



# Integration of peer-to-peer strategies and SOAP/XML for inter-domain, user-driven provisioning in an ASON/GMPLS network

C. Pinart, G. Junyent

Publication:	in in OSA Journal of Optical Networking
Vol.:	5
pp.:	246-262
No.:	4
Date:	Barcelona (Spain). April 24, 2006

This publication has been included here just to facilitate downloads to those people asking for personal use copies. This material may be published at copyrighted journals or conference proceedings, so personal use of the download is required. In particular, publications from IEEE have to be downloaded according to the following IEEE note:

©2007 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

# Integration of peer-to-peer strategies and SOAP/XML for inter-domain user-driven provisioning in an ASON/GMPLS network

Carolina Pinart

*Centre Tecnològic de Telecomunicacions de Catalunya, Avinguda Canal Olímpic s/n,  
08860 Castelldefels, Barcelona, Spain*

*carolina.pinart@cttc.es*

Gabriel Junyent

*Universitat Politècnica de Catalunya, Jordi Girona 1-3, 08034 Barcelona, Spain*

*junyent@tsc.upc.edu*

RECEIVED 1 AUGUST 2005; REVISED 11 JANUARY 2006;  
ACCEPTED 20 JANUARY 2006; PUBLISHED 23 MARCH 2006

We focus on the integration of peer-to-peer strategies and simple object access protocol/extensible markup language (SOAP/XML) for inter-domain, user-driven provisioning in an automatic switched optical/generalized multiprotocol label switching (ASON/GMPLS) network. To the soft-permanent connection (SPC) defined in ASON networks, a user-management interface is added to allow direct negotiation between the users and the management system. Moreover, the management system combines SOAP/XML and simple network management protocol (SNMP) to communicate with a GMPLS control plane that is responsible for rapid connection establishment, including routing and resource management. The management system is distributed and integrates a communication interface (the internal management interface, I-MI) based on SOAP and on the peering strategy Chord to perform provisioning related actions. Experimentation results with an ASON/GMPLS testbed assess potential suitability of the overall approach for service provisioning due to its low delays and high flexibility. © 2006 Optical Society of America

*OCIS codes:* 060.4250, 060.4510.

## 1. Introduction

The historical context in which optical networks have been designed and deployed provides much appreciation for why new requirements, such as efficient, timely provisioning, and services (on-demand, quality-of-service (QoS) enabled) are prevailing challenges today. New network requirements invalidate the assumptions upon which legacy networks were founded. Today's communications networks were designed primarily for private lines and voice service using circuit switching. Capacity was partitioned as  $N \times 64$  kbits/s ( $N$  times the size of an uncompressed voice channel) using multiple layers of hierarchy. Service deployment typically took several months, which was an acceptable time frame because the traffic demand was quite predictable and assumed to remain static for years at a time. The "traditional" building elements of optical networks procured by carriers were, and still are, data transmission services, routers, and terminals, and therefore the function of the network was to provide point-to-point connectivity. As the business case for providing data services became attractive, service providers retrofitted their network typically by yet another layering of protocols to support multiple data-service interfaces and networks, such

as asynchronous transfer mode (ATM) or Internet protocol (IP). Now, new building elements such as dark fibers, optical transmission equipment, or advanced switches result in the possibility of providing a new network function: end-to-end optical circuits. Such circuits can lead to advanced services such as bandwidth on demand and optical virtual private networks, provided services can be deployed in a rapid way.

A new technology that will simplify service provisioning is the optical control plane, which represents a common set of control functions and interconnection mechanisms that allow for unified communication, routing, and control across disparate types of underlying transport technologies [IP, ATM, synchronous optical network/synchronous digital hierarchy (SONET/SDH), and dense wavelength division multiplexing (DWDM)]. In the circuit-switching paradigm, the automatic switched optical network (ASON) standard defined by the International Telecommunication Union (ITU) in late 2001 [1] describes the set of control plane components that are to be used to manipulate transport network resources in order to provide the functionality of setting up, maintaining, and releasing optical connections. The ASON control plane can make use of generalized multiprotocol label switching (GMPLS) [2], defined by the Internet Engineering Task Force (IETF), which is an extension of multiprotocol label switching that allows for rapid packet and spatial circuit switching.

In parallel with the advances on the control plane, user-empowered networks are becoming a broad phenomenon. Examples of this are the wide deployment of wireless fidelity (Wi-Fi) or the extension of Ethernet beyond the local area network, which is resulting in the installation and operation of broader-reach computer networks. In the optical networking context, this phenomenon is known as customer-empowered fiber (CEF) networks. CEF networks are becoming a reality due to the access to dark fiber resulting from the liberalization of leased line provisioning. New fiber providers are arising among companies that deploy railways and highways, and those that distribute power, fuel, or gas, as well as cable television companies. A major reference in the research field of user-empowered optical networks is the Canadian Advanced Network for Research, Industry, and Education (CANARIE). CANARIE was the first to identify the key challenge of customers having total visibility of the network. This means that a user requesting a service could be the one to negotiate with the separate, independently managed networks involved, without using inter-domain services, which assume carrier-to-carrier signaling [3]. In CANARIE, no control plane is considered as a must for the service provisioning process. Instead, user management and control of network resources, called user-controlled lightpath provisioning (UCLP), is done directly via management in the form of a web services architecture, as well as with intercarrier routing, such as the standard border gateway protocol (BGP).

Nowadays, service delivery in the Internet across different network provider (carrier) domains usually adopts a hop-by-hop, cascaded model for the interactions between carriers both at the application and network (IP) layers, which involves BGP to a great extent. Within a single carrier, networks are typically partitioned to promote scaling of the intradomain routing protocols or to demarcate administrative or vendor boundaries. According to ITU Recommendation G.805 [4], a management domain defines a collection of managed objects that are grouped to meet organizational requirements according to geography, technology, policy, or other structure, for a number of functional areas, or for the purpose of providing control in a consistent manner. Management domains can be disjoint, contained, or overlapping. As such, the resources within an administrative domain can be distributed into several possible overlapping management domains. The same resource can therefore belong to several management domains simultaneously, but a management domain shall not cross the border of an administrative domain. The internal details of a domain are usually not shared externally, or they are shared to a very limited extent. Therefore, deploying services across multiple domains within a carrier has traditionally been centralized (Fig. 1). With the advent of new users and services, and with the trend toward easily “pluggable”

and highly dynamic optical elements, peering strategies in intracarrier inter-domain provisioning are becoming attractive.

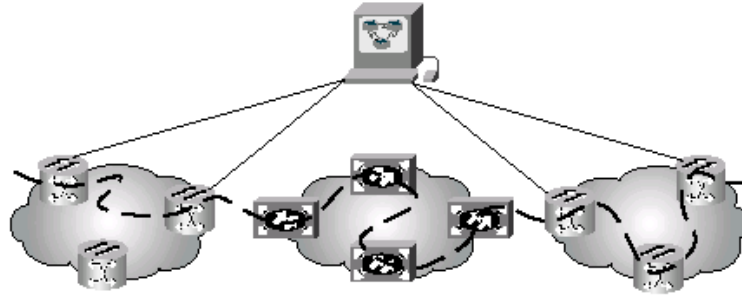


Fig. 1. Service provisioning across multiple domains.

To facilitate user-driven provisioning requests, users or applications shall be offered management, control, or even ownership of network resources, from processing and storage capacity to bandwidth allocation, within one or more domains. User-driven provisioning of an optical service encompasses combining existing and emerging technologies, adapting optical network elements and making protocols and communication mechanisms cooperate, the goal being that a user, who may be agnostic to optical network resources and management, can “point and click” a service. In this work we propose to integrate peer-to-peer strategies and simple object access protocol/extensible markup language (SOAP/XML) for inter-domain user-driven provisioning in an ASON/GMPLS network. The remainder of the paper is organized as follows. Section 2 describes the framework of the distributed management system that performs user-driven provisioning and inter-domain communication among peer management systems, called the distributed optical manager (DOM). Section 3 deals with design and implementation issues of the DOM application and inter-domain peering (integrated management interface, or I-MI). In Section 4 we outline the experimental performance of the DOMs. In Section 5 we draw conclusions.

## 2. Distributed Management System

The two main features of the distributed management system proposed are the enabling of user-driven provisioning and the integration of peering strategies oriented toward provisioning. Figure 2 illustrates this framework. Steps 1 to 4 (user request) are described in Subsection 2.A. Step 5 is the peering process among DOMs to provide an inter-domain service and is described in Subsection 2.B. Steps 6 to 8 encompass the forwarding of the request to the GMPLS-based control plane, and the signaling of the connection by GMPLS (ASON soft-permanent connection as defined in ITU-T Recommendation G.8080 [1]). Step 9 is the response of the ASON/GMPLS network to the user, and Step 10 is the transmission of data by the user equipment. In this context, the end-to-end setup delay is the time elapsed between Steps 1 and 10, whereas the network setup delay is the time span between the reception of a user request (Step 4) by a DOM and the sending of a response to the user (Step 9).

### 2.A. User-Driven Provisioning

The main advantages of the UCLP concept are both the enabling of users to control interfaces on network elements for the purpose of establishing end-to-end connections across optical networks, and the provision of a flexible definition of functionality and services.

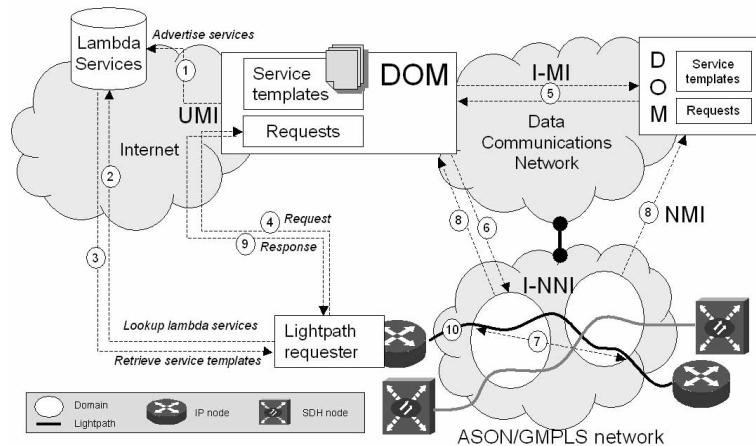


Fig. 2. User-driven service provisioning across multiple domains.

UCLP uses SOAP and the web services description language (WSDL), which are an open standard for signaling, communications, exception handling, and service definition. On the reverse side, UCLP software is required in all involved elements, which may not be the case for already-deployed networks, i.e., carriers. In this work, we put together the principle of empowering users, brought about by UCLP, and the capabilities of the emerging control plane (carrier perspective). This involves combining management and control in such a way that users can drive provisioning in a fast, automatic way, using the powerful functions of the ASON control plane, which we consider as the best of the UCLP and ASON frameworks. A description of the main features of our approach follows.

- **Advertising optical services:** If users are to request services autonomously, they should know which services are offered by the network. A way of doing this is advertising. This corresponds to Step 1 in Fig. 2, in which the DOMs are responsible for advertising the services available in the network through service templates, such as setting up and tearing down wavelength connections. Templates also include the service level agreement (SLA) the users must adhere to, which will result in a given service quality.
- **Triggering the provisioning process:** Whenever users need to set up an optical service, they follow Steps 2 to 4 depicted in Fig. 2. First, they look in the database for the available optical (lambda) services. This database can either be centralized or distributed. Then, they retrieve a template of how they should request the service. Finally, they send a request, according to this template, to a DOM. For example, in the web services architecture, these steps are publishing the web service (publish), retrieving the WSDL definition, and invoking the web service (bind).

In Fig. 2 we observe as well that users may retrieve templates for requesting services (Step 3). Figure 3 illustrates a template for setting up a transparent optical connection, implemented in this work [5]. Note that the object `ExplicitRoute` is optional, because the control plane performs routing and wavelength allocation. Retrieval can be done by using the dynamic, their static code paradigm, or a combination of both. In the dynamic paradigm, code is sent from one location to another to be executed. Examples are Java applets (code sent from a server to a client) or intelligent agents (agents decide where to be sent, whether client or server). In the static paradigm, code is functionality in a certain location (usually a server) and can be accessed by clients

using proxies. Static and dynamic paradigms can complement each other, depending on the sense (latency or cost, for instance) of sending amounts of code. For example, a common object request broker architecture (CORBA) server implementing a database management system would not send the entire database code to a client, but instead a Java applet implementing the user interface functionality that subsequently would access the database. In this work we use the dynamic paradigm.

```

<s:complexType name="SetupRequest">
  <s:complexContent mixed="false">
    <s:extension base="s0:Request">
      <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="TrafficType" type="s:int" />
        <s:element minOccurs="1" maxOccurs="1" name="RequestID" type="s:int" />
        <s:element minOccurs="1" maxOccurs="1" name="DstClientID" type="s:int" />
        <s:element minOccurs="1" maxOccurs="1" name="ContinuousLsp" type="s:int" />
        <s:element minOccurs="1" maxOccurs="1" name="BidirectionalLsp" type="s:int" />
        <s:element minOccurs="1" maxOccurs="1" name="Peakrate" type="s:int" />
        <s:element minOccurs="1" maxOccurs="1" name="SwitchType" type="s:int" />
        <s:element minOccurs="1" maxOccurs="1" name="Enc" type="s:int" />
        <s:element minOccurs="1" maxOccurs="1" name="ExplicitRoute" type="s:int" />
      </s:sequence>
    </s:extension>
  </s:complexContent>
</s:complexType>

```

Fig. 3. WSDL-setup requests.

- **Interfaces:** We define three interfaces in the DOM. The user management interface (UMI) interconnects users with the “requests” function of the DOMs (Fig. 2). The user view of a distributed system may be on the protocol level, such as common management information protocol (CMIP) or simple network management protocol (SNMP), or the protocol may be hidden. In the latter, only object interfaces are exposed to the user, which results in defining objects regardless of underlying protocols, whereas the former defines a protocol that users must observe in order to communicate with the management system. Our approach is interface-centric, and hence UMI hides management protocols to users, offering a higher level of abstraction. Therefore, the UMI requires a universal access method, such as SOAP. The management interface (internal, I-MI, and external, E-MI) interconnects DOMs belonging to the same (I-MI) or different networks (E-MI). Section 2 describes the I-MI, whereas the E-MI encompasses intercarrier communication, which is beyond the scope of this work. Finally, the network management interface (NMI) serves as a communication mechanism between the management system (DOMs) and the management agents located in optical network elements (of the control and transport planes, that is, NMI-A and NMI-T of G.8080 [1]).

## 2.B. Peer-to-Peer Interdomain

Figure 4 illustrates the reference points and domains of a control plane enabled optical network scenario for a model based on a client–provider relationship, where the client is a user and the provider is a network operator (carrier). The clouds in the figure illustrate the control plane, which has an interface to the management plane (NMI), another with the users (user–network interface, UNI) and another with other control planes (network–node interface, NNI) within a carrier (internal or external intracarrier, I-NNI and E-NNI, respectively) or between carriers (intercarrier E-NNI). The ITU G.8080 standard defines as well an interface between the control and transport planes, known as the connection controller interface (CCI) [1]. Moreover, Fig. 4 includes user–provider (UPI) and provider–provider (PPI) interfaces in the management plane. Compliant with this scenario, this work

proposes a distributed architecture for the ASON management plane, through designing an intra-autonomous system (intra-carrier-like) interface within the management plane (I-MI), in which the DOMs are peers, in a similar way to the I-NNI within a distributed control plane, which does not appear in Fig. 4. The traditional “client” is what we name the user, and the “provider” is the optical network. A user is then a client that may own part of the infrastructure, such as CEF. Therefore instead of UPI, we consider universal user and management-system interfaces, which we name UMI and I/E-MI, respectively, as described in the previous subsection.

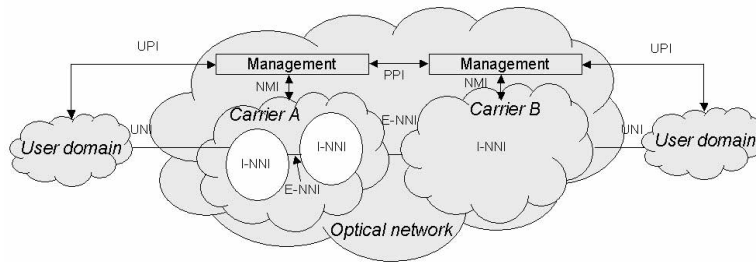


Fig. 4. Reference points in an optical network.

The rationale behind the management flat structure proposed for peer-to-peer provisioning across multiple domains is illustrated in Fig. 5. On the left side, the traditional management approach used in optical networks is shown. A top-level manager (UM) delegates management operation to midlevel managers (M), which are usually domain managers (element management system, EMS). These midlevel managers, in turn, manage optical devices (WDM and SDH equipment), which have embedded management agents. Usually, such agents do not cooperate with one another, but perform queries from the top- or midlevel managers, and produce alarms and notifications to higher-level entities. Note that agents may be delegated (DA) if they act as a “proxy” to other management technologies, such as converting SNMP requests to Transaction Language 1 (TL1) messages. Figure 5 (right) depicts our approach; the DOMs are comparable to traditional midlevel managers, and agents are embedded not only in optical (active) hardware, as done today, but also in control plane elements. Moreover, managers and agents cooperate. Since distributed management technologies, such as CORBA, Java remote method invocation (RMI), or open distributed management architecture (ODMA), among others, are used in a hierarchical approach, current management planes are mostly based on vertical delegation. As depicted in Fig. 5 (left), there is little horizontal delegation. In contrast, our approach encompasses horizontal delegation, as well as cooperation among agents. Therefore, the architectures and technologies to be used must follow cooperative paradigms [6].

One may argue that our approach keeps the manager-agent paradigm, and that by doing so, the approach is rather hierarchical. We include the manager-agent paradigm in our work because it is the dominant approach in current optical networks, and one of the goals of our design is to provide a smooth migration to future optical networks, while introducing new concepts to cope with emerging users, scenarios, and applications that did not exist in the past. Moreover, we enforce horizontal delegation and cooperation, and we eliminate vertical delegation as much as possible, taking the manager-agent paradigm toward distribution. Note that although at first sight the interworking of DOMs may resemble the typical network or domain management systems relation, there is no hierarchy among them.

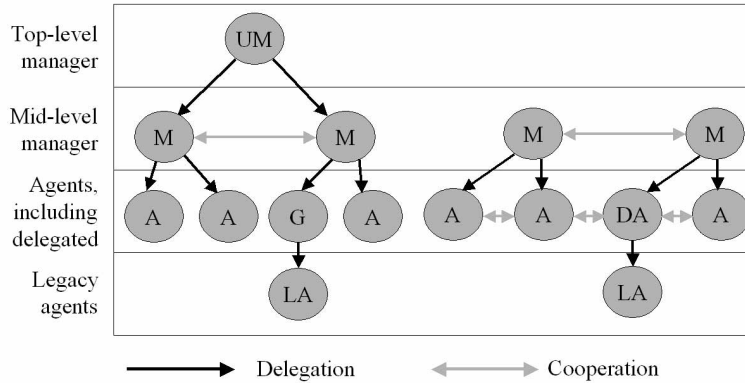


Fig. 5. Traditional versus proposed management.

### 3. Distributed Optical Manager

The distributed management system proposed in this work is composed of management agents located in relevant elements and DOMs that perform as control plane, transport plane, and resource managers, and that “proxy” with user-driven requests. In this section we describe the main building blocks of the DOMs, as well as their behavior with respect to the functionalities of an ASON management system [1, 7].

#### 3.A. Design Challenges

Each DOM gathers information of a network partition, such as an administrative domain, and interacts with other DOMs through the I/E-MI interface. Moreover, the DOMs are a proxy between users (UMI interface) and optical nodes (NMI interface). The DOMs have three functional blocks; configuration includes the mechanisms necessary for triggering SPCs, as well as the interfaces I/E-MI, UMI, and NMI. In order to avoid duplication of functions, in this work the ASON management plane does not support call setup or tear-down and connection restoration functions. Information duplication is avoided as well by letting the DOMs have a high-level view of optical connections, such as the one illustrated in Fig. 6. Should they require further details about the network, they would query the optical elements through the NMI. This block contains as well configuration functions for the control and transport elements that are manageable, especially for initial and after-failure configuration.

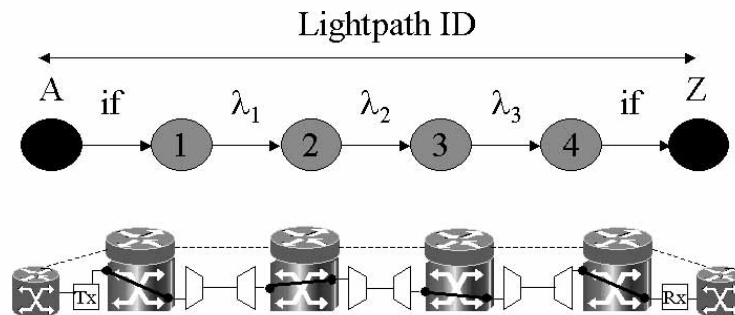


Fig. 6. DOM and control-plane views of optical services.

Monitoring and performance management have always been the realm of the management plane, and the ASON framework does not change this. In this work, the management plane performs service monitoring, through an online, nonintrusive monitoring system [8]. The DOMs are responsible for processing the notifications and alarms from the monitoring system and for informing the control and transport planes about reactive or proactive actions, if needed. In general, the control plane (or transport plane) should handle protection and restoration with little intervention from a management system. Currently, in the IETF organization, the control plane is being designed to provide efficient and timely recovery mechanisms in a distributed and automatic manner, so that a centralized information correlation via a management system is not required in order to carry out a successful recovery after network faults. The goal is to meet a stringent network restoration time and to prevent a single bottleneck from slowing down the overall recovery operation. Although in this approach a management system does not participate in the recovery operations, it can still correlate relevant protection and restoration data for other network management tasks. To this end, the DOMs have a fault logging block, which may also perform basic recovery in the event of severe control plane failures.

Note that since this work does not consider a hierarchical management system, there is no umbrella network management system (NMS) and several EMSs. Therefore, efficient interfacing among the DOMs (cooperation) is crucial for peering. This is so because there is no centralized entity that “sees everything,” but instead a series of distributed systems. Then, relevant information must be exchanged among them in order to ensure service provisioning. The following scenarios are taken into account for designing the I-MI:

1. **Request from a user belonging to the domain:** This is the simplest case, where a DOM acts like an EMS.
2. **Request from external user:** In this scenario, the user requesting a service is not recognized by the DOM (user not registered), which means that the user does not belong to the domain of the DOM or that the user is not registered in any domain. Here the DOM must apply security patterns to authenticate the user, and then perform the request or forward it to another DOM, which would permit the user to be registered. In Subsection 3.B we give an example of how DOMs search for information hosted by other DOMs.
3. **User (un)registration:** Since users may wish to have temporary “contracts” with the optical network, they can register and unregister from the management system. Users may be charged during the time they are registered or for the duration of their active connections.
4. **Notification of connection setup or teardown:** Through the I-MI, DOMs forward information about new and erased services, so that other DOMs have global information about the status of optical services.
5. **Status change of a resource:** Similarly, the monitoring system forwards information about relevant changes in the status of resources to the DOMs, in order to avoid requests involving these resources, which would be rejected.

### 3.B. Implementation

Figure 7 illustrates the scheme of the design modules of the DOM application. SnmpEngine manages SNMP messages (NMI interface); LocalDatabase contains information of the network, users, and lambda services; and Requests contains setup and teardown requirements from the UMI and I-MI interfaces. The module LocalDatabase contains the information of established lambda services, the users “known” by the DOM, and the free identifiers for the

agents in the DOM domain. The functions of SnmpEngine are the dispatching and processing of SNMP messages. The Requests module contains the classes for performing lambda service requests and responses as described in Section 2, which are Request, Reponse, SetupRequest, TeardownRequest and StatusRequest. Figures 8, 9 depict the class diagrams of LocalDatabase and Requests. Different instances of the DOM application, programmed in C#/.NET, run on different machines to form the distributed management system.

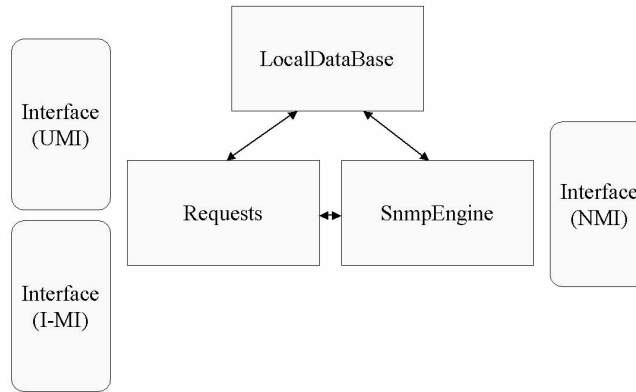


Fig. 7. Functional blocks of the DOM application.

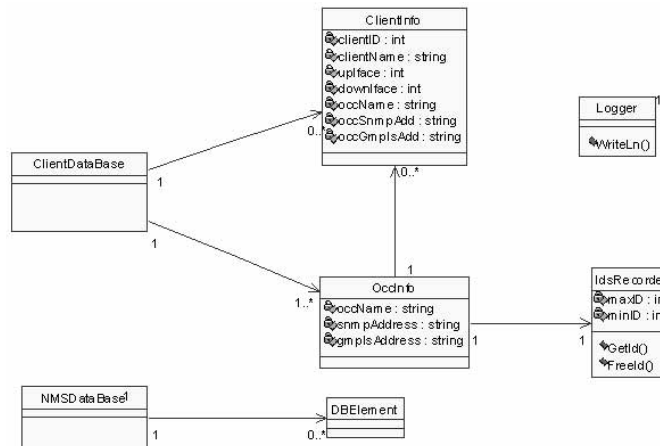


Fig. 8. DOM class diagram of LocalDatabase.

The NMI is based on SNMP [9]. Basically, to request an SPC [1], a DOM sends a Set message [10] to the management agent located in the head-end GMPLS node, which contains the SetupRequest or TeardownRequest objects illustrated in Fig. 9. The UMI, based on SOAP, uses WSDL to define service templates, such as the one depicted in Fig. 3 (setup). Architecturally, the I-MI interface is similar to the UMI and also uses WSDL as the interface description. In order to define the elements in WSDL, we considered the scenarios described in Subsection 3.A. Case 1 is a typical user-driven request, so no I-MI messages are needed. Case 2 encompasses forwarding the request to other DOMs, which allows the requesting user to be registered. There are several options to forward requests, with flooding and sequential forwarding being the most common ones. In principle, cases 3 and

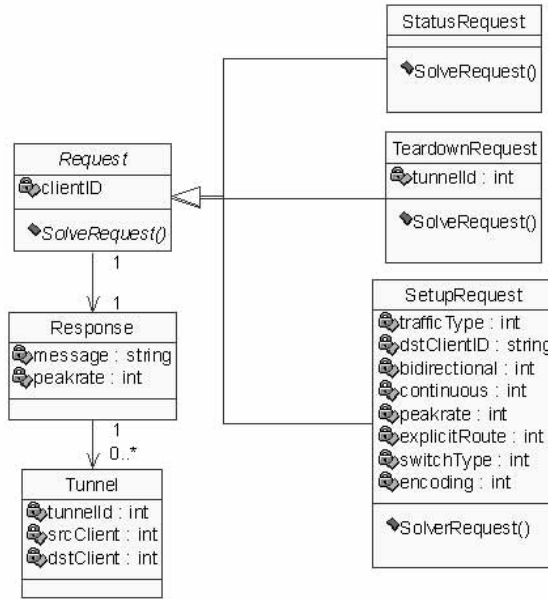


Fig. 9. DOM class diagram of Requests.

4 are flooding-like, such as in routing protocols, because the information involved is to be stored in the local databases of the DOMs. Case 5 is similar to case 3, and involves the NMI interface.

In order to forward information among the DOMs, two main alternatives were studied: flooding and distributed hash tables, in particular, Chord [11], which is a distributed look-up protocol for peer-to-peer applications. The main advantage of Chord is that the number of operations required is logarithmic with respect to the number of nodes (DOMs) in the management network. Another advantage is its good behavior in the event of changes in the number of nodes. Chord builds up a circular list of nodes (ring), ordered by their identifiers, which are calculated by applying a hash function  $H$  to the address of each node. A node (DOM)  $k$  in the management network stores the keys of the objects whose identifier is in the range  $[\text{identifier}_{k-1}, \text{identifier}_k]$  in the ring. This way, in order to locate an object from a key, we only need to find the successor in the list ( $\text{successor}(k)$ ). To calculate the identifier of an object, Chord uses the  $H$  function.

The basic problems in a distributed system are the bandwidth needed for forwarding information and the convergence time. In a flooding strategy, such as the open shortest path first protocol, the DOMs would “know” the status of the network by sending messages massively in order to locate nodes that respond (hit) the DOM’s queries. The advantage of flooding is that there is no need for a centralized element or static lists in the DOMs. However, flooding results in greedy use of bandwidth (in the context of this work, the IP control channel of the data communications network), it needs a mechanism to limit the number of messages flooded, and it is difficult to scale. On the contrary, Chord can make use of a name service, which is a mechanism that, when given an object (name), returns the address of the resource that contains it. In the Internet, the domain name service consists of a hierarchical system. This is the approach considered in this work. In short, the main advantages of Chord are the following:

- **Decentralization:** There is no need for centralized servers, and each “node” is in the same level (M in Fig. 5 right).

- **Availability:** Although the system may experience changes, for instance, due to failures in communications or changes in the nodes, identifiers can always be located.
- **Scalability:** The cost of a search grows logarithmically with the number of nodes in the Chord network.
- **Load balance:** The hash function distributes the identifiers uniformly.
- **Low bandwidth:** When a DOM has to upload a user with a given name, it applies the hash function and obtains the key. Then, it registers the key in the corresponding node (DOM), together with the user address ( $\text{Key}_{\text{user}}, \text{Addr}_{\text{user}}$ ), so that the user can be located by simply coding the user's name and querying the DOM that uploaded it. Note that since the DOMs are based on web services, universal resource identifiers are used instead of IP addresses. Also note that although the hash function might issue the same key for two different names, there is a statistical guarantee to avoid this, which depends on the length of the key used. Otherwise, the DOMs could store the triplet  $(\text{Key}_{\text{user}}, \text{Addr}_{\text{user}}, \text{Name}_{\text{user}})$ . We assume that we can estimate the maximum number of names in the optical network, so that we assign a sufficient length to the keys.

In short, each DOM maintains a table of neighbor DOMs, known as a finger table. When a DOM performs a lookup on the peer-to-peer system, it will use its finger table to determine what DOMs to send queries to. Chord is designed so that the size of the finger table is of the order of  $O(\log(M))$ , where  $M$  is the number of DOMs. The Chord algorithm  $n.\text{find\_successor}(id)$  for mapping and retrieving keys is the following (ask DOM  $n$  to find the successor of an identifier):

```

if  $id \in (n, \text{successor}]$ 
  then return ( $\text{successor}$ )
  else  $\left\{ \begin{array}{l} n' = \text{closest\_preceding\_node}(id) \\ \text{return } (n'.\text{find\_successor}(id)) \end{array} \right.$ 

```

And the algorithm for searching the local table for the highest predecessor of an identifier  $id$  ( $n.\text{closest\_preceding\_node}(id)$ ) is

```

for  $i = m$  downto 1
  do  $\left\{ \begin{array}{l} \text{if } \text{finger}[i] \in (n, id) \\ \text{then return } (\text{finger}[i]) \end{array} \right.$ 
return ( $n$ )

```

where  $\text{finger}[k]$  is the first node on the Chord ring that succeeds  $(n + 2^{k-1}) \bmod 2^m$ , with  $1 \leq k \leq m$ ,  $\text{successor}$  is the next DOM on the identifier ring ( $\text{finger}[1].\text{node}$ ), and  $\text{predecessor}$  is the previous DOM on the identifier ring.  $m$  is the number of bits used for the identifier. Reference [11] provides an example of the behavior of Chord.

A Chord ring was built with the DOM identifiers, as well as the keys of management objects (users and lambda services). Each key was assigned to the DOM whose identifier was equal to or bigger than the key ( $\text{successor}(k)$ ). This  $\text{successor}(k)$  is the first DOM on the ring starting from the key  $k$  clockwise. Note that the use of the finger table avoids the worst case of the algorithm  $n.\text{closest\_preceding\_node}(id)$ , which is to look up in all the DOMs, because it would take a time linearly growing with  $M$ . This way, when a DOM receives a request to locate a key, it looks up in its finger table the DOM that has the bigger identifier that is smaller than the key to search, and forwards the request to that DOM through the I-MI. Since the number and location of the DOMs may vary with time, in order to ensure

that the lookups always return correct keys, we use a stabilization protocol that updates the values in the finger table, as well as the successor of the DOM. This protocol consists of the functions Create(), Join(), Stabilize(), Notify(), FixFinger(), and VerifyPredecessor(). When a DOM is activated, it tries to locate the Chord ring. If there is none, the DOM creates one with its function Create(), with no predecessor and with itself as successor, so that other DOMs can join the ring in the future. Otherwise, the DOM will join one of the DOMs already registered on the ring, and will notify the successor (using the function Notify()) that this DOM will be its new predecessor. The function Join() is as follows:

```
Join(Node belong) {
    predecessor = NULL;
    successor = belong.find_successor(id);
    successor.Notify(thisNode);
}
```

Note that a DOM joining or leaving the ring is the most particular situation, when it comes to searching for successors right after a change, because Chord is based on the fact that each node on the ring “knows” its successor. To cope with this scenario, each DOM may store a list of  $n$  successors, instead of only the following successor. If the probability of a DOM failing (leaving the ring) is  $p$ , the probability of all successors failing is  $p^n$ ; then, the higher  $n$  is, the lower is the probability of not finding a successor. If we use  $n > 1$ ,  $n.closest\_preceding\_node(id)$  will have to be modified accordingly. Obviously,  $n > 1$  results in a faster search of a successor. Note as well that the Join() function is also used by a DOM with identifier  $i$  to notify its successor of the keys smaller than  $i$ . The function Stabilize() is key to guaranteeing the proper behavior of the protocol. This function verifies, on a regular basis, that all the DOMs know their predecessor on the ring. The objective of FixFinger() is to verify that the values in the finger table are those of the DOMs present on the ring. Finally, for each DOM  $i$ , VerifyPredecessor() verifies that the predecessor ( $i$ ) is active and, if not, notifies the next candidate to be predecessor.

As for the data model, the I-MI interface was implemented by creating the class library InterDOMEngine, which was added to the functional blocks depicted in Fig. 7. InterDOMEngine basically manages the Chord ring and registers and locates keys. The modules used by the UMI are used to send and receive SOAP packets through the I-MI that result from the use of the Chord algorithm. Figure 10 illustrates the classes of this library.

#### 4. Experimental Performance

Some of the scenarios defined in Subsection 3.A were tested in a laboratory ASON/GMPLS setting. The goal of these experiments was to assess the suitability of the approaches proposed for rapid setup of optical connections. Therefore, the goodness of the system’s performance was based on the setup delay; the lower, the better. Moreover, flexibility aspects about peering among DOMs were assessed qualitatively. Note that blocking probability is not taken into account because it is caused by lack of resources, which are managed by the control plane, and therefore the management plane only increases setup delay, not blocking.

##### 4.A. ADRENALINE Testbed

The ADRENALINE testbed (all-optical dynamic reliable network handling IP/Ethernet gigabit traffic with QoS) is a circuit-switched optical WDM network that deploys reconfigurable optical add-drop multiplexers (OADMs) based on various technologies and tunable lasers. In ADRENALINE, end-to-end lightpaths are set up and torn down dynamically and in real time by means of a GMPLS-based control plane and the distributed management

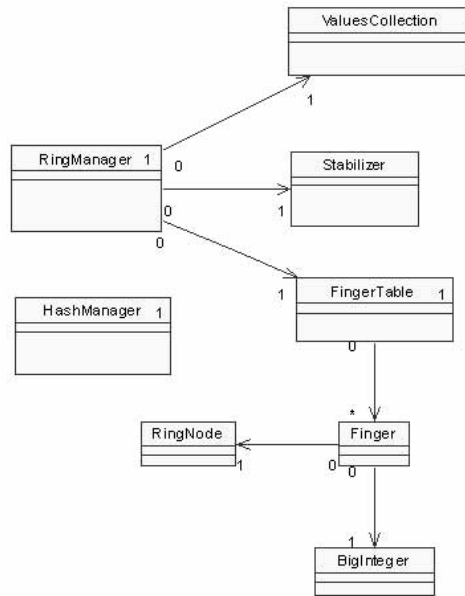


Fig. 10. Class diagram of the I-MI.

system presented in this work. That is, user-driven connections, or soft-permanent connections (SPCs), are established by combining user management of lightpaths and control plane technologies. ADRENALINE is a hybrid platform, developed at the labs of the Centre Tecnològic de Telecomunicacions de Catalunya, which combines both real and emulated optical nodes and links [12]. Control and management messages are based on fast Ethernet carried out-of-band ((f) in Fig. 11). The peculiarity of combining real and emulated links allows the dynamic configuration of several types of ring or mesh networks by simply switching cables and modifying the delays in the emulated links. In this work the topology used is the ring, with a constant link span of 35 km, and with nine nodes. Table 1 lists the main parameters of the ADRENALINE testbed.

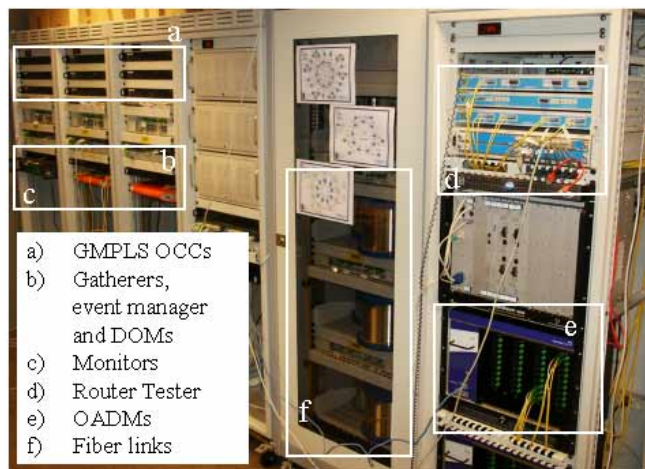


Fig. 11. Elements of the ADRENALINE testbed.

**Table 1. ADRENALINE Network Parameters**

Description	Value	Typical Value in Metro
Network diameter	105–315 km	up to 600 km
Number of nodes	3–9	up to 15
Number of wavelengths	8 per link	up to 16
WDM channel spacing	0.8 nm	20 nm (CWDM)
Optical channel speed	1–2.5 Gbits/s	155–622 Mbits/s (STM-1/4)
Control channel speed	100 Mbits/s	Depending on the DCN

The intelligence of the network is implemented through a distributed GMPLS-based control plane, with the resource reservation protocol (RSVP) and the open shortest path protocol (OSPF), both with traffic engineering and GMPLS extensions, for signaling and routing, respectively. GMPLS-based optical connection controllers (OCCs) are implemented in Linux PCs acting as routers [12] ((a) in Fig. 11). The transport plane used in this work is composed of nine  $4 \times 4$  OADM s configured by virtual divisions of micro-electro-mechanical system (MEMS) -based optical switches, which are remotely controllable and manageable by TL1 ((e) in Fig. 11). The infrastructure of the management plane consists of three Windows PCs that host the DOMs ((b) in Fig. 11). Moreover, SNMP agents are embedded in the Linux PCs hosting the GMPLS OCCs. A router tester is used to generate IP traffic ((d) in Fig. 11). Finally, ADRENALINE’s users are enabled with a SOAP application (“lightpath requester” in Fig. 2), which discovers the services offered by the optical network through a discovery agency entity, and communicates with the DOMs using the UMI.

#### 4.B. Interdomain User-Driven Provisioning

In order to verify the performance of the DOMs for user-driven provisioning we analyze the cases listed in Subsection 3.A. For case 1, ADRENALINE is organized as a domain with nine nodes. Figure 12 illustrates the end-to-end setup and teardown delays of user-driven connections within a single domain of the ADRENALINE network. Connection requests (SPCs) arrive according to a Poisson model, and holding times are exponential, with a mean of 5 s. The distribution of traffic is uniform among the nine nodes. The delays obtained, below 1.5 s for setup, are end-to-end network delays, that is, they include UMI, NMI, CCI (based on TL1), and GMPLS signaling delays (I-NNI), according to the scenario depicted in Fig. 2. The DOMs and SNMP agents embedded in the OCCs represent about 40 ms, and hence over 90% of the setup delay is due to interworking with the link resource management module hosted in each of the GMPLS-based OCCs, that is, routing and wavelength assignment procedures.

To study cases 2 to 5, we organize the ADRENALINE network in three administrative domains of three nodes each, in order to investigate the behavior of the interworking among different DOMs. The setup of this experiment is illustrated in Fig. 13. Three instances of the DOM application (DOM1, DOM2, and DOM3) were built, one for each of the domains depicted in Fig. 13. Once we launch the DOMs, they join the I-MI ring as described in Subsection 3.B. This process is called RingWalk and has the following outcome:

```
http://10.1.1.152/DOM/DOM.asmx
http://10.1.1.150/DOM/DOM.asmx
http://10.1.1.54/DOM/DOM.asmx
```

That is, the DOMs are reachable through the I-MI. If a user requests a connection establishment to a given DOM whose database does not contain it (case 2 in Subsection 3.A),

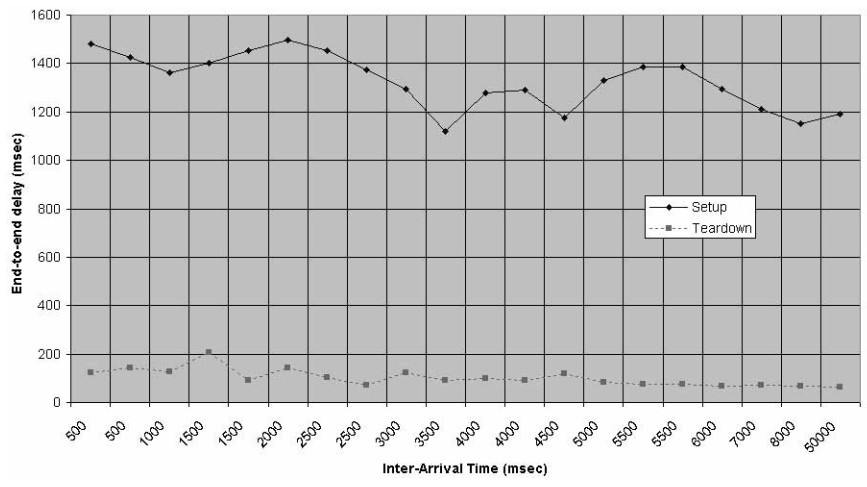


Fig. 12. End-to-end delays for user-driven requests.

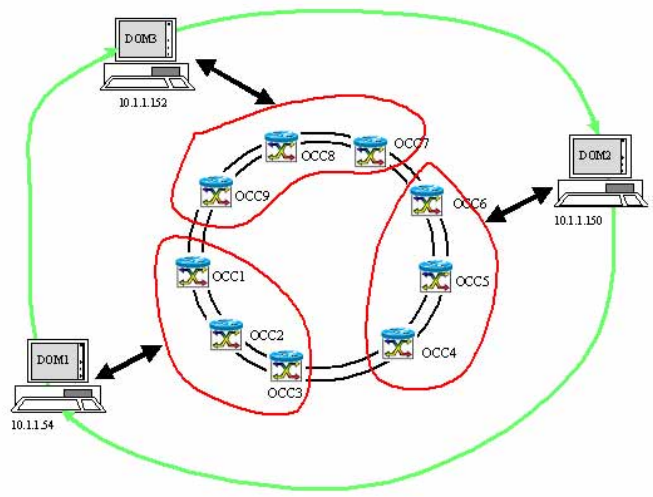


Fig. 13. Setting of the DOMs. The exterior line is the I-MI.

we add an extra delay to that of Fig. 12, which results from searching for this user or information about it, and optionally forward this request to the adequate DOM. Continuing with the previous example, a user requests a connection to DOM1 from user 0 to user 30. User 0 is attached to node 1, controlled by OCC1 (Fig. 13), and therefore is known by DOM1. User 30 is attached to node 8, and therefore belongs to another domain, that of DOM3. Then, DOM1 searches for this user in the I-MI ring (I-MI request):

```
PERFORMING I-MI REQUEST: 0-30
```

The I-MI request reaches DOM3 following the direction of the I-MI ring, depicted in Fig. 13. DOM3 stores the identifier of the user requesting the connection, so it forwards the request and sends back the information requested by the DOM1, which is the IP address of the destination OCC needed to send the SPC requests through the NMI, as described previously. This process takes a few milliseconds even in the worst case (two-hop search in the I-MI ring and forwarding of request), which results in the setup delay being augmented by 1%–3%.

The distributed creation and deletion of users (case 3 in Subsection 3.A) involves the creation and registration of keys, which is only part of the setup delay if an unauthorized user requests a connection, and the system must register the user first. Once the DOM that is contacted by the user verifies that this user is not registered in the Chord ring, following the steps described in the previous case, it must register the associated key. An example of this is the addition of a user in DOM1 after activating DOM3 (with IP address 10.1.1.152):

```
Registering key: 160567886100249482591293 from node  
http://10.1.1.152/DOM/DOM.asmx
```

This delay is the sum of the case 2 delay and the registration delay in the appropriate DOM, which may augment the setup delay by 3%–5%. Finally, cases 4 and 5 involve the registration of keys about connections and resources, which do not affect the setup delay. Chord is very flexible in this context, because any object can be translated into a key and then stored in the Chord ring, as described in case 3. In fact, the strength of Chord is the distributed hash lookup, which allows for this cooperative sharing and forwarding of information among the DOMs.

## 5. Conclusions

Control planes of next-generation optical networks can make use of GMPLS-related protocols and their extensions for optical networks, which raises novel management challenges. User-driven provisioning is arising in parallel to control plane research. With the advent of new optical components and technologies that allow for rapid provisioning of on-demand services, inter-domain aspects arise as an important issue. A universal and flexible technology that can be of interest in this context is SOAP/XML, which is emerging as a de facto standard to exchange information over heterogeneous systems. Results of combining SOAP/XML user-driven requests with ASON/GMPLS provisioning offers setup delays of less than 1.5 s. Applying peering strategies based on distributed hash tables to the management system only adds between 1% and 5% to the setup delay, and allows for distribution of the management system, which is an interesting approach for inter-domain provisioning.

## Acknowledgments

The authors would like to thank C. Herrero for his support in the development of the DOM application. This work is part of the NetCat research project, supported by the Centre Tecnològic de Telecomunicacions de Catalunya.

## References and Links

- [1] "Architecture for the automatically switched optical network (ASON)," ITU-T Recommendation G.8080/Y.1304 (International Telecommunication Union, November 2001).
- [2] E. Mannie, "Generalized multi-protocol label switching (GMPLS) architecture," IETF RFC 3945 (International Engineering Task Force, October 2004).
- [3] B. Saint-Arnaud, Jing Wu, and B. Kalali, "Customer-controlled and -managed optical networks," *J. Lightwave Technol.* **21**, 2804–2810 (2003).
- [4] "Generic functional architecture of transport networks," ITU-T Recommendation G.805 (International Telecommunication Union, March 2000).
- [5] C. Pinart, R. Muñoz, and G. Junyent, "Experimental implementation of distributed management for service provisioning in an ASON/GMPLS testbed," in *Proceedings of the 9th IEEE International Conference on Communications Systems* (IEEE, 2004), pp. 376–380.
- [6] J. P. Martin-Flatin and S. Znaty, "Two taxonomies of distributed network management paradigms," in *Network and Systems Management: Emerging Trends and Future Challenges* (Plenum, 1999), Chap. 3.
- [7] "Framework for ASON management," ITU-T Draft New Recommendation G.7718/Y.1709 (International Telecommunication Union, December 2004).
- [8] C. Pinart, R. Martínez, and G. Junyent, "Experimental implementation of dynamic in-service performance monitoring for lambda services," presented at the 31st European Conference on Optical Communications, Glasgow, UK, 25–29 September 2005.
- [9] C. Pinart and G. Junyent, "Experimental test of management integration in GMPLS enabled metro WDM networks for service provisioning," presented at the 30th European Conference on Optical Communication, Stockholm, Sweden, 5–9 September 2004.
- [10] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A simple network management protocol (SNMP)," IETF RFC 1157 (International Engineering Task Force, May 1990).
- [11] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. Frans Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for Internet applications," *IEEE Trans. Netw.* **11**, 17–32 (2003).
- [12] R. Muñoz, C. Pinart, R. Martínez, J. Sorribes, and G. Junyent, "ADRENALINE testbed: user management of lighpaths over intelligent optical WDM networks through GMPLS and XML," in *First International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities* (IEEE, 2005), pp. 252–261.