ALMA MATER STUDIORUM — UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA
IN
Ingegneria Elettronica, Informatica e delle Telecomunicazioni

Cicle XXI
Disciplinary Sector: ING-INF/03

DESIGN OF OPTICAL NETWORKS FOR ADVANCED APPLICATIONS

Candidate

Dott. Ing. ALDO CAMPI

Coordinator:

Chiar.mo Prof. Ing. PAOLA MELLO

Supevisor:

Chiar.mo Prof. Ing. GIORGIO CORAZZA

Co-Supervisors:

Prof. FRANCO CALLEGATI

ACADEMIC YEAR 2007–2008

Contents

1 Introduction

Nowadays, computing is migrating from traditional high performance and distributed computing to pervasive and utility computing based on heterogeneous networks and clients. The current trend suggests that future IT services will rely on distributed resources and on fast communication of heterogeneous contents. The success of this new range of services is directly linked to the effectiveness of the infrastructure in delivering them. The communication infrastructure will be the aggregation of different technologies even though the current trend suggests the emergence of single IP based transport service. This concept is formalized in the ITU-T Next Generation Networks (NGNs) [KNT05], which defines a service stratum responsible of network service provisioning and a transport stratum responsible of IP based packet forwarding. In summary, future network architectures must be able to offer complex and converged services, which integrate the traditional QoS-enabled connectivity (a matter of network resources) and novel features, such as the handling of non-network resources (storage and computing), with the ability for user to select and invoke suitable services among the offered ones in on demand fashion.

Optical networking is a key technology to answer the increasing requests for dynamic bandwidth allocation and configure multiple topologies over the same physical layer infrastructure, optical networks today are still “far” from accessible from directly configure and offer network services and need to be enriched with more “user oriented” functionalities. However, current Control Plane architectures only facilitate efficient end-to-end connectivity provisioning and certainly cannot meet the aforementioned future network service requirements, e.g. the coordinated control of resources.

The integration and management of resources and services within distributed, heterogeneous and dynamic environments is a key issue. NGN addresses the management and creation of network services, in order to enable direct invocation of network resources with proper grade of service [SCLJ07]. At the same time the range of the aforementioned applications (for instance Grid computing) has similar management problems regarding distributed application resources. Unfortunately, networks and, in particular their control planes, regardless of the specific implementation, deal with topology, capacity, connectivity and routing, while application services deal with application resources. As a consequence the application and network worlds have very different perspectives and talk very different languages.

The vision in broad terms is to design a set of functions that close the gaps between the user needs and optical network resources. Today optical networks are controlled by operators thanks to control planes that “see” the network resources (for instance GMPLS [Far05]) while the more advanced IT services are developing specific languages to express the needs of the application instances (for instance JSDL in grid networks [ADF$^+$05]). These logical layers are not communicating to date and service provisioning is not available “on-the-fly” at the user but is a rather “static” operator task. So far, a direct link between services and optical network control is still missing and the applications usually rely on virtual overlay topologies that provide connectivity according to a pre-defined scheme and are built a-priori in the network.

The overall objective is to provide the network with the improved usability and accessibility of the services provided by the Optical Network. More precisely, the definition of a service-oriented architecture is the enable technology to allow user applications to gain benefit of advanced services over an underlying dynamic optical layer. The definition of a service oriented networking architecture based on advanced optical network technologies facilitates users and applications access to abstracted levels of information regarding offered advanced network services. In this scenario, Service Oriented Networking plays an important role in order to close this gap investigating and designing a general purpose Architecture capable of doing (i) network and IT resource virtualization, i.e. support of a high-level interface to access these resources (ii) service abstraction, i.e. translation of a high-level interface into a low-level (technology-dependent) interface (iii) service composition for the delivery of complex services to both the service provider and the final user.

To this purpose, protocols and languages play a fundamental role in the aforementioned architecture. In particular, this work has been focused on the use of the SIP protocol as a inter-layers signalling protocol for defining a Session plane in conjunction with a description language. The study of the SIP protocol has been achieved under different uses profile. SIP network inherits embedded functionalities like Location, service profile, authentication etc. as well as many issues like NAT traversal, interoperability, security etc. During this thesis some of these issues have been faced studding novel SIP collaboration architecture like for instance H-SIP. However even if the study of SIP protocol is a part of the thesis work it is not fully reported in this document since it hasn’t been reached the necessary maturity to be published in this context.

On the other hand, an advantage optical network must accommodate high data bandwidth with different granularities. Currently, two main technologies are emerging promoting the development of the future optical transport network, Optical Burst (OBS [CM99]) and Packet Switching (OPS [YDM00]). Both technologies respectively promise to provide all optical burst or packet switching instead of the current circuit switching. However, the electronic domain is still present in the scheduler forwarding and routing decision.

Because of the high optics transmission frequency the burst or packet scheduler faces a difficult challenge, consequentially, high performance time focused design of both memory and forwarding logic is need. This open issue has been faced in this thesis proposing an high efficiently implementation of burst and packet scheduler. The main novelty of the proposed implementation is that the scheduling problem has turned into simple calculation of a min/max function and the function complexity is almost independent of on the traffic conditions.

1.1 Thesis organization

The thesis is organized as following. In chapter 2 the trend for deploying the Future Internet is presented focusing on ITU NGN architecture, which is the new standard under way, and the related optical transport infrastructure. Chapter 3 faces the transport data scheduling issues for OBS and OPS network proposing the implementation of a novel scheduling architecture.

Form chapter 4 the thesis starts dealing with the formalization and implementation of a Service Oriented Networking which is the service layer able to bring the application awareness into the network according with the next generation Internet needs. To this purpose chapter 3 defines the architectural models, the relevant building blocks and the protocols and languages needed. In particular, the Network Resource Description Language (NRDL) is presented in details in chapter 5 which is a novel description Language formalized during this thesis work. The NRDL is a novel tool proposed here in order to outline or negotiate the network activities in a Service Oriented Networking.

Chapter 6 concerns all the operations regarding the resources information exchange (i.e. publication and discovery) as well as resources reservation. Particular attention is given to resources reservation model of the session protocol chosen for the Service Oriented Networking, that is the SIP protocol. In the same chapter a deeply analysis of the SIP reservation mechanism with QoS shows the weakness of the current SIP management framework and consequentially a novel extended version of a resource management framework with QoS in SIP is argued. Chapter 7 reports the detailed results of some experimental activities conducted with two distinct Service Oriented Networking prototype implementations and finally the conclusion are drawn in chapter 8.

2 Future Internet

The international race to invent the next Internet generation or Future Internet is under way. The fast evolution of the Internet with over 1.5 billion users worldwide suggests the neede of increasingly forms for supporting more and more connection functionalities. The current Internet is a simple end-to-end transport network service able to interconnect different and etherogenious systems. The current Internet architecture is centred on a flat network layer that is being to route packets from diffrent location, without any delivery guaranty.

Becouse of the cuirrent simple paradigms, the network has promoted the push of complexity into the endpoints, allowing the Internet to reach an impressive scale in terms of inter-connected devices. However, while the scale has not yet reached its limits, the growth of functionality and the growth of size have both slowed down. It is now a common belief that the current Internet is reaching both its architectural capability limits and its capacity limits (i.e. in addressing, in reachability, for new demands on QoS, Service and Application provisioning, etc).

Although the current Internet has been extraordinarily successful because of the simple ubiquitous and universal means for communication and computation, many challenges with fundamental aspects are emerging. Many of these aspects were not easily foreseen at the beginning of the Internet deployment, but now a lot of issues are coming up. The very success of the Internet is now creating obstacles to future innovation of both the networking technology and the services that use it. This is obvious thinking about the transport protocol ossification that makes the introduction and deployment of new network technologies and services very difficult and very costly (e.g. IPv6). The current network trend has shown a static network layer with the a lack of supporting any facilities to build non-native functionalities.

Because of the huge internet successful, Trust management, security with privacy and data-protection mechanisms is needed. Internet is still missing of an adequate addressing scheme for supporting network mobility, services and devices where user identity and devive location are not mixed up in the same address.

Nowadays, more and more applications look into the network for facilities to support Quality of Service (QoS), Service Level Agreements (SLAs) and for large scale provisioning and deployment of both services and management services. In other words, there is the emerging of supporting higher integration between services and networks. This integration can be considered as fundamental brick for the addition of new functionality, including capability for activating a new service on-demand, network functionality, protocols (i.e. addressing the ossification bottleneck) and inherent network management functionality, specifically self-management functionality. In conclusion, the overall missing mechanisms aim to support a high efficiency networking with a strict separation of mechanisms and policies.

All these considerations come from the current service-aware networks trend. Service awareness itself has many aspects, the delivery of content and service where the optimisation of the network resources during the delivery is a fundamental characteristic for Quality of Service (QoS) applied with a predefined Service Level Agreements (SLA). Conversely, services themselves are becoming network-aware. Networking-awareness means that both services and network resources can be managed uniformly in an integrated way allowing the services to be treated within a network environments.

The design of both Networks and Services is moving forward to include higher levels of virtualizzation and composition. In order overcome the ossification of the current Internet and to achieve the objective of being service-aware and network-aware both service and network resources must be aware of the relevant environment conditions as well of there own state.

We belive that the Future Internets must be built as service-aware providing built-in funcionalities and architectural aspects able to drive communication resources and services to an enhanced awareness and decision capabilities enabled by aware-sensing.

2.1 Infrastructure

The new infrastructure consists in an enhancing evolution of the current Internet able to support configuration and deployment of any resources (i.e. including virtual resources) of both networks and services. This is the first step tovards a more sofisticated aforamentioned transport delivery service. In particular the network is composed by transport and service functional nodes:

- Core Nodes for the provisioning of high-speed, high volume traffic flows for data processing functions, including flexible control and management capabilities;
- Edge Nodes and Service Nodes for the programmatic provisioning of the transport and content resources needed to deploy wide-area services and new network functionality, including programmability of the network forwarding functions and flexible control and management capabilities.

These nodes are accomplished by virtualisation components which are able to virtualize resources involved in the creation and management of a virtual slice of network in

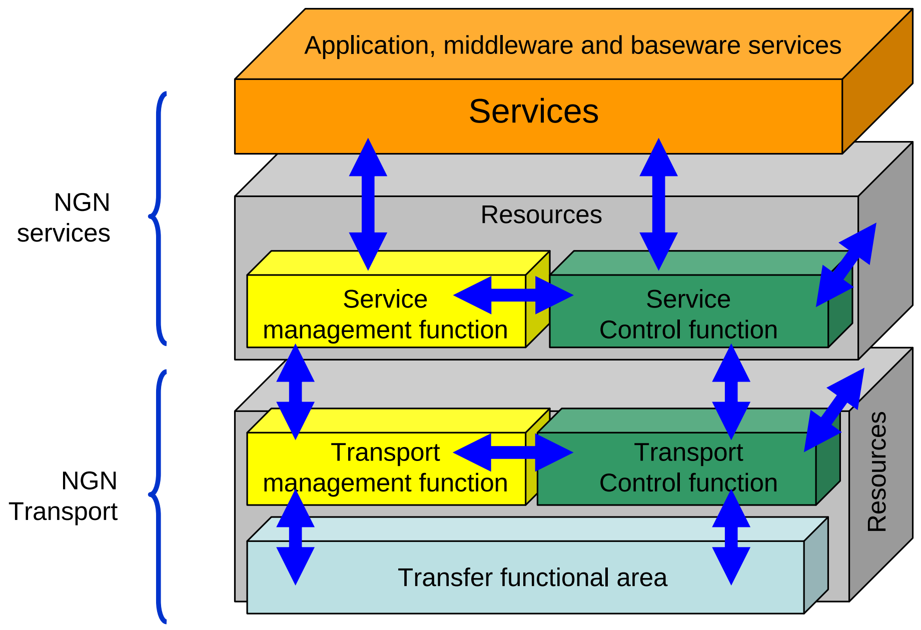

Application, middleware and baseware services
Services
NGN services
Resources
Service management function
Service Control function
NGN Transport
Transport management function
Transport Control function
Resources
Transfer functional area

Figure 2.1: Next Generation Network functional model

support of a service. This new globally infrastructure services results in a Context-centric networks approach where information are centric to the network and ubiquitous connectivity supports the infrastructure. As a consequence the Future Internet needs more powerful and flexible control mechanisms than the current transport network. These new needs push a new uniform open control frameworks to be scalable and dynamic by means of control compositions of both resources and systems.

To this purpose the fundamental step is the explicit decoupling of the control (i.e. basic routing, content-based routing, source-influenced routing and value added functions) and transport (i.e. forwarding) planes and new tuneable protocols for different layers of the protocol stack in support of cleaner cross-layer interaction and dynamic service composition. Consequentially, new mechanisms and interfaces to accommodate flexible and cost effective operations of service platforms over core and edge transport networks are required.

The current bulding block of the future Internet infrastructure are under way. The main wired transport tecnology is without any doubts the optical network which is able to accomodate high traffic bandwidth. About transport architacture, many efforts have been alredy spent in order to outline the new delivery platform. Focusing on ITU-T standardization, the most significant example is the Next Generation Network (NGN) architecture [JY02]. This new architectural approach tries to virtualize generalized transport objects (including L2 and L3, in addition to L1) through a standard framework for high-level triggering of network services.

2.2 Next Generation Network

Next Generation Network (NGN) is a packet-based network able to provide telecommunication services and able to make use of multiple broadband, QoS-enabled transport technologies and in which service-related functions are independent from underlying transport-related technologies. It enables unfettered access for users to networks and to competing service providers and/or services of their choice. It supports generalized mobility which will allow consistent and ubiquitous provision of services to users [KNT05][ooN04][raotNr06][Racfingn06].

NGN envisage provisioning of pervasive and utility computing to heterogeneous clients connected to whatever network infrastructure. Consequentially, the communication infrastructure will be the aggregation of different technologies under an ubiquitous IP based transport service. This new network architecture defines a service stratum responsible of network service provisioning and a transport stratum responsible of IP based packet forwarding, figure 2.1. This concept is a fundamental point in the NGN, which define a strong separationd between functional layers. In particular, the service stratum responsible of network service provisioning and the transport stratum responsible of IP based packet forwarding.

Moreover, the integration and management of resources and services within distributed, heterogeneous and dynamic environments is a key issue. NGN addresses the management and creation of network services, in order to enable direct invocation of network resources with proper grade of service [SCLJ07].

NGN architecture is composed by two sets of functionalities.

- Service Stratum whose main functional entity is the *Service Control Function (SCF)* which "...include resource control, registration, and authentication and authorization functions at the service level..." [ooN04] and trigger the network resources reservation by interacting with the Transport Stratum [KNT05]. The SCF is responsible for extracting service requirements from service signaling and sending a request to the Transport Stratum for network resource authorization and reservation.

- Transport Stratum provides the user functions that transfer data and the functions that control and manage transport resources to carry such data between terminating entities and provides IP-based connectivity services for the benefit of SCFs.

Service and transport stratum include a number of functional sub-blocks on which we will not deal here and having the goal to formalize all the various functions that must be performed to achieve the final goal of service oriented networking. Through the logical partitioning of functions into a Service Stratum and Transport Stratum, NGN is capable of abstracting the connectivity services by virtualizing the network resources.

2.3 Optical networking

Today, optical networking is the main wired transport tecnology and probably one of the most important research issues to ensure high networking capacity at a low price. Optical tecnologies have been used from access to core networks with different success. Due to the high costs of optical equipments currently the optical tecnologies are mainly used as static broadband transport infrastructure. Metro and backbone networks represent the two optical transport network mostly deployed as communication infrastructure.

At present, backbone networks exploit the large bandwidth made available by optical transport with static or quasi-static configurations, typically lacking scalability and flexibility. Current backbone segments are usually based on single fiber using WDM transmission equipment by means of multiple SONET/SDH rings interconnection [Pin02] [Ram02]. In this approach, which is called first-generation or opaque optical networks, optics is only used as point-to-point high-speed. In fact, optics is relegate as pure transport interconnection while all forwarding decisions are still made in elettronic domain. The state of the art of the optical networking is represented by the second optical networks generation.

The second generation or transparent optical networks consist of all optical meshed interconnection pattern. Data is routed in the optical domain without undergoing OEO conversions [CGK92] [CFZ96]. Optical Cross Connects (OXC) or Reconfigurable Optical Add and Drop Multiplexers (ROADM) are able to dynamically establish and tear down interconnections (called lightpaths) at the wavelength layer. In this scenario a management framework is need in order provide lightpaths dinamic configuration. In particular, Generalized Multi-protocol Label Switching (GMPLS) has been developed and standardized as management and control frameworks [Far05].

Currently most of the research on the next optical broadband network tecnology is involved in the third generation optical networks, which promise to develop all optics high-performance switching capabilities with different granularities. Optical Burst Switching (OBS) and Optical Packet Switching (OPS) network have been proposed as new solutions to provide respectively burst or packet switching intead of the cuuret circuit switching [CQY04] [CCR02] [Pat05] [EBS02]. Currently, these tecnologies are consided effective solutions for medium- and long-term next generation optical network.

The architecture of future optical core networks, as well as their integration in the existing networking environment will necessarily be hierarchical. It is not realistic to foresee a whole migration of the existing infrastructure to a single, homogeneous network. More likely networks of different kinds and based on different technologies will co-exist and co-operate, following the same architectural principle of the Internet, that is a network made of interconnected clouds, often designed in an optimized way according to specific applications and environment. Networks will be multi-service, multi-domain (i.e., under the control of different operators), multi-vendor, and multi-layer (i.e., built upon multiple technologies and protocols). Network is logicqally divided inmto two main planes: data and control plane. Both next-generation Control and data plane must face the challenges already outlined.

2.3.1 Control plane

The control plane (CP) is aimed at dynamically handling connection requests: in particular through the coordination of distributed network entities it guarantees fast and reliable connection set up, maintenance, and recovery in case of failure. Control plane is a crucial issue in the deployment of optical networks. Nowadays, the control plane for optical networks was based on the IP routing protocol and on a static service delivery where the control plane is responsible for control connections, disseminate connectivity information and the Calculate optimal path.

Both International Telecommunication Union (ITU) and the Internet Engineering Task Force (IETF) have spent some effort in order to define Control Plane architecture and functionalities. ITU has developed a control plane reference architecture called Automatically-Switched Optical Network (ASON) framework [G.801]. it retains a high level of abstraction and is expressed in terms of functional components and the interactions between them. This leaves the architecture open for application to a variety of network scenarios and actual control plane protocols.

At the same time IETF has undertaken a similar task, but with a different approach. The target is the development of a common distributed control plane platform, called GMPLS [Far05], able to jointly manage several data plane layers (e.g., the IP/MPLS packet layer, the SONET/SDH TDM layer, and the optical layer). However, starting from the actual Control Plane implementation and standardization the Next generation Control architectures have to evolve the control capabilities tovards service awareness functionalities.

Of course different networks have different control planes with different capabilities, thus the CP functionalities are strongly technology dependent, for instance a control plane for optical network is tailored to reliable high bandwidth and connection oriented data transmission, but lacks fast bandwidth provisioning while Wi-Fi is able to provide fast mobile access but with a limited reliable bandwidth segmentation. However, a first step to the configuration of networks of devices has been done by the IETF with the standardization of NETCONF [Enn06]. NETCONF has been proposed as homogeneous network configuration protocol.

2.3.2 Transport plane

An optical transport layer provides the basic mechanisms for reserving resources, and the service layer builds a set of services over the basic optical transport layer. There are multiple reasons to deploying an “intelligent” optical transport layer; advanced network flexibility in order to deal with the sharp emergence of new services and the constantly bandwidth increasing, flexible and re-configurable network elements are required to support efficient networking under unpredictable conditions.

Moreover, to realize efficient solutions to recover optical channels in case of network element failures the optical networking technology can enable complex node and management functions. Based on advanced management functions and automatic switching flexibility advanced resilience techniques like shared protection and restoration can be efficiently implemented. The recent initiative targeting a Global Environment for Networking Innovations (GENI) [iG06] supported by the National Scientific Foundation (NSF), recommend that network operators be able to configure multiple services over a common infrastructure, to expedite Internet evolution.

Currently, many emerging next-generation Internet applications, such as videoconferencing, multimedia distribution, are also characterized by high bandwidth requirements, multi-source and multi-destination communication paradigms, strict QoS requirements with respect to delay and loss, and high reliability and survivability requirements. In order to support these diverse requirements, optical networks can be used to manage resources in a flexible manner.

3 WDS Scheduling Algorithm Implementation

Burst Switching (OBS) [CM99] and Optical Packet Switching (OPS) [YDM00] have been considered by the research community as the most promising paradigms to increase bandwidth efficiency in IP over DWDM networks. Due to the statistical nature of optical burst/packet multiplexing, OBS/OPS nodes may be affected by contentions for the available switching and transmission resources, because of contemporary requests by bursts/packets. Therefore a contention resolution problem arises and cannot be addressed by means of queuing techniques as it would be in electronic networks.

A viable solution to this issue is to design algorithms that are able to proactively schedule the transmission of the bursts/packets, thus avoiding contention as much as possible. Such scheduling algorithms have similar characteristics when designed for OBS and OPS nodes, especially when OPS nodes must deal with asynchronous, variable-length packets. In both cases it is necessary to calculate the exact transmission time of the burst/packet just after the forwarding decision, planning the resource usage in order to minimize the chance of contention.

3.1 Related works

Many scheduling algorithms have been proposed so far in the literature, where the optimal target is to provide efficient resource utilization and minimum burst loss [FC07].

The very first one [S.99] defines a time horizon for each output wavelength, as the instant after which no burst occupies the channel. Any channel with a horizon smaller than the arrival time of the burst payload (or of one of its copies delayed by the FDL buffer, if present) is available to accommodate the incoming burst and the algorithm selects the channel with the latest horizon in order to minimize the bandwidth wasted due to voids. For this reason the same scheduling technique is also called Latest Available Unscheduled Channel (LAUC) [XVC00]. A major improvement in terms of performance is achieved by adding void filling capabilities to the Horizon policy, resulting in the Latest Available Unused Channel with Void Filling (LAUC-VF) algorithm [XVC00]. Already scheduled channels are now included in the search, given that they are unused for a period (i.e. void) starting before the (possibly delayed) payload arrival time and large enough to accommodate the entire incoming burst. The unscheduled channels are considered as a particular case of voids with infinite length. The algorithm selects the suitable void with the latest starting time, minimizing the gap left in front of the incoming burst so that the achieved throughput can be considered as an upper bound of the OBS node performance. Since LAUC-VF must keep track of all the voids in the output channels, the algorithm is computationally more complex and time-consuming than LAUC. However, other variants of void filling scheduling policies, such as MinSV, MinEV, BestFit [XQLX04] or HVF [MRZ04] achieve the same loss ratios as LAUC-VF with a significantly reduced complexity [CQY04], thanks to the use of efficient data structures and smart search techniques. The most efficient scheduler compared in [CQY04] has a complexity of $O(logK)$, where K is the number of scheduled bursts.

Recently, the hardware implementation of an OBS scheduler based on burst resequencing, which is able to achieve optimal scheduling in $O(1)$ complexity, has been proposed [CTM07]. This scheduler does not process burst headers immediately as they arrive. Instead, it delays and reorders them according to the respective payload arrival time. Then, by applying a simple Horizon policy, it is possible to schedule bursts that would have required a void filling algorithm in sequential header processing. However, this scheduler is applicable to the bufferless case only, since it is able to merely exploit the voids created by different offset times and not by burst payloads delayed by FDLs. Furthermore, due to the delayed header processing, data bursts need an additional latency.

The overall time required to perform the scheduling algorithm is not easily predictable and may turn to be large enough to make it a system bottleneck [CCXV99]. In order to limit the scheduling execution time, solutions based on a binary tree search have been proposed [XQLX03]. In order to implement a scalable OBS scheduler, the dependence of the algorithm execution time from the switch size should be as low as possible. This can be achieved adopting a parallel processing scheme, as the one described in a recent paper [CCC07] which presents a specific formulation of the scheduling problem and a simulation of a viable hardware implementation with the resulting response time. Nevertheless, all approaches previously described address the scheduling problem by searching fast and/or parallel algorithms, for processing one single burst header. However, headers are still processed sequentially, which brings two persistent drawbacks: (1) a sequential approach is greedy, (2) the system has to be dimensioned for a worst case situation, with a high number of headers to be processed in a short time period.

3.2 Contention resolution in OBS/OPS networks

At arrival time the burst control packet (BCP) or the packet header is analyzed by a processor to determine the forwarding path. Then the forwarding information is passed

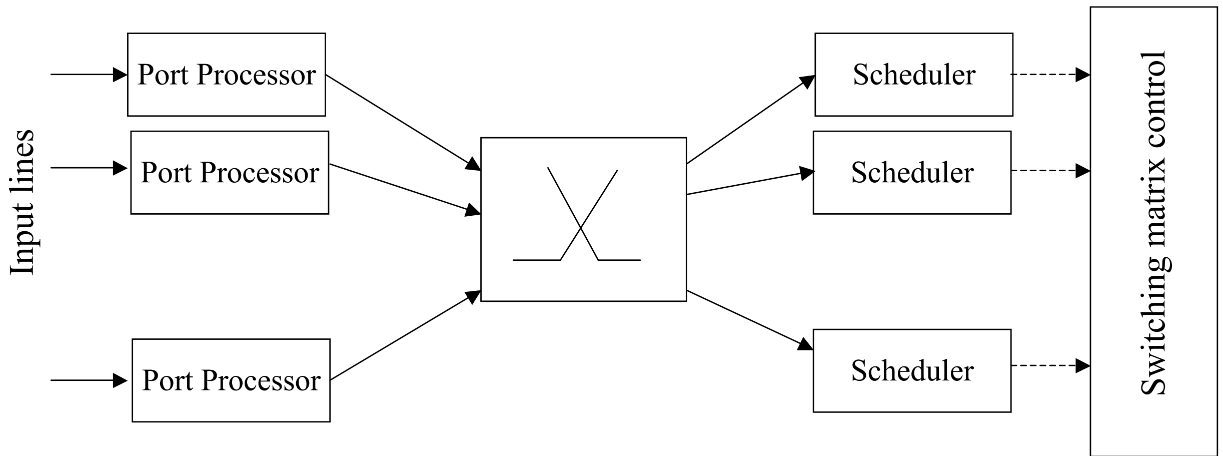


Figure 3.1: Block diagram of the control logic of an OBS/OPS switch.

to a scheduler that must schedule the burst/packet according to output channel conditions. Both functions could be centralized, but this is not the most effective solution in terms of computational efficiency. If processing speed has to be optimized, the functions can be parallelized, performing at the same time the forwarding operations at each input line and the scheduling operations at each output line. This is the solution suggested by the control architecture of an OBS/OPS node shown in figure 3.1. In the following we will focus on the concept and implementation of the scheduler.

It is assumed that the switching systems have to deal with variable-length data payloads arriving asynchronously and contending for the output ports. Because of contemporary requests for a given port, a contention resolution problem arises. To solve such issue the time, wavelength and space domains may be exploited. The approach to balance the load over the wavelengths of a fiber has been shown to provide a very significant performance improvement [S.99] [CMP$^+$98].

We place the focus on combining wavelength and time congestion resolution, exploiting full range wavelength conversion [S.J96] and some sort of optical buffering [HCAM98]. It is assumed that, once the decision on the output fiber has been taken, the scheduler takes care of contention resolution, by deciding:

- the wavelength on the designated output fiber that will be used to transmit the burst/packet, in order to properly control the output interface;
- the delay, if any available, that will be assigned to the burst/packet in case all wavelengths are busy at the arrival time;
- to drop the burst/packet if no wavelengths and delays are available.

The choice of the most effective wavelength for load balancing as well as the choice of the delay, if needed, are correlated and make the so-called Wavelength and Delay Selection

(WDS) scheduling problem, addressed by a suitable WDS algorithm [FWCP04]. This is a sort of optimization, where bursts/packets are scheduled in a given time window over a set of wavelengths. The time window is related to the number and length of delays available and the set of wavelengths to the degree of WDM implemented on the input/output fibers. In general the WDS algorithm may depend on the architecture of the switching node. This work does not focus on any specific node architecture, thus considering WDS algorithms that are applicable to any switching architecture that logically keeps a virtual queue per output wavelength.

It is well known that delay buffers put constraints on the time when a burst/packet may be ready for transmission. The WDS algorithm must schedule bursts/packets over the wavelength set according to such constraints. Let us call W the number of wavelengths per fiber and B the number of delays (including the zero-delay cut-through path), providing a discrete set of access points to the future transmission time framework with a first-come-first-served scheduling.[1]

The typical behavior of the WDS algorithm is to look for a possible scheduling time for a burst/packet, either by choosing the cut-through path if immediate transmission is possible, or by using the available delays. In an ideal switching matrix, where information units can be freely switched in space and wavelength, the instants at which a generic burst/packet arriving at time t may be transmitted on wavelength j are $s_{ij} = t + iD$, with $i = 0, \ldots, B-1$, $j = 1, \ldots, W$. Let us call $\mathcal{S}$ the set of such instants, $s_{ij} \in \mathcal{S}$. The cardinality of $\mathcal{S}$, i.e. the number of elements in the set, is then $|\mathcal{S}| = BW$.

However, depending on hardware and/or logical limitations, bursts/packets may be scheduled at times $s_{ij} \in \mathcal{S}'$, where $\mathcal{S}'$ is a subset of $\mathcal{S}$. Indeed $\mathcal{S}' = \mathcal{S}$ when full wavelength conversion is available at the switching matrix and all delays can be used by all bursts/packets. For the sake of simplicity, in the following it will be assumed that there are no limitations on the scheduling and therefore the algorithm uses the full set $\mathcal{S}$. However, this assumption does not affect the implementation proposed, that can be effectively applied to algorithms using $\mathcal{S}'$ as well.

The most important issue regarding WDS is that it may be impossible to schedule a burst/packet just after the end of the transmission of a previous one. Therefore some "gaps" between transmitted bursts/packets are created [TYC+00]. It has been shown that these gaps have a very detrimental effect on performance since, basically, they cause a sort of reduction in the output lines transmission capacity [F.00].

The WDS algorithms are based on heuristics that differ in the definition of the optimal value of $s_{ij} \in \mathcal{S}$, and can be classified as [XVC00] [CCC01]:

- delay-oriented algorithms (D type), that aim at minimizing the latency and therefore send a burst/packet to the wavelength requiring the shortest delay;

[1] Throughout this work the case of a degenerate delay buffer is considered [LLF99], i.e. delays are consecutive multiples of a given time unit D, ranging from 0, to D, $2D$, ..., up to $(B-1)D$. Nonetheless, the concepts presented here are applicable to different buffer arrangements as well.

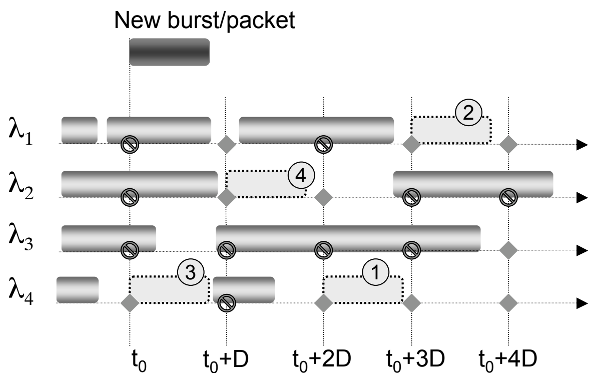


Figure 3.2: Example of WDS algorithms: (1) D type noVF, (2) G type noVF, (3) D type VF, (4) G type VF.

- gap-oriented algorithms (G type), that aim at minimizing the gaps between packets (i.e. maximizing the line utilization) and therefore send bursts/packets to the wavelength providing the minimum gap.

Moreover a WDS algorithm may or may not try to fill as much as possible the gaps between bursts/packets, with a technique known as void filling [TYC$^+$00]. Therefore WDS algorithms can be:

- D or G type without void filling (noVF), just exploring the scheduling times after the last scheduled burst/packet on each wavelength;
- D or G type with void filling (VF), exploring all scheduling times, including those between other scheduled bursts/packets, to see whether the newcomer may fit in.

An example of WDS scheduling is shown in figure 3.2. A burst/packet has to be scheduled at an output interface where other bursts/packets have already been scheduled according to the picture. The figure shows a simple example with $W = 4$ and $B = 5$, therefore $|\mathcal{S}| = 20$ and s_{ij} are all the cross-points between delays and wavelengths from t_0 onwards. It happens that 10 points out of 20 are currently busy (a transmission is already scheduled at that time) and 10 are available (outlined with the gray diamonds). The WDS algorithm is supposed to choose the "best" of these 10 scheduling possibilities. The figure shows examples of the four combinations of D and G type algorithms with or without void filling.

In the past literature several algorithms have been proposed and studied belonging to the various classes previously outlined. The problem in implementing these algorithms, as well as other WDS algorithms, is that they need to search "on the fly" at BCP or packet header arrival for the best scheduling time. Generally speaking, this implies to scan a data structure that records the arrival and departure times of each previously scheduled burst/packet. The complexity of this data structure, as well as the number

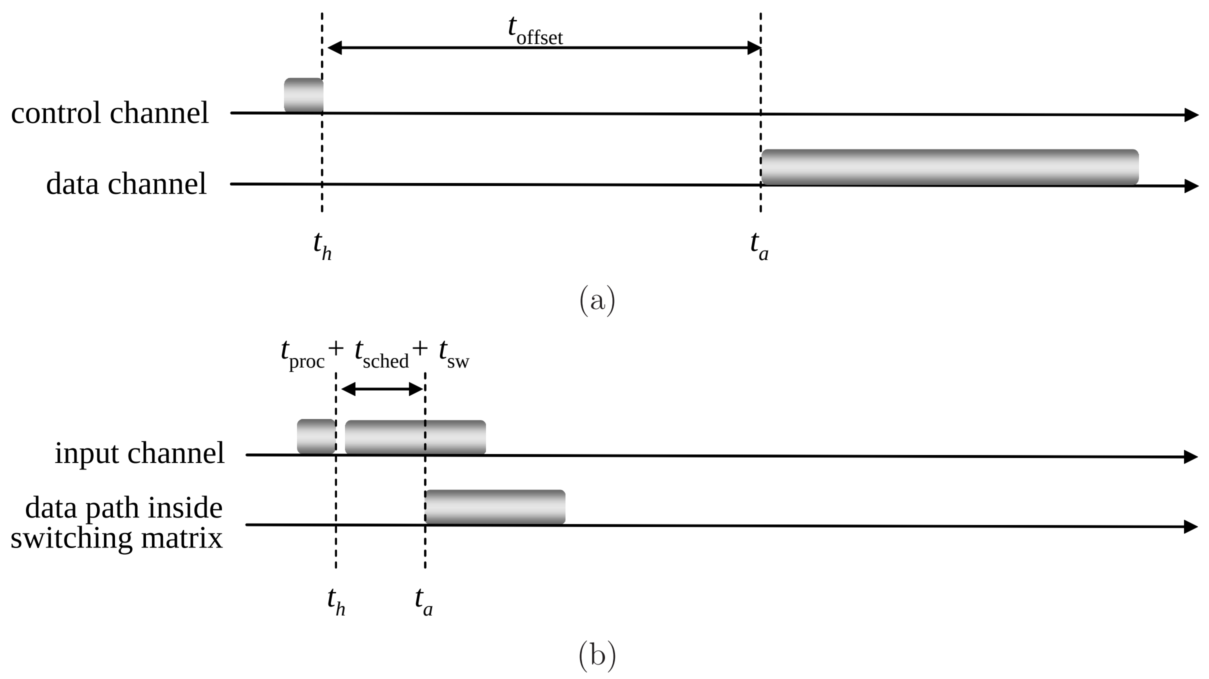

t_{offset}
control channel
data channel
t_h
t_a
(a)
$t_{\text{proc}} + t_{\text{sched}} + t_{\text{sw}}$
input channel
data path inside switching matrix
t_h
t_a
(b)

Figure 3.3: Timing for OBS (a) and OPS (b) scheduling.

N of scheduling points to search from, may be fairly large, depending on the number of wavelengths and delays. Moreover such complexity, especially for the VF algorithms, varies according to the traffic conditions.

3.3 Scheduling problem formulation

This work assumes that the optical switching matrix has to deal with bursts/packets with asynchronous arrivals and variable size, with minimum and maximum values L_m and L_M respectively (as usually happens with network protocols). With relation to figure 3.1, this section refers to a single scheduler working on a given forwarding destination that, in the most general case, is represented by a bundle of fibers carrying a number of wavelengths and by a given set of delays per wavelength. For the sake of simplicity in the notation and without loss of generality, it is assumed that there is only one fiber per forwarding destination carrying W wavelengths. Therefore the scheduling algorithm must return the scheduling point $s_{ij} \in \mathcal{S}$ that satisfies some optimality criterion.

Let t_h be the time when a burst control packet or a packet header has arrived at the switch. After the input port processing, at time $t_0 = t_h + t_{\text{proc}}$, the correct output fiber has been determined and the forwarding information is passed to the corresponding scheduler. So t_0 is the time when the scheduling begins. Let assume that t_{sched} is the time required for the scheduling, t_{sw} the switch reconfiguration time and t_{offset} the offset time in the OBS case. We can consider $t_{\text{offset\_min}}$ as minimum offset. The burst offset time is computed at the ingress node in order to include the processing, scheduling and switching times for all the nodes to be crossed along the path toward the egress node, while in the packet case the optical payload is typically delayed to allow for header processing, scheduling and switch configuration. Therefore, as shown in figure 3.3, the instant when actual data will arrive at the configured switching matrix is either

$$t_a = t_0 + t_{\text{sched}} + t_{\text{sw}}$$

for a packet or

$$t_a = t_h + t_{\text{offset}}$$

for a burst. Let x be the burst/packet duration (i.e. the burst/packet size expressed in time units), with $L_m \le x \le L_M$. Let us define the *buffer delay times*, i.e. the times when the burst/packet could be available at the output after being delayed, as

$$d_i(t_a) = t_a + iD, \quad i = 0, \dots, B-1.$$

These values include the cut-through path ($i = 0$) and consider the switching matrix propagation time negligible. Finally, the *scheduling time window* is defined as

$$d_B(t_a) = d_{B-1}(t_a) + L_M.$$

The scheduling formulation proposed here requires the gaps on each wavelength to be arranged in a list, for a total of W lists per output fiber. The number of lists are exactly the number of scheduling points, $|\mathcal{S}|$. Each gap in the list is represented by an element including the gap starting time and the gap ending time $v_{ij}(b, e)$. Lists are ordered based on the gap starting time. In order to have a quick access to relevant gaps, each list, in a Wavelength, is managed through a number of pointers equal to the number of available delays. Each pointer refers to the first gap that overlaps the corresponding delay. In general, pointer p_{ij} refers to the first gap on wavelength $j = 1, \dots, W$ such that the gap starting and ending times b and e satisfy one of the following conditions:

$$v_{ij}(b) \le d_i(t_a) \le v_{ij}(e) \le d_{i+1}(t_a) \tag{3.1}$$
$$d_i(t_a) \le v_{ij}(b) \le v_{ij}(e) \le d_{i+1}(t_a) \tag{3.2}$$
$$d_i(t_a) \le v_{ij}(b) \le d_{i+1}(t_a) \le v_{ij}(e) \tag{3.3}$$
$$v_{ij}(b) \le d_i(t_a) \le d_{i+1}(t_a) \le v_{ij}(e) \tag{3.4}$$

where $i = 0, 1, \dots, (B-1)$. As an example, let us consider the scheduling problem in figure 3.4, where $B = 5$, $W = 4$ and $L_M = 3D$. Without loss of generality we assume $D = 1$ and $t_a = 0$. The gap lists to be used in this case and has 20 pointers, the position of the p_{ij} pointers are shown in figure 3.5. Since there is no gap on wavelength $j = 4$ overlapping delay $i = 1$, pointer p_{14} is undefined. In case without void fillings the gap list

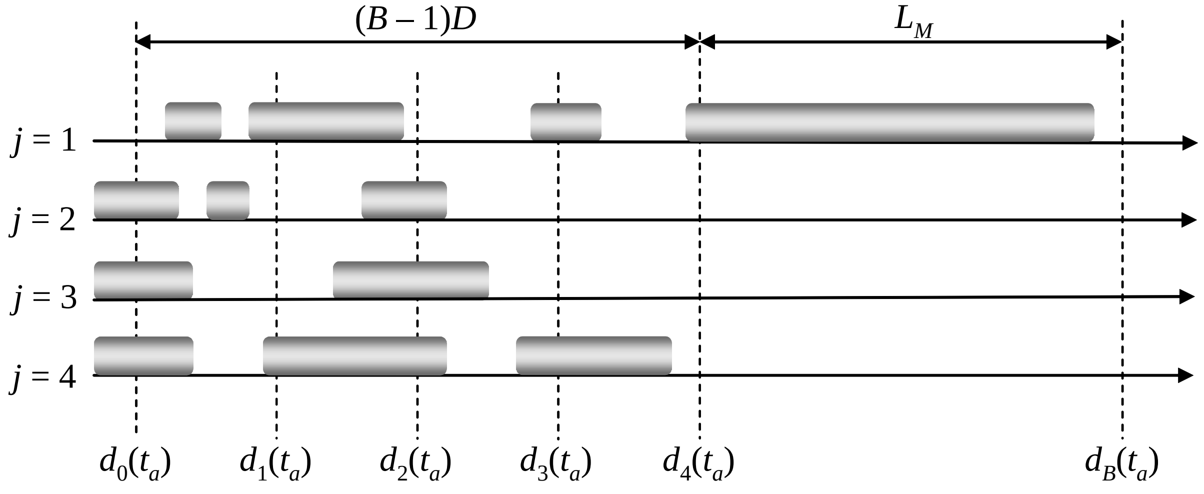


Figure 3.4: Buffer status at time t_a.

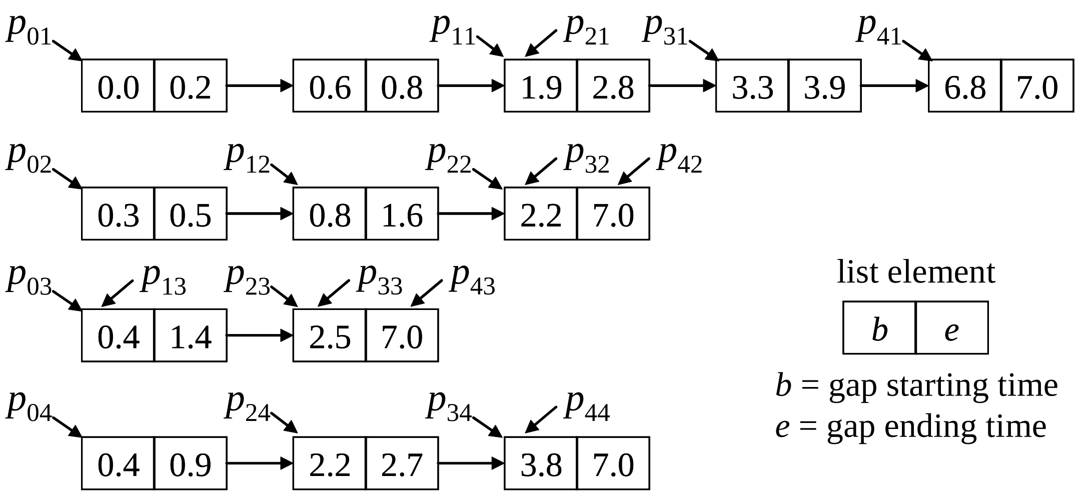


Figure 3.5: Gap lists and pointers corresponding to the status of figure 3.4 with Void Filling.

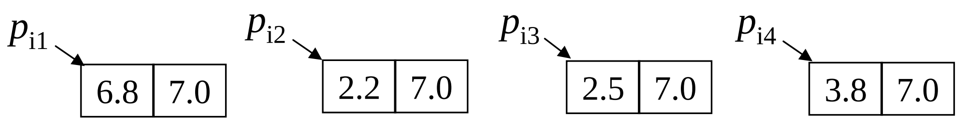


Figure 3.6: Gap lists and pointers corresponding to the status of figure 3.4 without Void Filling.

is much more simpler because gaps between packets can not be considered, thus only the gap at the end of the buffer is taken into account and all the pointers are referred to that gap. Figure 3.6 gives an example.

The memory list must be updated while the time is running in order to delete the gaps already past, that is the set of gaps less that $t_0 + t_{sched} + t_{sw}$ in packet switching and $t_0 + t_{\text{offset\_min}}$ in burst switching, where $t_{\text{offset\_min}} \leq t_{sched} + t_{sw}$. Consequentially, pointer position must be updated to the first list element. Moreover, the pointer position must be "updated" every time a new packet/burst must be scheduled. This is quite straightforward in the packet case, since it requires only to shift pointers p_{ij} forward as long as they reach the first gap overlapping the corresponding delay.

The burst case is different, since t_a represents the time reference when the burst will actually arrive once the offset time has elapsed. However, future BHP arrivals may refer to bursts with smaller offset time and the time reference must be updated according to this value. This means that in the OBS case the memory list must be updated every time a new BCH arrives but the set of gaps between t_0 and t_a must be remain in memory for possible future use. This results in a simple shift of all memory pointers to the current t_a in order to forecast the buffer situation.

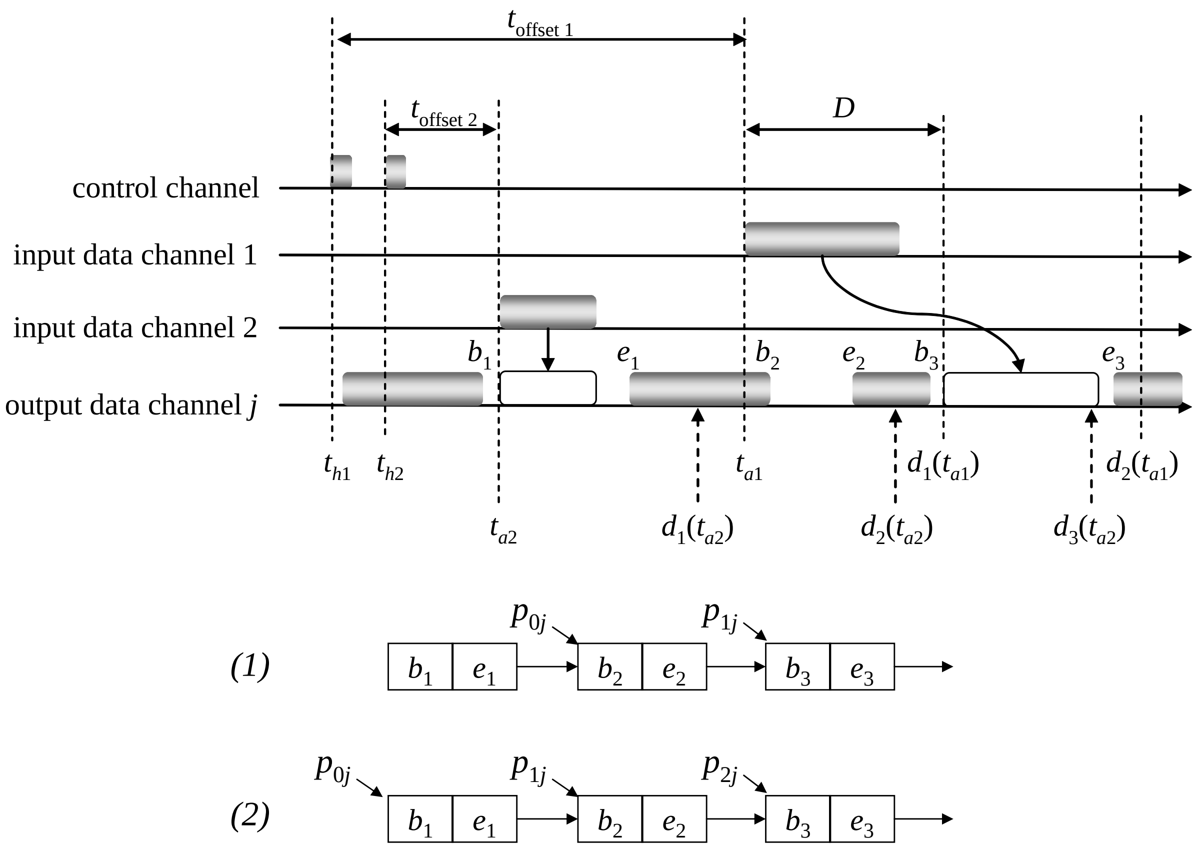
Because of the memory lists non correlation, all memory operations can be performed in parallel on every single list pointer, $p_{i,j}$, thus improving the scheduler speed.

In order to find a scheduling point for a new arrived Burst/packet the algorithm presented below must be applied. The function **GapTest**(i, j), performed for each wavelength j and delay i, gives as output the *Availability Space* A which collects all the available scheduling points for a particular burst or packet.

In order to understand how to use the gap lists, let us consider a burst/packet with length x arriving at time t_0. After the memory update, let c_{ij} be the pointer to the first element in list v_{ij} representing a gap that is not expired yet. This element is always the first one such that $c_{ij}(e) > t_0$, but in the OBS case, if a minimum offset time $t_{\text{offset\_min}}$ is defined, this element is the first one such that $c_{ij}(e) > t_0 + t_{\text{offset\_min}}$. All the gaps represented by elements preceding the one pointed by c_{ij} cannot be used by the scheduler since they are already expired. For each delay i and each wavelength j, the function **GapTest**(i, j) follows three different phases;

1. the first if statement controls into the first element of the lists c_{ij} if the end of the gap, $p_{ij}(e)$, is less that the *buffer delay times* $d_i(t_a)$;

2. the second if statement controls that the beginning of the gap, $p_{ij}(b)$, found in the first phase is bigger that of the *buffer delay times* $d_i(t_a)$ plus the length of the burst/packet x. If the statement is *true* means that the burst/packet is too long for that particular gap. If the statement is *false* the algorithm calculate the *Head matrix* H_{ij} and the *Tail matrix* $T_{ij}(x)$. A negative value in the *Head matrix* H_{ij} indicates that there is a overlap between the beginning of the gap and the burst/packet. A



(b)

Figure 3.7: Example of OBS network (a) buffer (b) updated list at time t_{a1} *(1)* and t_{a2} *(2)*.

negative value in the *Tail matrix* $T_{ij}(x)$ detects the overlap with the burst/packet at the end of the gap.

3. in the third if statement only positive values of both matrix are selected and inserted in the matrix of *Availability Space* A.

```
1: **GapTest**$(i, j)$
2: **if** $c_{ij}(e) < d_i(t_a)$ **then**
3:    **if** $c_{ij}(b) > d_i(t_a) + x$ **then**
4:       $H_{ij} := -\infty$
5:       $T_{ij}(x) := -\infty$
6:    **else**
7:       $H_{ij} := d_i(t_a) - c_{ij}(b)$
8:       $T_{ij}(x) := c_{ij}(e) - d_i(t_a) - x$
9:    **end if**
10:   **if** $H_{ij} \geq 0$ **and** $T_{ij}(x) \geq 0$ **then**
11:      Insert$(c_{ij}, \mathcal{A})$
12:   **end if**
13: **end if**
```

Basically, given a gap in the scheduling time window of a given packet, T and H measure how much a packet to be scheduled overlaps with the two packets already scheduled that create the gap. If $c_{ij}(b)$ and $c_{ij}(b)$ are the starting and ending times of the gap respectively, t_a is the packet arrival time (or the delayed packet time if FDLs are used) and x its length, we define:

1. $c_{ij}(b) < t_a < t_a + x < c_{ij}(e) \rightarrow H > 0, T > 0 \rightarrow$ no overlapping

2. $t_a < c_{ij}(b) < t_a + x < c_{ij}(e) \rightarrow H < 0, T > 0 \rightarrow$ head overlapping

3. $c_{ij}(b) < t_a < c_{ij}(e) < t_a + x \rightarrow H > 0, T < 0 \rightarrow$ tail overlapping

4. $t_a < c_{ij}(b) < c_{ij}(e) < t_a + x \rightarrow H < 0, T < 0 \rightarrow$ head and tail overlapping

We have also the case when $t_a < t_a + x < c_{ij}(b) < c_{ij}(e)$, but in this case there is total overlapping of the packet to be scheduled with the one preceding the gap, so it's of no use and we set H and $T = -\infty$.

In figure 3.7 an example in an OBS network is reported with two input data channel and only one output data channel.

At time t_{h1} a new BCH arrives on the control channel, the burst will arrive at time t_{a1} on input data channel 1 after t_{offset_1}. Following the function already presented all the buffer lists must be update at time t_{a1}. Figure 3.7.b shows the gap list at time t_{a1} only the single output data channel. The first element of the memory list, p_{0j}, point at (b_2, e_2), obviously the first gap, (b_1, e_1), can not be taken into account because it is in the past of

t_{a1}. The function **GapTest**(i, j) starts running in the pointer p_{0j}. The gap (b_2, e_2) is too small for the burst, it overlaps both the head and the tail (i.e. $H_{ij} < 0$, $T_{ij}(x) < 0$). In parallel, the function is also performed to the pointed p_{1j} of the first delay line $d_1(t_{a1})$. The pointer points to the gap (b_3, e_3) which is the first gap in the list. This gap fits well with the burst and it is a possible scheduling point, it will be added to the *Availability Space*.

A second BCH arrives in the control channel regarding a burst on the second input data channel. This new burst has a smaller offset t_{offset_2}, in this case it is easy to verify that the first gap (b_1, e_1) could aldo be a valid scheduling point.

With reference to the example in figure 3.4 3.5, assuming $x = 0.3$, the availability space in the void filling case is:

$$\mathcal{A} = \{(1,2),(1,3),(2,1),(3,2),(3,3),(4,2),(4,3),(4,4)\}$$

As previously discussed, in case of scheduling algorithms that do not use void filling, the gap lists are much simpler because the gaps inside the packets can not be considered, figure 3.6. This leads to a smaller availability space.

$$\mathcal{A}' = \{(3,2),(3,3),(4,2),(4,3),(4,4)\}$$

Depending on the scheduling algorithm chosen, search functions may be defined on $\mathcal{A}$ and applied to the elements in the gap lists. To this purpose, in the next sections some scheduling algorithm will be analyzed.

3.3.1 G-type VF scheduling algorithm

For each $(i, j) \in \mathcal{A}$ let us define the function

$$G_{ij}(x) = H(i, j) \tag{3.5}$$

For each scheduling point available in $\mathcal{A}$, this function measures the residual gap that would be created between the incoming burst/packet and the following one, in case of immediate transmission (i.e. when $i = 0$), or the previous one, in case of delayed transmission (i.e. when $i > 0$). Therefore, the optimal scheduling point for the G-type VF algorithm is $s_{dw} \in \mathcal{S}$ such that

$$G_{dw}(x) = \min_{(i,j)\in\mathcal{A}} G_{ij}(x) \tag{3.6}$$

When multiple scheduling points provide the same smallest residual gap, equation (3.6) gives multiple solutions in $\mathcal{A}$. In this case, the one with the smallest value of i is selected, meaning that the earliest optimal scheduling point is chosen. When there are still multiple alternatives, a random choice is made. The values of $G_{ij}(0.3)$ for the reference example are shown in table 3.1. In this case, the best scheduling point is s_{21}.

Table 3.1: $G_{ij}(0.3)$ function

$G_{ij}(0.3)$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
$j=1$			0.1		
$j=2$		0.2		0.8	1
$j=3$		0.4		0.5	1
$j=4$					0.2

3.3.2 D-type VF scheduling algorithm

Given the function defined in (3.5), for each $(i,j) \in \mathcal{A}$ let us define

$$D_{ij}(x) = T_B(t_a) - t_i(t_a) - G_{ij}(x) \tag{3.7}$$

This function measures the distance between each scheduling point available in $\mathcal{A}$ and the end of the scheduling time window, reduced by the residual gap measured by $G_{ij}(x)$. Therefore, $D_{ij}(x)$ is maximized when the smallest delay available is assigned to the burst/-packet. In case such delay is available on more than one wavelength, the further decrease of the residual gap causes the highest value of $D_{ij}(x)$ to correspond to the smallest gap. Then, the optimal scheduling point for the D-type VF algorithm is $s_{dw} \in \mathcal{S}$ such that

$$D_{dw}(x) = \max_{(i,j)\in\mathcal{A}} D_{ij}(x) \tag{3.8}$$

In case equation (3.8) gives multiple solutions in $\mathcal{A}$, corresponding to multiple scheduling points with smallest delay and same residual gap, a random choice is performed. The values of $D_{ij}(0.3)$ for the reference example are shown in table 3.2. In this case, the best scheduling time is s_{12}.

Table 3.2: $D_{ij}(0.3)$ function

$D_{ij}(0.3)$	$i=0$	$i=1$	$i=2$	$i=3$	$i=4$
$j=1$			4.9		
$j=2$		5.8		3.2	2
$j=3$		5.6		3.5	2
$j=4$					2.8

3.3.3 G-type noVF scheduling algorithm

Since gap lists now are made by at most a single element, a measure of the residual gap between the current burst/packet and the previous one can be written as

$$G'_{ij}(x) = H_{i,j} \tag{3.9}$$

for each $(i,j) \in \mathcal{A}'$. Therefore, the optimal scheduling point for the G-type noVF algorithm is $s_{dw} \in \mathcal{S}$ such that

$$G'_{dw} = \min_{(i,j)\in\mathcal{A}'} G'_{ij} \tag{3.10}$$

In case of multiple solutions, the same observations made for equation (3.6) apply. The values of G'_{ij} for the reference example are shown in table 3.3. In this case, the best scheduling time is s_{44}.

Table 3.3: $G'_{ij}(0.3)$ function

(i,j)	$(3,2)$	$(3,3)$	$(4,2)$	$(4,3)$	$(4,4)$
G'_{ij}	0.8	0.5	1	1	0.2

3.3.4 D-type noVF scheduling algorithm

Similarly to what done in section 3.3.2, it is possible to define

$$D'_{ij} = t_B(t_a) - t_i(t_a) - G'_{ij} \tag{3.11}$$

for each $(i,j) \in \mathcal{A}'$. The optimal scheduling point for the D-type noVF algorithm is $s_{dw} \in \mathcal{S}$ such that

$$D'_{dw} = \max_{(i,j)\in\mathcal{A}'} D'_{ij} \tag{3.12}$$

The values of D'_{ij} for the reference example are shown in table 3.4. In this case, the best scheduling time is s_{33}.

Table 3.4: $D'_{ij}(0.3)$ function

(i,j)	$(3,2)$	$(3,3)$	$(4,2)$	$(4,3)$	$(4,4)$
D'_{ij}	3.2	3.5	2	2	2.8

3.4 List management

The list management is a fundamental part of the proposed algorithm. Every time a burst/packet arrives the scheduler runs getting all the buffer status information from the memory lists. Moreover, every time a new burst/packet is routed to an output port the buffer must be update according. In this section the most important management functions will be presented. In order to do this, we need to define the current time limit

t_l which is minimum time to be consider for scheduling, depending on OBS or OPS the t_l can assume different values:

$$OBS : t_l = t_a = t_0 + t_{sched} + t_{sw} \quad (3.13)$$
$$OPS : t_l = t_0 + t_{\text{offset\_min}} \quad (3.14)$$

Trigger Function

As previously outline, the memory list must be updated while the time is running in order to delete the gaps already past. This is mandatory since as soon as a new burst/packet arrives the function **GapTest**(i, j) process the first values in each lists. The trigger function deals with this particular issue. Each pointer has a **Trigger** function, shown below, which controls that the end of the current pointed gap does not overlap the current time t_l. Moreover, the trigger function keeps the first list element up to date.

```
1: Trigger$(i, j)$
2: while $True$ do
3:    if $c_{ij}(e) < d_i(t_l)$ then
4:       if $c_{ij}(e) < t_l$ then
5:          Drop: $c_{ij}$
6:       end if
7:       $c_{ij} := c_{ij} \rightarrow$ next value
8:    end if
9: end while
```

In the OPS case the **Trigger** function is enough to keep updated the memory lists and to allow the function **GapTest** to process on the fly any new packets. The **Trigger** function runs continuously in parallel for every list pointers.

Pointer function update

This particular function is involved only in OBS scheduling algorithm. OBS has burst data delayed by an offset, thus before running the **GapTest** the memory list must be update at the t_a value. This operation is performed by the **PointerUpdate** function, this function provides a simple pointers shifting in order to update the list pointers and doesn't provide any burst drop or any memory values modification.

```
1: PointerUpdate$(i, j)$
2: while $c_{ij}(e) < d_i(t_a)$ do
3:    $c_{ij} := c_{ij} \rightarrow$ next value
4: end while
```

The **PointerUpdate** function must be called for each wavelength and fibre delay lines before running the **GapTest** function only in the case of OBS.

Gap modification

When the scheduler has found a valid scheduling point forwards the burst/packet to the proper output wavelength and fibre delay line. Consequentially, the gap selected for the new burst/packet insertion is modified. The gap can be completely fitted by the burst/-packet or it can happened that after the insertion new gaps can be created. The following algorithm shows the gap modification phases supposing x the length of the burst/packet and c_{ij} the selected gaps for the insertion.

```
ModifyGap($i, j$)
if $c_{ij}(e) - c_{ij}(b) - x = 0$ then
  Drop: $c_{ij}$
  $c_{ij} := c_{ij} \rightarrow$ next value
else
  if $c_{ij}(b) = d_i(t_a)$ then
    $c_{ij}(b) := c_{ij}(b) + x$
  end if
  if $c_{ij}(b) < d_i(t_a)$ then
    if $c_{ij}(e) = d_i(t_a) + x$ then
      Drop: $c_{ij}$
      $c_{ij} := c_{ij} \rightarrow$ next value
    else
      $c_{ij}(b) := c_{ij}(b) + x$
    end if
    Create new gap: $n$
    $n(b) = c_{ij}(b)$
    $n(e) = d_i(t_a)$
    $n \rightarrow$ next pointer $:= c_{ij}$
    $n \rightarrow$ previous pointer $:= c_{ij} \rightarrow$ previous pointer
  end if
end if
```

In particular, if the length of the burst/packet x is equal to the gap size the gap is dropped and the next gap became the first list gap. Otherwise, if the burst/packet doesn't fit with the gap, the insertion produces another gaps. Mainly, two new gaps can be created; the first before the beginning of the burst/packet and a second gap at the end. After the scheduling the **ModifyGap**(i, j) function must be called only for the gap selected for the forwarding (i, j) in order to fix the gap list head.

3.5 WDS scheduling implementation

In order to implement the WDS scheduling according to the formulation given in the previous sections, the major issues to be solved are related to the gap lists management, the computation of the function H and T and to the optimum search function. This section suggests a possible scheduling implementation showing some prototyping results obtained by FPGA simulation. The simulation have been done by Altera Qartus II 8.0 Web Edition using Stratix II and III as referenced board. All the simulation have been made using 8 bit register, input and output variables and real timing simulation parameters without any fitter optimization.

As previously outline OPS and OBS scheduler is different in some parts. The gap lists v_{ij} and the values of $d_i(t_a)$ (delay buffer times) and $d_B(t_a)$ (scheduling time window) depends on the instant time chosen as reference. Such instant is typically the burst/packet arrival time, therefore when a new arrival occurs for instance at t_a, the first task is to update $d_i(t_a)$ and each gap list head.

In the OPS case this task is unnecessary since the Trigger Function keeps all the values lists already updated during the time. On the other hand, in the OBS case the value of t_l could be quite far from the curret time because of the t_{offset}. In this case, the function (**PointerUpdate**(i, j)) shifts the pointer head list to the current memory value of t_a.

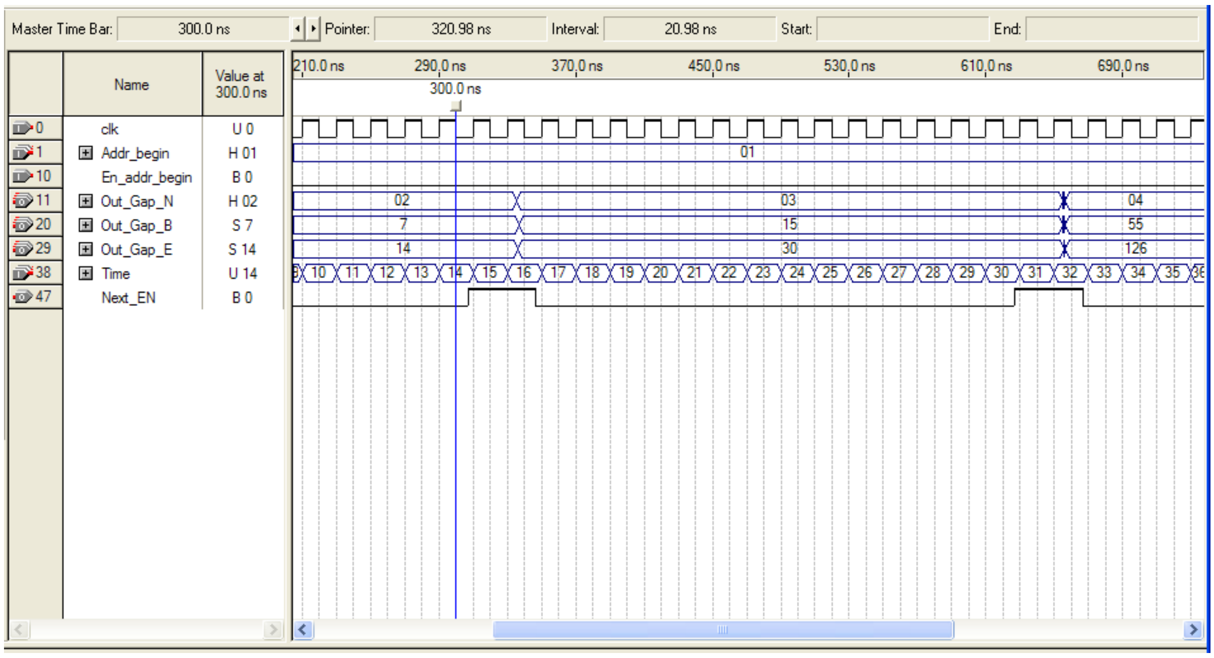
Updating the values of $d_i(t_a)$ is very straightforward, since they can be obtained with a simple summation; in the OBS case, $d_i(t_a) = t_h + t_{offset} + iD$ and $d_B(t_a) = t_h + d_{B-1}(t_a) + L_M$ while in the OBS case these values are already updated.

Updating the gap lists is just a little more complicated. First of all, note that all the values $v_{ij} < t_l$ are obsolete (the bursts/packets related to gaps in those lists have already been transmitted) and can simply be deleted, while new empty lists are created to store future gaps that could be the result of the scheduling of the new packet. This operation can be implemented with a simple shift of memory pointers and it is fairly simple. After having shifted the lists (if necessary), by means of the Trigger Function (section 3.4), all the already past elements are removed and the gap redistributed according to the rules introduced in section 3.3.

All these operations can be made in parallel per wavelength (index j of the v_{ij}). Since previous works [CCB+06] suggest that, for a given value of $|\mathcal{S}|$, a large number of wavelengths is more effective for congestion resolution than a large number of delays, it should be $j >> i$ and therefore parallel processing of the lists should highly improve the time needed to perform this task.

Pointer shifting operation

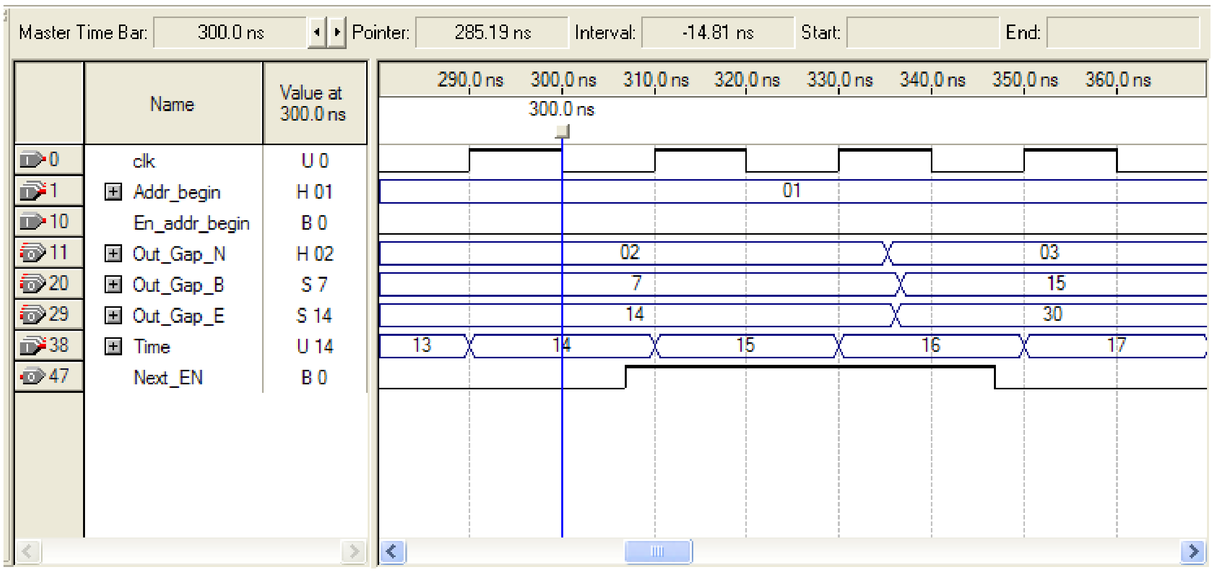
In figure 3.8.(a) and (b) is shown the simulation results of pointer shifting operation. The input parameter "Time" represents a generic $d_i(t_a)$ value, while the time is running the list head c_{ij} must be maintained up to date. To this purpose the list head pointer must



Master Time Bar: 300.0 ns Pointer: 320.98 ns Interval: 20.98 ns Start: End:

Name	Value at 300.0 ns
clk	U 0
Addr_begin	H 01
En_addr_begin	B 0
Out_Gap_N	H 02
Out_Gap_B	S 7
Out_Gap_E	S 14
Time	U 14
Next_EN	B 0

(a)



Master Time Bar: 300.0 ns Pointer: 285.19 ns Interval: -14.81 ns Start: End:

Name	Value at 300.0 ns
clk	U 0
Addr_begin	H 01
En_addr_begin	B 0
Out_Gap_N	H 02
Out_Gap_B	S 7
Out_Gap_E	S 14
Time	U 14
Next_EN	B 0

(b)

Figure 3.8: Pointer shifting operation simulation waveforms (a) portional view (b)

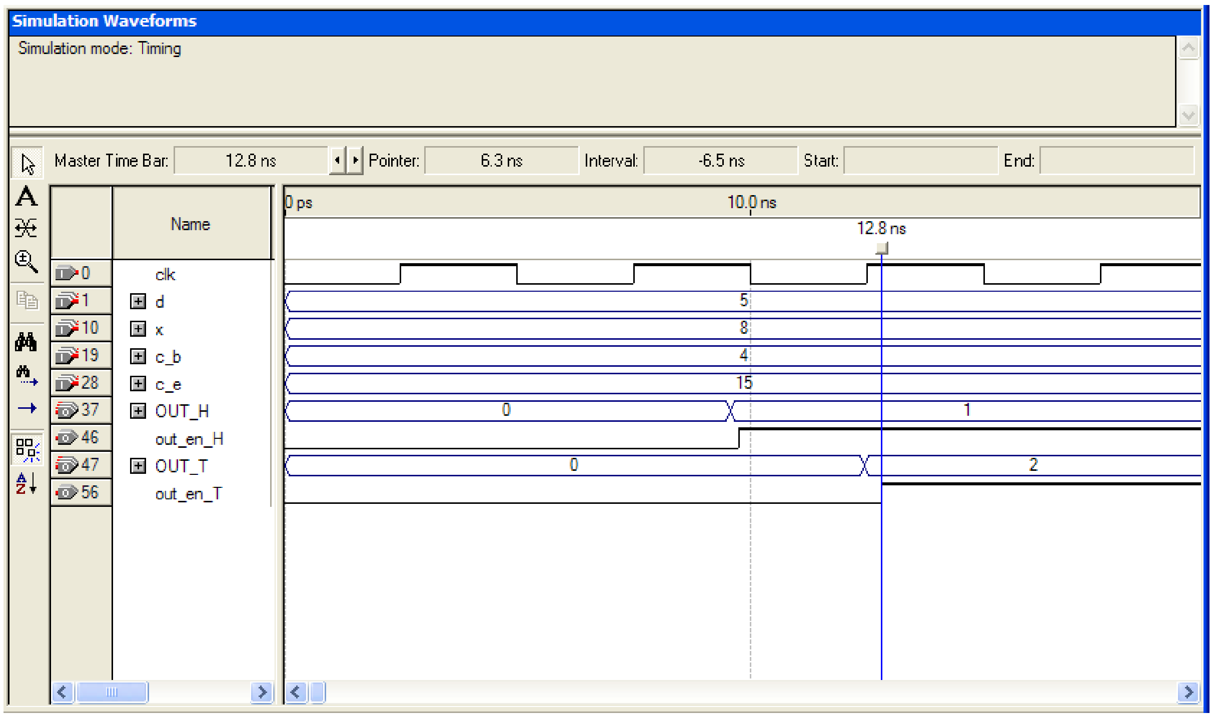


Figure 3.9: Simulation waveforms of H and T calculation

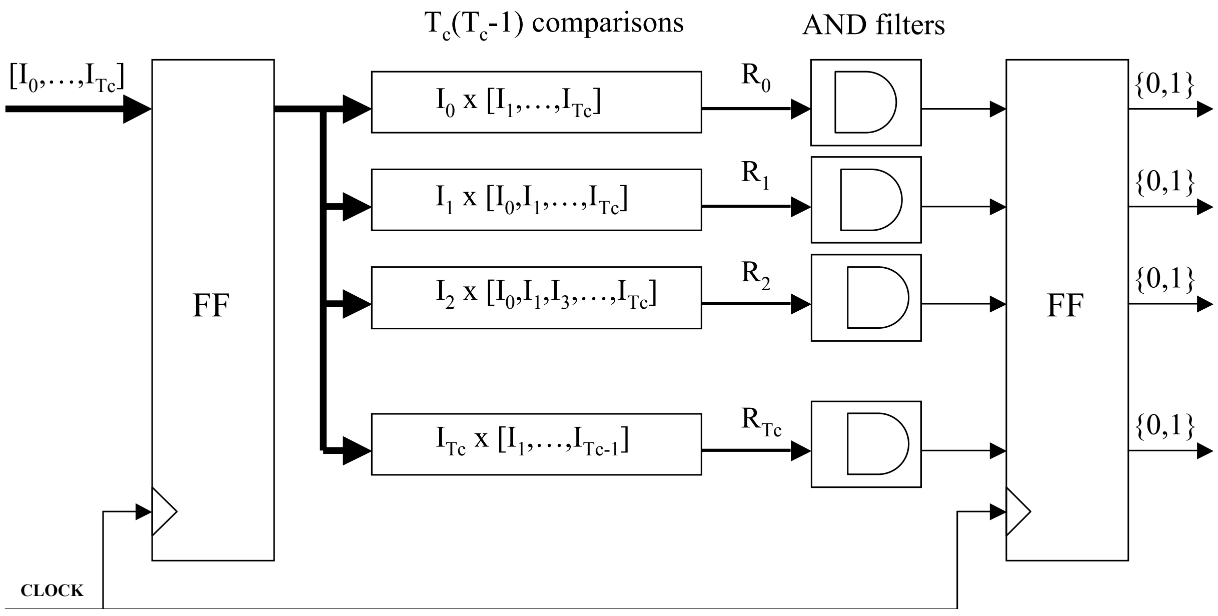
be updated periodically. c_{ij} is represented by three output registers, "*OUT_Gap_N*", "*OUT_Gap_B*" and "*OUT_Gap_E*" which represent respectively the pointer to the next gap, the starting gap value and the ending gap value.

At the beginning of the simulation the c_{ij} values are: "*OUT_Gap_N*"=2, "*OUT_Gap_B*"=7, "*OUT_Gap_E*"=14. When the "Time" arrives at the same value of the end of the gap (i.e. "*OUT_Gap_E*"), the gap became obsolete and the scheduler must update the head list with the new gap (pointed by the "*OUT_Gap_N*" value). The "*Next_EN*", which is the enable signal, rises up and the scheduler loads the next value pointed by the "*OUT_Gap_N*" register. Therefore, the new values are loaded into the registers becoming the new list head, "*OUT_Gap_N*"=3, "*OUT_Gap_B'*"=15, "*OUT_Gap_E*"=30.

This is shown in details in figure 3.8.(b). As soon as a gap become obsolete the scheduler takes two clock cycles in order to load the new values. The simulation has been simulated with a max clock frequency of 155.09 Mhz (6.448 ns) in a Stratix III board. It is also important to point out that this result has been obtained without any fitter optimization, thus further improvement can be easily achieved.

H and T calculation

As presented in section 3.3 the calculation of H and T involves the list head, the length of the burst/packet x and $d_i(t_a)$. Figure 3.9 present simulation results where "d" is a generic



$T_c(T_c\text{-}1)$ comparisons
AND filters
$[I_0,\ldots,I_{Tc}]$
FF
$I_0 \times [I_1,\ldots,I_{Tc}]$
$I_1 \times [I_0,I_1,\ldots,I_{Tc}]$
$I_2 \times [I_0,I_1,I_3,\ldots,I_{Tc}]$
$I_{Tc} \times [I_1,\ldots,I_{Tc\text{-}1}]$
R_0
R_1
R_2
R_{Tc}
FF
{0,1}
{0,1}
{0,1}
{0,1}
CLOCK

Figure 3.10: Example of hardware algorithm

$d_i(t_a)$, "x" is the the burst/packet lenght, "c_b" and "c_e" are respectively the starting and the ending value of the list head gap. The scheduler calculates the value of both H and T in few clock cycles. In particular the "OUT_en_H'' signal takes two clock cycles to rise up allowing the read of the new H value in the "$OUT\_H = 1$" register. Regarding the T function, it takes three clock cycles in order to have the signal output "OUT_en_T" and "$OUT\_T = 2$" calculated. This test has been simulated using a Stratix II board with a maximum clock frequency of 241.31 Mhz (4.144 ns).

Search function

As for the implementation of the optimum search function, the execution of the WDS scheduling algorithm is accomplished by simply computing the minimum or maximum over a given set of values. This can be realized by dedicated hardware. The hardware structure shown in figure 3.10 is composed by a set of $S(S-1)$ comparators, that take two input values I_1, I_2 and returns a binary output. For instance, while searching a maximum, the comparator work as follows:

- $C_{max}(I_1, I_2) = 1$, if $I_1 \geq I_2$
- $C_{max}(I_1, I_2) = 0$, if $I_1 < I_2$

As suggested by the schematic in figure 3.10, for each input I_i the hardware structure determinates if I_i is a maximum of the set. Such operations can be performed in parallel by arrays of comparators. Therefore the time needed to compare all these values is approximately *constant and in the order of a single computational step.* The result set is composed by all the optimal values, according to the algorithm chosen. The final result can be chosen by means of an additional criteria or simply by a random choice.

We have designed and simulated the behavior of the structure in figure 3.10 assuming an FPGA implementation with 32 bits encoding for the values of the time instants. Preliminary results suggest a processing time, to obtain the best scheduling, in the order of 10 ns with a number of inputs of a few tens. This sounds like a fairly reasonable performance, assuming an average length of the optical bursts/packets in the order of the microsecond or more [CMP+98], [CCXV99].

4 Service Oriented Networking

Service Oriented Networking aims at defining a user-oriented architecture that enable user applications to benefit of advanced services over an underlying dynamic optical layer. The optical service layer builds a set of services over the basic optical transport layer which provides the basic mechanisms for reserving resources

The core of the proposed architecture is to provide full application aware networking by means of a single signalling protocol, thus providing advanced applications (for instance Grid computing) with the opportunity to request a given application resources (i.e. remote storage space, remote computation, media stream, etc.) and communication facilities (i.e. Bandwidth, Delay etc.) required to connect to it with proper grade of service fully relying on the network infrastructure.

For this purpose, the open challenges are addressed in terms of:

- Resource Virtualization, that is the capability to make network and non network resources visible to the end-user application in a unified and “understandable” language. Resource Virtualization aims at representing physical objects and equipments as logical resources. Moreover, virtualization can also deals with service abstraction decoupling the service in functional parts. This is the case of Service Composition where complex function can be split up into simple operations;

- end-to-end service provisioning, that is the capability to match application requests to resource availability, automatically and on demand;

The step forward proposed here is based on the idea to implement a signalling architecture that can make the applications and the network evolve into a fully integrated infrastructure while still allowing to keep a clear separation of the roles of users and applications, as well as of different network domains. The idea is to the boundaries between application, network control and information transport disappear thanks to an integration layer provided by the new signalling infrastructure.

To this purpose, the following action points have been studied to this end:

- make the application capable of exchanging semantically rich messages with the network to issue service requests for the negotiation and reservation of the needed resources;

- choose a semantically rich language that is general enough to be useable by any sort of application for describing network requirements;
- define a "technology independent" methodology to control network resources by mapping the service request into network control plane directives.

4.1 Previous works

Currently, some aspects of the Service Oriented Networking have already been addressed with different flavours, but with the lack of generality and purpose of the whole Service architecture. In particular, Resource Virtualization has recently became an interesting topic for many researchers.

One of the first attempts to virtualize network resources was the concept of User-Controlled Light-paths (UCLP) [WZC+04]. The UCLP is based on a web service-based infrastructure in order to allow users to manage optical network resources in a user-centric approach. UCLP has also been recently extended to manage Layer 1 Virtual Private Network (VPN) [WSC+06] but in a more network-centric approach where end-user still utilize the UCLP based approach but to manage VPN already provided by the L1VPN administrator. Thus, the atomic elements the end-user can play with in [WSC+06] are the VPN and not the lightpaths of which they consists. Similarly, in the framework of the ITU-T there has been effort in this direction trying to standardize high-level triggering of network services. The most significant example is the ITU-T NGN architecture, see section 2.2. Through the logical partitioning of functions into a Service Stratum and Transport Stratum, NGN is capable of abstracting the connectivity services over a IP-based service-oriented network architecture.

Moreover, projects regarding End-to-end reservation of network and non-network resources are emerging especially in the Grid research area. Three major projects are currently focusing on developing mechanisms for end-to-end coordinated reservations across network domains of geographically distributed high-end computing resources and scientific instrumentation Phosphorus and Enlightened reflect the NREN (National Research Network), Research infrastructure views concerning network and service development and interaction. All the projects are specifically focusing on Grid and data intensive scientific applications.

- *PHOSPHORUS* [PHO]: This project addresses some of the technical challenges in enabling on-demand end-to-end network services across multiple heterogeneous domains. In the Phosphorus' implementation the underlying network will be treated as first class Grid resource. PHOSPHORUS will develop integration between application middleware and transport networks. Coordination of network and IT resources will be achieved through enhancements of the GMPLS Control Plane (GMPLS) to provide optical network resources as first-class Grid resource

- *Enlightened Computing* [Com]: Enlightened Computing designs and develops a Grid framework that allows applications to dynamically request computing and storage resources along with the necessary dedicated bandwidth. To achieve this, Enlightened develops Grid middleware that views the network as a Grid resource at the same level as the compute and storage resources. The underlying network(s) is GMPS controlled.

- *G-Lambda* [GL]: G-Lambda establishes a standard Web service interface (Grid Network Service/Web Service Interface GNS-WSI) between Grid resource manager and network service manager. It defines a standardized network interface in a Grid context including APIs for users and multi-domain inter-working interface.

The first two approaches are user-centric since the user can have knowledge and in some cases control of the network topology and resources, while the G-Lambda has a network centric approach. All these works provides very valuable contributions and ideas but with the lack of generality to be applicable to a general application environment on a general and public networking infrastructure. They are either limited to specific networking scenarios or to specific application scenarios. Moreover in case of a public network operated by networks operators they do not allow an easy separation between user and operator responsibilities.

4.2 Concepts and Objectives

The Service Oriented Networking proposed here is based on a novelty approach, the key concept is to migrate into the network some of the intelligence needed to support application services, making the network node capable of interacting with the network in an application oriented language. This is in line with the NGN proposal (section 2.2), shifting the current view of the network from pure transport facility to a more intelligent service oriented infrastructure.

Moreover, the Service Oriented Networking could represent the first steps towards the implementation of the NGN architecture where the aforementioned goals are achievable if service and transport stratum may operate according to a uniform point of view.

The key building block to support network services is an application signalling language able to:

- accept and understand the publication of information about the availability of IT resources and the related access policies;

- accept and understand the application requests;

- search if the resources required are available and negotiate the best possible answer to the requests of the specific application;

- provide access to the network resource in order to transport information of various sizes as effectively and efficiently as possible.

Accomplishing all these tasks at once is not easy, most of all because they span over several logical layers of the network stack. Today's solutions usually focus "horizontally" on a subset of them, while an integrated "vertical" solution is missing. To this end it is necessary to define protocols, languages and functional models capable of dealing with the interaction of different layers. In particular, the Service Oriented Networking is defined by the following concepts.

- A language able to represent application and network resources as well as attributes of a given communication to be used in a service composition context. This work describes the Network Resource Description Language (NRDL) 5 as the tool that allows application to communicate with the network.
- A session protocol to exchange statefull information between network's components. To this purpose, the Session Initiation Protocol (SIP) (section 4.3.2) has been used as the medium to support the signalling. By exploiting the session management message flows already provided by SIP, it is possible to implement application oriented functions such as resource publication, resource discovery and resource reservation, including communication facilities (i.e. QoS) 6. As a consequence, every equipments must be embedded with SIP capabilities in order to interact with the other elements.
- An interaction model between SIP and NRDL where the high level application and network languages are embedded into the payload of SIP messages at due time. The session management messages can carry both a application protocol (section 4.3.3) and a general resource description payload that contains the application request to the network in a user oriented language (i.e. NRDL).
- The main goal achieved by the interaction between the session protocol and the NRDL language is the definition of a new fundamental opaque layer, the *Session Plane* (section 4.3.1). The resulting session plane represents the statefull communication channel in the middle of the application and network layer. Exploiting the session plane the boundaries between application, network control and information transport almost disappear allowing an overall service view.
- An architecture with ad-hoc functional elements (section 4.4). The proposed architecture is based on a "network-centric" approach where resource virtualization and end-to-end service provisioning is supported by the introduction of a novel specific network module, called Application Oriented Module (AO-M) (section4.4.1), which operates at the logical interface between the end User (Applications) and the network. The main function of the AO-M is to interface, by means of a Session Layer,

the applications and the network Control/Data Plane, in order to co-ordinate the use of both network and IT resources.

- the result of this coordination activities is the basis of a new functional plane called *Service Plane* (section 4.3.1) which is the architectural and functional element of the Service Oriented Networking. In other words, the service plane, by means of the AO-M, is commissioned to interface applications, session plane, and optical network control plane provides network and non-network resource abstraction and co-ordination.

It is worth mentioning that for communications spanning over multiple network domains the problem of signalling is in general more complex that in a single network domain. The obvious reason is that each administrative entity involved along the path has responsibility on its domain and may not be fully coordinated with the others. So far, we do not consider the multi-domain issue, which will be subject of further investigation, and focus on QoS guarantee within a single network domain.

4.2.1 The players

The players of the Service Oriented Networking are mainly Application layer and Network control plane.

In one side, Network control plane (section 2.3.1) represents the interface to manage the transport layer. Talking about the network control plane we have already pointed out the importance to address heterogeneous networks with different levels of capacity, flexibility and intelligence. Of course different networks have different control planes with different capabilities. Because, the Service Oriented Networking aims at defining a set of correlated procedure between application and network, the interaction with the control plane represent a fundamental action.

To the other side, we have high-end advanced applications in constant evolution, it is important to point out that not all kinds of applications can take advantages by the Service Oriented Networking. Due to the fact that the Service Oriented Networking exposes enriched network functions and functionalities, in general only statefull distributed and heterogeneous applications with complex network requirements can take a significant gain by all these features.

As already mentioned, the large variety of applications represent the final user of the Service Oriented Networking. Because only a set of application can take the maximum gain from the underling advantage network, in the following we will consider only those applications well suited for our goals. In this case, applications particularly indicated could be Grid computing as well as Video on Demand services which are distributed applications with stringent QoS requirements. Moreover the experimental activities 7 are based on these particular applications as a proof of the aforementioned concepts.

Video on Demand

Video-on-demand (VoD) services are emerging as a fundamental multimedia application in both wired and wireless network environments. The essence of video-on-demand services is the timely display of multimedia objects, and this requires that the number of concurrent sessions should be maintained at below a system's capacity. In particular, video-on-demand (VoD) has stringent response-time constraints and requires the differentiation and dynamic QoS adaptation. Therefore, streaming servers generally use mechanisms in order to admit as many requests as possible up to system capacity, under the assumption that the current resource utilization is below the system capacity and admitting a new request will not hurt the quality of service (QoS) of the existing sessions significantly.

However, it is not easy to verify this assumption without having a portion view of the transport network. This is where the Service Oriented Networking comes into play allowing the session control with a network oriented prospective This approach meets the needs of Service providers and network operators giving methodologies and mechanisms for managing runtime overall QoS.

Grid computing

Grid computing is migrating from traditional high performance and distributed computing to pervasive and utility computing based on heterogeneous networks and clients. The success of this new range of services is directly linked to the effectiveness of the networking infrastructure used to deliver them.

Grid computing services aim at integrating and manage resources according to application requirements and within distributed, heterogeneous, dynamic environments. On the other hand, networks and, in particular their control planes, regardless of the specific implementation, deal with topology, capacity, connectivity and routing. In general, these two worlds have very different perspectives and talk very different languages. So far the direct link between the application and the network is still missing and the applications usually rely on virtual overlay topologies that provide connectivity according to a pre-defined scheme and are built a-priori in the network.

The present typical scenario is that, as proposed for instance by the Open Grid Forum [For], applications talk end-to-end and resource management is provided by means of web-services, an approach that has the merit to provide a standard and flexible communication [TCF+03][FBD+04]. This is a typical overlay approach with associated merits and drawbacks. In particular it may result in inefficiencies in the use of the network resources and in limited functionalities with respect to what achievable with a more integrated approach. Moreover limitations exist for implementation targeting large populations of users in terms of scalability, fault tolerance, performance, quality of service provisioning etc.

An alternative approach may be that of migrating into the network the intelligence needed to support the Grid computing services, making the network node capable of interacting with the network in an application oriented language. The realization of this goal requires the disintegration of numerous barriers that normally separate application services, IT resources (computing systems), underplaying network and make the data network infrastructures application-aware. Up to now the application layer services (e.g. resource discovery, reservation, etc.) have been deployed with the same centralized mechanisms based on web services without any knowledge of the transport services. The result is that, in most cases, the Grid signalling is segregated on a separate infrastructure while the transport network is used as a best effort transport to carry the application data almost independently by the Grid signalling.

The step forward proposed here is a signalling architecture that can make the grid applications and the network evolve into a fully integrated infrastructure where the boundaries between application, network control and information transport will almost disappear.

Exploiting the Service Oriented Networking we will show how it is possible to provide full application aware networking for Grid networks by means of a single signalling protocol, thus providing the application with the opportunity to request a given resource (i.e. remote storage space, remote computation, media stream, etc.) and the communication facilities required to connect to it with proper grade of service fully relying on the network infrastructure. Obviously, because the main target is to deploy a General Service Oriented Network Grid computing is not the only application environment which can take advantage by the network. However a significant number of test have been made using the Grid computing as main proof of concept application.

4.3 The building blocks

In this section we outline the general architecture of the proposed solution and related building blocks. The starting point is the consideration that Service Oriented Networking is the results of several coordinated actions between application and network. Moreover the Quality of Service issue is more and more important since, in general, high-end advanced applications like VoD or Grid computing (section 4.2.1 and 4.2.1) have rather stringent QoS requirements and communication without QoS guarantees may be rather meaningless. Because of these reasons we believe that the migration of application oriented functions into the network must adopt a state-full approach to be able to track and manage the state of the communication and the associated QoS.

Basically our proposal is to implement a session layer with the capability to maintain an end-to-end state of the communication that is more general than the single (TCP for instance) connection. The session can be spread over several connections and/or data flows, interrupted, retrieved, re-routed etc. and a session oriented protocol like SIP provides all the messages and primitives to perform these actions.

Therefore the session layer match very well with the need to manage a connection oriented communication between remote application entities with several general features. It is rather trivial to exploit the SIP protocol together with the implementation of a few ad-hoc modules to make the Service Layer support these functions, with a rather general approach that proves to be extensible and very flexible. The key point is the capability of SIP to carry whatever payload we like in the body of the signalling messages (such for instance the session initiation message INVITE) and of the SIP server entity to store "users profiles" i.e. information about rights of access, service policies etc [1].

Building on these capabilities the technical solution described in here is based on the following general concepts:

- users and resource are seen as SIP terminals (User Agents - UA) that can register to a SIP server entity (section 4.4.1) providing information on their network location and state at the time of registration;
- the SIP server entity keeps the information for later use, in particular for resource discovery and reservation;
- a service request is issued as a session set-up request, including in the payload of the signalling messages higher layer protocols that specify all the requirements of the application (type and amount of resources needed, including network resources);
- the SIP server entity will establish the session if it is possible to satisfy the requirements, possibly re-negotiating them.

How these actions are performed is a matter of implementation. Surely it is easy to understand that they can be accomplished if:

- one or more application protocols to exchange application related information between the terminals and the SIP server entity exists. Protocols are carried in the payload of the signalling messages and can be interpreted by ad-hoc modules in the server entity;
- is defined a model of interaction, together with the related protocols, to transfer the application requests to the network, in order to identify and reserve the network resources requested by the communications.

Before entering into the details of a possible implementation let us first review the three basic problems to be addressed: session management using the SIP protocol, interacting with the network control plane and interacting with applications by means of application level languages.

[1]SIP supports these two features oriented to the VoIP application that is the most typical field where it is used. Basically the SIP proxy must be able to authenticate a user registering to the proxy and the Session Description Protocol (SDP) [HJ98] is used as the application language used to specify session attributes.

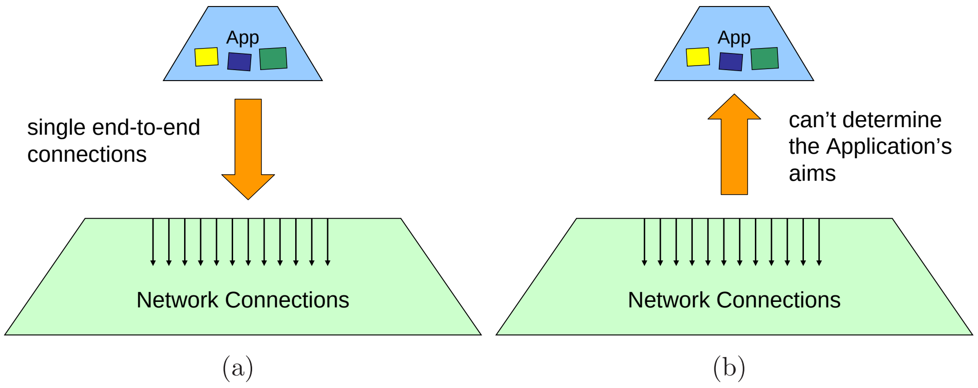

App
single end-to-end connections
Network Connections
(a)
App
can't determine the Application's aims
Network Connections
(b)

Figure 4.1: Layer interaction; (a) Application point of view (b) Network point of view

4.3.1 Service and Session plane

To avoid confusion and misunderstandings in the rest of the work , it is important to point out the differences between these two functional layers. The session plane is capable of maintaining in a single plane both application and network state. The interface to interact with the session plane is the signalling protocol (i.e. SIP). Obviously, because the service plane is responsible for activating and managing application and network resources in a single functional plane, the interface to interact with the service plane is the session plane.

As already outline, the aim of the Service Oriented Networking proposed here is to fill the gap between Application's functions and network's operations. On one side communication between applications is a logical entity that does not necessarily match with the transport facilities, to the other side communication in the network is based on "connections" by means of uncorrelated packets or streams. Figure 4.1.(a) shows a simple logical view of this situation where applications apply for end-to-end network connections without the capability to express any further requirements. On the other hand figure 4.1.(b) represent the same scenario from the network point of view where network provides connections without any ability of defining any service communications. In this scenario, the transport network generally does not know the applications requirements for communication while applications do not see the network connections.

This clean separation between application and network layer is at basis of the today's networks. Currently, a lot of effort was already spent to define some Quality of Service (QoS) mechanism in order to try to differentiate the communication's services. Indeed, today QoS is becoming more and more important since the network evolution trend is evolving in the direction of grouping every communication services in a single IP packet based network. Thus, a single packet base transport network should be enriched by the capability to guarantee quality and reliability suitable for a large variety of applications. However, the network QoS can only deal with the transport reservation mechanism defining a different treatment of the packet streaming, in other words QoS only extend the

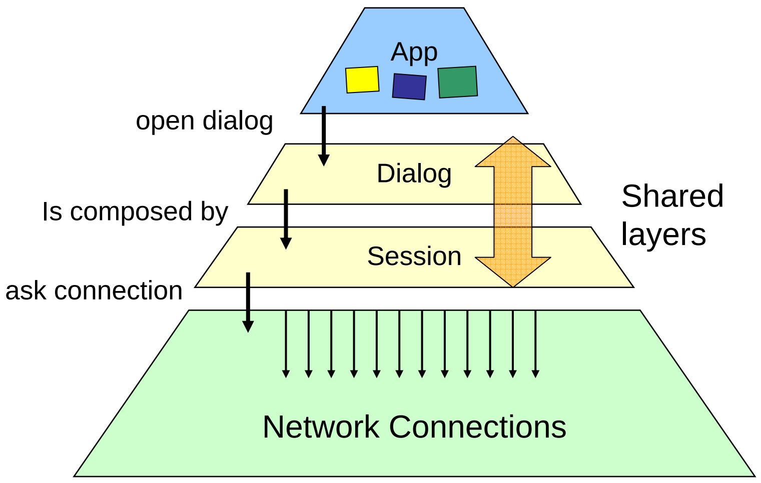


Figure 4.2: Logical view of the shared layer between Network and Appliaction

versatility of the transport network.

The Service plain aims to overcome these issues by defining interaction mechanisms between user, application and network. This goal is achieved by bringing the concept of communication session into the network. The network control plan is enriched by a session layer on top it. The session layer is able to keep the session state during the communication and share the state to both application and network. Moreover, the session layer has the capability to describe "the effect of the communication" in term of describing the logical and physical object involved in the "communication relationship". This characteristic becomes more and more important if we think that in a generalize Service Oriented Networking the final goal is the user experience. This means that any services must be treat as an object constituted by many different parts. The transport service and the application layer must be provided by shared "interfaces" in order to describe the overall status of the communication. In order to have a single logical service view, the session plane implements service logical layers.

Logical layers

Figure 4.2 shows two more logical layers between application and network environment, *Dialog* and *Session*.

The *Dialog* represent logically the service relation between end-points, in other words the Dialog is the logical wrapper of the communication, every interaction between the involved end-points is logically included in the same dialog with a unique dialog identifiers. The dialog represent the higher identifier of the communication which permits to logically group all the communication instances. For instance, an application open a dialog with one or more other applications, e.g. Video on Demand dialog, all the communication flows between the end-points, e.g. audio and video streaming, can be identified as streaming in the same logical context. This can be particularly useful in a distributed environment where applications and consequentially the data exchanged between the participants can be spread over different network portions. The capability to identify a group of communication messages under the same Dialog identifier can boost the semantic expressiveness of the network.

The second layer added represents the *Session*. Sessions are the logical entities representing the communication services delivered between end-points in a Dialog. Each Dialog can expresses one or more sessions which are distinct logical elements with different scopes. For instance a Video on Demand dialog can be composed by many different sessions, e.g. audio session, video session, chat session etc. All the sessions define communication services, this means that a session is a further logical wrapper where a group of connection are referred as a unique entity. Each sessions can be composed by one or many connections which are the logical representation of the packet streams in the transport network.

The Session plain can be accomplished by adding the two aforementioned shared logical layers between application and network environment, this is done in practice using a generic resource description language (chapter 5) and a common signal protocol (section 4.3.2). On the other hand, the Service plain is achieved by the using of the Session plain with functional elements able to interact with network control plane (section 4.4.1).

4.3.2 Session Initiation Protocol (SIP)

The Session Initiation Protocol (SIP) [RSC+02] is an IETF application layer protocol used for establishing and managing sessions. The concept of session is well known in networking but also in more general real life and is related to a set of activities performed by a user that can be logically correlated. In networks several exchanges of information (either in parallel or series) may be part of a single session.

The session may be manipulated by the user or the network according to the needs, for instance a session may be suspended, retrieved etc. SIP deals with session-oriented mechanisms, regardless the scope(s) of a session. It specifies the message flows required to initiate, terminate and modify sessions. In other words, SIP does not provide services but provides primitives that can be used to implement services on top of sessions. For example, SIP can locate a user and deliver an opaque object to its current location. It is also neutral to the transport protocol and can run on top of almost all existing protocols (TCP, TLS, UDP). Thanks to these characteristics SIP scales well, is extensible, and sits comfortably in different architectures and deployment scenarios. Because of these features SIP has become the core protocol of the IMS architecture [PMKN06] that promises to pave the path towards ubiquitous communication over heterogeneous networks.

SIP limits itself to a modular philosophy and focuses on a specific function set, thus maximizing interoperability with existing and future protocols and applications. In the perspective of this work SIP is used at its best to manage the sessions:

- sessions are used to handle the communication requests and maintain their state;
- sessions are mapped into a set of networking resources with QoS guarantees;
- the SIP protocol is used to manage the sessions, to provide user authentication, session set-up, suspension and retrieval, as well modification of the service by adding or taking away resources or communication facilities according to the needs.

For all these reasons SIP has been the chosen protocol to provide session management protocol in the Service plain (section 4.3.1). SIP is used to exchange information about service requirements, based on an Offer/Answer schema and Service states in both application and network layer.

How is it possible that SIP supports network related functions such as network resource reservation and QoS management when the network control plane uses primitives that are not directly accessible by SIP and may depend on the network technology. What is needed is a framework to describe the network resources in an abstract way in the SIP elements and then implement a mechanism that maps this abstract representation into control plane directive specific to the given networking technology. The framework must be general enough to be able to identify a wide variety of resources as well as resources' state. In section 6.6 the framework will be presented extensively.

4.3.3 Application Layer Protocols

Applications do communicate with their specific languages to implement a service instance. Protocols exist to express the end-user needs, find the resources a user wants and reserve them for use. Of course, the application protocols are related to a particular application.

For instance in the experimental activities 7 Grid application is used extensively as proof of concept application Consequentially, Grid application protocol is used by the end-users to find the computational resources needed and exchange the description of requirements of jobs to be dealt with by the computational resources. The Job Submission Description Language (JSDL) [ADF+05]) is the protocol currently existing to this end. JSDL is a language to describe the requirements of computational jobs in Grid environments. The JSDL language contains a vocabulary and XML Schema that facilitates the expression of requirements as a set of XML elements. The specification focuses on the description of computational tasks to be submitted to traditional high-performance computer systems. JSDL describes the submission aspects of a job as well as job state with the following general categories:

- Job identification requirements as Job name, description, etc.;
- Resource requirements, such as total RAM available, total swap available, CPU clock speed, number of CPUs, Operating System, etc.;

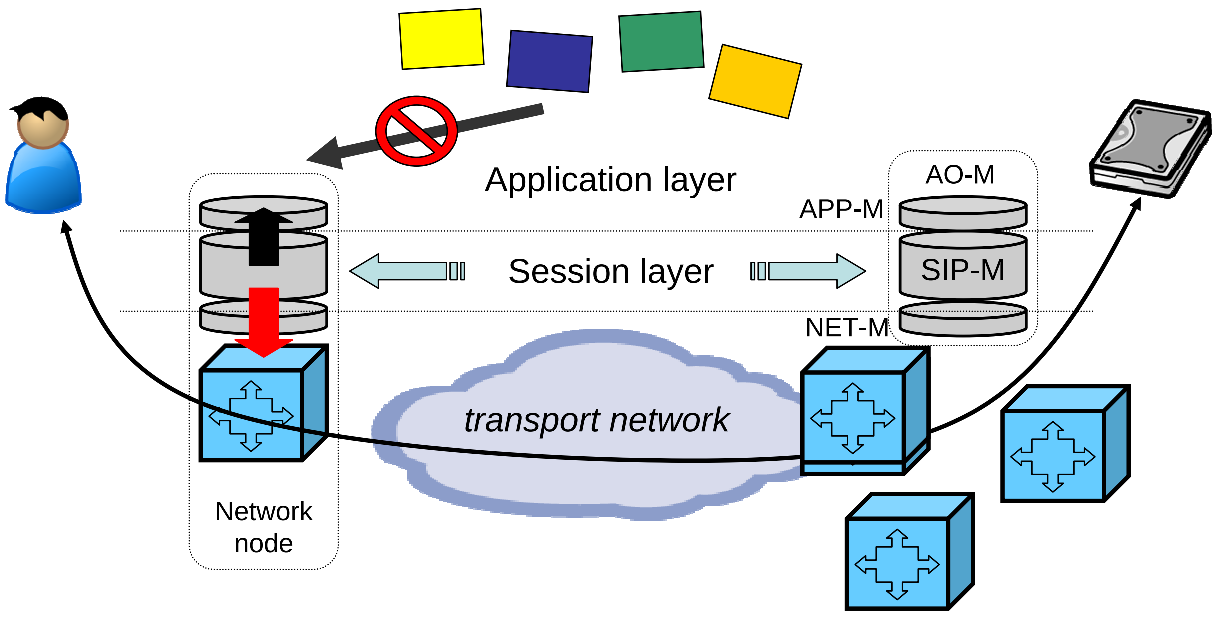


Figure 4.3: Service Oriented Networking Architecture

- Data requirement, such as the maximum amount of CPU time, wallclock time, or memory that can be consumed, etc.

JSDL does not cope with the network resources needed to support the communication related to a given grid service. JSDL will be encapsulated into SIP messages in order to activate Grid sessions.

4.4 Service Oriented Networking Architecture

This section describes the proposed architecture and the relevant building blocks. The logical architecture is presented in figure 4.3 where three logical layers are in evidence, an application layer application, a session layer managing the signalling and the network layer which provides the mapping between service and network requests by means of the network control plane. The coordination between the Session layer and the Application Oriented Modules realize the Service plane, that is the function plane of the Service Oriented Networking. From an external point of view, the interface to interact with the Service Oriented Networking, by means of its Service plane, is the Session layer.

It is important to point out that there is no any external interface to control the service or resource provisioning, a generic application or user can not invoke any Service Oriented Networking functions from outside, as shown in figure 4.3.

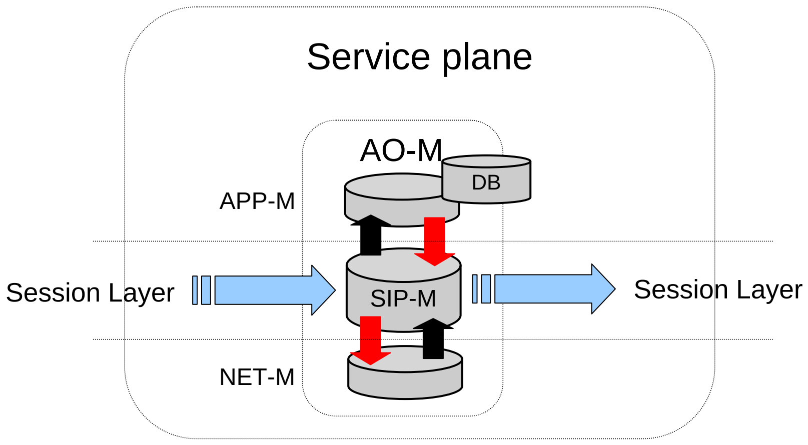


Figure 4.4: Application Oriented Modules

4.4.1 Application Oriented Modules

The Application Oriented Modules (AO-M) is the block responsible of the Service Oriented Networking. The AO-M represents the node where all application and network requests are routed, by means of the session plane, in broad terms the AO-M module is the network service element. It can be deployed stateless or statefull network module. The logical scheme of the AO-M is that shown in figure 4.4. It is logically sub-divided into three modules.

SIP module

SIP module (SIP-M) is built around a SIP proxy implementing standard SIP communication facilities, having the task to map the communication needs into sessions. SIP-M is, in same way, the hart of the AO-M, it is responsible for managing the Session Layer, find the next hop and forward any SIP messages to the destination.

The SIP-M is enhanced with interfaces towards the upper and lower module. Combining the communication capabilities of APP-M and SIP the AO-M may assist applications into publishing, searching and reserving resources 6, thus understanding the related communication needs and mapping them into sessions. Thanks to the “network oriented” module the SIP-M may trigger the network control plane, by means of the NET-M, into creating the connections required to transport the data flow according to the service profile of a given session.

Application module

Application module (APP-M) parses and partially understands the application protocols that may be encapsulated in the SIP messages. APP-M represent the application oriented part of the AO-M. The APP-M is equipped with a relational database where it is possible to store some kind of application or network data. When an application publish it own availability to the network through the AO-M the APP-M store the availability into its own database. With the same procedure the APP-M can also store the documents carried by SIP protocol into its body for describing the communication requirements. The APP-M's database became a container of all relevant network communication activities allowing an easy network management and monitoring.

Network module

Network module (NET-M) is able to interact with the network control plane in order to activate the transport striatum with a predefined service. The NET-M modules is the only technology dependent module in the APP-M. In fact the NET-M interact to the network control plane according to type of transport network, for instance GMPLS control plane. The functions relegated to this module are mainly oriented to the transport stratum activation and management. NET-M has a direct interface to the network Control plane to check and reserve resources.

The sequence of actions to be performed in the network module are mapped into:

- check by means of the network Control Plane if enough resources are available;
- reserve the network resources providing the QoS required;

Moreover, the NET-M adds transport information into the Network Resource Description Language 5, enriching the network descriptor with detailed information.

4.4.2 Architectural models

The SIP driven session layer and the network transport layer controlled by the network control plane can co-exist with different levels of integration, figure 4.5. It is resonable to define different architectural approaches considering the possibility to have two different trasport networks; a low bandwith legacy IP based network and an high bandwith (e.g. OBS, GMPLS, etc.) optical network. In broad terms, we support the possibility to have the signaling and the data carried by two physical distinct network. This assumption allow different logical and psysical integration models between the AO-M and the network equiopments.

Overlay model

Figure 4.5.a, where the legacy networks and the OBS transport network co-exist with functional separation.

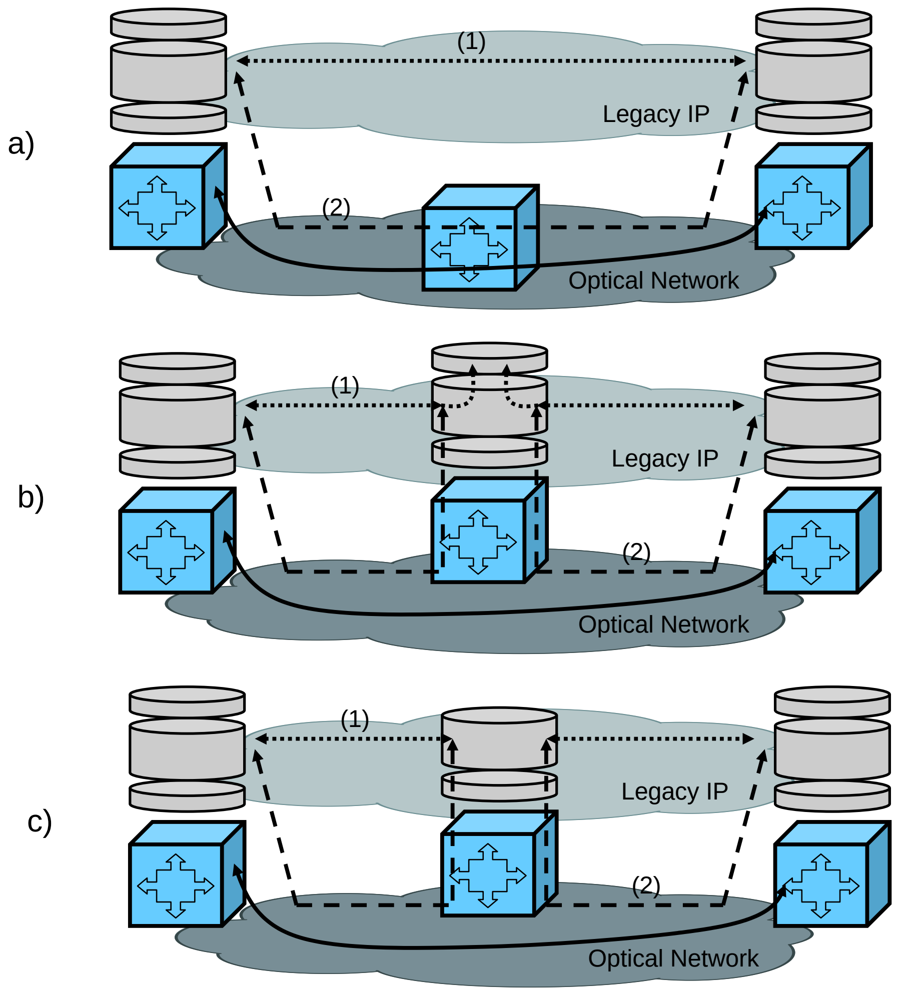


Figure 4.5: Architecture models: a) Overlay b) Partially Integrated c) Fully Integrated

- *Physical* (1). The Session layer, by means of SIP signalling, and the data are carried on separated infrastructures. The legacy IP network based on conventional electronic routing carries the SIP signalling while the high bandwidth transport plane is used to tranfer the data.
- *Logical* (2). The AO-Ms are placed into the edge routers only and the application resources and network resources are managed separately in an overlay manner. The users (i.e. the application) use the SIP protocol to negotiate the IT communication session. When the session is set the SIP-M triggers the NET-M that is responsible to request a data path between the edge routers involved in the session to the optical network control plane. Then the session data cut through the optical transport plane.

Partially Integrated model

Figure 4.5.b, no legacy networks are into play any more. SIP signalling and data share the same networking infrastructure

- *Physical* (1). All data flows are switched through the optical networks, either signalling (both Session and Transport signalling) or user data in a unified manner. The bandwidth available on the optical layer is completely shared between all communication needs.
- *Logical* (2). The optical control plane is enriched with SIP functionalities, to realize a pure network interworking with the session layer. All network nodes are equipped with fully Session layer functionality.

Fully Integrated model

Figure 4.5.c, between the two solutions just presented this solution still segregates most of the intelligence of the SIP layer at the boundaries of the optical network while exploiting physical integration.

- *Physical* (1). All data flows are switched through the optical network as in the integrated solution.
- *Logical* (2). The AO-M in the edge nodes are fully functional and logically identical to those mentioned before. On the other hand the AO-M in the core node is equipped with a subset of functionalities, to satisfy the best performance/complexity trade-off. For instance the AO-M could be limited to a SIP-M functioning as a light proxy with forwarding capabilities and to the NET-M with network resource management functions.

5 Network Resource Description Language

NRDL aims at expressing communications relationship and equipments virtualization in both the Service and the Transport stratum. NRDL wants to express network service virtualization and not only single network elements like equipments, link, data flow, etc. This follows the NGN 2.2 ideas of looking at the network as an infrastructure which is able to provide communication services instead of bare transport.

Here we need to cope with network resource. Therefore we need a generic syntax that is able to describe “communication needs”, general enough to provide network information exchange independently of the networking technology. NRDL is based on the Resource Description Framework (RDF) ontology model [MM04]. RDF is a general-purpose language, defined by W3C, providing a general method of modelling information through a variety of syntax formats, allowing the formalization of a wide variety of resources as well as resources state. In the case of interest here the RDF syntax must be able to describe communication needs, general enough to provide network information exchange independently from the networking technology.

It is important to point out that NRDL aims at defining a technology independent container for describing both services description and an exchange information protocol. The definition of every objects and devices is out of the scope of NRDL. Only the interaction between elements is the object of the language.

5.1 Logical Schema

The NRDL schema has been splitted in Figures 5.1, 5.2 and 5.3 while the NRDL schema is reported in appendix A.3. The schema consists of a set of classes and proprieties; classes are shown as ovals while properties are represented as labeled arrows. Below we give a short description of the classes.

The two main classes are:

- **ServiceStratum** represents the network service layer as outlined by the NGN ar-

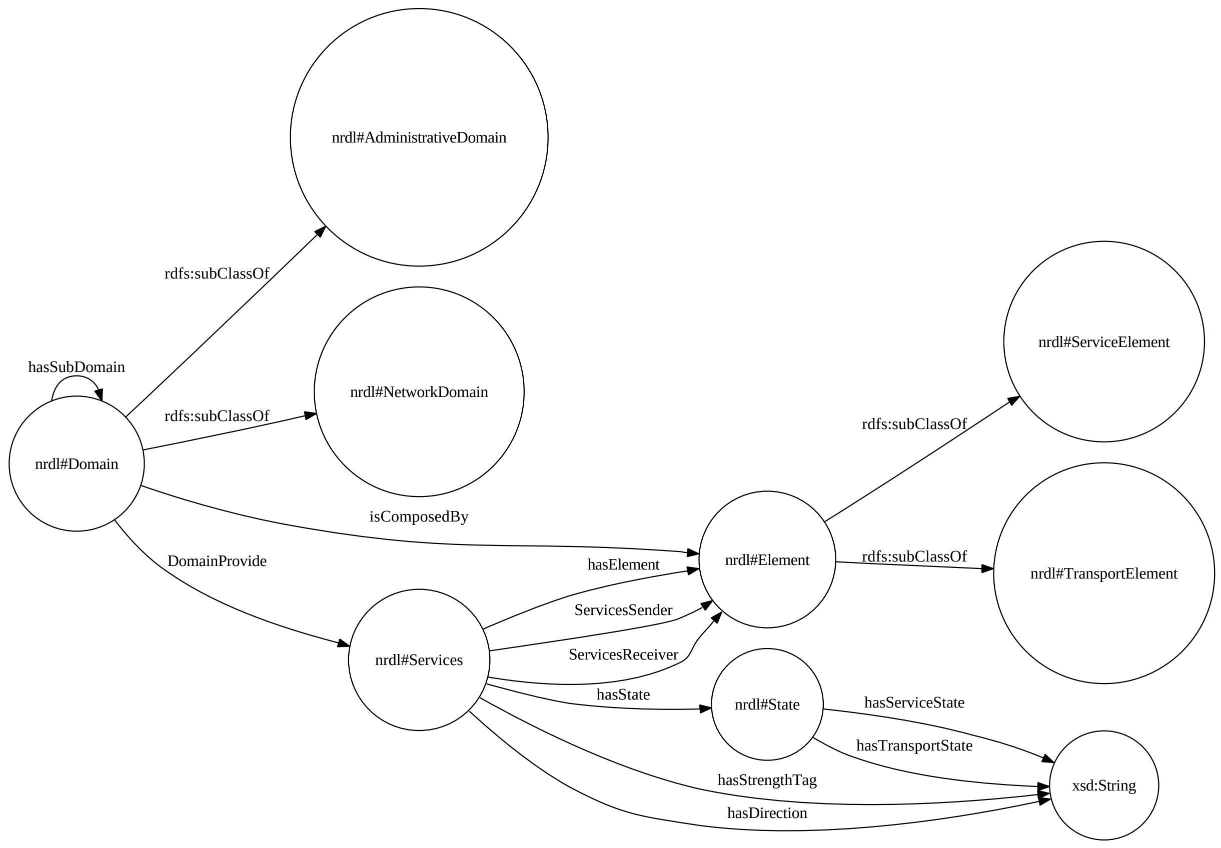


Figure 5.1: NRDF schema: Domains, Services and Elements

chitectures, in broad terms the Service stratum is a functional network overlay view;

- **TransportStratum** defines the elements and functions coupled to the network layer, in road terms the Transport stratum is a functional overlay view of the network control plane.

5.1.1 Domains and Elements

In figure 5.1 is represented the schema of the NRDL core part. The administrative entities are represented by the *Domain* class which is specialized in two subclasses; *AdministrativeDomain* representing the entity that manages a collection of service resources and *NetworkDomain* representing the entity that manage a collection of network resources. Both Administrative and Network domains can be further specialized using for instance the NDL Domain Schema [vdHDT+06] or other ontology. The *Domain* class collects a set of resources and can express a set of Services. Moreover, The property *hasSubDomain* permits to express domain partitions.

The resources are expressed by means of the *Element* class specialized by two subclasses; *ServiceElement* that collects all the resources in the ServiceStratum and *TransportElement* that defines the network resources in the TransportStratum. The detailed definition of the elements is out of the scope of NRDL. In fact, this part is technology dependent, every Domains can be composed by a wide variety of Service (i.e. SIP equipments, Web Services, HTTP server, etc.) and Transport elements (i.e. IP, ATM, Wirless, etc.) depending on the technology used and the services exposed. For example regarding the elements in the Transport stratum, the devices can be expressed by means of the NDL *Device* class.

Example

Regarding the NRDL schema (see Appendix A.3) a simple example is given on how the NRDL semantic can be used for describing a collection of elements in a realm. Because of the excessive length of complex NRDL file just very simple examples are listed below.

An administrative domain *tlc.deis.unibo.it* is composed by two distinct transport network: *Publicservice* and *StudentLab*. For sake of simplicity and without lacking in generality we suppose that *ServiceElement* is sub-specialized by 4 different classes: AO-M, HTTP server, VoD server and SIP elements (i.e. SIP proxy, Registrar and Locator server) (see listing 5.1). As previously outlined, we don't provide any details about objects or elements.

Listing 5.1: Service elements description

```
<rdfs:Class rdf:ID="AO-M">
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
```

```
    >AO-M</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Service"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="HTTPserver">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >HTTPserver</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Service"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="VoDserver">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >VoDserver</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Service"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="SIPelement">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >SIPelement</rdfs:label>
    <rdfs:subClassOf>
      <rdfs:Class rdf:ID="Service"/>
    </rdfs:subClassOf>
  </rdfs:Class>
```

The realm exposes an HTTP and VoD server in the *Publicservice* and has also an AO-M for those users interesting of having QoS sessions (see listing 5.2).

Listing 5.2: Example of Service elements composition

```
  <Network rdf:ID="PublicService">
    <isComposedBy>
      <AO-M rdf:ID="AO-M.tlc.deis.unibo.it">
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >AO-M.tlc.deis.unibo.it</rdfs:label>
      </AO-M>
    </isComposedBy>
    <isComposedBy>
      <HTTPserver rdf:ID="www.deisnet.it">
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >www.deisnet.it</rdfs:label>
      </HTTPserver>
    </isComposedBy>
    <isComposedBy>
      <VoDserver rdf:ID="VoD.deisnet.it">
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >VoD.deisnet.it</rdfs:label>
      </VoDserver>
    </isComposedBy>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >PublicService</rdfs:label>
  </Network>
```

While in listing 5.3 the *StudentLab* class is provided by a simple SIP testing network (i.e. SIP proxy, Registrar and Locator server).

Listing 5.3: Second example of Service elements composition

```
<Network rdf:ID="StudentLab">
  <isComposedBy>
    <SIPelement rdf:ID="locator.deisnet.it">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >locator.deisnet.it</rdfs:label>
    </SIPelement>
  </isComposedBy>
  <isComposedBy>
    <SIPelement rdf:ID="registrar.deisnet.it">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >registrar.deisnet.it</rdfs:label>
    </SIPelement>
  </isComposedBy>
  <isComposedBy>
    <SIPelement rdf:ID="proxy.deisnet.it">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >proxy.deisnet.it</rdfs:label>
    </SIPelement>
  </isComposedBy>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >StudentLab</rdfs:label>
</Network>
```

Following the same principle, the *Domain* subclasses can be specialized by detailed transport elements.

5.1.2 Services

In NRDL, the set of network Services are represented with the *Services* class which can be specialized by subclasses depending on the particular service that the network can provide (i.e. QoS, VPN, LSP, etc...). Figure 5.1 doesn't show any service subclasses because the aim of this work is to provide a logical schema of a possible resource description language and not a formalized implementation for a specific service. For instance, the NDL schema could express network services dealing with implementation of a lightpath, etc. The *Services* is connected by a property to the *Element* class in order to specify which resources are engaged in the service provisioning.

For instance, it is possible to list the equipments involved in a VoD service: AO-M and SIP proxy in the service layer while two routers and one firewall in the transport layer. See the NRDL example in listing 5.4.

Listing 5.4: Example of a list of equipments involved in a VoD service

```
<Services rdf:ID="VoD1234">
  <hasElements>
    <Firewall rdf:ID="Firewall1">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Firewall1</rdfs:label>
    </Firewall>
  </hasElements>
  <hasElements>
    <Router rdf:ID="Router2">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Router2</rdfs:label>
    </Router>
  </hasElements>
  <hasElements>
    <Router rdf:ID="Router1">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Router1</rdfs:label>
    </Router>
  </hasElements>
  <hasElements>
    <AO-M rdf:ID="AO-M.tlc.deis.unibo.it">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >AO-M.tlc.deis.unibo.it</rdfs:label>
    </AO-M>
  </hasElements>
  <hasElements rdf:resource="#proxy.deisnet.it"/>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >VoD service</rdfs:label>
</Services>
```

All services are equipped by a service strength tag to express the resource reservation priority, which defines the significance of the service. The strength tag is a key parameter because defines the importance of the communication. The service strength tag is defined by the *hasStrengthTag* property and can have the following values:

- None: no resource reservation is needed;
- Optional: the session can continue regardless of whether or not the service provision is possible;
- Mandatory: session establishment can not continue without the service resource reservation;
- Required: the equipments involved in the service provisioning must have the capability to reserve resource, but the reservation is Optional;
- Obligatory: if and only if the equipments have the capability to reserve resource, the resource reservation is mandatory; otherwise if the equipments involved in the

service provisioning don't have the capability to reserve resource the session can continue.

Obviously, any communication service can be reserved uni-directionally or bi-directionally. The direction attribute is stored in the *hasDirection* property and it can have values like *send*, *recv* or *sendrecv* to indicate the flow direction. Each services has also a service status, which represent the communication progress. Because the NRDL impose a layering in the communication, the communication *State* class has two properties; *hasServiceState* which represent the communication state in the service stratum and *hasTransportState* which represent the communication state in the Transport stratum.

Example

As a simple example we can consider a generic one way VPN connection between router $R1$ and router $R2$. This VPN connection is considered mandatory. The connection is running (i.e. connection state is in progress) and two switches ($S1$,$S2$) are also involved in the VPN connection along the path, see listing 5.5.

Listing 5.5: Example of the description of VPN network services

```
<Services rdf:ID="VPN1234">
  <hasDirection rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >send</hasDirection>
  <hasState>
    <State rdf:ID="State5678">
      <hasTransportState rdf:datatype="http://www.w3.org/2001/XMLSchema#
          string"
      >progress</hasTransportState>
      <hasServiceState rdf:datatype="http://www.w3.org/2001/XMLSchema#
          string"
      >none</hasServiceState>
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Working Status</rdfs:label>
    </State>
  </hasState>
  <hasStrengthTag rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >mandatory</hasStrengthTag>
  <hasElements>
    <Switch rdf:ID="S2">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Switch2</rdfs:label>
    </Switch>
  </hasElements>
  <hasElements>
    <Switch rdf:ID="S1">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Switch1</rdfs:label>
```

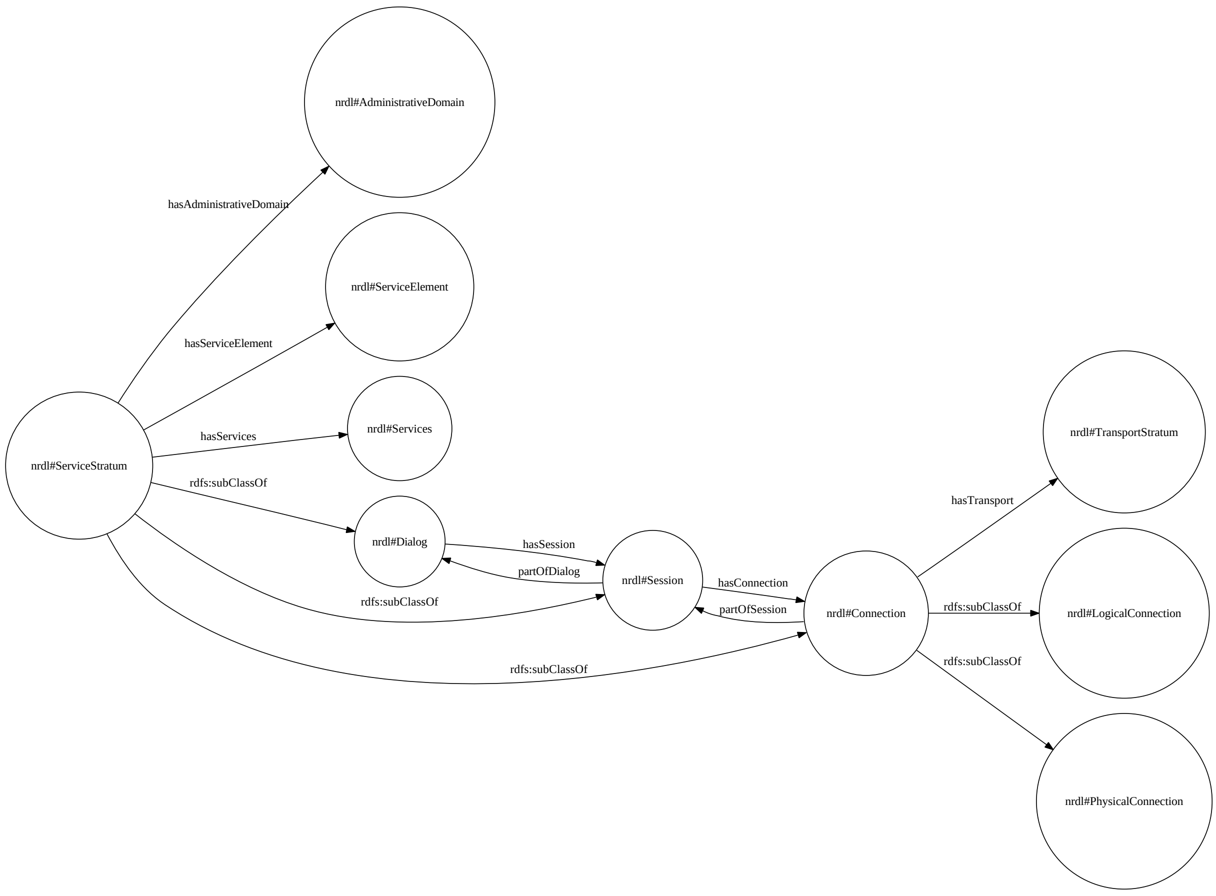


Figure 5.2: NRDF schema: Service stratum

```
    </Switch>
  </hasElements>
  <ServiceSender>
    <Router rdf:ID="R2">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Router2</rdfs:label>
    </Router>
  </ServiceSender>
  <ServiceReceiver>
    <Router rdf:ID="R1">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Router1</rdfs:label>
    </Router>
  </ServiceReceiver>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >VPN service</rdfs:label>
</Services>
```

5.1.3 Service stratum

Referring to the *ServiceStratum*, figure 5.2, a communication between two end-points can be specialized in three different subclasses:

- *Dialog*; the dialog represents logically the service relation between end-points. For instance, a multimedia dialog relates two multimedia applications. Within the Dialog one or many sessions can be established.

- *Session*; the session is the logical entity representing the services provided into the dialog. For instance, a multimedia dialog could be composed by Audio session, Video session etc... Each session could be independent by the others and could be treat separately. Sessions are also composed by one or more connections.

- *Connection*; the connection is the logical path in which the data are exchanged. For instance, Audio session could have data streams and a control channels. The connection is generally represented by an IP address, port and transport protocol. Following the nowadays Internet paradigm connections are isolated packets flows without any correlation.

All these three subclasses have proprieties in order to specify the communication layer relationship. Moreover, the *ServiceStratum* class has property to be related to:

- *AdministrativeDomain*; define the Administrative entities of the Service stratum;

- *ServiceElement*; addresses the network resources used in the Service stratum;

- *Services*; collects the Services provided by the Service stratum;

The *Connection* subclass can be also specialized in logical and physical connections. *LogicalConnection* subclass reflects the overlay view of the connection defining the connection between logical entities while *PhysicalConnection* subclass define the real route path storing the physical resources used. The architecture of the NDRL schema permits the representation of many layers, see the layering capability of NRDL (see section 5.2).

Example

Let us consider a Dialog (Dialog1234) composed by a VoD service (VoD5678) with two distinct sessions (Session1359,Session7853). Each session is specialized by a service (i.e. Video6245 and Audio9378) and a physical connection. The physical connection represents the container of transport service (i.e. QoS1634, VPN1234). The NRDL file of this simple example is listed in appendix A.1.

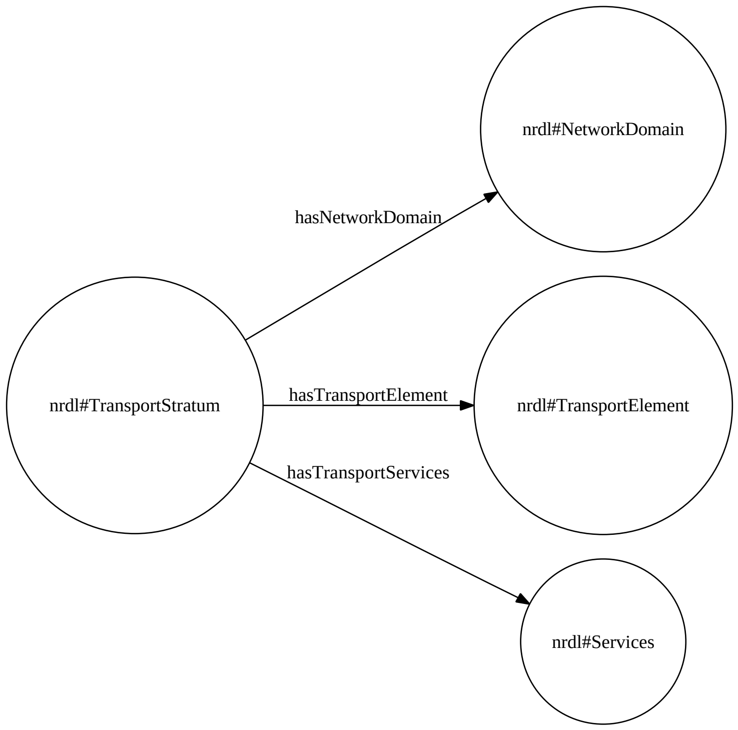


Figure 5.3: NRDF schema: Transport stratum

5.1.4 Transport stratum

Connection subclass is engaged by a property with the *TransportStratum* class, figure 5.3, in order to map the Connections to the transport resources.

In fact, *TransportStratum* class has property to be related to:

- *NetworkDomain*; define the network entities of the Transport stratum;
- *TransportElement*; addresses the network resources used in the Transport stratum;
- *Services*; collects the Services provided by the Transport stratum;

5.2 Language applicability

The NRDL can be used with two different scopes:

- *Services Description Language*: reflecting the current ontology model, NRDL is able to describe generic objects and/or resources like network logical or physical equipments, connections or network services. Thanks to the RDF, which NRDL is based on, the ontology expressiveness follows the semantic web paradigms. Each

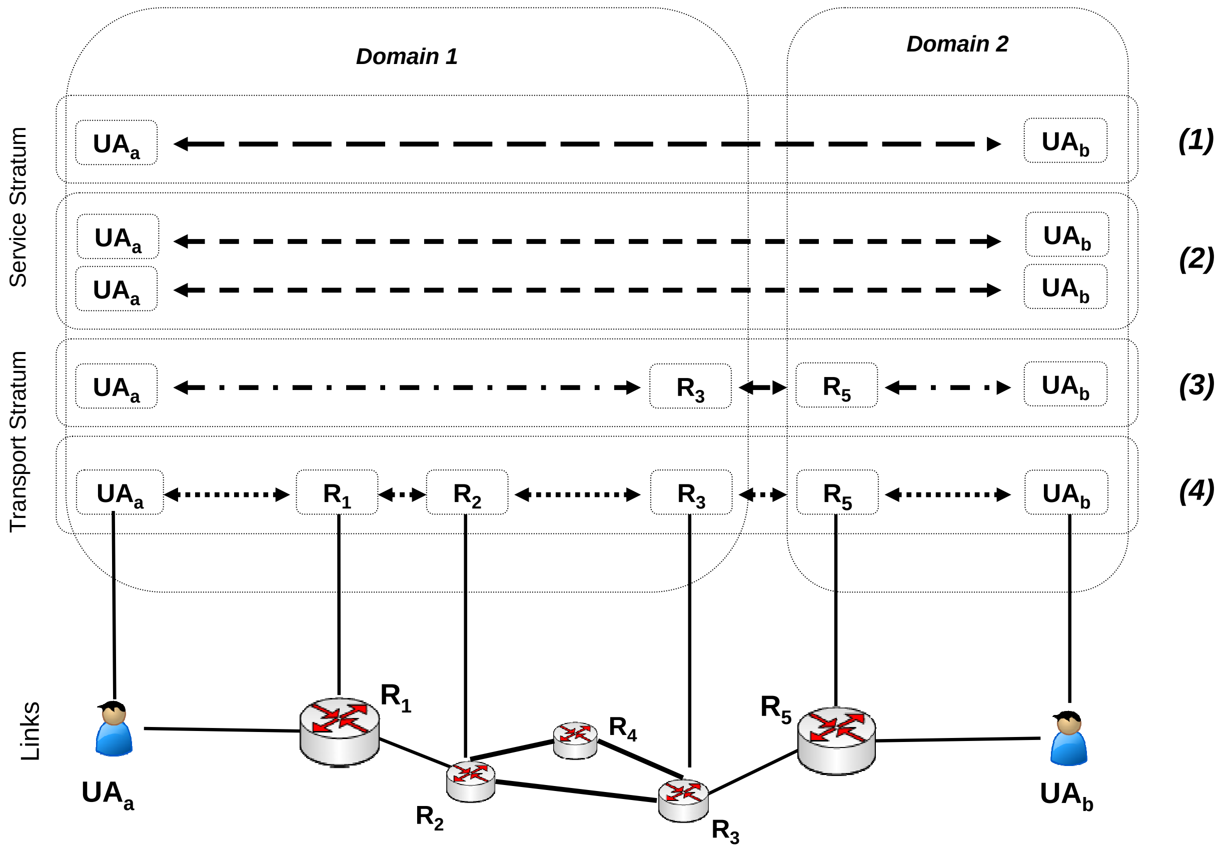


Figure 5.4: NRDF layering capabilities

communication entities can be described by means of NRDL ontology and can be stored in a relational database for easy monitoring and consultation.

- *Network Protocol*: exploiting the NRDL descriptive characteristics it is possible to generate descriptive files regarding generic services requests. On the other hand, using the same semantic it is possible to provide descriptive files of services requests states. This NRDL peculiarity can be exploited in order to generate services offers and answers. Thanks to the layering capabilities of NRDL, the services expressiveness can be related to any overlay or physical connection in both service and transport stratum.

In general NRDL can be use in every situation in which a communication services must be perform with the interaction of service and transport stratum. In broad terms NRDL is a language, which describes the effect of communication in both layers giving a point of contact to the network service.

5.2.1 Network service description and monitoring

In figure 5.4 a simple layering example between two User Agents, belonging to different domains, is given. For instance, UA_b is providing a service to UA_a. This service opens a dialog between the two end-points, figure 5.4.1, the dialog is composed by two different sessions, figure 5.4.2.

The sessions could generate a bulk of connections in the transport stratum, for sake of simplicity only a single connection is shown. The connections are data blocks streamed across different domains. As a consequence of that, the connection is spitted in:

1. a domain logically view, figure 5.4.3;
2. a physical network view, figure 5.4.4, where data are sent though the network equipments.

Referring to the description capabilities of the language, a single NRDL document is able to describe the entire example presented. Thus, a set of NRDL documents could be used to collect and represent the relevant communication activities into the network. In particular, any connection, either logical or physical, can be express by means of NRDL documents which can be stored in the AO-M relational database representing a statefull descriptor of the current network services.

This approach is in line with the NGN structure where transport and service stratum are separated but the network services are the result of coordinating activities between the two stratums. An expamle of the logical connection (only for Domain1) presented in figure 5.4 is listed in appendix A.2.

5.2.2 Network services activation

NRDL can also be used as simple network protocol to exchange information and to activate network services. The NRDL schema has been structured as offer/answer protocol. Figure 5.5 shows a logical view of the network services activation process. In this example we use concepts and objects already described, in particular we will refer to the AO-M, see 4.4.1, as end-point to show how the NRDL activation process works. UA_a prepares a NRDL document describing the network services from its point of view, thanks to the NRDL layering capabilities, UA_a can define the network services at different physical and logical levels (i.e. Dialog, Session, and Connection).

Then, UA_a encapsulates the NRDL document in a state-full connection and sends the document to the UA_b, figure 5.5.1. Regarding the state-full connection, we suggests to use a subscription mechanism in order to exchange NRDL document between two or more participants. In particular we suggest to use the SIP subscription mechanism for creating a resource management framework, as reported in section 6.6. During the path to the destination, the connection goes thought the AO-Ms in the Service stratum. Each AO-M can extracts and parse the NRDL document in order to verify if the User Agent

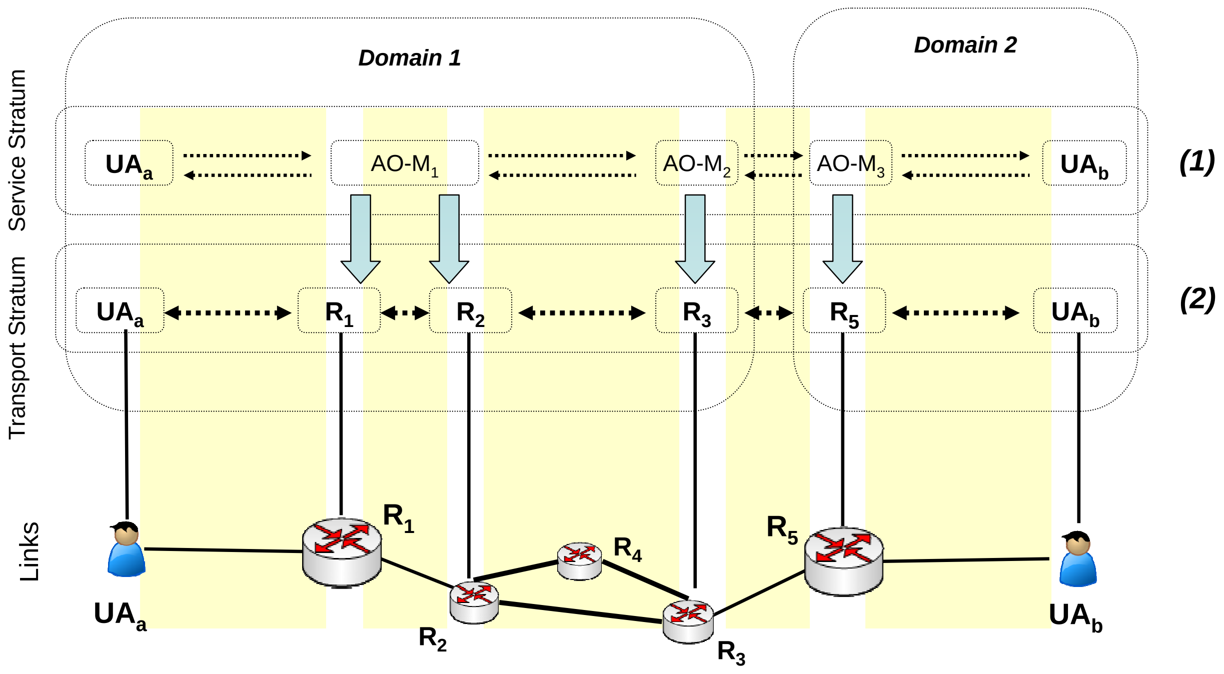


Figure 5.5: Logical view of the services activation chain

has requested network services in its network or administrative domain. If so, the NET-M must provide the requested services, according to the service strength tag presented in the NRDL. In order to do this, each NET-M prepares the network triggering the appropriate equipments (R_i) in the transport stratum (figure 5.5.2). Then, the NET-M can enrich the NRDL document with detailed information about the service, for instance adding physical information about the connections.

The updated NRDL document is forwarded to the proper destination (i.e. another AO-M or final end-point), by means of another connection (i.e. SUBSCRIBE message), and finally, with the network service offer, at the destination, UA_b. UA_b extracts and parse the NRDL in order to verify the network services requested by UA_a and accordingly updates the NRDL document as answer. The answer represent the acknowledgment and describes the network services from the UA_b point of view. The updated NRDL document is sent back to the UA_a using the same path. The NRDL document goes thought the same AO-Ms and allows the network services activation in the opposite direction. Obviously, because network services can be composed in several steps the information exchange between the two UAs can continue until the network services are established or finally cancelled.

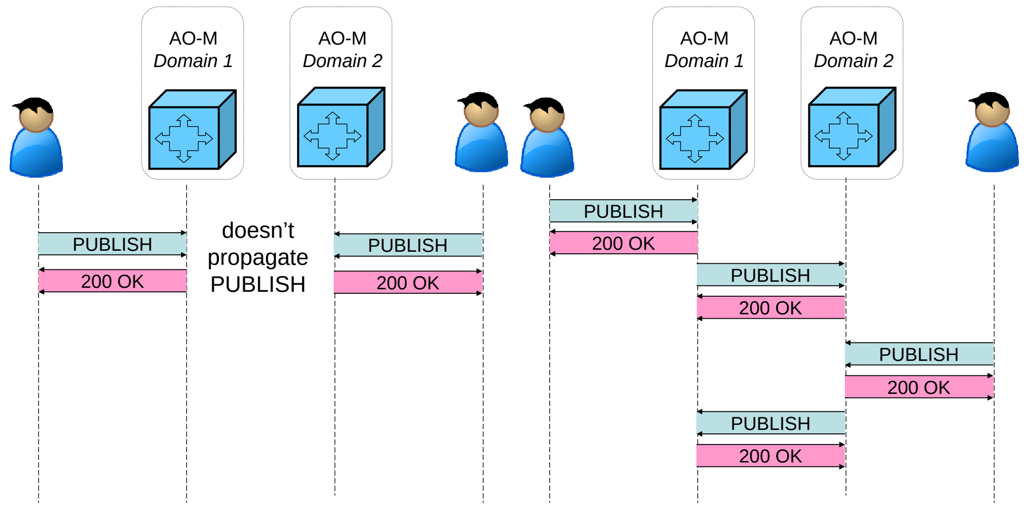
The network services activation process is represented by a subscription chain along the service path. Each network portion (both logical or physical) is treat as an independent subscription service where service requests and notify answers remain bounded to a single network portion. NRDL is the collector of all these portions showing up the entire session status. By means of the NRDL every nodes are informed about the entire session status but each node deals only with a single resource reservation network portion. As previously outline, this network centric approach guarantee a minimum users involvement in the QoS reservation process. The end-users must only describe their QoS needs in the NRDL and let the network provides them.

6 Resource Management

The resource management concerns all operations regarding the resources information exchange (i.e. publication and discovery) as well as resources reservation. As already outlined, we propose to use the session layer with the SIP signaling to support resource management in an integrated manner within the network. To this end we assume that resources and users are divided into domains. A domain is an organization that is particularly indicated in a real networks oriented to consumer applications where a large number of resources and users can be present. With the general term of resources we intend both application and network resources. Since this research is applied to Grid Networks the application resources (i.e. storage space, computational power, etc.) will be addressed as general Grid applications. On the other hand, network resources are all the resources concerning the network layer (bandwidth available, etc.). Thus, referring to a generalized resource, the proposed resource management operations follow in principles three different phases:

- *Resource publication:* each available resource must publish its own availability to the AO-M. The publication phase allows the collection of network and application resources into the network nodes equipped with AO-Ms.
- *Resource discovery:* allows users to find a set of available resources into the network including their network location and terms of availability.
- *Resource reservation:* allows the reservation of both application resources in the application domain, and network resources in the network domain. Resource reservation can follow different paradigms depending on the network capability to reserve resources with a defined QoS and depending on the interaction between application and network reservation. The interaction model between application and network is envisaged in the following.

The resources are managed by processors behaving as SIP User Agents, i.e. are able to perform the typical signaling of a standard SIP terminal. In the following we discuss how the various phases can be implemented using SIP.



(a) Localized approach (b) Distributed approach

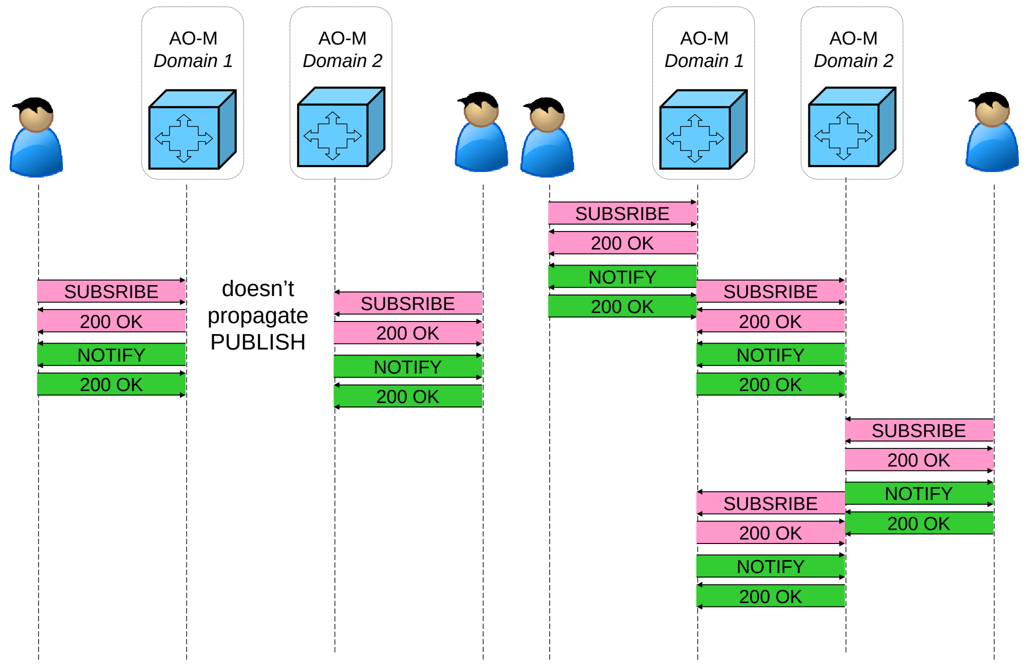
Figure 6.1: Schematics of the resource publication

6.1 Resource publication

Resources (application and network) are announced to the AO-M by means of a SIP PUBLISH message, figure 6.1, where a standard SIP message encapsulates into the body a resource descriptor. The resource descriptor must address the resource (i.e. resource's SIP address), capability, availability and state. To this purpose a NRDL document (see section 5) can be successfully used as a generic resource descriptor. When the state of a resource changes, the SIP UA responsible of the resource prepares a NRDL document reporting the detailed resource state. The document is encapsulated in a SIP PUBLISH message and sent to the AO-M. Here it is parsed by the SIP-M and the NRDL payload in the body is sent to the APP-M where updated resource capability and availability are stored into the APP-M database.

Depending whether the information must be propagated in the network the PUBLISH message may be forwarded to adjacent AO-M or not. Depending on the domain two approaches are possible, *Localized* or *Distributed*.

Every time the state of a particular resource changes, the SIP UA responsible of the resource can update the resource status by means of a new PUBLISH message. Obviously to avoid network overload some sort of time windowing can be implemented to limit the number of PUBLISH messages since resources may change their state very rapidly. Moreover in order to avoid scalability problems an intelligent trade of between Localized and Distributed approach should be done in real networks.



(a) Localized approach (b) Distributed approach

Figure 6.2: Schematics of the resource discovery

Localized approach

Figure 6.1.(a) In order to avoid uncontrolled messages propagation, PUBLISH messages can be forwarded only in the UA's Home domain. A PUBLISH message can not reach any AO-M in a different domain or subdomain. This approach is particularly indicated in domains with few resources where the probability to find a free resource is acceptable.

Distributed approach

Figure 6.1.(b) In order to spread the resources state into the network SIP PUBLISH messages can be propagated to any AO-M nodes, even crossing SIP domain borders. To avoid unnecessary messages if a PUBLISH arrives in an AO-M that already know the resource with the same parameters, the AO-M module sends back a SIP error response and the PUBLISH propagation stops in that node. The distribute approach should be used only in a federated inter-domain environment where predefined agreements can restrict uncontrolled messages propagation.

6.2 Resource discovery

Any SIP UA can request a set of particular resources exploiting the SIP protocol presence notification mechanism [Roa02]. A resource is associated with a SIP address (i.e. a UA of the SIP network) and has a set of proprieties with a state. A SIP UA sends a request by issuing a SIP SUBSCRIBE message, figure 6.2 to the AO-M, the characteristics of the request are included in the SIP message body by means of a description language.

The description language must have the capability to express a set of requirements in both network and application domain. Again the specific description language used is not a major matter since the APP-M can be extended to understand any new application descriptor. As mentioned before, without lacking in generality, we assume that a generic application protocol (e.g. JSDL for Grid application) and network protocol (e.g. NRDL) can be embedded in the SIP SUBSCRIBE message.

Depending on where the database of the available resources is maintained the SUBSCRIBE arrives to the closest AO-M node where the message is parsed by the SIP-M module. The message body is forwarded to the APP-M where it is processed. The APP-M starts a seeking phase into the resources database where all the published resources are stored. Whatever the requested resources are found or not, the AO-M node send a NOTIFY message back to the UA, figure 6.2. The NOTIFY message encapsulates the NRDL which describes the position and the availability of a free set of resources or an information massage. The information massage notifies the UA about the state of the research with a detailed semantic information. For instance, an information massage can report information like: seeking not complete, seeking in progress, resource not available, resource not present, etc In this work, we will not go though a detailed description of the information massage. The two approaches allowed in the resource discovery follow in principle the approaches used in resource publication, Localized and Distributed.

Obviously, the NOTIFY message are effected by congestion control issues. In fact, an update NOTIFY message is sent from the AO-M modules to the used every time the state of the subscribed resources change. Congestion control mechanism should be apply to cope with messages overload.

Localized approach

Figure 6.2.(a) SUBSCRIBE message can not be forwarded out of the Home Domain or Subdomain. Using the Localized approach the probability to find a free resource is reduced, on the other hand since the SUBSCRIBE massage is not effect by the propagation phase, the final NOTIFY response is given in a shorter period. Moreover, the Localized approach guarantee a fully controlled environment with trusted resources.

Distributed approach

Figure 6.2.(b) In order to have a high probability to find a free resource the SUBSCRIBE message can be propagate by any AO-M nodes to any Domains or Subdomains. The SUBSCRIBE distributed approach is effected by the same issues advised in the resource publication section 6.1. Moreover, because every search phase is followed by a notification one, by means of a SIP NOTIFY messages, the waste of bandwidth and consequentially the network congestion can became intolerable if the Distributed approach is applied widely.

6.3 Resource reservation

The resource reservation will be performed by means of the SIP request used to established a dialog between two UAs [RSC+02]. This is the time when the network QoS issue should also be addresses, since the reservation could be subject to the availability of given network resource. To make the presentation more effective, addressing one problem at a time, this section does not deal with QoS. The QoS management issue will be dealt with in the dedicated section that will follow (see section 6.4). Therefore for the case here presented the reservation will be provided by a standard SIP INVITE without the SIP extension for the QoS [J.02].

Encapsulated into the INVITE there is either an application document (for instance JSDL describing Job requirements) or a NRDL document or even both. The INVITE message goes though the network and arrives at the UA destination, that acknowledges the request by sending back a 200 OK positive response. Otherwise a response message error is raised depending of the resource state. A SIP ACK message follows to close the INVITE transaction. The SIP INVITE message can be used to implement resource discovery and reservation in two different approaches.

6.3.1 Single phase

The single phase approach is presented in figure 6.3, where both resource discovery and reservation mechanisms are performed at the same time. In some way the single phase approach defines an anycast procedure for reservation, in fact, the user with data to be processed remotely sends a request to the closest AO-M in the form of an INVITE SIP message. Encapsulated into the INVITE there is a description language describing the request, for instance in Grid network is the JSDL document which describes job requirements, figure 6.3.(b). Obviously, the first phase is to publish Grid resources to the AO-M, by means of a SIP PUBLISH message (figure 6.3.(a)) as already explained in section 6.1.

The SIP-M module passes the job request to its application middleware (APP-M) that performs a resource discovery algorithm to find out whether there are enough computing

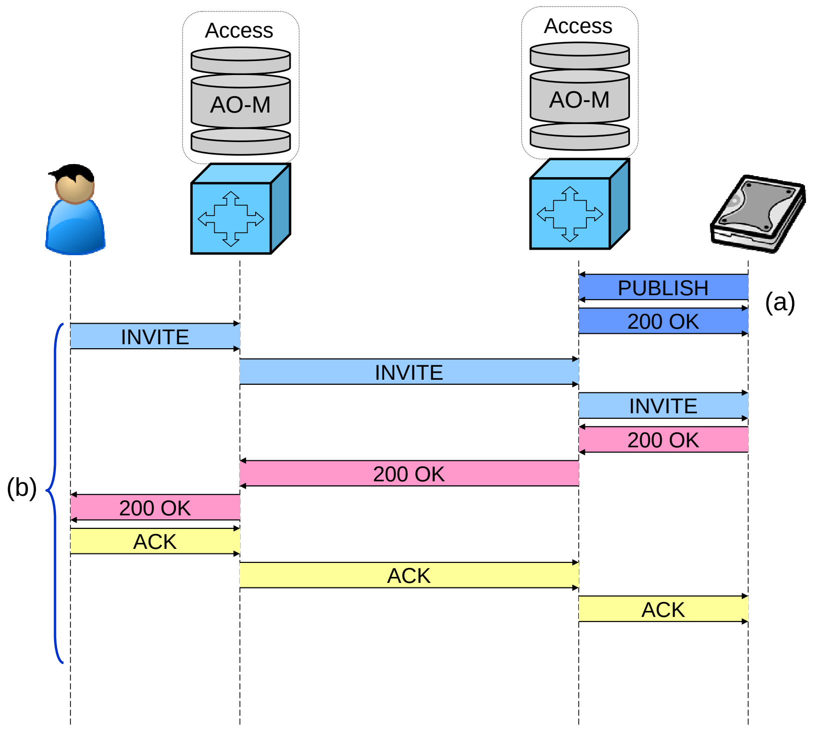


Figure 6.3: Schematics of the single phase resource reservation approaches

resources available within its known resources. If the answer is “yes” the AO-M would start establishing the session, if it is “no” it would forward the message to the other AO-M module in the network to look for the requested resources until the message arrive to an available resource or it is dropped. If a resource is found, the INVITE is acknowledged and a session between user and resource is created, thus reserving the resource usage. Depending on the interaction model between network and application many resource reservation with qos requirements can be applied (see section 6.4).

6.3.2 Two phases

The two phases approach is shown in figure 6.4. At first a resource discovery with a notifications mechanism is performed then a direct reservation by the client follows to complete the reservation. In this approach the Grid resources are published to the attached AO-M with a PUBLISH message, Fig 6.4.(a). A user requesting resources sends a SUBSCRIBE message to the nearest AO-M. The AO-M checks the status of its own resources and if they can satisfy the request then notifies the client. Otherwise, the SUBSCRIBE message is propagated to the other known AO-M (with a Localized or Distributed approach) either by utilizing sequential or parallel forking in order to discover the requested resources, figure 6.4.(b). The AO-M with available resources sends a NOTIFY message back to UA.

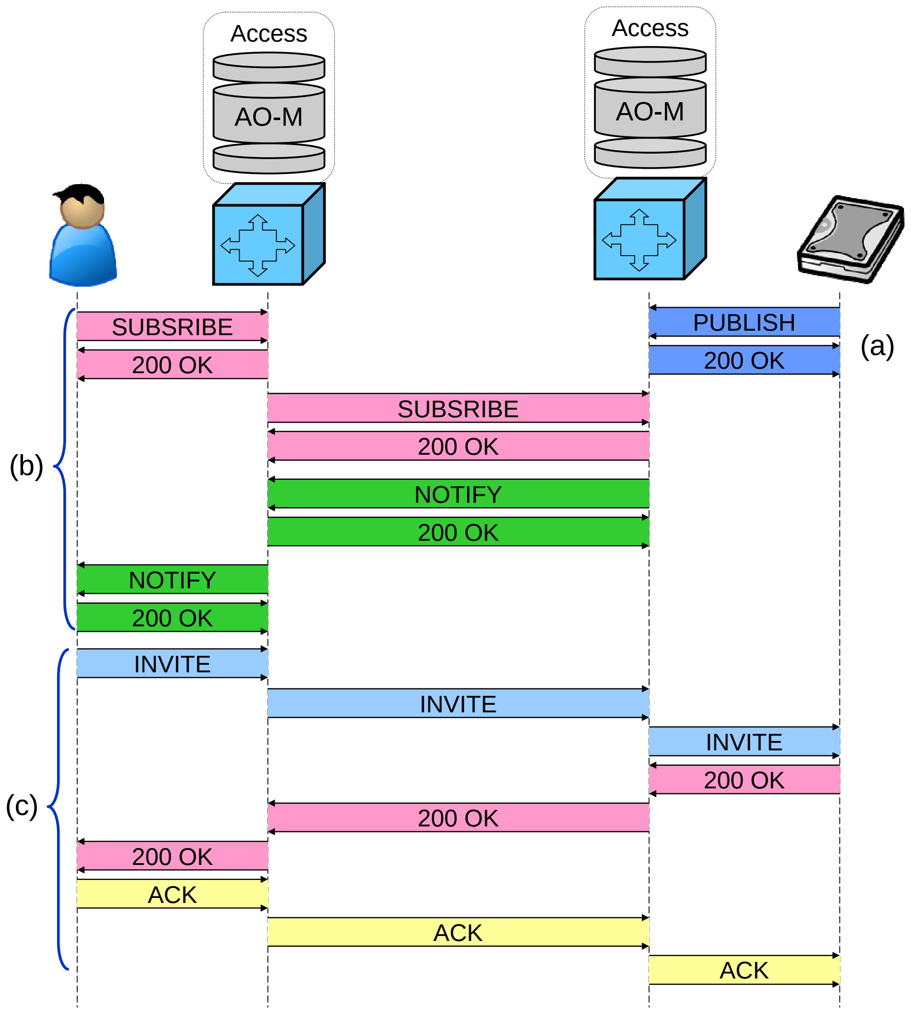


Figure 6.4: Schematics of the two phases resource reservation approaches

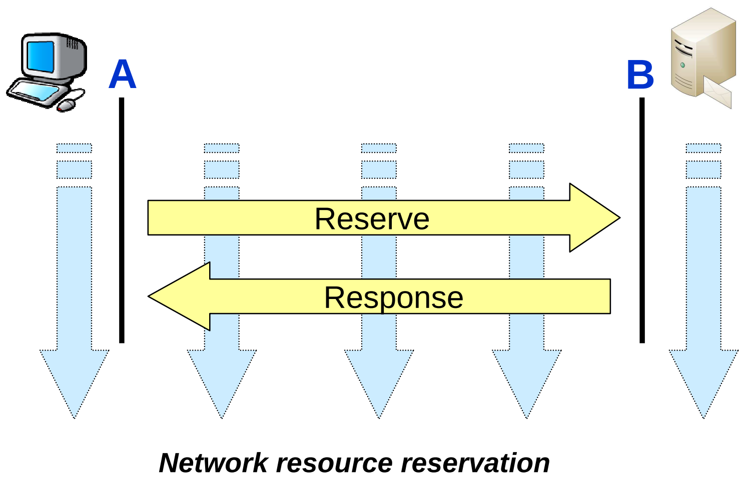


Figure 6.5: QoS operations overview

The NOTIFY is used to communicate to the UA the availability and location of the resources (i.e. address of the end point or AO-M or domain). After the resource discovery the user knows the location (SIP name or network address) of the resource and can attempt a direct reservation by an INVITE message, figure 6.4.(c)

6.4 Resource reservation with SIP QoS Management

Nowadays, two main models are possible to finalize SIP QoS resource reservation Framework [J.02]:

- The end-to-end model, requiring that the peers have the capability to map the media streams of a session into network resource reservations. For example, IP Multimedia Subsystem (IMS) [PMKN06] supports several end-to-end QoS models and terminals may use link-layer resource reservation protocols (PDP Context Activation), Resource ReSer-Vation Protocol (RSVP), or Differentiated Services (DiffServ) directly. Anyway, today the option of using RSVP has been removed (or is not used by anyone in practice) and DiffServ is being used as transport QoS mechanism (mapping PDP context QoS information to DSCP parameters).

- A core oriented approach where the transport network already provides QoS oriented service classes (for instance using DiffServ) and the sessions are mapped directly into this classes by the network itself.

The approach presented here is in line with the current reservation models, but we extend the current SIP QoS resource reservation scheme in order to achieve a communication service QoS Framework, section 6.5.

The main issue is how the search and reservation of application and network resources are combined in time during the session start-up phase. The importance of this is due to the fact that the completion of the session could requires alerting the end-user to ask whether the session is acceptable or not (phone ringing in a VoIP call for instance). Whether this has to be done before, while or after the network resources required for a given QoS communication are reserved is a matter to be discussed and tailored according to the specific service.

It is worth mentioning that for communications spanning over multiple domains, QoS support is not straightforward. Each administration entity involved along the path has the responsibility of reserving resources in its controlled subnet. Thus, reservation must be performed in a multi-hop fashion where each administration is responsible for reserving resources and guarantee QoS, depending on both the demand and the agreements between domains. In the remainder of this work we do not consider the multi-domain issue, that will be subject of further investigation, and focus on QoS guarantee within a single network domain.

As already outlined, a generic "Communication service" involves both network and application layer, figure 6.5. Depending on the type of application and network two different layers can be addressed.

The former, can be represented as an horizontal application layer where specific request and response application messages can be used for reserving application resources. In fact, from an application point of view, the reservation mechanism is based on a proper end-to-end messages exchange in order to negotiate the service. In particular, the sequence of logical actions to be performed in the application layer are the following:

- *check* if the application resources are available;
- *reserve* the resources for the desired session;
- *set-up* the application session and consequently prepares and starts the data streams.

The latter layer is related to the QoS activity of the network. It can be represented as an obscure vertical layer which guaranties an QoS path between the participants. From a network point of view, the QoS activation is a vertical flow. In fact, the control plane is the only one responsible for the network activation and management, consequently any interaction with the network moves from the network control plane to the data plane. In particular, the sequence of logical actions to be performed in the network layer are the following:

- *check* by means of the network Control Plane if enough resources are available
- *reserve* the network resources providing the QoS required
- *set-up* the network data flows

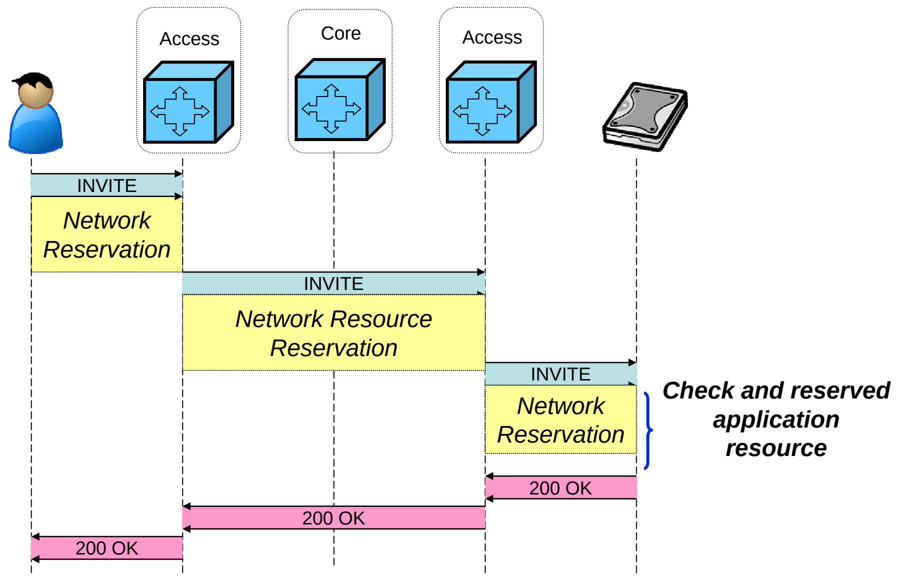


Figure 6.6: Network reservation during session set up

As already outlined, end users may interact with the network control plane but to date this interaction can not be general and technology independent. Nonetheless is possible to imagine interaction models between the application and the network layer to guarantee the establishment of a session with network QoS guarantee. In the following, given that a session set-up phase is always started with an INVITE message sent by the caller to the callee, some examples of different approaches are given exploiting the SIP messages syntax.

Network reservation during session set up

Figure 6.6. While the INVITE message goes through the network (e.g. with anycast procedures) a network path is reserved before the INVITE message reach the resource destination. Network reservation is performed step by step and always before application reservation. Each network portion is independent and reserved on the fly during the INVITE path. This method can be used for very fast provisioning purpose.

Network reservation before application reservation

Figure 6.7. As soon as the INVITE message arrives at the destination and application resources are checked, the network reservation starts. After the network resources are reserved the application resource is reserved and the session started. In this case, the network reservation is performed before application reservation as a whole network path. Moreover, the application domain must wait the end of the entire network reservation before reserving any resources. This is the case of VoIP calls.

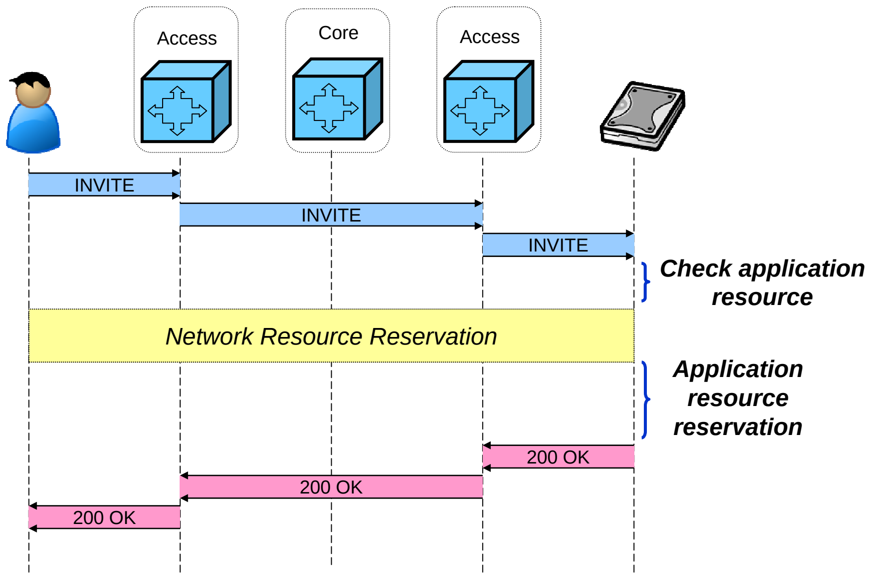

Access
Core
Access
INVITE
INVITE
INVITE
Check application resource
Network Resource Reservation
Application resource reservation
200 OK
200 OK
200 OK

Figure 6.7: Network reservation before application reservation

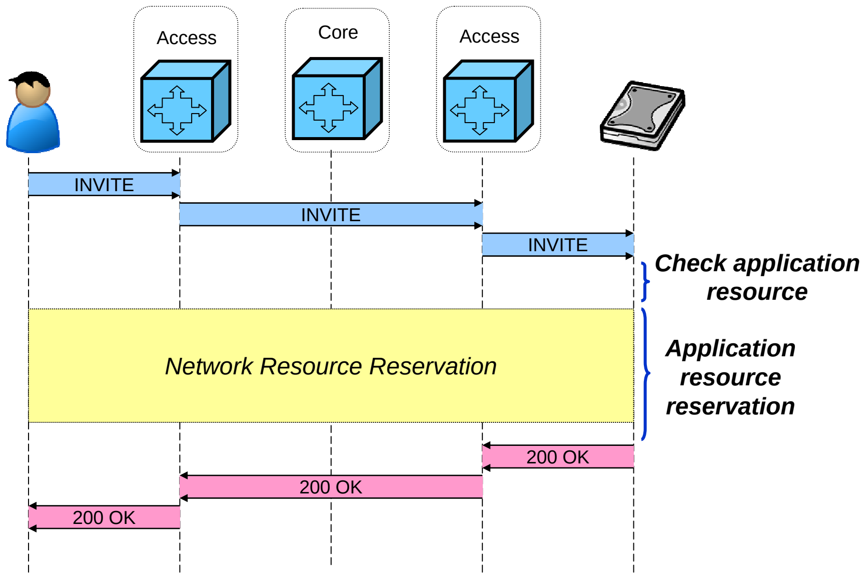

Access
Core
Access
INVITE
INVITE
INVITE
Check application resource
Application resource reservation
Network Resource Reservation
200 OK
200 OK
200 OK

Figure 6.8: Network reservation before session set up

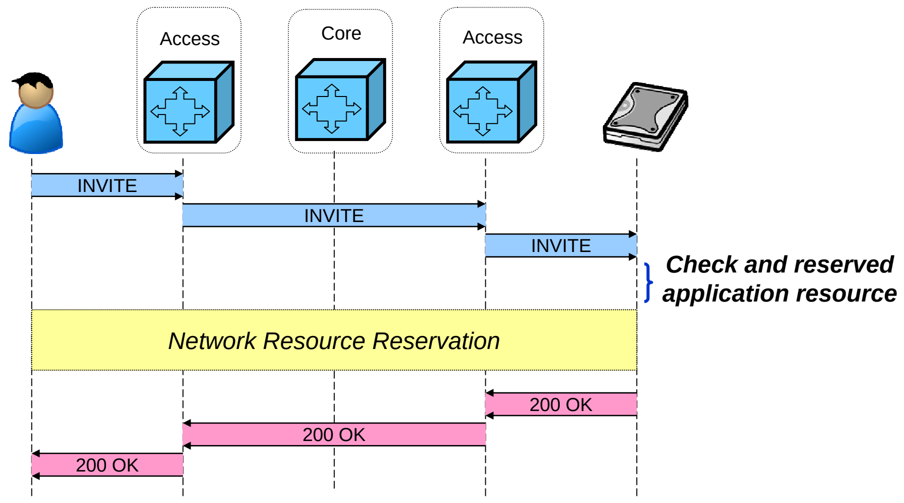

Access
Core
Access
INVITE
INVITE
INVITE
Check and reserved application resource
Network Resource Reservation
200 OK
200 OK
200 OK

Figure 6.9: Network reservation after application reservation

Network reservation before session set up

Figure 6.8. As soon as the INVITE message arrives at the destination, and the application resources are found, the application and network resource reservations are started in parallel. Once both reservations are completed independently and both the application resource and the network path are available the session is started. In this case the whole session can starts only when both application and network resources have been reserved. This can be the case of standard GRID sessions.

Network reservation after application reservation

Figure 6.9. The INVITE message arrives at the destination and application resources are checked and reserved. Then, network reservation starts into the network and as soon as the path between user and resource is established, the application session is started. This can be used when few application resources are available and the probability to find a free resource application is low.

Network reservation after session set up

Figure 6.10. The INVITE message arrives at the destination and application resources are checked and reserved, immediately the application session can start without having any network resources reserved. Then, network reservation starts and as soon as the path between the participants is established, the application session already started can take advantage moving the transmission state from best-effort to QoS enable. This can be used when the QoS is not crucial for the application (or at least not in the beginning part of the session).

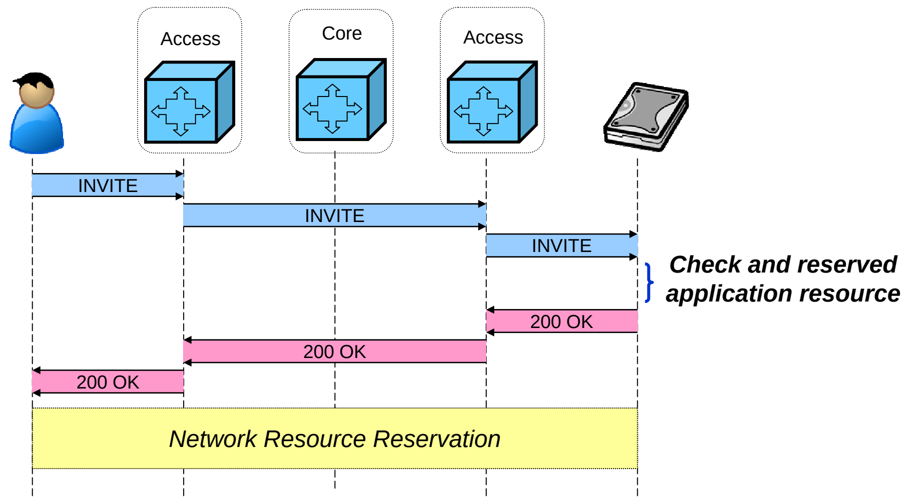

Access
Core
Access
INVITE
INVITE
INVITE
Check and reserved application resource
200 OK
200 OK
200 OK
Network Resource Reservation

Figure 6.10: Network reservation after session set up

6.5 SIP Resource Management framework

The management of the QoS issue is not part of the standard SIP protocol but the issue is there and, a Resource Management framework was defined for establishing SIP sessions with QoS.

6.5.1 Standard framework

The current standard framework proposed in [J.02] exploits the concept of pre-condition, by means of extended SDP parameters [HJ98]. The pre-condition is a set of "desiderata" that are negotiated during the session set-up phase between the two SIP UAs involved in the communication. If the pre-conditions can be met by the underlining networking infrastructure then the session is set up, otherwise the set-up phase fails and the session is not established.

The pre-conditions simply requires that participants use end-to-end existing resource reservation mechanisms to reserve what is needed before establishing the session (e.g. RSVP, PDP context activation etc.). The framework requires the SIP user agents to reserve network resources before establishing the session, so that the user is not alerted of the incoming communication if the network resources required are not available.

This scheme is mandatory because the reservation of network resources frequently requires learning the IP address, port, and session parameters of the callee. Moreover, in a bidirectional communications the QoS parameters must be agreed between caller and callee. Therefore the reservation of network resources can not be done before the exchange of information between caller and callee has been finalized. This exchange of information is the result of the initial offer/answer message exchange at the beginning of the session start up. The information exchanged set the pre-conditions to the session. If the pre-conditions can be met the session is then established.

6.5.2 Weakness of the current standard framework

The current framework says that the QoS pre-conditions are included into the SDP message, using two state variables: *current status* and *desired status*. Consequently, the SIP UA treats these variables as all other SDP media attributes [16]. The current and desired status variables are exchanged between UAs using offers and answers in order to have a shared view of the status of the session. The framework has been proposed for VoIP applications and the chosen solution is tailored to this specific case. Consequentially it has some limitations when considering a generalized application aware environment, enumerated in the following.

- Both session and quality of service parameters are carried by the same SDP document. Thus, a session established with another session protocol (i.e. Job Submission Description Language (JSDL)) is not possible.
- The pre-condition framework imposes a strict "modus operandi", since pre-conditions must be met before alerting the user. Therefore network reservation must always be completed before service reservation.
- Since the network reservation is performed by the UAs at the edge of the network, only a mechanism based on end-to-end reservation is applicable.
- QoS is performed on each single connection, without the ability of grouping connections resulting from sessions established by others.
- Since SDP is a protocol based on lines description, it has a reduced enquiring capability. Moreover, only one set of pre-conditions can be expressed by SDP. Multiple negotiations of pre-conditions are not possible.
- The SDP semantic is rather limited. It is not possible to specify detailed QoS parameters since the SDP lines are marked with "qos" parameter without specifying any additional detail (i.e. delay, jitter, etc...).

Given these issues, we believe that an extended resource management framework is needed in order to achieve a general framework for QoS for application oriented network for supporting advantage applications, such as Grid computing, and network reservation. In section 6.6 we propose an extension to the framework to generalize its applicability in a generalized Service Oriented Networking.

6.6 The extended SIP Resource Management framework

The extension to the existing framework can be implemented exploiting the following ideas:

- keep the QoS mechanisms out of SIP protocol. The specific end-user application will manage the interaction model, see section 6.4.
- do not limit the framework to use only SDP protocol but allow more general protocols in the SIP payload at session start-up;
- separate the information about the request and requirement related to the application layer from the information about request and requirements at the network layer, using two different protocols to carry them;
- allow as many re-negotiation as possible both at session start-up and while the session is running.

Application and Network have different timings and mechanisms. The resource reservation in the application layer generally has a simple messages exchange based on request and response mechanism. On the other hand the network Control Plane could need a number of operations in order to established QoS path resulting in a significant delay.

Because of all these reasons, the extended Resource Management framework for QoS is based on an approach like *network service subscription*. This different approach is motivated by the wiliness of keeping the QoS framework out of the current SIP mechanism for reservation. Regarding the protocols to declare session requirements we propose to make a distinction between application and network protocol. In the following:

- Application Description Protocol (ADP): is used to describe application requirements. Obviously, it is application-dependent, for instance JSDL which is a consolidate language for describing Job submission for Grid networks.
- Network Resource Description Language (NRDL): is used to describe network requirements and QoS requests. The use of a complete different document descriptor for QoS allows the use of many ADPs for describing application requirements, and not only the SDP.

The NRDL is a general method of modeling information and has the capability to describe both resources and the resources state. For this reason NRDL can be used instead of SDP as a more general description language for network resources. Furthermore, because NRDL is a structured language, it is possible to enrich its semantic at will with the detailed information about QoS (i.e. Bandwidth, jitter, delay, etc...). The meaning of using NRDL for describing network requirements is motivated by maintaining the QoS approach technology independent.

The main advantage of this approach is that the network does not need the capability to understand the ADP since the QoS requirements are only described in the NRDL. The main drawback is that a link between ADP and NRDL is needed; and therefore some sort of “interface” has to be implemented in the UA on top of a standard SIP UA.

The extension is implemented exploiting concepts and messages already present is SIP [Roa02]:

- SUBSCRIBE message: The general concept is that every SIP entities (i.e. UAs or AO-M) in the network can subscribe to network resources or call state for various network resources, and those entities can send notifications, by means of NOTIFY messages, when those states change.
- NOTIFY message: a message that allows the exchange of information related to a state of a particular subscribed network resource. Any UA or AO-M can act as notifier sending a NOTIFY message to a subscriber to inform the subscriber of the state of a resource.

Both SUBSCRIBE and NOTIFY messages encapsulate into their body the NRDL which gives semantic expression to provide detailing information about network services as well resource state.

The NRDL is general enough to provide network information exchange independently of the networking technology, both as services description language and service negotiation protocol based on an offer/answer communication paradigm. Thanks to the layering capabilities, the NRDL acquires the function of describing the “communication needs” as effect in both Service and Transport stratum.

The generalized SIP QoS mechanisms proposed here is based on two strict separated mechanisms for:

- Application Reservation, by means of an INVITE message;
- Network QoS Reservation, by means of a subscription mechanism (i.e. SUBSCRIBE message)

Two distinct approaches, called Implicit and Explicit, are possible depending on the fact that resource reservation in both application and network layer is performed separately or in conjunction.

6.6.1 Implicit approach

The Implicit approach follows in principle the concept of “pre-condition” known from the current SIP resource management framework where QoS is achieved by an INVITE

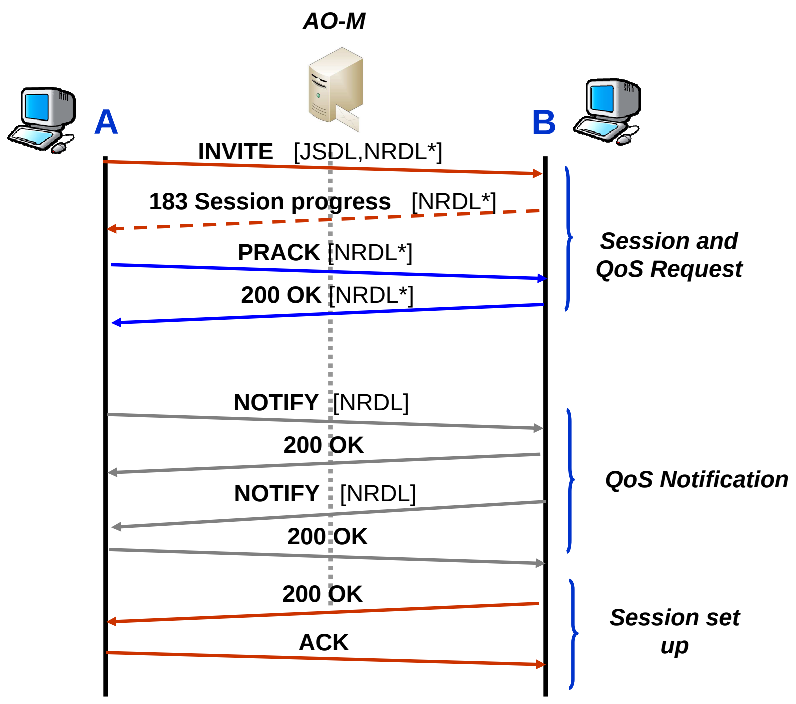


Figure 6.11: Extended Resource Management framework: Implicit approach

mechanism subordinated to a network resource reservation as precondition. This approach which is in line with the current standardization has the merit to extend the current framework to every ADP and not only to a SDP descriptor. The Implicit approach is based on an INVITE multi-body message able to carry more than one protocol in the payload.

In figure 6.11 is presented an overview of the call flow in an end-to-end scenario where the network resource reservation is a pre-condition to the session set up. User A sends an initial INVITE multi-part message including both the network QoS requests and an application protocol for the application requests. The QoS requests are expressed by means of an NRDL document while the application protocol depends on the type of application (i.e. SDP for VoIP, JSDL for Grid computing, etc...)

A does not want B to start providing the requested service until the network resources are reserved in both directions and end-to-end. B starts checking if the service is available, if so it agrees to reserve network resources for this session before starting the service. B will handle resource reservation in the $B \rightarrow A$ direction, but needs A to handle the $A \rightarrow B$ direction. To indicate so, B returns a 183 (Session Progress) response with an NRDL document describing the quality of service from its point of view. This first phase goes on with the two peers exchanging their view of the desired QoS of the communication, thus setting the “pre-conditions” to the session in term of communications quality. Then both A and B start the network resource reservation. When an UA finishes to reserve resources in one direction, it sends a NOTIFY message to the other UA to notify the current reservation status by a NRDL document in the message body. The NOTIFY message is used to specify the notification of an event. Only when the network channel meets the pre-condition B starts the reservation of the desired resources in the application domain, and session establishment may complete as normally. Once the service is ready to be provided B send back a 200 OK message to A. Data can be exchanged by the two end-point since both application and network resourced have been reserved. BYE message close the dialog.

The main differences with the current approach are:

- use of a separated NRDL document in an multi-part INVITE message. The INVITE message starts a SIP dialog between parties. The NRDL can be carried by requests like INVITE or PRACK and by provisional (i.e. 183, etc.) or final responses (i.e. 200).
- use of NOTIFY messages instead of UPDATE. Since ADP and NRDL are separated (even if carried by the same INVITE message) the UPDATE message can be sent only by UA A and can be used to update both ADP and NRDL. The NOTIFY can be sent by both UAs and is related to NRDL only and guarantees that the issues of network resource reservation is addressed independently from that of application resource reservation. The NOTIFY messages have Dialog-ID, From and To tag equal to the INVITE message.
- The INVITE message implies the opening of a logical relationship between the peers (a subscription) that ends with the dialog tear down. Each time that the network reservation state changed a NOTIFY message is used to notify the changes for a specific subscription.

6.6.2 Explicit approach

The Explicit approach makes a clear separation between application and network reservation. As extensively shown in section 6.4 different approaches depend on the type of application, as a consequence the reservation schema must be chosen tailored to the specific application needs. The Explicit approach deals only with networking issues while application reservation depends on the application type. In figure 6.12 is shown the logical call flow. It is easy to distinguish two separate flows: the first regards application session, by means of proper reservation messages. The second deals with network QoS reservation, by means of SIP subscription mechanism. These interaction of these two distinct phases are managed by the user end point.

A prepares a NRDL document specifying the "effect" of the network service requested and starts network reservation process sending a SUBSCRIBE message to the AO-M

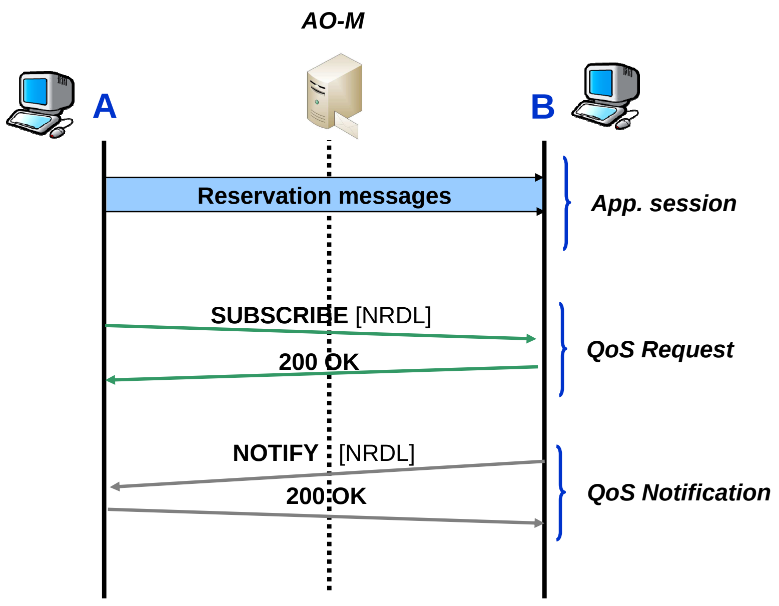


Figure 6.12: Extended Resource Management framework: Explicit approach

coupled to him (e.g. the user's gateway). The NRDL document is encapsulated into a SUBCRIBE message, which is used to create a subscription to the network control plane. In other words, the starting point of the network reservation phase is the creation of a NRDL document which is the statefull reservation descriptor and the use of a statefull subscription mechanism as information exchange protocol in the network session plane.

The AO-M receives the message, by means of the SIP module, extracts and parse the NRDL, by means of the NET-M module and finally activates the transport stratum, by means the NET-M interface to the network control plane. The network dependent part of the NET-M sends configuration messages to the control plane in order to activate the QoS reservation described in the NRDL. The messages exchanged between NET-M and network CP can be device dependent or more likely NETCONF messages [Enn06].

During this phase the SIP-M modules has to inform the User Agent *A* about the current reservation state. To this purpose, the SIP-M module prepares a NRDL document collecting the information about the service layer, this document specify in details the state of the overall communication as well as the state of the local reservation phase. The NRDL document is encapsulated into a NOTIFY message and sent to the caller. Every time the state of a resource changes the SIP-M modules sends another notification message to update the caller point of view resource state. This mechanism defines a network reservation chain as described in section 5.2.2 exploiting the subscription mechanism of the SIP protocol. A SUBSCRIBE message with the parameter "expires" equal to zero delete the network resource reservation and deactivate any network control plane reservation.

7 Experimental Activities

This chapter reports the experimental activities conducted in order to validate the concept of the Service Oriented Networking. Mainly, the tests have been conducted in collaboration with the University of Essex in England and with the Scuola Superiore degli Studi di Perfezionamento Sant'Anna in Pisa. All the software used for the experiments has been developed by the University of Bologna while the test-bed implementation is the result of coordinated activities with the aforementioned University. Two main optical networks have been used as proof of concepts: a full functional OBS and GMPLS network.

The first group of tests have been conducted implementing a real life SIP-OBS test-bed. The work stands as a successful international collaboration. The AO-M, by means of a single simple SIP-M module, was developed by the University of Bologna from scratch and the Optical Burst Switching test-bed implemented at the University of Essex. Thanks to staff members mobility the two parts were integrated into a single full functional test-bed and the various concepts previously described experimented.

Basically, two main experiments are described in this first group of tests; the former had the aim to validate the concept in terms of feasibility and consistency of signaling timing using a generic Grid application (section 4.2.1); the latter wants to demonstrate a more realistic situation where an high definition VoD application (section 4.2.1) is requested and the SIP-OBS network delivers it accordingly. Because of its significant complexity, the OBS test-bed has a rather simple network topology with two edge nodes and a single core node. In practise this topology does not provide any significant challenge in terms of networking complexity, therefore it was not meaningful to try a comparison of the various logical architectural alternatives mentioned earlier in section 4.4.2. Nonetheless the full functionalities of the concepts described have been tested and their feasibility proved.

The second group of tests have been conducted developing a more sophisticated and advantage AO-M on top of a GMPLS network in the Sant'Anna's laboratory. The second experiment have shown that the network can be made "service oriented" rather easily by exploiting existing technology and modules. Moreover, we have also shown that the architecture proposed is well in line with the ITU-T Next Generation Network (NGN) specification, see section 7.2.1.

In this second experiment we have experimented a VoD application with stringent

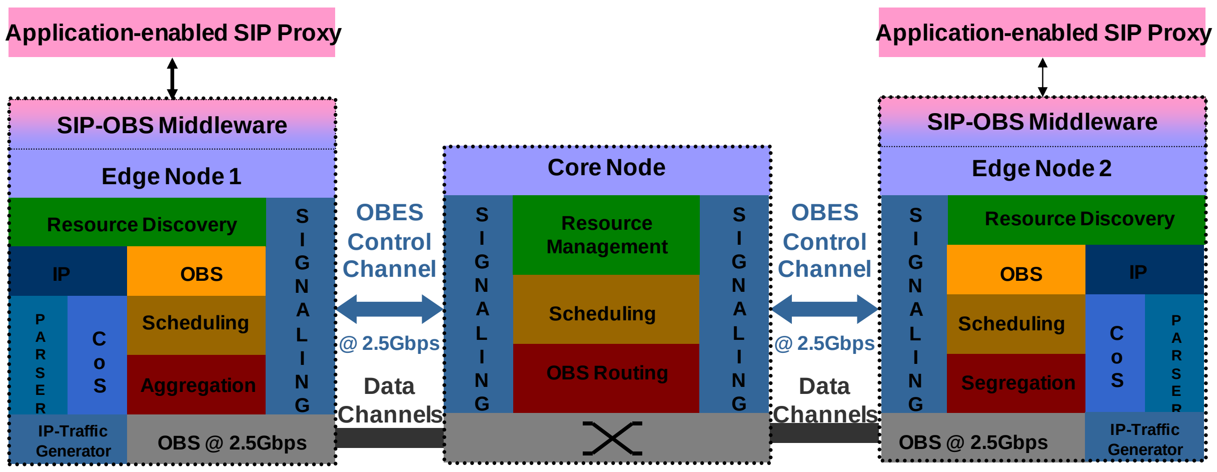


Figure 7.1: Functional block diagram of the SIP enabled control plane in the proposed OBS network architecture. JIT-SIP signaling goes through OBES channel

QoS requirements. The AO-M has been rewritten partly, exploiting an Open Source Standard SIP proxy [Ser] the APP-M and NET-M has been rewritten and integrated in the SIP proxy modules. Moreover, in order to be technology independent the SIP service requests have been translated into consistent and network-specific directives thanks to a service architecture for CP-enable transport network, called Service Oriented Optical Network (SOON), see section 7.2. SOON, developed by the Scuola Superiore degli Studi di Perfezionamento Sant'Anna in Pisa, allows network technologies to be independent from future evolution of the services and the Control Plane to be unburdened of the service-related functionality.

7.1 OBS TestBed

The test-bed architecture is based on OBS network technology utilizing SIP-enabled OBS routers and JIT-SIP signaling protocol as shown in figure 7.1 and 7.2. The test-bed comprises two SIP-enabled edge routers, one SIP-enabled core router and JIT-SIP control protocol as described below. The SIP-enabled edge OBS router is utilizing PowerPC processor and FPGA (Field Programmable Gate Arrays) and is able to process, classify and differentiate application layer sessions according to their network (bandwidth) and non-network (computing) resource requirements [GYR$^+$08][ZNSO06a]. It can aggregate jobs according to their resource requirements into bursts and allocate a suitable wavelength and Burst Control Header (BCH) to them. The SIP-enabled core OBS router comprises a fast switch. It utilizes network processor and FPGA and it can process application layer information on the fly to allocate switch resources.

The proposed JIT-SIP protocol utilizes SIP functionalities to negotiate and manage

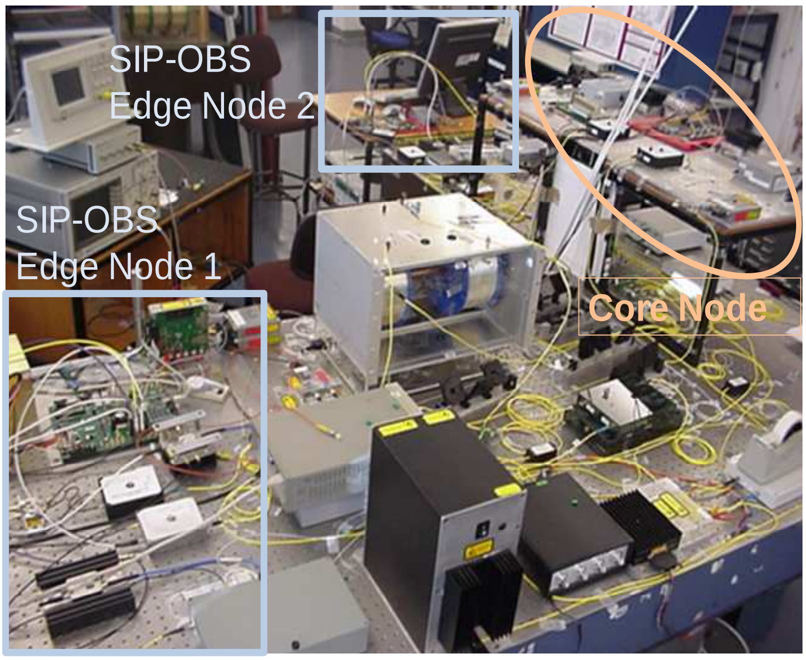


Figure 7.2: Overall picture of the OBS network architecture

the application sessions (i.e. non-network resource discovery and allocation) as described in section 6 and JIT signaling to reserve optical network resource and manage the physical layer connections. In the proposed architecture, SIP messages are carried in the optical domain with JIT signalling. In this network architecture the SIP functionalities are handled by SIP proxies (i.e. SIP-M module) that are coupled with OBS edge nodes. On the other hand SIP-enabled core OBS routers, where the processing speed is critical, are equipped with a light subset of SIP functionalities to provide the best performance for the proposed network in terms of resource discovery and connection establishment. User Agent is based on standard SIP UA properly modified to have the capability to embedded Grid application protocol into the SIP messages body. Moreover the SIP UA used in this experimental activity can publish and discovery a generic Grid resource. Both SIP Proxy and User Agent are based on a SIP stack called pjsip [pjs] and runs on a PC coupled to the Edge Routers.

The OBS test-bed operates at 2.5 Gbps for both the data plane and control plane. All routers utilize a Xilinx high-speed and high-density VirtexII-Pro FPGA with embedded network processor. The OBS control channel is transmitted on dedicated wavelength channel in the same fibre using the proprietary Optical Burst Ethernet Switched (OBES) transport mechanism [ZNSO06b]. The data plane transports variable sized bursts with variable time intervals and operates in bursty mode. Edge router (ingress side) utilizes a SIP engine and one fast and widely tunable SG-DBR laser. Details of both edge and core router functionalities are described in following paragraphs.

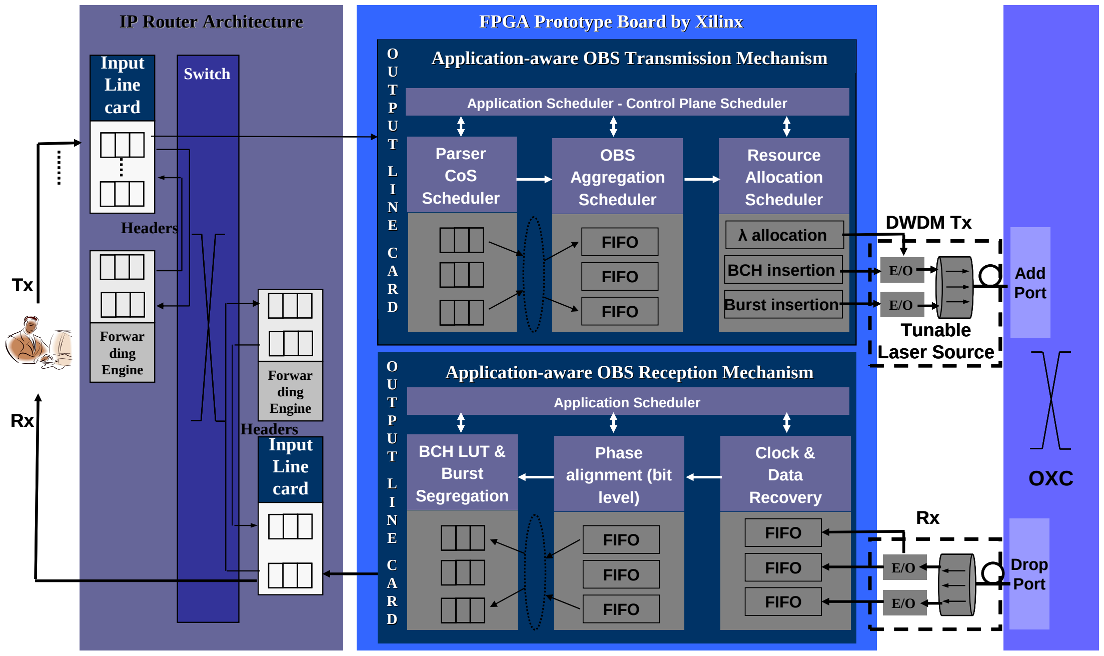

IP Router Architecture
Input Line card
Switch
Headers
Forwarding Engine
Tx
Rx
FPGA Prototype Board by Xilinx
OUTPUT LINE CARD
Application-aware OBS Transmission Mechanism
Application Scheduler - Control Plane Scheduler
Parser CoS Scheduler
OBS Aggregation Scheduler
Resource Allocation Scheduler
FIFO
λ allocation
BCH insertion
Burst insertion
DWDM Tx
E/O
Tunable Laser Source
Add Port
OXC
Application-aware OBS Reception Mechanism
Application Scheduler
BCH LUT & Burst Segregation
Phase alignment (bit level)
Clock & Data Recovery
Rx
Drop Port

Figure 7.3: SIP enabled Edge router architecture

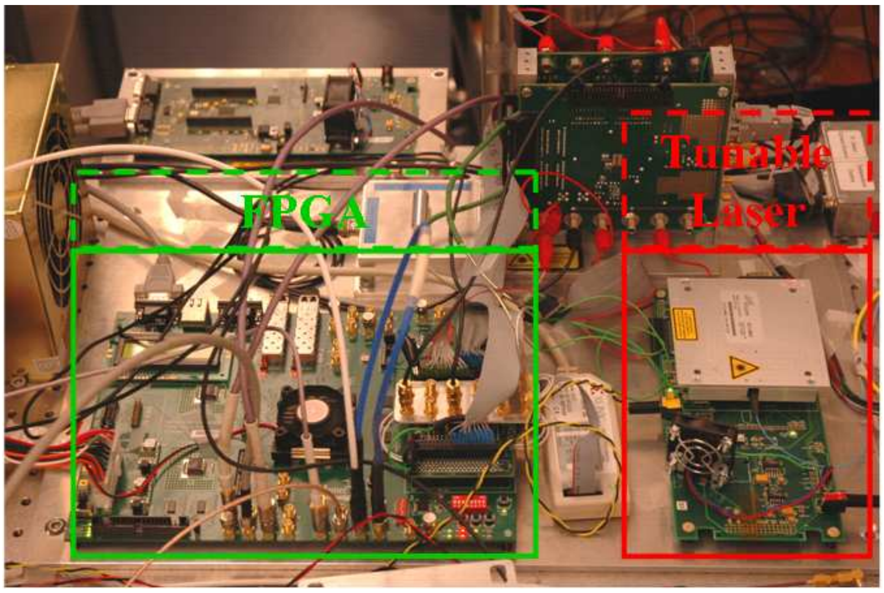


Figure 7.4: Picture of SIP enabled Edge router

7.1.1 OBS Edge router

The SIP-enabled OBS edge router architecture proposed here incorporates an IP router followed by a generic input/output IP-OBS line card that is attached to an all-optical switch, figure 7.3 and 7.4. It represents the electro-optic transit point between the legacy network and the all-optical backbone network.

Functionality and algorithms of the edge router are presented, beginning with a packet arrival at an input line card of the IP router. Its header information is extracted and passed through the switch to a forwarding engine while the remainder of the packet remains on the IP input line card. The forwarding engine using the header information determines how and where to forward the packet and then sends the required forwarding instructions back to the inbound line card. The packet is then sent to the output OBS line card for further processing and transmission. The output OBS line card can be logically divided into three main blocks.

The first section named as Parser and Class of Service (CoS) Scheduler is equipped with a parser, a classification mechanism and buffers for storing packets until the parsing results become available and the packets can be forwarded to the next block. It detects the type of application out of TCP or UDP port understands if any higher layer protocols is encapsulated (e.g. SIP) and parse header details to SIP Scheduler. The SIP Scheduler identifies the type of SIP message that has been received. If that message has to do with resource discovery (e.g. SUBSCRIBE), reservation (e.g. NOTIFY) then it informs the second and third stage so as to encapsulate the SIP message into a BCH message and forward it to the appropriate output port. In case of receiving session activation message (e.g. ACK) the SIP Scheduler process the network requirements (e.g. latency, loss, jitter) of that session and controls aggregation scheduler and resource allocation scheduler accordingly.

The second block is the OBS aggregation scheduler. It incorporates a scheduling control unit responsible for handling the buffers/memories required to shape bursts. The third block - the resource allocation scheduler - incorporates resource look-up-tables (LUT) required to efficiently allocate optical resources to short lived paths (assembled bursts). It also comprises output buffers for contention resolution when all wavelengths are reserved. Finally, it has electro-optic interfaces (high-speed agile tunable laser interfaces not shown on the figure) in order to map the IP traffic to the all-optical backbone network.

The input OBS line card is also divided into three sections. The first and prerequisite hardware block (from the right) incorporates a Clock and Data Recovery (CDR). Then a synchronization section is important to align phase and frequency out of received burst control headers (BCH) as well as bursts to be able to process them further. Finally a BCH processor and a burst segregation mechanism is deployed to forward IP packets back to the legacy network. The outbound line card of the OBS edge router is based on a high-speed reconfigurable hardware platform (Xilinx FPGA- Virtex II-Pro) and a fast widely tunable sampled grating distributed Bragg reflector (SG-DBR) laser able to emit

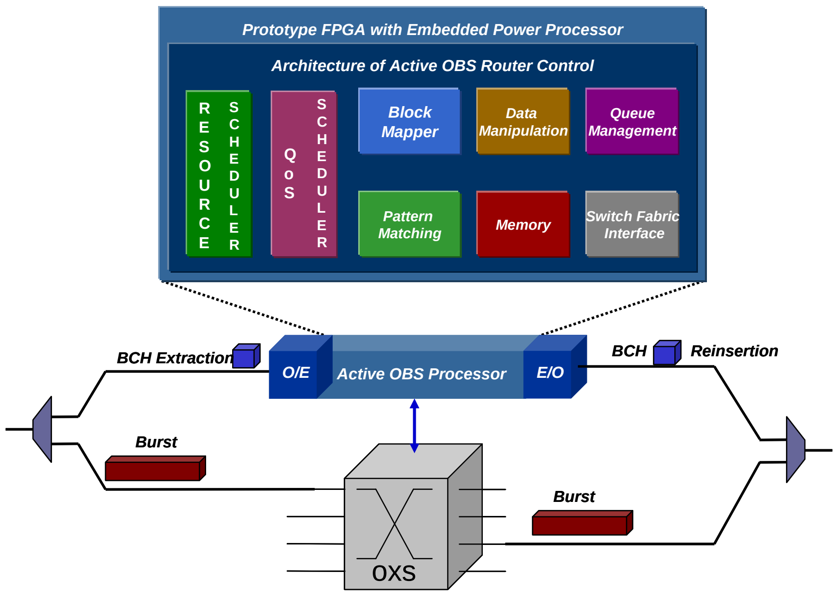


Figure 7.5: SIP enabled Core router architecture

and tune between 85 ITU channels with 50 GHz spacing in less than 100 ns (for 75% of them).

7.1.2 OBS Core router

The core router integrates an FPGA with integrated PowerPc processor and an optical crosspoint switch (OXS) matrix. The SIP-enabled OBS core router architecture illustrated in figure 7.5 7.6 is introduced to support the Just-In-Time (JIT) OBS reservation protocol for network resources as well as algorithms, mechanisms to support discover and reservation of non-network resources based on SIP protocols explained in section III. The header (BCH) that is transmitted out-of-band, is first converted to electronic domain and then its type is identified. BCH can be used to encapsulate SIP messages for resource discovery and reservation mechanisms or in the traditional way to reserve a path for a particular burst. In the first case the the BCH is forwarded to the next SIP-OBS router/s. In the second case the BCH blocks (offset, burst length, destination, CoS) are thoroughly processed in order to establish cross-connections in order to allow transparent bust transmission through the 4x4 OXS. The functional blocks of the overlay electronic control of the core node are listed below:

- At the input stage (Pattern Matching) the FPGA extracts BCH for identification (BCH-SIP or traditional BCH).
- The Block Mapper part provides efficient means to pass BCH between different components. The mechanism is used to speed-up the BCH and in turn burst handling

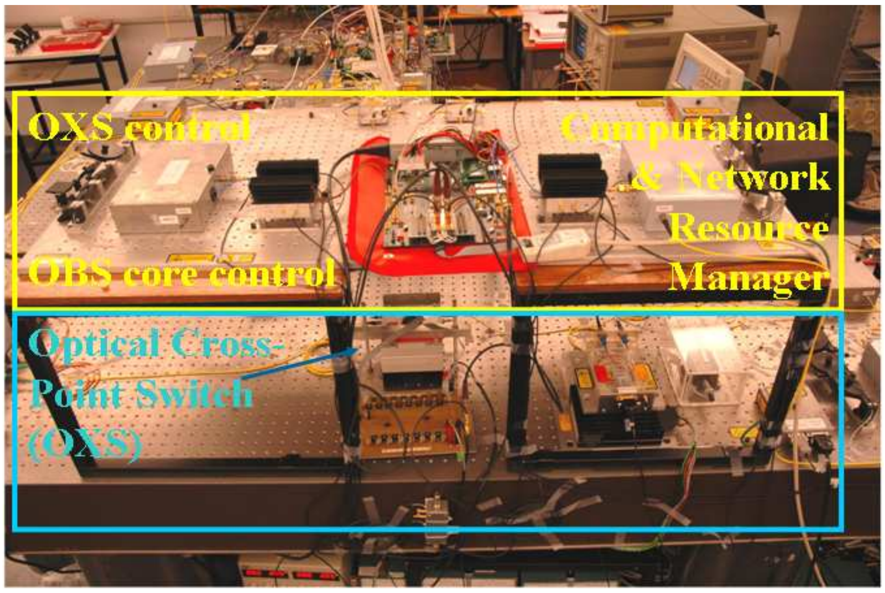

OXS control
OBS core control
Computational & Network Resource Manager
Optical Cross-Point Switch (OXS)

Figure 7.6: Picture of SIP enabled Core router

on the node.

- The Data Manipulation part modifies the BCH, (e.g. it decrements the Time to Live field in the IP header block, recalculates the CRC check and performs burst encryption).
- The Memory stores the resource availability, lookup tables, packet data and Queue information.
- The Queue Management schedules and stores the BCHs to provide QoS priority queuing.
- The Switch Fabric Interface provides an interface to the switch fabric and set-up the appropriate in-out ports for routing purposes.
- The SIP scheduling is responsible for ensuring that the network service provisioning conforms to the session requirements.
- Resource scheduling handles the resource capability and availability of the core router.

This router architecture offers the opportunity to take into consideration both network and IT resources into consideration in a single step and in tern provide application layer functions in OBS control plane. The 44 OXS matrix consists of 16 interconnected optical switch cells. Each switch cell connects two passive waveguides that intersect each other perpendicularly as input and output signal buses, and two active vertical couplers (AVC) are formed at each cell by stacking an active waveguide layer on top of the passive waveguides, enabling the function of switching by carrier induced refractive index and gain-absorption changes in the AVCs. Extremely low crosstalk levels of ¡-60 dB and fast chip switching time of 1.5 ns have be achieved.

7.1.3 First experiment

The first experiment aimed at demonstrating the feasibility of the concept. The architecture implemented is fully overlay. The SIP signaling is carried over a conventional IP network (Fast Ethernet) to the SIP-OBS test-bed while the data bursts are carried over optical bursts. The emulated data traffic is generated within the OBS and are not related to any specific application. The network concept demonstrated in the test-bed provides application layer resource discovery and routing of application data or user's jobs to the appropriate resources across the optical network. The SIP based control plane architecture comprises a resource discovery stage and a traditional OBS signalling stage using Just-In-Time (JIT) bandwidth reservation scheme.

The user with data to be processed remotely (e.g. Grid user, e-Science) sends a request to the AO-M of the ingress edge router to the OBS cloud. The request carries the job specification and resource requirements (i.e. computational and network) in the payload, in the form of a Job Submission Description Language (JSDL) document. The SIP server classifies the incoming job requests (JR) and then either processes it or forwards it to the next AO-M which has knowledge of the available resources. Client applications are connected to the edge router 1 while resources are connected to the edge router 2. Therefore all messages with requests are sent by the clients to AO-M 1 coupled with edge router 1. Since AO-M 1 does not have any available resource, the requests are forwarded to the AO-M 2, coupled with edge router 2, from where the resources are reachable. The message-timing flow is presented in figure 7.7.

After the INVITE message is processed at AO-M 2 the user is informed about the results of the resource discovery, with either a positive or negative reply, depending on whether resources are available or not. In the experiment a positive reply is the 200 OK message, that is received by AO-M 1 $TS_1 = 14.45ms$ (experimentally measured) after sending the INVITE message, as indicated in figure 7.7. In this case AO-M 1 forwards the OK message to the client and triggers the application to sent the job. At the same time a computational resource reservation signaling (ACK) message is sent to AO-M 2 (after $TS_2 = 17.95ms$). This is for acknowledging the session establishment. The time elapsed between the arrival of the OK message and the departure of the ACK is due to the signalling between AO-M 1 and the client and for triggering the application to transfer the data referring to the communication session under negotiation.

Regarding the timings of the OBS network, in figure 7.7 the time $Edge1.1$ is for user acknowledgment, job submission, aggregation and transmission over OBS, it is variable

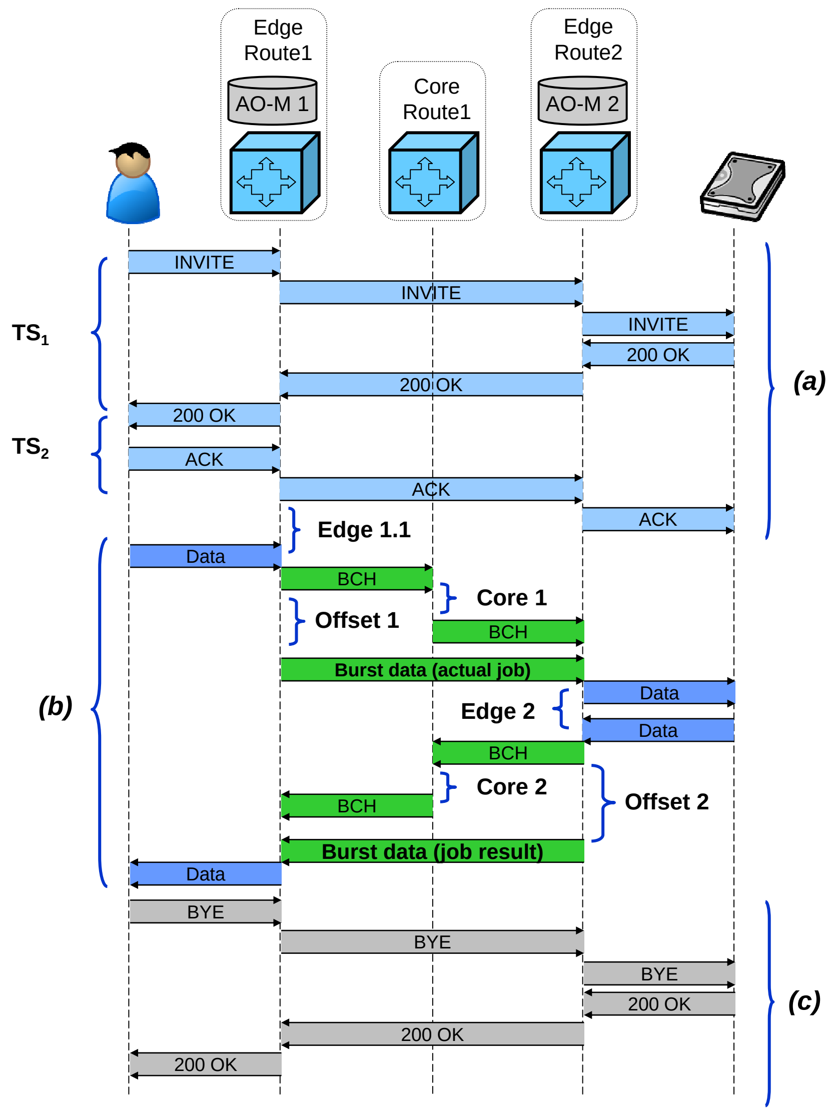


Figure 7.7: Examples of application oriented functions supported by the SIP-OBS network: INVITE reservation (a), communication over Just-In-Time (JIT) OBS (b) and session tear down (c).

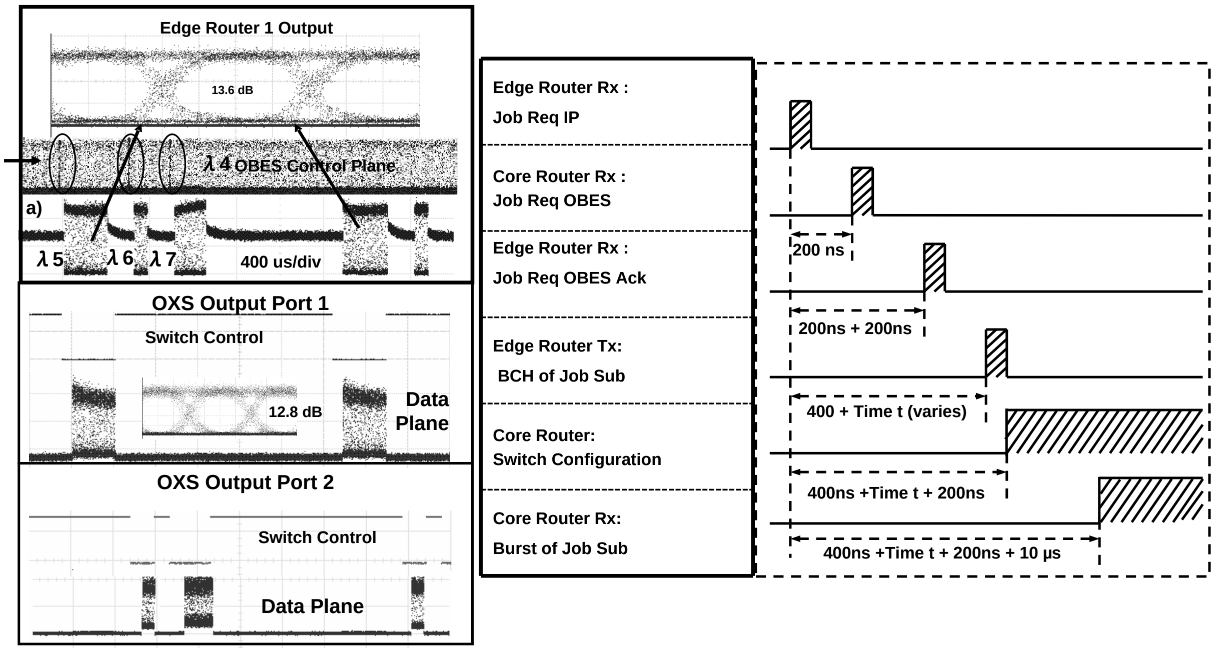


Figure 7.8: Test-bed demonstration results: Three variable sized burst transmitted at Edge router 1 a), first burst is sent through output port 1 and received at Edge router 2 by controlling Port 1 of OXS and two bursts are sent over output port 2 for a different edge router by controlling Port 2 of the OXS

and depends on incoming traffic and burst size. $Core1$ and $2 = 200ns$ is the time for BHC processing and reinsertion in the core nodes. $Edge2.2$ is the time for processing incoming actual job and the transmission of the result over OBS, it can be variable and depends on processing time. The time $Offset1$ and 2 is the offset required to implement JIT transport protocol, BCP processing time at core switch and setup time of optical crosspoint switch (OXC).

In the data transport part of OBS test-bed, variable length optical bursts (from 60 μs up to 400 μs) with their associated BCHs (figure 7.8.a), in three different wavelengths has been demonstrated ($\lambda 5 = 1538.94nm$ and $\lambda 6 = 1542.17nm$, $\lambda 7 = 1552.54nm$ for data bursts). The generated optical bursts with variable offset times are shown in figure 7.8.a together with the OBES control plane. Control signals to route burst 1 through port 1 is shown in figure 7.8.b together with eye diagram of the output port 1. The results show that $13.6dB$ extinction ratio (figure 7.8.a) can be achieved by the edge router and $12.8dB$ extinction ratio after output port 1 of the OXS and at the received point. Figure 7.8.c shows the two further bursts which are switched to output port 2 of the OXS.

7.1.4 Second experiment

The second experiment implements a full functional test application based on Video on Demand (VoD). The video server publishes a list of the available movies (resources) at the AO-M integrated at the SIP-OBS-2 node with a PUBLISH SIP message, figure 7.9.(a) The client application is connected at SIP-OBS-1 and requests the list of available movies (using the SUBSCRIBE message), figure 7.9.(b). SIP messages exchange between edge nodes are encapsulated into JIT signaling in order to be transmitted as embedded OBS signaling.

An direct INVITE SIP message is used to request and activate the VoD application service (video server) to stream video over OBS, figure 7.9.(c). The JIT-SIP control protocol performance has been evaluated by measuring the resource reservation time (from the time the INVITE is sent from user side to the time ACK is received to resource site) which is 60 $msec$. This value is mostly dependent of the end host performance used for AO-M and not the actual OBS test-bed. Independent evaluation of the AO-M with a single dual 2.4 Ghz Xeon PC with 2 Gbyte of RAM virtualizing neighbor AO-Ms has proven that a speed on 0.2 ms is required to process the INVITE message.

The JIT-SIP messages encapsulated in BCHs are sent over OBS control plane, and the generated optical bursts over data plane, figure 7.9.(d). Figure 7.10.a shows the bursts over data plane (a), (b), (c). The VoD application generates traffic (video streaming) of around 1Mbps with fixed size 1370-byte UDP packets. FTP background traffic of around 20Mbps is also generated and added to the video traffic in order to emulate the current internet traffic behavior (between TCP and UDP data). The aggregation developed is hybrid and combines both size and time threshold. The maximum size threshold is set to 5000 bytes and time limit to 2ms. These are set based on the total incoming traffic load

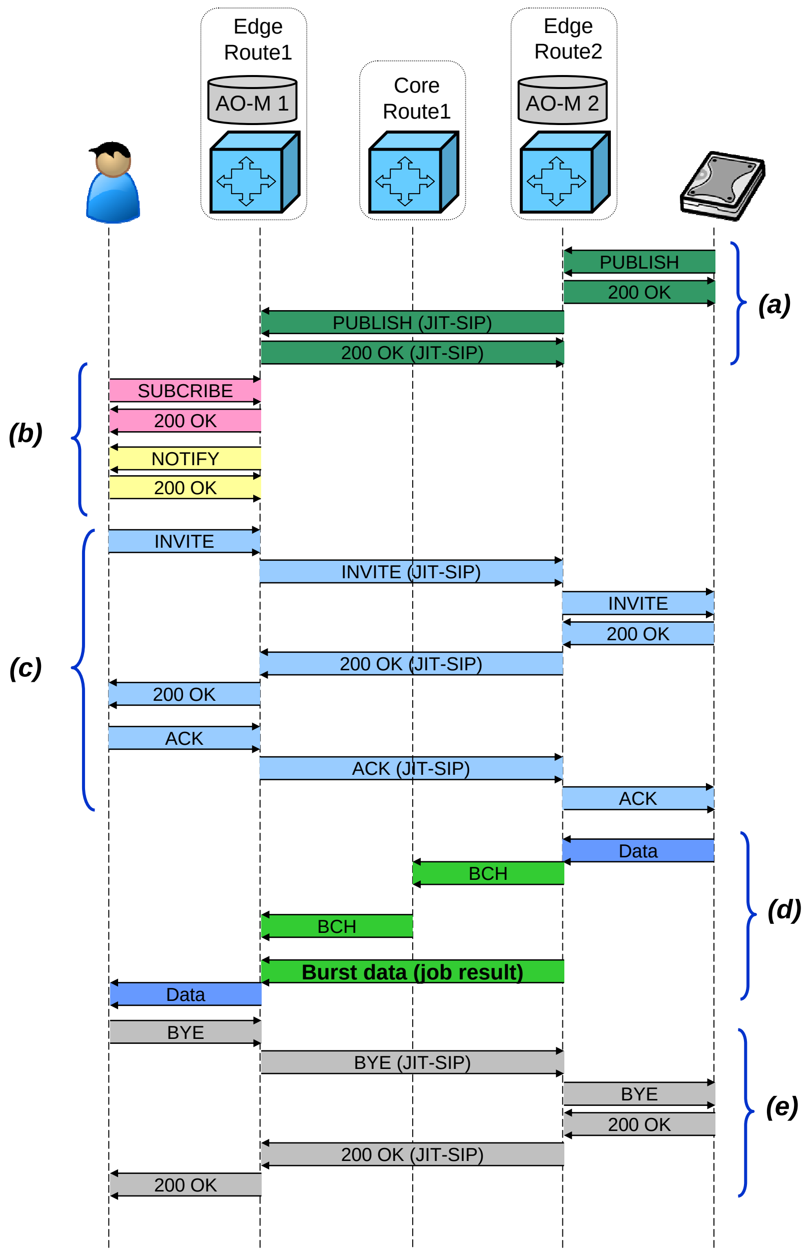
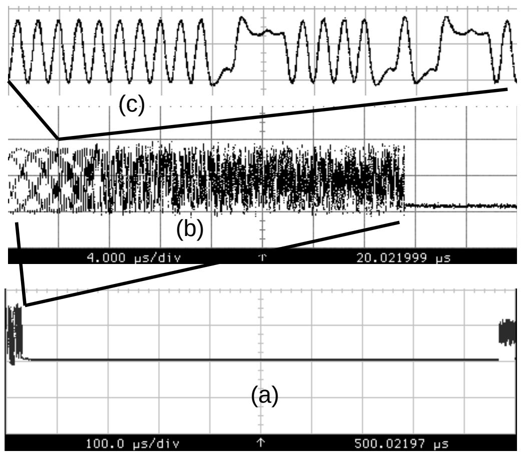
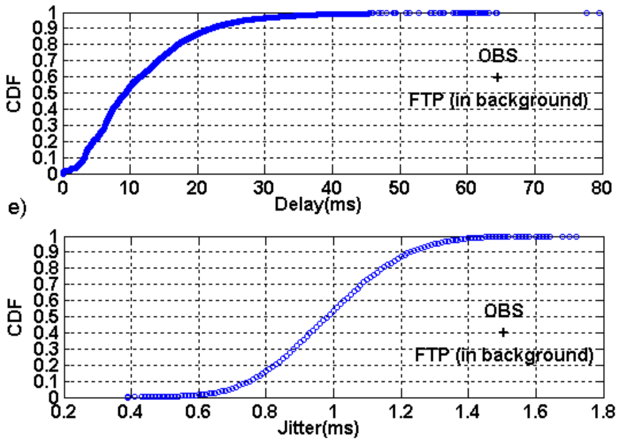


Figure 7.9: Examples of VoD application functions supported by the SIP-OBS with JIT-SIP signaling network: resource pubblication (a), resource discovery (b), direct reservation (c), data streaming over JIT-SIP (d) session tear down (e)



(a)



(b)

Figure 7.10: Test-bed demonstration results: a) bursts over data plane b) delay and jitter measurement of VoD application over OBS

(21Mbps) and the low latency requirement. The video server generates MPEG-2 fixed size UDP packets (1370 bytes).

Figure 7.10.b shows that more than 95% of the UDP packets have a delay of less than 30 ms with a maximum delay of just over 80ms which is well within the acceptable level. This figure also shows that the jitter also remains below 1.8 ms for 100% of the traffic, also a well accepted value. The packet loss of the OBS network is zero for the whole amount of data.

Table 7.1: Timing results (ms) of SIP messages on a set of SIP-Ms

Messages to Proxy/ Number of SIP-M	1	2	5	10	20	40
INVITE → 404 Not Found	0,206	0,311	0,316	0,759	1,371	2,472
PUBLISH → 200 OK	0,169	0,243	0,317	0,737	1,405	2,037
SUBSCRIBE → NOTIFY	0,291	0,368	0,602	0,824	1,326	2,432

7.1.5 Scalability issues

One last issue that could not be addressed with the available test-bed for dimension reasons is scalability. The experiments presented above showed the feasibility of the concepts but in a rather simple and small network topology due to the constraints on availability of optical hardware. We have then made some evaluations to test the scalability of the proposed SIP-OBS approach.

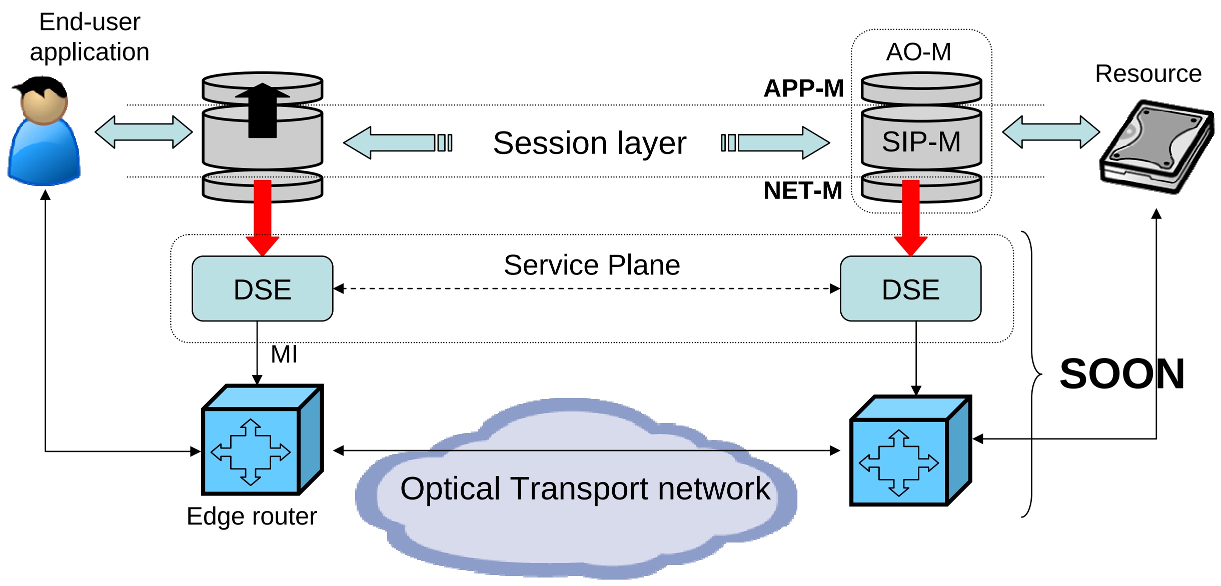


Figure 7.11: Application oriented network architecture, showing the various modules and their interaction

Regarding publishing (PUBLISH), discovering (SUBSCRIBE), and reserving (INVITE) resources, the processing time of these single messages and the total amount of time required to forward messages to the neighbour AO-Ms is one of the main issues. The results shown in table 7.1 projects the time between the request of a User Agent and the AO-M's processing time to forward the message to all neighbour AO-Ms, thus the total amount of time to broadcast a message into the network. The tests were performed with a single dual 2.4 Ghz Xeon PC with 2 Gbyte of RAM virtualizing neighbour proxies. In addition to this, table 7.1 shows the processing time between the request of a User Agent (UA) and the AO-M response for all possible SIP messages on stand alone AO-M.

These numerical results prove that the proposed solution can scale quiet well. The results reported are compatible with real life applications for several tenths of nodes, in spite the simulations have been performed with standard off-the-shelf computers.

7.2 GMPLS TestBed

The second experimental activities has been achieved in collaboration with Scuola Superiore degli studi di Perfezionamento Sant'Anna in Pisa.

The logical architecture is presented in figure 7.11 where three logical layers are in evidence. A session layer managing the application signalling, the service plane managing the mapping between service and network requests and the GMPLS-enabled optical network. The network coupled with the service layer constitutes the Service Oriented Optical

Network (SOON) architecture. The test-bed is composed by three main components as shown in figure 7.11.

- User Agent: based on the same implementation made in the OBS experiment, see section 7.1.
- AO-M: the AO-M was completely rewrote based on a standard Open SIP server [Ser]. This choice has been made in order to drive the AO-M implementation towards a more versatile deployment. The SIP-M part is managed directly by the SIP proxy while the NET-M part has been written as a proxy's module extension. The NET-M, which is the network technology dependent modules of the AO-M, has been tailored to the SOON architecture. Consequentially, the AO-M achieves a further abstraction level, thanks to the SOON the NET-M is not related any more to the Control plane technology becoming network independent.
- SOON architecture: this innovative network equipment create an abstraction layer of the network Control Plane.
- GMPLS network; based on GMPLS Junipper router [Net].

This experiments has been focused on the QoS reservation mechanism for a VoD application case. The resource location has not been tested since the aim of the test was to proof QoS reservation the concept in a NGN fashion.

Overview of SOON

SOON has been develop completely in the Scuola Superiore Sant'Anna Pisa and it is used a further network service plane.

SOON is a service architecture for (G)MPLS optical transport networks conceived to fulfill connectivity service requests issued by applications through a distributed signalling among "service nodes" called Distributed Service Elements (DSEs) thus masking the technology-dependent details from the abstract request of the service, i.e. in terms of end-host addresses and perceived quality of service.

The SOON [MBC05] mediates the interaction between qualified applications, e.g., Proxy SIP Server, and (G)MPLS networks for the invocation of QoS-enabled and high-bandwidth connectivity services. The SOON architecture implements a lower network Service Plane (SP), acting on top of the (G)MPLS CP (see figure 7.11). Specifically the SP exploits the source-initiated GMPLS CP signalling for providing on-demand transport network services with a level of abstraction suitable for being invoked by applications, i.e. in terms of end-host address and perceived quality of service. This is achieved through a distributed signaling among "service elements" that consists in one Centralized Service Element (CSE) and a number of Distributed Service Elements (DSEs), one for each edge network node. The CSE, not shown in the figure, performs application identification and

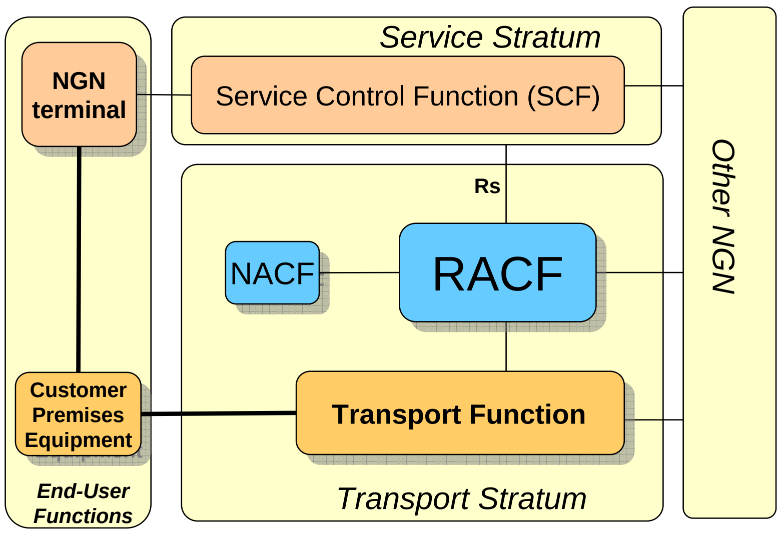


Figure 7.12: Schematic of the NGN architecture and main sub-blocks

authorizes the relevant service requests using the information stored in its Service Level Agreement (SLA) database.

The DSE processes network service requests issued by applications, AO-M in this work. To fulfil the service request, the DSE triggers a signalling message exchange with the CSE to authenticate and authorize the application, and with the DSEs involved in the service provisioning in order to agree on the network -wide settings to be performed on the edge network nodes. Then the involved DSEs issue the technology-dependent primitives to the controlled edge network nodes. These primitives may include configuration commands, e.g., circuit set-up, as well as the resource status monitoring, e.g., link bandwidth availability check. The involved DSEs are obtained from the information stored in the Network Topology Resource Database (NTRD). The implementation of DSE and CSE and how the NTRD is filled has been described in [BMMC07].

7.2.1 Mapping to NGN

This section provides the mapping between implemented building blocks with NGN functionalities [KNT05] to show the correspondence of this research work with standard NGN principles [ooN04]. Referring to figure 7.12, the NGN architecture is composed of two sets of functionalities, named Service Stratum and Transport Stratum.

To the purpose of this work, the main Service Stratum functional entity is the Service Control Function (SCF). The SCFs *"include resource control, registration, and authentication and authorization functions at the service level"* and trigger the network resources reservation by interacting with the Transport Stratum [raotNr06]. The SCF is responsible for extracting service requirements from service signaling and sending a request to RACF for network resource authorization and reservation. In practical terms, the SCF represents the signalling within a service provider infrastructure among, e.g., SIP Proxy Server and Video Server, for the control of application resources involved in the content transmission, i.e. codec, delay bound, max bandwidth, during a SIP-based call or multimedia transfer. From a functional point of view the SCF corresponds to the AO-M, specifically to the SIP-M and NET-M.

The Transport Stratum provides IP-based connectivity services for the benefit of SCFs with the support of the Network Attachment Control Function (NACF) and the Resource and Admission Control Function (RACF). NACF support RACF with information related to subscription profile. RACF *"provides an abstract view of transport network infrastructure to the SCF [...] such as network transport technology, network topology, connectivity, resource utilization and QoS mechanisms. The RACF executes policy-based transport resource control upon the request of SCF, determines transport resource availability, makes admission decision, and applies controls to transport functions."* [Racfingn06]. From this we can argue that from a functional point of view the RACF and NACF correspond to the SP for (G)MPLS-enable transport networks [BMMC08].

7.2.2 TestBed description

This section presents the test-bed used to validate the proposed architecture and examines an example of signalling flow. The test-bed architecture is illustrated in figure 7.13, while the physical setup is shown in figure 7.14. We start with a few implementation details:

- the SIP-M has been built using an open-source SIP proxy [Ser];
- the APP-M and the NET-M have been implemented by means of ad-hoc opensips external modules which define an NRDL parser and the interface to SOON.

The SIP User Agents (UA) used as end-user terminals are based on the PJSIP stack [pjs] enhanced with the capability to send a multipart INVITE massage and an NRDL module. The NRDL modules have been implemented exploiting the Raptor RDF parser [par]. In this proof of concept the NRDL has been provided with the capability to request a predefined bandwidth and Class of service only. The service plane is composed by 3 Linux Boxes, each of them running an instance of the DSE module and equipped with Linux v2.6 operating system and have public IP addresses to assure the complete reachability. The DSEx controls the ERx (with x = 1, 2, 3) issuing NETCONF directives [Enn06]. Each Linux Box has been provided with Ethernet interface connected to the controlled ER and set with a public IP.

The metro-core network comprises six IP/MPLS routers interconnected as shown in the figure 7.13. The routers called ER1, ER2 and ER3 are configured as provider Edge Router (ER). The routers named CR are configured as core routers. All the six routers are equipped with Fast Ethernet (FE) interfaces and support MPLS, OSPF, and RSVP

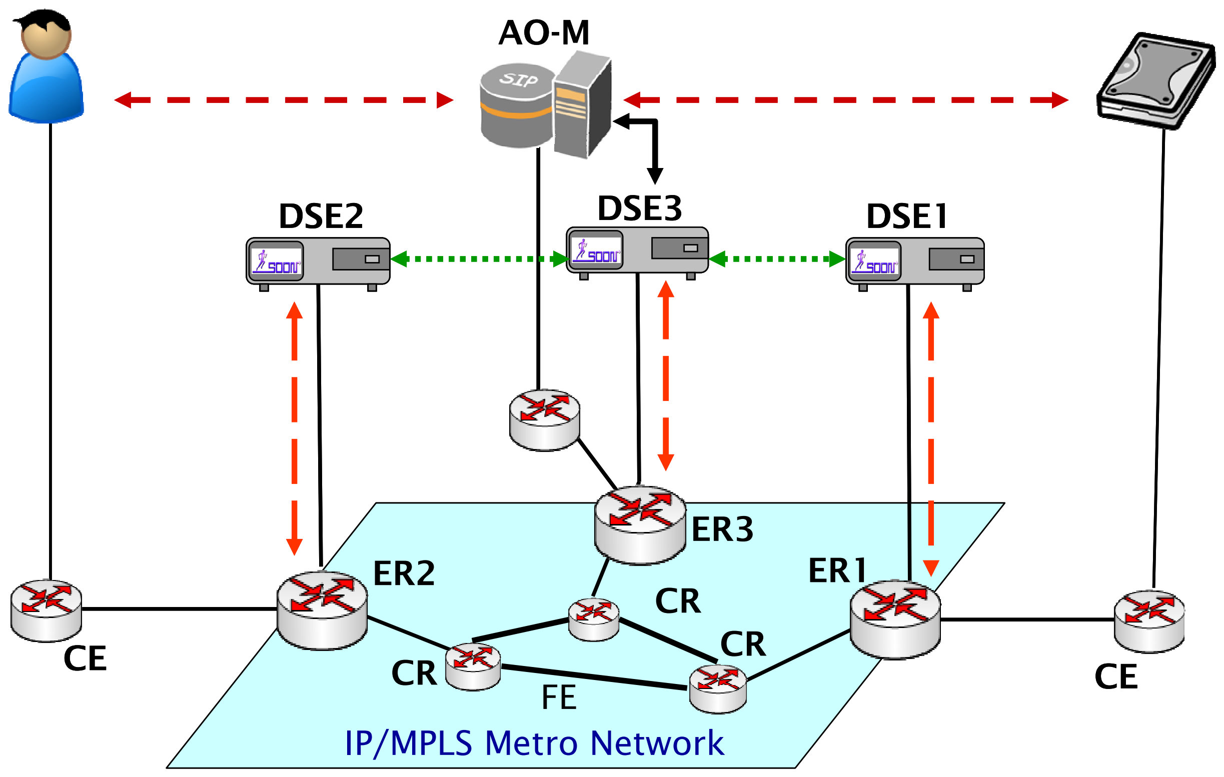


Figure 7.13: SIP-SOON test-bed architecture

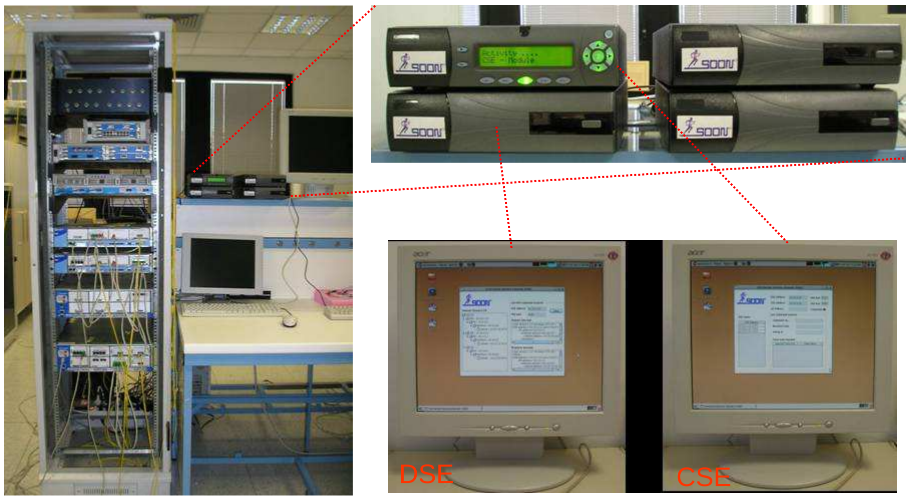


Figure 7.14: SOON test-bed picture

protocols with extension for DS-TE [FL03]. Without lacking in generality, we consider only one AO-M couples to DSE3.

The SIP UAs are connected to the transport network via a router configured as Customer Edge (CE) node (CE1 for UA1 and CE2 for UA2), representing the gateways out of the customer premises. They are connected to the transport network through VLANs, namely "VLAN1" for CE1 and "VLAN2" for CE2, that segregate the traffic flow of different application categories, e.g., video, voice, data. Without lack of generality, we consider the case of video traffic.

In order to enable the packet transfer between UAs with the required QoS, a mechanism is needed to reserve network resources through the MPLS network. Such mechanism exploits DS-TE capabilities that allows the advertisement and reservation of resources based on traffic class while optimizing the use of network resources. In fact, by reserving the required portion of link bandwidth during the LSPs set-up and by accordingly setting queue scheduling policies, TE capabilities can adjust network resources among LSPs in a way that packet flows switched along such LSPs can experience the required treatment. As part of test-bed set-up, an LSP is established between ER1 and ER2 (LSP-video) with reserved bandwidth of 80Mbit/s and with queue schedulers accordingly configured in both core and edge routers to guarantee rate assurance to video packet delivery.

7.2.3 Use case

The message flow of the experiment is shown in figure 7.15. The whole procedure is started with an INVITE message (1) from UA2 to the AO-M carrying a NRDL document that requests the HD video service. The SIP-M receives the INVITE and passes the NRDL to the APP-M. The APP-M parses it and triggers the NET-M to ask the DSE3 to arrange the network resources reservation across the MPLS network domain, by means of a *"Service Request"* (2). The service request consists of an XML file including the UA IP addresses and the service characteristics, i.e., service type and quality level.

DSE3 obtains the addresses of the relevant ERs (i.e., ER1, ER2), and of the DSEs connected to those ERs (i.e., DSE1, DSE2) thanks to the information stored in NTRD. Then, DSE3 translates the service parameters into network resource requirements, i.e. needed bandwidth (e.g., 20Mbit/s for HD video transfer) and category of traffic (e.g.., delay-sensitive traffic). Subsequently it sends a *"Resource Availability Request"* (3) to DSE1, linked to ER1, containing such requirements.

Upon receiving the *"Resource Availability Request"*, DSE1 sends a *"Bandwidth Availability Request"* (4) to the ER1 to get the bandwidth that is currently used by existing voice packet transfers between ER1 and ER2. Specifically, the transmit rate is requested on a queue dedicated to transfer delay-sensitive packets at the output interface towards ER2. When receiving the *"Bandwidth Availability Response"* (5), DSE1 compares the available bandwidth with the requested bandwidth and sends the *"Resource Availability Response"* (6) to DSE3. If the bandwidth is available, DSE3 sends the *"Connectivity*

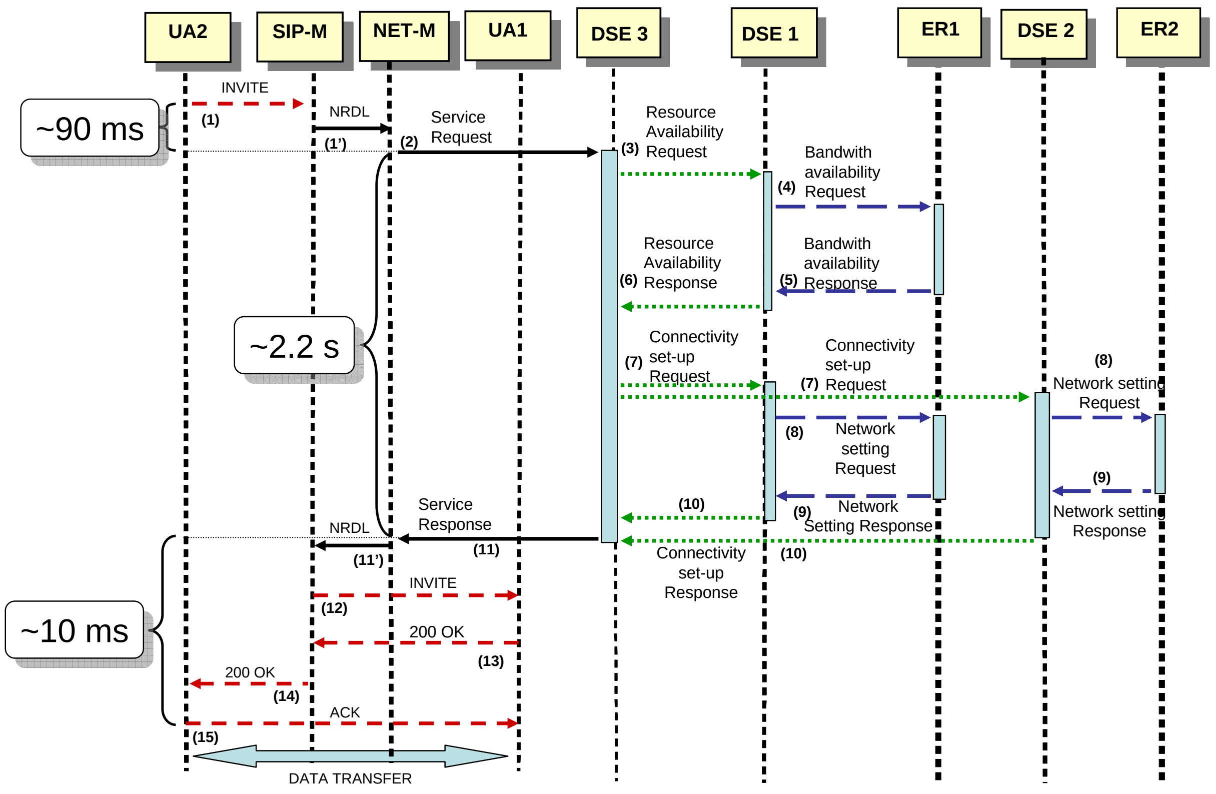

UA2
SIP-M
NET-M
UA1
DSE 3
DSE 1
ER1
DSE 2
ER2
~90 ms
~2.2 s
~10 ms
INVITE
(1)
NRDL
(1')
(2) Service Request
(3) Resource Availability Request
(4) Bandwith availability Request
(5) Bandwith availability Response
(6) Resource Availability Response
(7) Connectivity set-up Request
(7) Connectivity set-up Request
(8) Network setting Request
(8) Network setting Request
(9) Network Setting Response
(9) Network setting Response
(10)
(10) Connectivity set-up Response
(11) Service Response
NRDL
(11')
INVITE
(12)
200 OK
(13)
200 OK
(14)
ACK
(15)
DATA TRANSFER

Figure 7.15: Call flow

Set-up Request" to DSE1 and DSE2 (7) to enforce the proper traffic policy that enables the media flow generated by UA terminals to be mapped on the path connecting ER1 and ER2.

Accordingly, DSE1 and DSE2 issue the NETCONF directives, i.e. *"Network setting Request"* (8), to the corresponding ERs that in turn apply the requested configuration and send the *"Network setting Response"* (8) back containing information about the success or failure of such configuration. Afterwards, DSE1 and DSE2 send the result of router configuration to DSE3 that in turn sends the *"Service Response"* (11) to NET-M with the value set to ACK if the configuration has been properly performed, otherwise it sends a NACK. When the resource reservation has been successfully realized, the NET-M sends a NRDL document with the current network status (11') back to SIP-M. The NRDL is reattached to the body of the INVITE message in order to keep the network status along the SIP path. Then, the INVITE message is forwarded to the destination (12). UAb replies with a 200 OK message to the SIP-M, encapsulating its own application protocol and an NRDL document, containing the network requirements from the UAb point of view (13), in the message body. The 200 OK message goes through the SIP-M which extracts the NRDL in order to perform the network reservation by means of NET-M. When the transport facilities have been reserved, the 200 OK is forwarded to UAa (14) and an ACK follows to successfully close the session (15). The data exchange can finally start into the reserved channel across the transport network.

The timings obtained from the test-bed experiment are shown in figure 7.15, demonstrating the feasibility of the proposed platform. The SOON response time is short enough to avoid timeouts at both the session and application levels and the additional SIP signaling introduces negligible latency.

8 Conclusion

This work presents the relevant studies of the issues of managing application and network resources in a Next Generation Network showing how it is possible to outline a Service Oriented Architecture using as relevant building blocks current technology equipments.

The starting concept is that a more effective approach to the solution of these problems could be achieved by delegating directly to the network some crucial functions such as resource publication, discovery and reservation. This is achievable by means of a general signalling infrastructure embedded in the network with the capability to interact directly with the end user applications, as well as with the network functional sub-blocks.

We have discussed the possibility to implement such a concept by means of a session layer that is part of the network control plane and may be implemented using the SIP signalling protocol coupled with a description language like NRDL. Moreover, an extension of the resource QoS framework in SIP has been presented as curial part of reserving network resources with a proper grade of flexibility.

Research works and experiment already published (see Appendix B) and ongoing show that SIP provides a high degree of flexibility and can be configured to support application and network oriented functions such as application resource advertisement and discovery, application and network resource reservation, QoS management etc. over a variety of networking technology. Moreover, these functional schemes can be adapted to support different application languages (JSDL, SDP etc.) as well as different level of network intelligence by suitably integrating SIP with the existing network control planes.

We strongly believe this proposal may be a breakthrough in the implementation of this sort of services since it solves a large degree of problems with a very limited implementation effort, since many of the basic building blocks are based on existing and well established technologies. However, a lot of work is still to do in order to achieve a wide experimentation of the aforementioned concepts.

Moreover, the second main result has been the formalization of a very computational effective implementation of scheduling algorithms applicable to OBS/OPS networks, as outcome of the scheduling studies. In fact, the complexity of such algorithms in general depends on the traffic conditions. This work presents a formulation of the problem that turns into simple min/max search and whose complexity is almost independent of this quantity. The implementation of this algorithm combines the efficient search typical of problems based on a static data structure with the large and dynamic storing capacity of a dynamic data structure. The thesis also discusses a possible implementation of some algorithm's blocks, mainly based on the use of combinatorial operations. Some preliminary simulations of the hardware implementation suggest a very limited computational complexity and a fast response time of the algorithm. However, a real deployment of the proposed algorithms is under way.

Other achievements like a novel SIP collaboration based on an Hybrid architecture (H-SIP) are not discuss here. Even if the work has been deeply focused on the study of SIP protocol and architecture, these studies are not fully reported in this document since it hasn't been reached the necessary maturity to be published in this context.

A
NRDL files and sintax

Listing A.1: Example of Dialog Sessions and Connections

```
<Dialog rdf:ID="Dialog1234">
  <hasSession>
    <Session rdf:ID="Session7853">
      <hasService>
        <Services rdf:ID="Audio9378">
          <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#
              string"
          >Audio9378</rdfs:label>
        </Services>
      </hasService>
      <hasConnection>
        <PhysicalConnection rdf:ID="PhysicalConnection9115">
          <hasService>
            <Services rdf:ID="QoS1634">
              <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#
                  string"
              >QoS1634</rdfs:label>
            </Services>
          </hasService>
          <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#
              string"
          >PhysicalConnection9115</rdfs:label>
        </PhysicalConnection>
      </hasConnection>
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Session7853</rdfs:label>
    </Session>
  </hasSession>
  <hasSession>
    <Session rdf:ID="Session1357">
      <hasConnection>
        <PhysicalConnection rdf:ID="PhysicalConnection7645">
          <hasService>
            <Services rdf:ID="VPN1234">
              <hasDirection rdf:datatype="http://www.w3.org/2001/
```

```
                XMLSchema#string"
            >send</hasDirection>
            <hasState>
              <State rdf:ID="State5678">
                <hasTransportState rdf:datatype="http://www.w3.org
                    /2001/XMLSchema#string"
                >progress</hasTransportState>
                <hasServiceState rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#string"
                >none</hasServiceState>
                <rdfs:label rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#string"
                >Working Status</rdfs:label>
              </State>
            </hasState>
            <hasStrengthTag rdf:datatype="http://www.w3.org/2001/
                XMLSchema#string"
            >mandatory</hasStrengthTag>
            <hasElements>
              <Switch rdf:ID="S2">
                <rdfs:label rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#string"
                >Switch2</rdfs:label>
              </Switch>
            </hasElements>
            <hasElements>
              <Switch rdf:ID="S1">
                <rdfs:label rdf:datatype="http://www.w3.org/2001/
                    XMLSchema#string"
                >Switch1</rdfs:label>
              </Switch>
            </hasElements>
            <ServiceSender rdf:resource="#R2"/>
            <ServiceReceiver rdf:resource="#R1"/>
            <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#
                string"
            >VPN service</rdfs:label>
          </Services>
        </hasService>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#
            string"
        >PhysicalConnection7645</rdfs:label>
      </PhysicalConnection>
    </hasConnection>
    <hasService>
      <Services rdf:ID="Video6245">
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#
            string"
        >Video6245</rdfs:label>
        </Services>
      </hasService>
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Session1357</rdfs:label>
    </Session>
  </hasSession>
  <hasService>
    <Services rdf:ID="VoD5678">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >VoD</rdfs:label>
    </Services>
  </hasService>
  <hasAdministrativeDomain>
    <Administrative rdf:ID="tlc.deis.unibo.it">
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >tlc.deis.unibo.it</rdfs:label>
    </Administrative>
  </hasAdministrativeDomain>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Dialog1234</rdfs:label>
</Dialog>
```

Listing A.2: Example of logical connections

```
<LogicalConnection rdf:ID="Connection0001">
  <hasTransportStratum>
    <TransportStratum rdf:ID="TransportStratum0001">
      <hasNetworkDomain>
        <Network rdf:ID="Network0001">
          <isComposedBy>
            <Router rdf:ID="R1">
              <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#
                  string"
              >Router1</rdfs:label>
            </Router>
          </isComposedBy>
          <isComposedBy>
            <Router rdf:ID="R3">
              <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#
                  string"
              >R3</rdfs:label>
            </Router>
          </isComposedBy>
          <isComposedBy>
            <Router rdf:ID="R2">
              <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#
                  string"
              >Router2</rdfs:label>
            </Router>
          </isComposedBy>
          <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#
              string"
          >Network0001</rdfs:label>
        </Network>
      </hasNetworkDomain>
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >TransportStratum0001</rdfs:label>
    </TransportStratum>
  </hasTransportStratum>
  <hasService>
    <Services rdf:ID="Services0001">
      <hasState>
        <State rdf:ID="State0000">
          <hasTransportState rdf:datatype="http://www.w3.org/2001/
              XMLSchema#string"
          >progress</hasTransportState>
          <hasServiceState rdf:datatype="http://www.w3.org/2001/XMLSchema
              #string"
          >none</hasServiceState>
          <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#
              string"
          >Working Status</rdfs:label>
        </State>
      </hasState>
      <ServiceReceiver rdf:resource="#R3"/>
      <ServiceSender>
        <Element rdf:ID="UAa">
          <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#
              string"
          >UAa</rdfs:label>
        </Element>
      </ServiceSender>
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Services0001</rdfs:label>
    </Services>
  </hasService>
  <hasAdministrativeDomain>
    <Administrative rdf:ID="Domain1">
      <isComposedBy rdf:resource="#R3"/>
      <isComposedBy rdf:resource="#UAa"/>
      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >Domain1</rdfs:label>
    </Administrative>
  </hasAdministrativeDomain>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
  >Connection0001</rdfs:label>
</LogicalConnection>
```

Listing A.3: NRDL schema

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns="http://www.deis.unibo.it/NRDL#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.deis.unibo.it/NRDL">
  <owl:Ontology rdf:about="">
    <owl:versionInfo rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Created with TopBraid Composer</owl:versionInfo>
  </owl:Ontology>
  <rdfs:Class rdf:ID="Element">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Element</rdfs:label>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Connection">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Connection</rdfs:label>
    <rdfs:subClassOf>
      <rdfs:Class rdf:ID="ServiceStratum"/>
    </rdfs:subClassOf>
  </rdfs:Class>
  <rdfs:Class rdf:ID="TransportStratum">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >TransportStratum</rdfs:label>
  </rdfs:Class>
  <rdfs:Class rdf:about="#ServiceStratum">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ServiceStratum</rdfs:label>
  </rdfs:Class>
  <rdfs:Class rdf:ID="TransportElement">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >TransportElement</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Element"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Service">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Service</rdfs:label>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Session">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Session</rdfs:label>
    <rdfs:subClassOf rdf:resource="#ServiceStratum"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Dialog">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Dialog</rdfs:label>
    <rdfs:subClassOf rdf:resource="#ServiceStratum"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="State">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >State</rdfs:label>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Domain">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Domain</rdfs:label>
  </rdfs:Class>
  <rdfs:Class rdf:ID="ServiceElement">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ServiceElement</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Element"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="LogicalConnection">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >LogicalConnection</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Connection"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Network">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Network</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Domain"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="PhysicalConnection">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >PhysicalConnection</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Connection"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Administrative">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >Administrative</rdfs:label>
    <rdfs:subClassOf rdf:resource="#Domain"/>
  </rdfs:Class>
  <rdf:Property rdf:ID="hasStrengthTag">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasStrengthTag</rdfs:label>
    <rdfs:domain rdf:resource="#Service"/>
  </rdf:Property>
  <rdf:Property rdf:ID="hasState">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasState</rdfs:label>
    <rdfs:range rdf:resource="#State"/>
    <rdfs:domain rdf:resource="#Service"/>
  </rdf:Property>
  <rdf:Property rdf:ID="hasServiceState">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#State"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasServiceState</rdfs:label>
  </rdf:Property>
  <rdf:Property rdf:ID="partOfSession">
    <rdfs:range rdf:resource="#Session"/>
    <rdfs:domain rdf:resource="#Connection"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >partOfSession</rdfs:label>
  </rdf:Property>
  <rdf:Property rdf:ID="hasService">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasService</rdfs:label>
    <rdfs:domain rdf:resource="#ServiceStratum"/>
    <rdfs:range rdf:resource="#Service"/>
  </rdf:Property>
  <rdf:Property rdf:ID="ServiceSender">
    <rdfs:range rdf:resource="#Element"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ServiceSender</rdfs:label>
    <rdfs:domain rdf:resource="#Service"/>
  </rdf:Property>
  <rdf:Property rdf:ID="partOfDialog">
    <rdfs:range rdf:resource="#Dialog"/>
    <rdfs:domain rdf:resource="#Session"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >partOfDialog</rdfs:label>
  </rdf:Property>
  <rdf:Property rdf:ID="hasAdministrativeDomain">
    <rdfs:range rdf:resource="#Administrative"/>
    <rdfs:domain rdf:resource="#ServiceStratum"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasAdministrativeDomain</rdfs:label>
  </rdf:Property>
  <rdf:Property rdf:ID="hasTransportElement">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasTransportElement</rdfs:label>
    <rdfs:domain rdf:resource="#TransportStratum"/>
    <rdfs:range rdf:resource="#TransportElement"/>
  </rdf:Property>
  <rdf:Property rdf:ID="hasTransportState">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#State"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasTransportState</rdfs:label>
  </rdf:Property>
  <rdf:Property rdf:ID="ServiceReceiver">
    <rdfs:range rdf:resource="#Element"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >ServiceReceiver</rdfs:label>
    <rdfs:domain rdf:resource="#Service"/>
  </rdf:Property>
  <rdf:Property rdf:ID="hasElements">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >haveElements</rdfs:label>
    <rdfs:range rdf:resource="#Element"/>
    <rdfs:domain rdf:resource="#Service"/>
  </rdf:Property>
  <rdf:Property rdf:ID="isComposedBy">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >isComposedBy</rdfs:label>
    <rdfs:domain rdf:resource="#Domain"/>
    <rdfs:range rdf:resource="#Element"/>
  </rdf:Property>
  <rdf:Property rdf:ID="hasTransportService">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasTransportService</rdfs:label>
    <rdfs:domain rdf:resource="#TransportStratum"/>
    <rdfs:range rdf:resource="#Service"/>
  </rdf:Property>
  <rdf:Property rdf:ID="DomainProvide">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >DomainProvide</rdfs:label>
    <rdfs:domain rdf:resource="#Domain"/>
    <rdfs:range rdf:resource="#Service"/>
  </rdf:Property>
  <rdf:Property rdf:ID="hasDirection">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasDirection</rdfs:label>
    <rdfs:domain rdf:resource="#Service"/>
  </rdf:Property>
  <rdf:Property rdf:ID="hasSubDomain">
    <rdfs:range rdf:resource="#Domain"/>
    <rdfs:domain rdf:resource="#Domain"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasSubDomain</rdfs:label>
  </rdf:Property>
  <rdf:Property rdf:ID="hasSession">
    <rdfs:range rdf:resource="#Session"/>
    <rdfs:domain rdf:resource="#Dialog"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasSession</rdfs:label>
  </rdf:Property>
  <rdf:Property rdf:ID="hasConnection">
    <rdfs:range rdf:resource="#Connection"/>
    <rdfs:domain rdf:resource="#Session"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasConnection</rdfs:label>
  </rdf:Property>
  <rdf:Property rdf:ID="hasNetworkDomain">
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasNetworkDomain</rdfs:label>
    <rdfs:range rdf:resource="#Network"/>
    <rdfs:domain rdf:resource="#TransportStratum"/>
  </rdf:Property>
  <rdf:Property rdf:ID="hasServiceElement">
    <rdfs:range rdf:resource="#ServiceElement"/>
    <rdfs:domain rdf:resource="#ServiceStratum"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasServiceElement</rdfs:label>
  </rdf:Property>
  <rdf:Property rdf:ID="hasTransportStratum">
    <rdfs:range rdf:resource="#TransportStratum"/>
    <rdfs:domain rdf:resource="#Connection"/>
    <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
    >hasTransportStratum</rdfs:label>
  </rdf:Property>
  <rdf:Statement rdf:ID="Statement_1"/>
</rdf:RDF>
```

B Publications

Journals papers

- Franco Callegati, Aldo Campi, Giorgio Corazza, Dimitra Simeonidou, Georgios Zervas, Reza Nejabati. SIP-empowered Optical Networks for Future IT Services and Applications, accepted for publication in IEEE Communications Magazine, 2009.
- G. Zervas, Y. Qin, R. Nejabati, D. Simeonidou, F. Callegati, A. Campi, W. Cerroni, SIP-enabled Optical Burst Switching architectures and protocols for application-aware optical networks, Computer Networks, Elsevier, Vol. 52, No. 10, pp. 2065-2076, July 16, 2008.
- N. Ciulli, G. Carrozzo, G. Giorgi, G. Zervas, E. Escalona, Y. Qin, R. Nejabati, D. Simeonidou, F. Callegati, A. Campi, W. Cerroni, B. Belter, A. Binczewski, M. Stroinski, A. Tzanakaki, G. Markidis, Architectural approaches for the integration of the service plane and control plane in optical networks, Optical Switching and Networking, Elsevier, Vol. 5, No. 2-3, pp. 94-106, June 2008.

Peer Reviewed Conference papers

- B. Martini, A. Campi, F. Baroncelli, V. Martini, K. Torkman, F. Zangheri, W. Cerroni, P. Castoldi, F. Callegati. SIP-based Service Platform for On-demand Optical Network Services. Optical Fiber Communication Conference (OFC), San Diego, March 2009
- A. Campi, W. Cerroni, G. Corazza, F. Callegati, B. Martini, F. Baroncelli, V. Martini, P. Castoldi. SIP-based Service Architecture for Application-aware Optical Network. The second International Conference on Information and Communication Technology & Accessibility (ICTA), Hamamet, 2009
- Georgios Zervas, Yixuan Qin, Reza Nejabati, Dimitra Simeonidou, Aldo Campi, Walter Cerroni, Franco Callegati. Demonstration of Application Layer Service Provisioning Integrated on Full-Duplex Optical Burst Switching Network Test-bed. Optical Fiber Communication Conference (OFC), San Diego, California, February 2008

- F. Callegati, A. Campi, W. Cerroni. H-SIP: Hybrid SIP Network. EEE GLOBECOM, New Orleans, November 2008
- Franco Callegati, Aldo Campi, and Walter Cerroni, Extension of Resource Management in SIP. GridNets, Pechino, October 2008
- G. Zervas, R. Nejabati, A. Campi, Y. Qin, D. Simeonidou, F. Callegati, M. OMahony, M. Reed, S. Yu, Demonstration of a Fully Functional Optical Burst Switched Network with Application Layer Resource Reservation Capability. 33nd European Conference on Optical Communication (ECOC 2007), September 2007, Berlin, Germany
- F. Callegati, A. Campi, W. Cerroni, A cost-effective approach to optical packet/burst scheduling, Proc. of 2007 IEEE International Conference on Communications (ICC 2007), Glasgow, Scotland, June 2007.
- F. Callegati, W. Cerroni, A. Campi, G. Zervas, R. Nejabati, D. Simeonidou, Application Aware Optical Burst Switching Test-bed with SIP Based Session Control, Proc. of 2007 Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom 2007), Orlando, FL, May 2007.
- D. Simeonidou, G. Zervas, R. Nejabati, F. Callegati, A. Campi, W. Cerroni, SIP-enpowered OBS Network Architecture for Future IT Services and Applications (Invited Paper), Proc. of IEEE Broadnets 2007, Raleigh, NC, September 2007.
- A. Campi, W. Cerroni, F. Callegati, G. Zervas, R. Nejabati, D. Simeonidou, SIP Based OBS networks for Grid Computing, Proc. of 2007 Conference on Optical Network Design and Modeling (ONDM 2007), Athens, Greece, May 2007.
- G. Sousa Pavani, H. Waldman, F. Callegati, A. Campi, W. Cerroni, Adaptive Routing in Optical Packet Switching Networks using Ant Colony Optimization, Proc. of 13th International Conference on Telecommunications (ICT 2006), Funchal, Madeira, Portugal, May 2006.
- F. Callegati, A. Campi, W. Cerroni, Efficient Implementation of Scheduling Algorithms for Optical Burst/Packet Switching, Proc. of 2006 Conference on Optical Network Design and Modelling (ONDM 2006), Copenhagen, Denmark, May 2006.

Other publications and presentations

- Franco Callegati, Aldo Campi, Walter Cerroni. A network service plain for QoS using SIP. SIP Conference. Paris, France, January 2009

- A. Campi, W. Cerroni, G. Corazza, F. Callegati, B. Martini, F. Baroncelli, V. Martini, P. Castoldi, SIP-based Service Architecture for Application-aware Optical Network, 2009 Italian Networking Workshop, Cortina D'Ampezzo, Italy, January 2009.
- F. Callegati, A. Campi, W. Cerroni. A novel network interoperability architecture for SIP mobility. In: SIP Conference. Paris, France, February 2007
- F. Callegati, A. Campi, W. Cerroni, A Practical Approach to Optical Burst Switching, 2007 Italian Networking Workshop, Bardonecchia, Italy, January 2007.
- F. Callegati, W. Cerroni, A. Campi, G. Zervas, D. Simeonidou, Signalling in Optical Networks, 2007 Italian Networking Workshop, Bardonecchia, Italy, January 2007.
- F. Callegati, A. Campi, W. Cerroni. SIP for Grid networks, April 2007, Invited speaker at Open Grid Forum (OGF21), 2007
- F. Callegati, A. Campi, W. Cerroni. SIP for Grid networks, September 2007, Workshop at ECOC 2007.
- Franco Callegati, Aldo Campi, Walter Cerroni. Efficient Implementation of Scheduling Algorithms for Optical Burst Switching. In: Examining the Case for Optical Burst Switching. Workshop at OFC San Francisco, CA, 2006

Book chapter

- Campi, A. and Callegati (Authors) F. Dr. Nick Antonopoulos, Mr. Georgios Exarchakos, D. M. L. and Liotta, D. A. (editor). Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications. Chpter title: SIP protocol for supporting Grid Computing. IGI Global, 2009, In printing.

Part I

Bibliography

Bibliography

[ADF$^+$05] A. Anjomshoaa, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva. Job submission description language (jsdl) specification, version 1.0, 2005.

[BMMC07] Fabio Baroncelli, Barbara Martini, Valerio Martini, and Piero Castoldi. A distributed signaling for the provisioning of on-demand vpn services in transport networks. In *Integrated Network Management*, pages 789–792, 2007.

[BMMC08] Fabio Baroncelli, Barbara Martini, Valerio Martini, and Piero Castoldi. Supporting control plane-enabled transport networks within itu-t next generation network (ngn) architecture. In *NOMS*, pages 271–278, 2008.

[CCB$^+$06] Franco Callegati, Walter Cerroni, L. H. Bonani, F. R. Barbosa, Edson Moschim, and Gustavo Sousa Pavani. Congestion resolution in optical burst/packet switching with limited wavelength conversion. In *GLOBECOM*, 2006.

[CCC01] F. Callegati, W. Cerroni, and G. Corazza. Optimization of wavelength allocation in wdm optical buffers. *Optical Networks Magazine*, 2(6):66–72, 2001.

[CCC07] F Callegati, A Campi, and W Cerroni. A cost-effective approach to optical packet/burst scheduling. *Communications, 2007. ICC'07. IEEE International Conference* , 2007.

[CCR02] F. Callegati, G. Corazza, and C. Raffaelli. Exploitation of dwdm for optical packet switching with quality of service guarantees. *Selected Areas in Communications, IEEE Journal on*, 20(1):190–201, Jan 2002.

[CCXV99] F. Callegati, H. C. Cankaya, Y. Xiong, and M. Vandenhoute. Desing issues for optical ip routers. *IEEE Communications Magazine*, 37:124–128, 1999.

[CFZ96] I. Chlamtac, A. Farago, and Tao Zhang. Lightpath (wavelength) routing in large wdm networks. *Selected Areas in Communications, IEEE Journal on*, 14(5):909–913, Jun 1996.

[CGK92] I. Chlamtac, A. Ganz, and G. Karmi. Lightpath communications: a novel approach to high bandwidth optical wan. *IEEE Trans. Commun.*, 40(7):1171–118, 1992.

[CM99] Qiao C. and Yoo M. Optical burst switching–a new paradigm for an optical internet. *Journal on High-Speed Networks*, 8:69–84, 1999.

[CMP$^+$98] Guillemot C., Renaud M., Gambini P., Janz C., and Et. Transparent optical packet switching: the european acts keops project approach. *IEEE/OSA Journal of Lightwave Technology*, 12(16):2117–2134, December 1998.

[Com] Enlightened Computing. NSF- http://www.enlightenedcomputing.org.

[CQY04] Y. Chen, C. Qiao, and X. Yu. Optical burst switching: A new area in optical networking research. *IEEE Network*, 18(3):16–23, 2004.

[CTM07] Y. Chen, J. Turner, and P.F. Mo. Optimal burst scheduling in optical burst switched networks. *IEEE/OSA Journal of Lightwave Technology*, 25(8), 2007.

[EBS02] T.S. El-Bawab and Jong-Dug Shin. Optical packet switching in core networks: between vision and reality. *Communications Magazine, IEEE*, 40(9):60–65, Sep 2002.

[Enn06] R. Enns. Netconf configuration protocol, 2006.

[F.00] Callegati F. Optical buffers for variable length packets. *Communications Letters, IEEE*, 4(9):292–294, 2000.

[Far05] Igor Bryskin Adrian Farrel. *GMPLS: Architecture and Applications*. Morgan Kaufmann, December 20, 2005.

[FBD$^+$04] I. Foster, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, H. Kishimoto, F. Maciel, A.Savva, F. Siebenlist, R. Subramaniam, J. treadwell, and J. Von Reich. Open grid service architecture v1.0. Technical report, Open Grid Forum, 2004.

[FC07] G. S. Pavani F. Callegati, W. Cerroni. Key parameters for contention resolution in multi-fiber optical burst/packet switching nodes. In *In Proceedings of IEEE Broadnets*, 2007.

[FL03] F. Le Faucheur and W. La. Requirements for support of differentiated service aware mpls traffic engineering, 2003.

[For] Open Grid Forum. http://www.ogf.org/.

[FWCP04] Callegati F., Cerroni W., Raffaelli C., and Zaffoni P. Wavelength and time domain exploitation for qos management in optical packet switches. *Comput. Netw.*, 44(4):569–582, 2004.

[G.801] ITU-T G.8080/Y.1304. Architecture for the automatic switched optical network (ason), 2001.

[GL] G-Lambda. JPN http://www.g-lambda.net.

[GYR$^+$08] Zervas Georgios, Qin Yixuan, Nejabati Reza, Simeonidou Dimitra, Callegati Franco, Campi Aldo, and Cerroni Walter. Sip-enabled optical burst switching architectures and protocols for application-aware optical networks. *Comput. Netw.*, 52(10):2065–2076, 2008.

[HCAM98] David K. Hunter, Meow C. Chia, Ivan Andonovic, and Senior Member. Buffering in optical packet switches. *IEEE Journal of Lightwave Technology*, 16:2081–2094, 1998.

[HJ98] M. Handley and V. Jacobson. Sdp: Session description protocol, 1998.

[iG06] NSF initative. GENI, 2006. http://www.nsf.gov/cise/geni/.

[J.02] Rosenberg J. Integration of resource management and session initiation protocol (sip), 2002.

[JY02] Cochennec J.-Y. Activities on next-generation networks under global information infrastructure in itu-t. *IEEE Comm. Mag.*, 2002.

[KNT05] Knightson K, Morita N, and T. Towle. Ngn architecture: Generic principles functional architecture, and implementation. *Communications Magazine*, 43(10), 2005.

[LLF99] Tancevski L., Tamil L., and Callegati F. Nondegenerate buffers: an approach for building large optical memories. *Photonics Technology Letters, IEEE*, 11(8):1072–1074, Aug 1999.

[MBC05] Barbara Martini, Fabio Baroncelli, and Piero Castoldi. A novel service oriented framework for automatically switched transport network. In *Integrated Network Management*, pages 295–308, 2005.

[MM04] F. Manola and E. Miller. Optical packet switching in core networks: between vision and reality, 2004.

[MRZ04] G. Muretto, C. Raffaelli, and P. Zaffoni. Effective implementation of void filling in obs networks with service differentiation. In *In Proceedings of WOBS*, 2004.

[Net] Juniper Networks. http://www.juniper.net/.

[ooN04] General overview of NGN, 2004. ITU-T Y.2001.

[par] RDF parser. http://librdf.org/raptor/.

[Pat05] Achille Pattavina. Architectures and performance of optical packet switching nodes for ip networks. *J. Lightwave Technol.*, 23(3):1023, 2005.

[PHO] PHOSPHORUS. EU-IST- http://www.ist-phosphorus.org.

[Pin02] Armando Pinto. Optical networks: A practical perspective, 2nd edition. *J. Opt. Netw.*, 1(6):219–220, 2002.

[pjs] pjsip. http://www.pjsip.org/.

[PMKN06] M. Poikselka, G. Mayer, H. Khartabil, and A. Niemi. *The IMS: IP Multimedia Concepts and Services.* second ed. John Wiley, 2006.

[Racfingn06] Resource and admission control functions in next generation networks, 2006. ITU-T Y.2111.

[Ram02] R. Ramaswami. Optical fiber communication: from transmission to networking. *IEEE Communications Magazine*, pages 138–147, 2002.

[raotNr06] Functional requirements and architecture of the NGN release 1, 2006. ITU-T Y.2012.

[Roa02] A. B. Roach. Session initiation protocol (sip)-specific event notification, 2002.

[RSC$^+$02] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. Sip: Session initiation protocol, 2002.

[S.99] Turner Jonathan S. Terabit burst switching. *J. High Speed Netw.*, 8(1):3–16, 1999.

[SCLJ07] J. Song, M.Y. Chang, S.S. Lee, and J. Joung. Overview of itu-t ngn qos control. *IEEE Communications Magazine*, 45, 2007.

[Ser] Open SIP Server. http://www.opensips.org/.

[S.J96] Yoo S.J.B. Wavelength conversion technologies for wdm network applications. *Lightwave Technology, Journal of*, 14(6):955–966, Jun 1996.

[TCF$^+$03] S. Tuecke, K. Czajkowski, I. Foster, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, and P. Vanderbilt. Open grid service infrastructure v1.0. Technical report, Open Grid Forum, 2003.

[TYC$^+$00] L Tancevski, S Yegnanarayanan, G Castanon, L Tamil, F Masetti, and T McDermott. Optical routing of asynchronous variable length packets. *IEEE Journal on Selected Areas in Communications*, 18(10):2084–2093, 2000.

[vdHDT$^+$06] J.J. van der Ham, F. Dijkstra, F. Travostino, H.M. Andree, and C.T.A.M. de Laat. Using rdf to describe networks. *Future Generation Computer Systems*, 2006.

[WSC$^+$06] Jing Wu, Michel Savoie, Scott Campbell, Hanxi Zhang, and Bill St. Arnaud. Layer 1 virtual private network management by users. *IEEE Comm. Mag.*, 2006.

[WZC$^+$04] J. Wu, H. Zhang, S. Campbell, M. Savoie, G. v. Bochmann, and B. St.Arnaud. A grid oriented lightpath provisioning system. *Globecom*, 2004.

[XQLX03] Jinhui Xu, Chunming Qiao, Jikai Li, and Guang Xu. Efficient channel scheduling algorithms in optical burst switched networks. In *In Proceedings of IEEE INFOCOM*, pages 2268–2278, 2003.

[XQLX04] J. Xu, C. Qiao, J. Li, and G. Xu. Efficient burst scheduling algorithms in optical burst-switched networks using geometric techniques. *IEEE Journal on Selected Areas in Communications*, 22(9), 2004.

[XVC00] Yijun Xiong, Marc V, and Hakki C. Cankaya. Control architecture in optical burst-switched wdm networks. *IEEE Journal on Selected Areas in Communications*, 18:1838–1851, 2000.

[YDM00] Shun Yao, Sudhir Dixit, and Biswanath Mukherjee. Advances in photonic packet switching: An overview. *IEEE Communications Magazine*, 38:84–94, 2000.

[ZNSO06a] Georgios Zervas, Reza Nejabati, Dimitra Simeonidou, and Mike OMahony. Demonstration of an application-aware hybrid optical burst/circuit switched ingress edge router for future optical networks. *ECOC*, 2006.

[ZNSO06b] Georgios Zervas, Reza Nejabati, Dimitra Simeonidou, and Mike O'Mahony. Qos-aware ingress optical grid user network interface: High-speed ingress obs node design and implementation. In *Optical Fiber Communication Conference and Exposition and The National Fiber Optic Engineers Conference*, page OWQ4. Optical Society of America, 2006.

Listings

List of Figures

List of Tables