility computation), which inside will have a sub-unit for Q-Tool evaluation. In order to perform the remote impairment check, FCM unit may send messages to FCM units on other nodes through a “switch interface” defined for this purpose. Moreover, FCM receives all resource change notifications from local Switch Instances Interface and send them to OSPF-TE daemon via “switch server interface”. Being RSVP-TE “switch control” removed, RSVP-TE needs to query Resource Manager to retrieve the feasibility check response, through “switch client interface”. A similar interface is defined for OSPF-TE daemon, in order to allow it to subscribe and receive resource availability notifications from Resource Manager.

Finally a new interface has to be defined between DRAGON and OSPF-TE daemons, in order to allow DRAGON to retrieve the information about network status (supervising requests) and above all the  $k$  candidate routes for a LSP requests. Furthermore, several modifications are required to all original DRAGON daemons and are identified below:

- DRAGON
  - Data structures for storage of the  $k$  candidate routes for a given requested LSP
  - Algorithm for the selection of the next candidate route in case of unfeasibility of the previous route
  - Thread processor extended in order to handle thread related to OSPF-TE
- RSVP-TE
  - Extension of the RSVP-TE packets in order to carry additional information (e.g., PLIs information) used for feasibility evaluation
  - Feasibility check request implementation (from RSVP-TE to Resource Manager)
  - Modification of the internal state machine (e.g., waiting for Resource Manager’s feasibility response, etc.)
  - Extension to the packet queue and processor in order to handle correctly messages incoming from Resource Manager
- OSPF-TE
  - Thread processor extended in order to handle threads related to interaction with DRAGON and Resource Manager daemons
  - Extension of OSPF-TE Link State Database in order to add monitoring over wavelength availability
  - Extension/Modification of the CSPF algorithm
  - Subscription mechanism to notification service provided by Resource Manager

All the above tasks are carried out in T5.2.

## 5. Conclusions

The two selected control plane architectures, namely hybrid and PCE based architectures are discussed. A high-level system design of these two control plane architectures based DRAGON open source GMPLS emulator is presented. The document identified the DRAGON modules that need to be extended to introduce PLI and other related information. It has also developed new modules and corresponding interfaces that are introduced in DRAGON emulator. These extensions and new modules are designed to reduce the complexity of overall design. Several interfaces are extended and several new interfaces are defined. The possible message encoding mechanisms are discussed for both architectures. As we go along with the implementation in T5.2 and WP6 (which are ongoing tasks), there might be several small changes in the actual design based on the designers choice in some of the interfaces is modules. Hence, we expect to update the document or present the final design in other related deliverables in November 2009. The purpose of the document is also to provide a detailed documentation of interfaces, so that different partners working on different architectures and other related modules can work independently. In addition it also serves as the reference while we do the porting of these architectures to testbed for demonstration activities. The detailed mechanisms to deal with possible active lightpath disruption and the parameters that are carried in various interfaces are provided in Appendix-A and B, respectively.

## References

- [1] D2.3: Initial report on control plane protocol extension prototype and design issues, April 2009
- [2] RFC 2205: R. Braden, et al., "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", Sep 1997.
- [3] RFC 3209: D. Awduche, et al., "RSVP-TE: Extensions to RSVP for LSP Tunnels", Dec 2001
- [4] RFC 3630: D. Katz, et al., "Traffic Engineering (TE) Extensions to OSPF Version 2", Sep 2003.
- [5] G. Bernstein, et al., "Routing and Wavelength Assignment Information Encoding for Wavelength Switched Optical Networks", IETF Draft, draft-ietf-ccamp-rwa-wson-encode, Mar 2009.
- [6] J. Ash, J.L. Le Roux, "Path Computation Element (PCE) Communication Protocol Generic Requirements", *RFC 4657*, Sept. 2006
- [7] J.L. Le Roux, "Requirements for Path Computation Element (PCE) Discovery", *RFC 4674*, Oct. 2006
- [8] H. Zang, J.P. Jue, and B. Mukherjee, "A review of routing and wavelength assignment approaches for wavelength-routed optical WDM networks", *Optical Networks Magazine*, 1(1):47–60, 2000.
- [9] Deliverable D2.2, "Planning and Optimization Aspects of Dynamic Optical Networks", *Technical report, DICONET FP7 Project*, 2008.
- [10] Deliverable D3.1, "Network Impairment in Transparent Networks and Definition of Monitoring Strategy", *Technical report, DICONET FP7 project*, September 2008.
- [11] Yabin Ye and *et al.*, "M2.2: Recommendations on Dissemination Physical Impairment Info across the Network", *DICONET FP7 Project*, April 2008.
- [12] J.L. Le Roux, J.P. Vasseur, Y. Ikejiri, and R. Zhang, "OSPF Protocol Extensions for Path Computation Element (PCE) Discovery", *RFC 5088*, January 2008.
- [13] R. Coltun, "The OSPF Opaque LSA Option", *RFC 2370*, July 1998.
- [14] R. Coltun, D. Ferguson, and J. Moy, "OSPF for ipv6", *RFC 2740*, December 1999.
- [15] K. Kompella and Y. Rekhter, "OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)", *RFC 4203*, October 2005.
- [16] A. Lindem, and et al., "Extensions to OSPF for Advertising Optional Router Capabilities", *RFC 4970*, July 2007.
- [17] L. Berger, I. Bryskin, A. Zinin, and R. Coltun, "The OSPF Opaque LSA Option", *RFC 5250*, July 2008.
- [18] IEEE, "IEEE Standard for Binary Floating-Point Arithmetic", *IETF*, 1985
- [19] DRAGON: A Framework for Service Provisioning in Heterogeneous Grid Networks
- [20] DRAGON: <http://dragon.maxgigapop.net/>

## Appendix-A

Remote Impairment Feasibility Check (Section 2.3.2.16) Service description in WSDL 1.1 format.

```
<definitions name="RemoteQFactorEvaluation"
  targetNamespace="http://create-net.org/diconet.wsdl"
  xmlns:tns="http://create-net.org/diconet.wsdl"
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
  xmlns:dico="http://create-net.org/diconet/schema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
    <schema targetNamespace="http://create-net.org/diconet/schema"
      xmlns="http://www.w3.org/2000/10/XMLSchema"
      xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
      xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">

      <complexType name="wavencoding">
        <documentation>
          Wavelength encoding as described in section 3.3 of
          http://www.ietf.org/internet-drafts/draft-ietf-ccamp-rwa-
          -wson-encode-01.txt
        </documentation>
        <sequence>
          <element name="grid" type="xsd:unsignedByte" use="required"/>
          <element name="cs" type="xsd:unsignedByte" use="required"/>
          <element name="wlnumber" type="xsd:short" use="required"/>
        </sequence>
      </complexType>

      <complexType name="link">
        <sequence>
          <complexType name="source">
            <complexContent>
              <restriction base="soapenc:Array">
                <attribute ref="soapenc:arrayType"
                  wsdl:arrayType="xsd:unsignedByte[]" />
              </restriction>
            </complexContent>
          </complexType>
          <complexType name="destination">
            <complexContent>
              <restriction base="soapenc:Array">
                <attribute ref="soapenc:arrayType"
                  wsdl:arrayType="xsd:unsignedByte[]" />
              </restriction>
            </complexContent>
          </complexType>
        </sequence>
      </complexType>

      <complexType name="lightpath">
        <complexContent>
          <restriction base="soapenc:Array">
            <attribute ref="soapenc:arrayType"
              wsdl:arrayType="link[]" />
          </restriction>
        </complexContent>
      </complexType>
    </schema>
  </types>
```

```
<message name="rImpChkReq">

  <part name="hops" type="dico:lightpath"/>
  <part name="wavelength" type="dico:wavencoding"/>
  <part name="bitrate" type="xsd:float"/>
  <part name="power" type="xsd:float"/>
  <documentation>
    MOD OOK=1: On-off keying.
    MOD DPSK=2: Differential Phase Shift Keying.
  </documentation>
  <part name="modulationformat" type="xsd:unsignedByte"/>
  <documentation>
    FEC NONE=0: No FEC.
    FEC RS255239=1: RS(255,239) coding.
  </documentation>
  <part name="fec" type="xsd:unsignedByte"/>

</message>

<message name="rImpChkRsp">

  <part name="qfactor" type="xsd:float"/>
  <part name="feasible" type="xsd:boolean"/>

</message>

<portType name="RemoteQFactorCheckService">
  <operation name="RemoteQFactorCheck">
    <input message="tns:rImpChkReq"/>
    <output message="tns:rImpChkRsp"/>
  </operation>
</portType>

<binding name="FCMSoapBinding" type="tns:RemoteQFactorCheckService">

  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>

  <operation name="RemoteQFactorCheck">
    <soap:operation soapAction="urn:diconet:remoteqfactorcheck"/>

    <input>
      <soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:diconet:remoteqfactorcheck"
        use="encoded"/>
    </input>

    <output>
      <soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:diconet:remoteqfactorcheck"
        use="encoded"/>
    </output>

  </operation>

</binding>

</definitions>
```

## Appendix-B

Various messages with their required parameters and the results are shown in the following tables:

<b>superReq</b>	Client_ID	<i>User ID</i>
	Client_Address	<i>User Address</i>
	Service_Request_ID	<i>Sequential Number</i>
	Service_Type	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• ACTIVE_LSPS</li> <li>• NETWORK_TOPOLOGY</li> <li>• RESOURCES_AVAILABILITY</li> </ul>
Service_Required_Parameters	<i>Void</i>	
<b>superRsp</b>	Service_Request_ID	<i>Service_Request_ID of the original request</i>
	Error_Occurred	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• NO_ERROR</li> <li>• <i>Other error codes (TBD)</i></li> </ul>
	Results	<i>Depending on the Service Type</i> ACTIVE_LSPS: <ul style="list-style-type: none"> <li>• <i>Route description (Node Address + Interface +status)</i></li> </ul> NETWORK_TOPOLOGY <ul style="list-style-type: none"> <li>• <i>Network nodes (Node Address)</i></li> <li>• <i>Links (Node source, Node destination, Interface)</i></li> </ul> RESOURCE_AVAILABILITY <ul style="list-style-type: none"> <li>• <i>Available wavelengths at each node (Node address, wavelength)</i></li> <li>• <i>Used wavelengths at each node (Node Address, wavelength, LSP_ID)</i></li> </ul>

<b>operReq</b>	Client_ID	<i>User ID</i>
	Client_Address	<i>User Address</i>
	Service_Request_ID	<i>Sequential Number</i>
	Service_Type	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• LSP_SETUP</li> <li>• LSP_RELEASE</li> </ul>
Service_Required_Parameters	<i>Depending on the Service Type</i> LSP_SETUP <ul style="list-style-type: none"> <li>• source/destination nodes</li> <li>• LSP setup parameters</li> </ul> LSP_RELEASE <ul style="list-style-type: none"> <li>• <i>to-be-released LSP ID</i></li> </ul>	
<b>operRsp</b>	Service_Request_ID	<i>Service_Request_ID of the original request</i>
	Error_Occurred	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• NO_ERROR</li> <li>• <i>Other error codes (TBD)</i></li> </ul>
	Results	<i>Void</i>

<b>netStatusReq</b>	Client_ID	<i>TE_Agent ID</i>
	Client_Address	<i>[TE-Agent socket]</i>
	Service_Request_ID	<i>Sequential Number</i>
	Service_Type	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• NETWORK_TOPOLOGY</li> <li>• RESOURCES_AVAILABILITY</li> </ul>
	Service_Required_Parameters	<i>Void</i>
<b>netStatusRsp</b>	Service_Request_ID	<i>Service_Request_ID of the original request</i>
	Error_Occurred	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• NO_ERROR</li> <li>• <i>Other error codes (TBD)</i></li> </ul>
	Results	<i>Depending on the Service Type</i> NETWORK_TOPOLOGY <ul style="list-style-type: none"> <li>• <i>Network nodes (Node Address)</i></li> <li>• <i>Links (Node source, Node destination, Interface)</i></li> </ul> RESOURCE_AVAILABILITY <ul style="list-style-type: none"> <li>• <i>Available wavelengths at each node (Node address, wavelength)</i></li> <li>• <i>Used wavelengths at each node (Node Address, wavelength, LSP_ID)</i></li> </ul>

<b>signReq</b>	Client_ID	<i>TE_Agent ID</i>
	Client_Address	<i>[TE-Agent socket]</i>
	Service_Request_ID	<i>Sequential Number</i>
	Service_Type	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• LSP_SETUP</li> <li>• LSP_RELEASE</li> </ul>
	Service_Required_Parameters	<b>See operReq</b> <i>Service_Required_Parameters</i>
<b>signRsp</b>	Service_Request_ID	<i>Service_Request_ID of the original request</i>
	Error_Occurred	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• NO_ERROR</li> <li>• <i>Other error codes (TBD)</i></li> </ul>
	Results	<i>void</i>

<b>routeReq</b>	Client_ID	<i>TE_Agent ID</i>
	Client_Address	<i>TE-Agent socket</i>
	Service_Request_ID	<i>Sequential Number</i>
	Service_Type	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• ROUTE</li> </ul>
	Service_Required_Parameters	<i>See operReq, LSP_SETUP</i> Service_Required_Parameters
<b>routeRsp</b>	Service_Request_ID	<i>Service_Request_ID of the original request</i>
	Error_Occurred	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• NO_ERROR</li> <li>• Other error codes (TBD)</li> </ul>
	Results	<i>Ordered sequence of k routes</i> <i>For every route, ordered sequence of its links (node+interface)</i>

<b>subRANReq</b>	Subscriber_ID	<i>Routing Module ID</i>
	Subscriber_Address	<i>[Routing Module socket]</i>
	Subscription_Request_ID	<i>Sequential Number</i>
	Service_Configuration_Parameters	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• WAVELENGTH_AVAILABILITY</li> </ul>
<b>subRANRsp</b>	Subscription_Request_ID	<i>Related Subscription_Request_ID</i>
	Subscription_Response	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• ACCEPTED</li> <li>• REFUSED (<i>different refusal motivation could be given: TBD</i>)</li> </ul>
<b>evtRANntf</b>	Subscription_Request_ID	<i>Related Subscription_Request_ID</i>
	Occurrence_Time	<i>Timestamp</i>
	Additional_Information	<i>Wavelength + availability</i>

<b>impChkReq</b>	Client_ID	<i>TE_Agent ID</i>
	Client_Address	<i>TE-Agent socket</i>
	Service_Request_ID	<i>Sequential Number</i>
	Service_Type	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• IMPAIRMENT_CHECK</li> </ul>
	Service_Required_Parameters	<i><b>OPD</b> object (retrieved from PATH message)</i>
<b>impChkRsp</b>	Service_Request_ID	<i>Service_Request_ID of the original request</i>
	Error_Occurred	<i>One of the following:</i> <ul style="list-style-type: none"> <li>• NO_ERROR</li> <li>• Other error codes (TBD)</li> </ul>
	Results	<i>Q-factor value</i> <i>Feasibility response (Q-factor above or under local tolerance threshold)</i>

<b>resChkReq</b>	Client_ID	<i>TE_Agent ID</i>
	Client_Address	<i>TE-Agent socket</i>
	Service_Request_ID	<i>Sequential Number</i>
	Service_Type	<i>One of the following:</i> <ul style="list-style-type: none"> <li>RESOURCES_CHECK</li> </ul>
	Service_Required_Parameters	<i>List of the available wavelengths common to all previous nodes in the route</i>
<b>resChkRsp</b>	Service_Request_ID	<i>Service_Request_ID of the original request</i>
	Error_Occurred	<i>One of the following:</i> <ul style="list-style-type: none"> <li>NO_ERROR</li> <li>Other error codes (TBD)</li> </ul>
	Results	<i>List containing the intersection of the "common to all previous nodes" available wavelengths set with the local available wavelengths (potentially void)</i>

<b>resAvailReq</b>	Client_ID	<i>TE_Agent ID</i>
	Client_Address	<i>TE-Agent socket</i>
	Service_Request_ID	<i>Sequential Number</i>
	Service_Type	<i>One of the following:</i> <ul style="list-style-type: none"> <li>RESOURCES_AVAILABILITY</li> </ul>
	Service_Required_Parameters	<i>void</i>
<b>resAvailRsp</b>	Service_Request_ID	<i>Service_Request_ID of the original request</i>
	Error_Occurred	<i>One of the following:</i> <ul style="list-style-type: none"> <li>NO_ERROR</li> <li>Other error codes (TBD)</li> </ul>
	Results	<i>List of local available wavelengths (potentially void)</i>

<b>cmdExeReq</b>	Client_ID	<i>TE_Agent ID</i>
	Client_Address	<i>TE-Agent socket</i>
	Service_Request_ID	<i>Sequential Number</i>
	Service_Type	<i>One of the following:</i> <ul style="list-style-type: none"> <li>PERFORM_XC</li> <li>RELEASE_XC</li> </ul>
	Service_Required_Parameters	
<b>cmdExeRsp</b>	Service_Request_ID	<i>Service_Request_ID of the original request</i>
	Error_Occurred	<i>One of the following:</i> <ul style="list-style-type: none"> <li>NO_ERROR</li> <li>Other error codes (TBD)</li> </ul>
	Results	<i>void</i>