An Automatic Model-based Reconfiguration and Monitoring Mechanism for Flexible GMPLS-based Optical Networking Testbeds*

Fermín Galán Márquez and Raül Muñoz

Centre Tecnològic de Telecomunicacions de Catalunya (CTTC)
Parc Mediterrani de la Tecnologia, Av. Canal Olímpic s/n, 08860 Castelldefels, Spain
{fermin.galan,raul.munoz}@cttc.es

Abstract. Testbeds play a key role in the evolution of GMPLS-based Intelligent Optical Networks (ION) proving grounds in which new optical networking research ideas (e.g., new constraint-based routing algorithms) can be tested and evaluated. In order to be a productive experimentation environment, a GMPLS optical network testbed should be flexible, allowing the reconfiguration of as many different network topologies and configurations as possible. But usually this flexibility comes at the expense of high management costs when switching from one scenario to another is performed through time-consuming error-prone manual procedures. This paper describes an automatic model-based deployment mechanism that overcomes the limitations of manual reconfigurable testbeds, allowing high flexibility without involving high management costs. The model-based approach is not only suitable for deployment (and undeployment) but also for monitoring. The practical application of the mechanism to ADRENALINE testbed (a GMPLS-based all-optical transport network developed at CTTC) with the ADNETCONF tool is also described.

1 Introduction

The accelerating growth of Internet traffic is motivating the research on dynamic transport networks based upon recent advances in optical networking technologies such as Wavelength Division Multiplexing (WDM), Reconfigurable Optical Add Drop Multiplexers (R-OADM), Optical Cross Connects (OXC) and tunable lasers, capable of providing reconfigurable high-bandwidth, end-to-end optical connections. The introduction of the dynamism or intelligence in future optical networks can be achieved by means of a distributed optical control plane (i.e. routing and signalling). This control plane can be based on Generalized Multiprotocol Label Switching (GMPLS) [1] protocol architecture, an extension of MPLS (Multiprotocol Label Switching) to cover circuit-oriented optical switching technologies such as WDM. One of the major applications of GMPLS is

* This work was partially funded by the MEC (Spanish Ministry of Science and Education) through project RESPLANDOR under contract TEC2006-12910/TCM.

constraint-based routing (CBR), which is used to compute paths that satisfy various requirements subject to a set of constraints.

Performance analysis of CBR algorithms in optical networks has been widely studied in the past through simulations. However there is a considerable lack of experimental performance evaluation of real GMPLS-based CBR implemented in optical network testbeds.This experiments require flexible testbed platforms with a high degree of reconfigurability in order to allow the deployment of not only different network topologies (e.g., NSFNet, Pan-European network, metro rings, etc.) but also, different configurations for each topology. (e.g., CBR algorithms to test, available resources per link, etc.).

Manual reconfiguration is the usual procedure to achieve such flexibility, but introducing commands and configurations manually in the different testbed devices has several important drawbacks. Firstly, is a high time-consuming task due to a lot of time is employed performing tedious and mechanics operations. Secondly, reconfiguration needs specific knowledge not related with the goal of the testbed itself. Third, humans tend to make errors typing commands and writing configurations. Finally, manual reconfiguration is not scalable. The aforementioned problems can be overcome implementing automatic reconfiguration procedures. This paper describes one of such mechanism, based on optical network modelling and the processing of models to perform automatic reconfiguration (deploy and undeploy) actions (in addition, model-based monitoring is also possible) and applies it to ADRENALINE testbed [2], a flexible GMPLS-based all-optical transport network developed at CTTC laboratories.

The rest of the paper is structured as follows. The ADRENALINE testbed is introduced in Sect. 2 as example of flexible GMPLS-based optical network testbed. Then Sect. 3 describes the proposed automatic model-based reconfiguration and monitoring mechanism. After that, Sect. 4 focuses on the application of the proposed approach to ADRENALINE testbed. Finally, Sect. 5 concludes de paper and presents future work lines.

2 ADRENALINE Testbed: An Example of Flexible GMPLS-based Optical Network

The ADRENALINE (All-optical Dynamic REliable Network hAndLINg IP/Ethernet Gigabit traffic with QoS) testbed [2] is a GMPLS-based Intelligent Optical Network (ION) developed at CTTC laboratories (Fig. 1). It is composed by an all-optical transport network constituted by a metropolitan DWDM bidirectional ring with three colourless R-OADM nodes and tuneable lasers, providing reconfigurable (space and frequency) end-to-end lightpaths. Each optical node is equipped with a PC Linux-based Optical Connection Controller (OCC) for implementing the GMPLS-based distributed control plane. These three OCCs are named *optical* OCCs. The control plane is responsible for handling dynamically and in real-time optical node's resources in order to manage automatic provisioning and survivability of lightpaths through signalling and routing protocols. ADRENALINE deploys three optical bidirectional pairs of fiber. The Data Com-

munication Network (DCN) employed for exchange signaling and routing packets between OCCs is based on control channels carried at 1310 nm with a line rate of 100 Mb/s using point-to-point links. Note that the optical transport topology is fixed to a ring network and can not be modified.

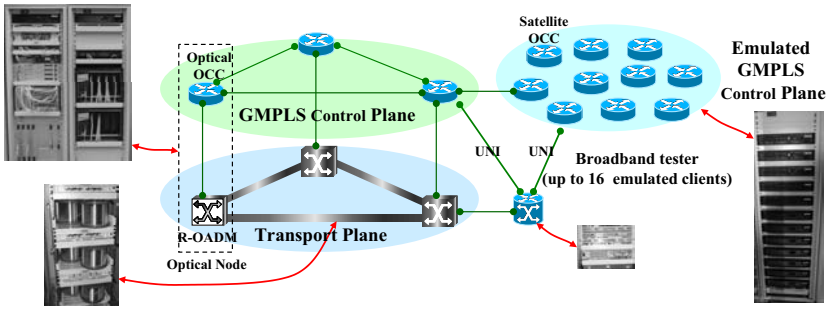

Fig. 1. ADRENALINE functional architecture

Given the fact that one of the focus of ADRENALINE testbed is the performance evaluation of GMPLS-based CBR algorithms and schemes, it was introduced a new set of ten OCCs named *satellite* OCCs. The difference with optical OCCs is that there is no optical hardware associated, that is, the optical hardware is emulated. The satellite OCCs introduce a new degree of flexibility, since there is no restriction neither on the optical network topology nor on the resources per link (e.g., number of available wavelengths, fibers, etc.). Regarding to the DCN, the satellite OCCs can be connected between themselves or with the fix ring of optical OCCs following any topology, through Fast Ethernet control channels. But in this case, the control channel is carried over emulated optical links between any pair of OCCs, allowing QoS constraints configuration (fixed and variable packet delays, packet losses, bandwidth limitations, etc.)[1]. In order to provide a flexible framework for DCN topology reconfiguration, the control channels are implemented using Virtual Local Area Networks (VLAN) 802.1q [3], configured in the layer 2 backbone Ethernet switches (named backbone nodes) and in the OCCs within the testbed. VLAN technology allows performing any layer 2 interconnections between network nodes absolutely decoupled of the physical infrastructure.

All OCCs, both optical and satellite, run the same set of protocols and processes: Resource Reservation Protocol Traffic Engineering (RSVP-TE [4]) for lightpath provisioning, Open Shortest Path First Traffic Engineering (OSPF-TE [5]) for topology and optical resources dissemination, Link Resource Manager (LRM) for management of node's optical resources, Single Network Management Protocol (SNMP [6]) implementing a management agent, and eventually[2], Optical Link Resource Manager (OLRM) for optical hardware control.

[1] Thanks to the Netem [7] emulation package installed in the OCCs.
[2] OLRM only runs in optical OCCs, but not in satellites.

Finally, ADRENALINE also includes client devices provided through a broadband tester that emulates a User Network Interface (UNI)-enabled IP router. It generates statistically UNI lightpath requests for Gigabit Ethernet, transmitting and analyzing IP packets once the lightpath is established.

3 Model-based Reconfiguration and Monitoring Mechanism

This section described the proposed model-based mechanism, first introducing scenario modeling concepts (Sect. 3.1), then detailing the model processing in depth (Sect. 3.2).

3.1 Scenarios and Models

From the model-based mechanism point of view, flexible GMPLS testbed is composed of a set of *network nodes* and *backbones nodes* (Fig. 2). The former are OCCs (optical and satellite) and client devices (note the same physical broadband tester emulates up to 16 separated clients), but not the optical hardware in the fixed transport plane. Backbone nodes are the Ethernet switches for DCN interconnection, supporting the 802.1q VLAN technology thus allowing (upon configuration) setting up different logical DCN topologies. Network nodes run processes (backbone nodes don't because their task is only providing VLAN-based interconnection). Processes running in each OCC may be different (e.g., satellite OCCs does not run OLRM, optical OCCs do) and, in different OCCs, the same process running on it can have different configurations (e.g., GMPLS OSPF-TE can be configured with different CBR algorithms in different OCCs).

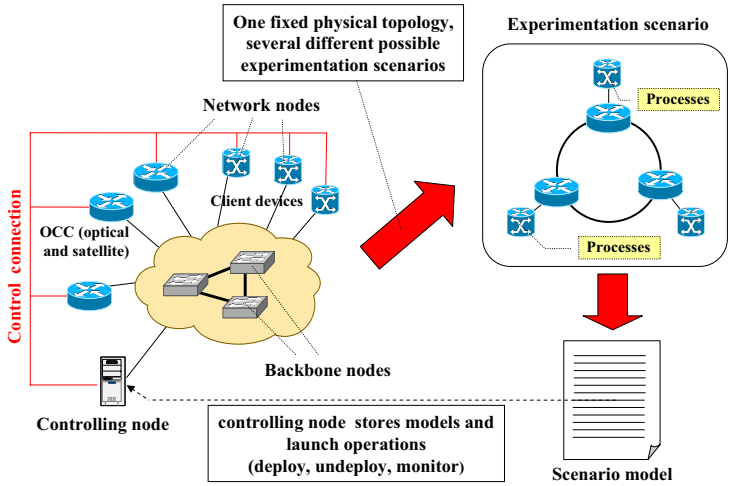

Fig. 2. Experimentation scenarios and models in optical flexible testbed

Experimentation scenarios are conceived by researchers, with their particular investigation goals in mind, obeying to diverse purposes (e.g., validating a theory, get measures, test new algorithm implementations, etc.). An experimentation scenario consists on a particular topology of network nodes (that will be set with the appropriated configuration of interconnection backbone nodes; for example, setting the proper VLAN configuration in switches and network elements operating systems) plus a particular configuration for each one of the processes running in those network nodes. A *scenario model* is a formal specification of an experimentation scenario, written in a particular language (with a particular syntax and semantic) so it can be processed automatically. Models include all the information regarding the network topology (network nodes involved in the scenario, the interconnection links among them, addressing issues, etc.) and the configuration of the processes running in each network node.

In theory any language can be used to write models. However, we will focus on XML-based ones. XML [8] is a W3C (World Wide Web Consortium) recommended general-purpose markup language for creating special-purpose markup languages. Currently, there are many well-known languages based on it (e.g., XHTML, RDF, SOAP, DocBook, etc.). The advantage of using XML is that, being a recognized standard with a wide support in the industry, there are many XML software components available that can be easily reused in order to build model processing tools. As a matter of fact, the ADNETCONF tool (the software that implements the model-based reconfiguration mechanism in ADRENALINE; described in detail in Sect. 4.1) reuses some of such XML modules.

3.2 Model Processing

The processing workflow is illustrated in Fig. 3. Firstly, the user writes the model of the desired scenario, either directly with a general purpose editor (XML is text-based human-readable) or, preferably, using a tool specifically designed for testbed model edition (like ADNETCONF, described in Sect. 4.1). Once the model has been created, it is ready to be processed. The processing engine (implemented with a software program) must run in a node (named *controlling node* in Fig. 2) physically interconnected to the testbed interconnection backbone. In addition, the controlling node must have pre-existing IP connectivity (*control connection* in the Fig. 2 and 3) to all the network and backbone nodes.

1. User launches the processing engine, using as input the model and the desired action to perform with it (deploy, undeploy or monitor). Typically, models are physically stored in the controlling node.
2. The processing of the model produces interaction with the network and backbone nodes through the control connection. There are two possible interactions: issuing commands (always) and installing configuration files (only in some cases during deploy). The nature of the commands depends on the interface provided by the node. In some cases (e.g., PC-based nodes) these commands are executed in the operating system of the node (using a remote shell). In others, they are issued using a proprietary API (Application Programming Interface) (e.g., vendor-specific broadband testers).

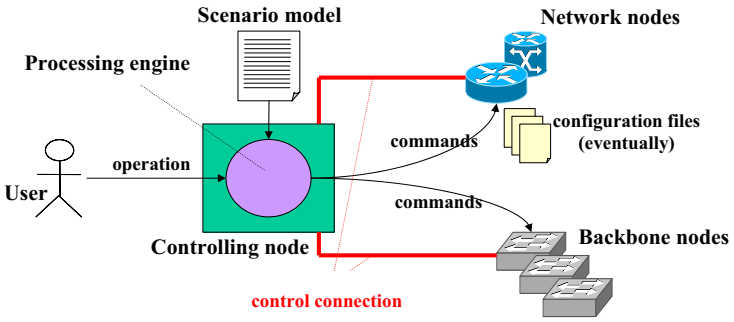

Fig. 3. Model processing

3. Processing engine reports the outcome of the action (i.e., whether the action was performed correctly or there was some error). In the case of monitoring, the result includes information regarding to scenario status.

There are three different processing actions (deploy, undeploy and monitor). The particular interactions performed during each action are described following:

Deploy. Processing of a model in order to set the corresponding scenario up in the testbed. The first commands issued to network and backbone nodes are for establishing the logical topology. It uses to imply enabling VLANs in backbone nodes and assigning VLANs and addresses (e.g., IP addresses) in network nodes. Then, commands are issued in order to start and configure the processes (only for network nodes, due to backbone nodes don't execute processes). The position (first setting the topology, next starting processes) is important, due to some processes need the topology configured in order to start properly (e.g., a routing process starting when no link has been established could fail). In the case some process needs a start-up configuration file, they are generated and installed before executing process starting commands.

Undeploy. Processing of a model in order to set the corresponding scenario down, reverting the testbed to a clear status (i.e., no scenario configured in the testbed). The scenario is supposed to have been previously deployed. Undeploy commands involve stopping of processes in first place, then removing the configuration of the logical topology. In this case configuration files are not required to be generated (they are needed for starting processes but not for stopping).

Monitor. Processing of a model in order to check the corresponding scenario status. The scenario is supposed to have been previously deployed. Monitor actions involve checking the status of links between network nodes and the aliveness of process (i.e., if the process is running or not). For each link and process, the status (up or down) is reported to the user, so he can know the status of the scenario accurately. Although deploy and undeploy are the main actions, monitor

is also a very useful auxiliary one that allows knowing if the scenario is working properly or, otherwise, easing the location and fixing of problems.

4 Application to ADRENALINE Testbed

This section presents the practical results of applying the proposed model-based reconfiguration mechanism (described in Sect. 3) to ADRENALINE testbed, first describing the ADNETCONF software tool (Sect. 4.1) and, secondly, showing the experimental results that demonstrates the benefits of the approach compared with manual reconfiguration (Sect. 4.2).

4.1 The ADNETCONF Tool

ADNETCONF (ADrenaline NETwork CONFigurator) is a software tool in charge of scenario model management in ADRENALINE testbed. It has been developed with two main objectives in mind. First, it provides and easy, quick and intuitive way of designing scenario models. Secondly, it implements deploy, undeploy and monitor operations (as described in Sect. 3.2) of such models. The structure of the tool resembles these goals; it is composed of two main modules, a graphic user interface (providing the model editor, in addition to the interface to launch operations) and a processing engine (implementing the actual operations).

As model editor, the GUI (shown in Fig. 4) provides a high-level perspective of the testbed. Models are created just "drawing" the network topology, using a library of items that includes the different network nodes in ADRENALINE testbed (OCCs and client devices) and configuring links among them. In addition, the configuration of the processes is performed just clicking in the particular element and editing a set of dialog boxes (one for each process). Model are stored in files, that can be saved, opened, modified, etc., following the same paradigm that any other conventional editor (e.g., .doc files for Microsoft Word).

Once the model has been completed, the user launches the desired operation clinking in the particular button on the GUI. The model is then serialized to a set of XML files that correspond to the formal representation of the scenario that the processing engine will understand. Up to six different XML files are produced; one describing the logical DCN topology (mandatory), the other five describing the configuration of the different processes (LRM, OSPF-TE, RSVP-TE, SNMP, OLRM), only needed when the particular process has been included in the scenario. This modularization allows a better implementation of the engine, composed of several modules each one specialized in validating and processing one XML file. In fact, XML files can be seen as an internal interface between the GUI (focused in model graphical design but not in processing) and the engine (focused in processing the model but not in designing aspects), hiding the complexity of the XML language to the user by the user-friendly GUI.

The ADNETCONF engine behaves like the one described in Sect. 3.2. The following is used to provide control connections: SSH (Secure Shell) for OCCs, telnet for Ethernet switches in the interconnection backbone and a RPC-based

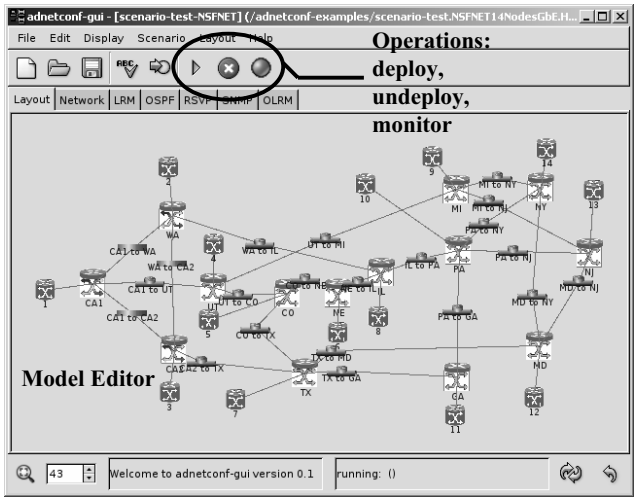

Fig. 4. ADNETCONF GUI

(Remote Procedure Call) vendor-specific proprietary API for the broadband tester implementing client devices. The commands issued in the most complete case (deploy) in ADRENALINE testbed include: VLANs set up for OCCs, client devices and switches; process start-up (LRM, OSPF-TE, RSVP-TE, SNMP and OLRM; two configurations files are generated and installed for OSPF-TE in each OCC before starting the process; one configuration file for LRM) and configuration of client devices in the broadband tester for connection requests, traffic generation and traffic analysis in the configured network topology.

4.2 Results

This section shows some practical results of the model-based automatic deployment method presented in this paper applied to ADRENALINE testbed, with the aim of illustrating its great benefits compared with conventional manual configuration.

The experiment consists in getting and analyzing some parameters when executing deploy and undeploy actions (monitor is left apart for the sake of briefness) with two different scenarios (one simple and the other one complex) using ADNETCONF. The simple one is composed of 3 OCCs (based on a ring topology), 3 client devices (each one attached to one OCC) and 3 links. The complex scenario resembles the mesh topology of the real NFSNet [9], composed by 14 OCCs, 14 client devices (each one attached to one OCC) and 20 links. The parameters analysed are execution time (how long lasts the completion of the operation), number of commands issued to network elements, number of configuration files generated, and their total size (the sum of all the lines in all the generated files). The summary of results is shown in Table 1.

Table 1. Experiment results

	Simple scenario (3 Nodes ring)		Complex scenario (14 Nodes NSFNet)	
	Deploy	**Undeploy**	**Deploy**	**Undeploy**
Commands	168	108	1046	722
Mean execution time	110 s	10 s	370 s	125 s
Commands per second (cmd/s)	1.52	10.8	2.82	5.78
Execution time of equivalent manual operation without errors (0.16 cmd/s)	1008 s (around 17 min)	648 s (around 11 min)	6276 s (around 105 min)	4332 s (around 72 min)
Configuration files	9	Not applicable	42	Not applicable
Configuration files lines	617	Not applicable	6492	Not applicable

One of the conclusions is that ADNETCONF can set up a scenario in the worst case (simple scenario deploy, around 1.52 cmd/s) much faster than any human administrator in the best case (supposing an average never-mistyping user[3], 0.16 cmd/s). Moreover, ADNETCONF ratio can be even higher in the case of undeploy (10.8 cmd/s) and when considering complex (and foreseeably more interesting) scenarios (deployment 2.82 cmd/s; undeployment 5.78 cmd/s).

More important than the improvement in execution time is that ADNETCONF removes the error debugging and fixing time of manual deployments (due to unnoticed typing errors introducing commands and configurations) since computer-based tools never performs such errors[4]. This time is difficult to estimate but usually is very high (it can take an entire day or even much more to locate and fix a bug) and proportional to the size of the scenario, so the tool is highly-scalable and especially valuable for complex scenarios (like the NFSNet).

Regarding to configuration files, ADNETCONF is able to generate 9 and 42 files respectively (involving 617 and 6492 lines) from a single model. Not only to generate all those files "by hand" can be an overwhelming task, but, even in the case they were created just once and reused after that, keeping the coherence in so many and so long files is difficult. Therefore, the ADNETCONF approach of generating all the needed files "on-the-fly" from the model is much better.

5 Conclusions and Future Work

The paper has described a reconfiguration mechanism for GMPLS-based optical testbeds in order to allow experimental performance evaluation of real GMPLS-based CBR under several topology and resource configurations. The method is based on modeling the desired experimentation scenarios and then processing models with automatic tools that deploy the desired configuration in the testbed

[3] Considering 30 wpm (words per minute) [10] and average 5-words commands.

[4] Software-caused errors impact is much smaller than human-caused. The former are deterministic and once fixed in the software they do not happen again. So, after a convenient software debugging time, the tool could be considered error-free.

(involving link connectivity and configuration for the processes running at each optical node) without human interaction. Therefore, the problems of manual reconfigurable testbeds (time consumption, high human error probability, specific knowledge on reconfiguration technologies -like VLAN- and scalability) are overcome, as proved with practical results in ADRENALINE testbed. Moreover, the generation of models does not need to be performed directly (in languages like XML) since specific user-oriented modelling tools can be developed (like ADNETCONF). Both factors (automatic reconfiguration and tool-oriented model management) lead to a highly productive and flexible experimentation environment for optical networking researches which key elements are *models*.

Furthermore, although this paper is focused in GMPLS-based optical testbeds, the mechanism is general enough to be applied to any other kind of testbed (optical or not) based on IP overlays, maybe using other interconnection technologies apart VLAN (e.g., GRE or IPSec tunnels). Such general model-based deployment method for IP networks is currently being patented by CTTC[5].

Currently, future work is focused in two lines. Firstly, the modelling technology and its performance should be further closely analysed in order to be enhanced. In addition, other fields in computers science (software engineering and knowledge engineering mainly) has long experience in models and meta-modeling and the technologies used there (like ontologies or model driven engineering) should be carefully studied. Secondly, ADNETCONF tool can be improved, increasing its usability (e.g., model coherence checking, real-time monitoring, etc.).

References

1. Mannie, E. (ed.): Generalized Multi-Protocol Label Switching (GMPLS) architecture. RFC 3945, IETF (2004)
2. Muñoz, R., *et al.*: The ADRENALINE testbed: integrating GMPLS, XML and SNMP in transparent DWDM networks. IEEE Communications Magazine, Vol. 43(8), IEEE (2005), s40–s48
3. IEEE 802.1Q Working Group: 802.1q: Virtual LANs IEEE (2001)
4. Berger, L. (ed.): Generalized Muti-Protocol Label Switching (GMPLS) signalling functional description. RFC 3471, IETF (2003)
5. Kompella, K., Rekhter, Y.: OSPF extensions in support of Generalized Multi-Portocol Label Switching. RFC 3471, IETF (2003)
6. Case, J., Fedor, M., Schoffstall, M., Davin, J.: A Simple Management Network Protocol (SNMP). RFC 1098, IETF (1990)
7. Hemminger, S.: Network Emulation with NetEm. LCA2005 (2005)
8. Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E.: Extensible Markup Language (XML) 1.0 (Second Edition). W3C Recommendation, W3C (2001)
9. Hülsermann, R., *et al.*: A set of typical network scenarios for network modeling. Procedings of 5th ITG-Workshop on Photonic Networks (2004)
10. Soukoreff, R., MacKenzie, I.: Theoretical upper and lower bounds on typing speed using a stylus and soft keyboard. Behaviour & Information Technology, Vol. 14, Taylor & Francis (1995), 370–379

[5] Method for Logical Deployment, Undeployment and Monitoring of a Target IP Network, PCT/EP2006/009960, October 2006.