ICTA'07, April 12-14, Hammamet, Tunisia

Service Plane-aided Visualization of Virtual Topology in transport networks

Valerio Martini[1], Fabio Baroncelli[2], Barbara Martini[2], Piero Castoldi[1],
Michele Guglielmucci[3], Aymen Garbouj[1], Siddharth Mishra[4]
[1]*Scuola Superiore S.Anna, Pisa, Italy;* [2]*CNIT, Pisa, Italy;*
[3]*Ministero Italiano delle Comunicazioni (ISCOM);* [4]*BTech IIT, Bombay, India*
{valerio.martini — castoldi}@sssup.it; {fabio.baroncelli — barbara.martini}@cnit.it

Abstract

Distributed applications such as Grid Computing, may benefit from the capability to dynamically request and display the status of the network for management purpose.

This work presents a graphical tool namely NetView that allows applications to display and monitor a generic network topology expressed as an XML-based file. In particular the NetView may be used for displaying the virtual topology, i.e. a topology of a Virtual Private Network (VPN), requested by applications to an instance of Service Plane (SP). The SP is a functional plane that provides network services in switched transport networks with a level of abstraction suitable for being invoked by applications. The advantage of the use of NetView and SP is that an application is able to visualize a virtual topology by issuing a simple request in terms of parameters such as the name of the VPN and without any reference to the transport network implementation details.

In addition, we validate our proposal with a testbed in which an application requests to the SP the topology of a VPN among a set of hosts and then displays it using an instance of the NetView.

1. Introduction

The telecommunication world is currently characterized by a significant request of connectivity with strict requirements in terms of bandwidth, resilience, and delay. In particular, advanced bandwidth-greedy applications, such as Grid Computing and Video On Demand, may benefit from the possibility to directly request the high-bandwidth connectivity services provided by the transport networks. The Automatically Switched Transport Network (ASTN) [1] is an architecture that introduces a new functional plane named Control Plane (CP) for enabling traffic engineering and on-demand Quality of Service (QoS) connectivity in transport networks. However, the inter-working between applications and the ASTN architecture is limited by the fact that applications require end-to-end network services whereas the ASTN is able to provide only edge-to-edge connectivity services via the User to Network Interface (UNI) and in addition without a level of abstraction suitable for being invoked by applications.

The Service Oriented Automatically Switched Transport Network (SO-ASTN), presented by the authors [2], enhances the ASTN architecture with a new functionality named Service Plane (SP) acting as a mediation layer between the applications and the Control Plane (CP) of the ASTN. The SP is able to coordinate and to compose the connectivity services provided by the CP via UNI in order to provide network service for the benefit of applications such as Virtual Private Network (VPN) facilities. The SP enables applications to request on-demand QoS network services without dealing with the network technology and topology details.

This work proposes a tool, named NetView, conceived for the graphical visualization of a generic network topology starting from input files expressed in the standard and XML-based GraphML format.

The NetView has a wide range of applicability since it may be used by application providers for the visual management of the IP-based connectivity established among hosts running distributed applications and by network operators for the management of their physical links as well.

In particular, we proposed the NetView to graphically represent a virtual topology (i.e., a topology relevant to a VPN) as obtained from an instance of SP by issuing a network topology request. The combined use of the SP and the NetView allows advanced applications to display for management purpose a VPN established among a set of hosts placed in different access networks without dealing with the network implementation details and without interacting with the ASTN architecture.

Since the NetView is implemented using Java technology, it is portable across different Operating Systems (OSs). Moreover, it may be customizable by simple text file according with the user needs and it is provided with algorithms that implement intelligent rendering of the network topology.

The paper is organized as follow. Section 2 reviews the Multi Protocol Label Switching (MPLS)-based VPN set-up and the SO-ASTN architecture. Section 3 describes the NetView software tools, its architecture, and the relevant implementation. Finally, section 4 presents a testbed in which an application accesses and monitors a virtual topology network established among a set of hosts thanks to the use of the NetView and of a SP prototype.

2. Provisioning of MPLS VPN services in transport networks

A Virtual Private Network (VPN) is a network that emulates a private Wide Area Network (WAN) by using shared or public infrastructures. In particular the BGP/MPLS VPNs [3] is a scalable and multi-domain Layer 3 solution for interconnecting access networks with secure tunneling based on IPv4 address schemes and IP/MPLS connectivity. It is standardized by the Internet Engineering Task Force (IETF) and foreseen the use of the MPLS forwarding technique and the Border Gateway Protocol (BGP) [4]. The MPLS is used for forwarding packets over the backbone. The BGP is used for distributing routing tables and for coordinating the Provider Edge (PE) nodes of the network. The primary goal of that VPN is to give IP connectivity services with the same scalability and flexibility of the private IP leased connectivity at much lower cost by using shared network infrastructure.

Advanced applications, such as Grid Computing, may benefit from the facilities offered by L3 VPN services in a WAN and in particular from the capability to directly trigger the connectivity services provided by the transport network. Unfortunately, the ASTN accepts on-demand connectivity requests only via the UNI that requires an intimate knowledge of the network infrastructure, that usually applications are not admitted to access for security reasons. In addition, the UNI signaling does not implement any mechanism for coordinating the simultaneous service set-ups among many Provider Edge (PE) nodes that may be needed, for example, to support Grid applications.

The SO-ASTN is an architecture offering a technology-independent invocation of network services with a level of abstraction typical of applications [2].

In this work the use of the SO-ASTN is twofold. On one hand, it provides on-demand VPN service to application in a network agnostic fashion (i.e., without deals with the technological and topology details of the network). On the other hand, it furnishes virtual topology information in a standardized format in order to be visualized by graphical software tools for management purpose.

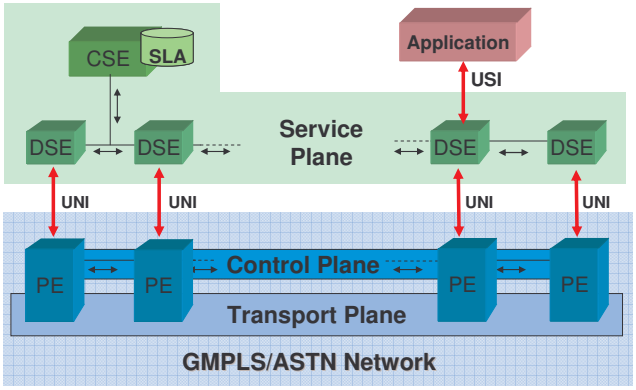

Figure 1: The SO-ASTN architecture

The SO-ASTN enhances the ASTN architecture [1] by introducing a new functional plane, named Service Plane (SP), logically located between the application and the CP, as shown in Fig. 1. Currently the SP architecture and implementation are object of study within the NOBEL 2 European project [5]. The SP composes and orchestrates the connectivity service provided by the CP at the boundary of the transport network and translates the network-agnostic application services request into a set of UNI-based directives to the network.

The SP architecture is constituted by one Centralized Service Element (CSE) and a set of Distributed Service Elements (DSEs), one for each PE in the network. The CSE is a centralized entities that identifies an application and authorizes the relevant service request using the information stored in its Service Level Agreement (SLA) database. The DSE is a distributed entity that accepts the on-demand network service requests issued by applications via the User to Service Interface (USI) and controls a specific PE provided with CP functionality for performing technology-specific network setting via UNI. The CSE and DSE entities are able to communicate thanks to a dedicated signaling. They may be located in a separate device as shown in the Fig. 1 or may be implemented as an extension of the current PE software equipment.

3. The NetView tool

The NetView is the graphical tool we realized to display and to monitor the topology of a network in a dynamic way and using standard topology input files. We aimed at the design a tool easily integrable with applications already deployed for the monitoring of different types of networks (e.g., LAN, WAN), without renounce to a simple and intuitive interface, a fast response time in terms of file elaboration, and an efficient use of the computational resources. Moreover, NetView has been conceived to be completely independent on the network technology, and to be portable across different Operating Systems (OSs).

3.1 The NetView topology files

The NetView represents a network as a graph expressed using the Extensible Markup Language (XML) [6]. Originally designed to meet the challenges of large-scale electronic publishing, XML is playing an increasing important role in the exchange of a wide variety of data on the Web. The power of XML consists in a large library available over all operating systems and programming languages that allows to easily parse, format, navigate and validate the XML-based information.

In order to be independent of a proprietary file format, the NetView is able to manage XML-based files consistent with the GraphML [7] technology. The GraphML is a standard XML-based syntax conceived to describe the structural properties of a generic graph.

In particular, the GraphML document defines the *graph*, the *node*, and the *edge* XML tags representing respectively a graph, its nodes, and the link between 2 nodes (named source and target). Each node has an identifier represented by the XML-Attribute *id*, which must be unique within the entire XML document. In addition, a node may be characterized by a set of attributes that are used to add details to the network topology description. This allows the NetView to represents the nodes with a number of attributes suitable to realize an exhaustive graphical visualization of a generic network topology.

An example of GraphML file representing a simple network with three nodes, named A,B,C, with two attributes each, interconnected with bidirectional links, is reported below.

```
<graph edgedefault="bidirected">
   <node id="A" attribute1="atr1a"
                attribute2="atr2a" />
   <node id="B" attribute1="atr1b"
                attribute2="atr2b" />
   <node id="C" attribute1="atr1c"
                attribute2="atr2c" />
   <edge source="A" target="B" attribute1="LinkAB"/>
   <edge source="B" target="C" attribute1="LinkBC"/>
   <edge source="C" target="A" attribute1="LinkCA"/>
</graph>
```

3.2 The NetView architecture

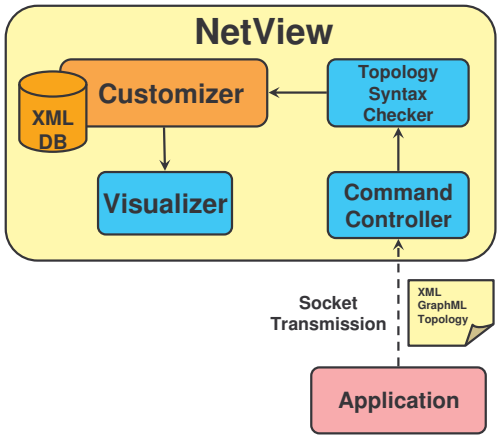

Figure 2: Network Viewer Functional Block

The NetView internal structure is presented in Fig. 2 and it is composed by the following functional blocks:

- The **Command Controller**: it manages the communication with the applications and accepts the relevant network topology files that need to be represented.
- The **Topology Syntax Checker**: it controls that the syntax of the XML topology files received from the Command Controller is consistent with the GraphML structure.
- The **Customizer**: it transforms the XML topology files in order to customize the graphical visualization of the network according to the information stored in configuration files provided by the user. For example, they may be used to indicate the images that should be used for representing the network entities or the filtering algorithms that should be implemented for representing specific portion of the network.
- The **Visualizer**: it performs the rendering of the topology using the information obtained from the Customizer. In particular, each time the user points the mouse arrow over a node or an edge element, the Visualizer is able to show the information expressed as attribute tags in the XML topology files. Moreover, the Visualizer is

provided with algorithms to arrange the network entity positions according to a customized hierarchical structure.

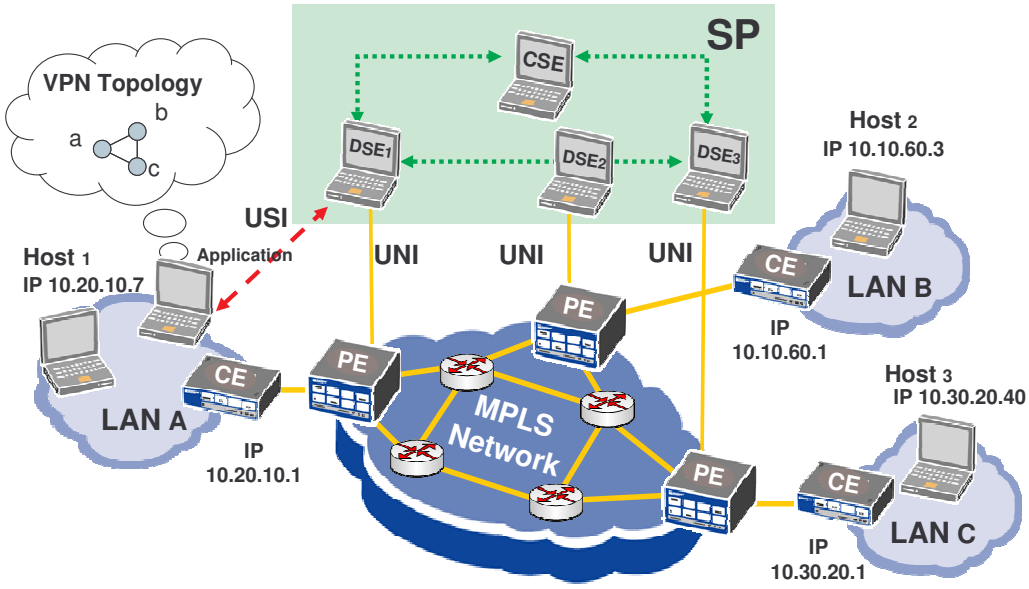

Figure 3: The Testbed

3.3 The NetView implementation

The NetView implementation presented in this work is based on the use of the Java technology and the Jung library. Java was chosen because it allows robust Object-Oriented programming, it permits to create portable code and it is provided with an extended library that simplifies the software development. Jung [8] is an open-source Java-based software library, that has been specifically developed as a common and extensible languages for the manipulation, analysis, and visualization of graphs and networks.

The communication between the NetView and the applications is based on a client/server paradigm and it is implemented in the Command Controller using TCP Socket programming. A TCP Socket is a two-way communication link between two applications running on the network based on the TCP protocol. This choice allows the NetView to communicate with applications distributed on the network and not necessary placed on the same host. Moreover, the TCP Socket is standardized thus the NetView is able to communicate with applications implemented using different programming language and running over different OSs.

The validation of the topology files give as input to the NetView is performed by the Topology Syntax Checker using the XML-Schema technology [9]. The XML-Schema is an XML-based language that defines the structure, the content and the semantics of a set of XML document. Since the topology files accepted by the NetView are XML-based, the XML-Schema technology supported by Java allows to easy check if the input files is consistent with the GraphML specifics and then to discharge invalid topology. In particular, the XML-Schema for the GraphML is available at [7].

4. Experimental Validation

The testbed shown in Fig. 3 has been configured to validate the NetView architecture and to verify the proposed integration with the network topology service provided by the SO-ASTN. In particular, the testbed reproduces a scenario in which an application needs to visualize the topology details relevant to the VPN established among 3 Hosts placed in different LANs and connected to the same MPLS-based transport network.

The testbed comprises a transport network, three LANs, a SP implementation prototype, and a PC running an application able to request network service to the SP and a NetView instance as well.

The transport network is composed by commercial routers provided with Fast Ethernet (FE) and Gigabit Ethernet (GE) interfaces that are interconnected as

shown in Fig. 3. All the routers have been configured to implement the MPLS, OSPF, and RSVP protocol stacks. In particular, the three edge routers have been configured as Provider Edge (PE) and thus configured to additionally implement the BGP protocol stack.

The LANs are composed by a set of hosts connected to the transport network through routers configured as Customer Edge (CE). Each LAN is configured as a Stubbed Network i.e. a network with private IP addresses where the relevant PE do not distribute the LAN route information outside the LAN itself. This means that the reachability of an Host that belong to the LAN is possible only using the NAT/PAT (Network Address Translation/Port Address Translation) functionality of the PE.

The possibility that the LANs could share the same private address is not considered in this work since it is still an open issue in VPN recommendations [3]. For the testbed purpose a BGP/MPLS VPN is established among three hosts placed in different LANs.

The SP is composed by 3 PCs, each of them running an instance of DSE and equipped with Linux Kernel v2.6. In addition, a different PC is used to run an instance of CSE. Each PC has been provided with Ethernet interface connected to the controlled PE and set with a public IP. Thanks to the public IP addresses the DSEs and CSE can inter-communicate across the transport network.

The SP implementation was presented in [10]. It is based on Java v1.5 programming language because it allows easy multi-threading and networking programming and code portability. All the SP messages has been implemented as XML documents. This choice allows the use of the XSLT [11] technology to easily perform data processing in the service mapping and of the XML Schema [9] for data structure checking in message validation.

Since current routers are not provided with UNI interface because not yet standardized, the DSEs used in the testbed communicate with the PEs via the XML-based configuration interface provided by several commercial routers. This choice is consistent with the SP principles because that kind of interface has features equivalent to the UNI such as the possibility to create, delete, update and query the connections, and the possibility to request information about the traffic and the network performance. Referring to Fig. 3 the application runs in an host placed in LAN_A. It requests to DSE_1 the topology relevant to the VPN previously established among the hosts 1,2,3. That request is issued via USI only in terms of the application identity and the VPN name (i.e., a label that identifies the requested VPN among the VPNs established in the network) and without any reference to the transport network details. The SP, according with its internal signaling, elaborates a response in which the VPN topology is expressed as an XML files consistent with the GraphML standard and reported below.

```
<graph edgedefault="bidirected">
  <node xmlns="" type="CE"
      id="10.20.10.1" ip="10.20.10.1"/>
  <node xmlns="" type="CE"
      id="10.10.60.1" ip="10.10.60.1"/>
  <node xmlns="" type="CE"
      id="10.30.20.1" ip="10.30.20.1
  <node xmlns="" type="Host"
      id="10.20.10.7" ip="10.20.10.7"/>
  <node xmlns="" type="Host"
      id="10.10.60.3" ip="10.10.60.3"/>
  <node xmlns="" type="Host"
      id="10.30.20.40" ip="10.30.20.40"/>
  <edge xmlns=""
      source="10.20.10.7" target="10.20.10.1"
      type="virtual" bandwidth="10M"/>
  <edge xmlns=""
      source="10.10.60.3" target="10.10.60.1"
      type="virtual" bandwidth="10M"/>
  <edge xmlns=""
      source="10.30.20.40" target="10.30.20.1"
      type="virtual" bandwidth="10M"/>
  <edge xmlns=""
      source="10.20.10.1" target="10.10.60.1"
      type="virtual" bandwidth="10M"/>
  <edge xmlns=""
      source="10.10.60.1" target="10.30.20.1"
      type="virtual" bandwidth="10M"/>
  <edge xmlns=""
      source="10.30.20.1" target="10.20.10.1"
      type="virtual" bandwidth="10M"/>
</graph>
```

The application transmits the topology files to an instance of NetView running on the same host via TCP Socket communication.

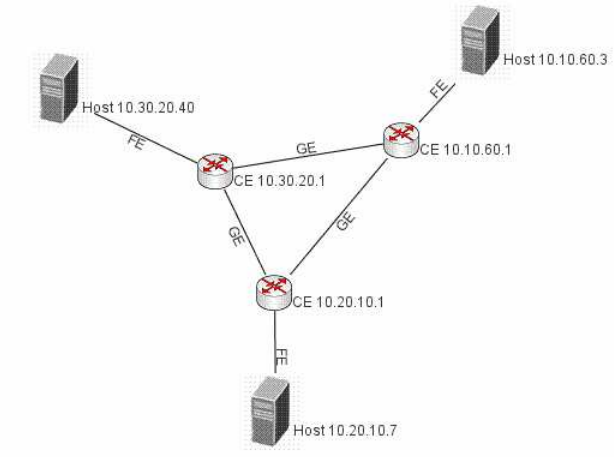

Figure 4: Snapshot of the NetView network visualization

A snapshot of the VPN topology visualized by the NetView is shown in Fig. 4. The visualization is very effective since all the VPN entities (the CE routers, the Hosts) are represented with their standards icons and are identified by their IP addresses. In particular, the lines represents the logical connections among the Hosts since the physical connection among CEs, PEs, and inner routers are completely hidden to applications. When the user clicks over a node or a link, the NetView is able to visualize the list of node and link parameters respectively expressed in the topology file as XML attributes.

9. Conclusion

This work has presented a software tool, named NetView, able to graphically represent a generic network topology for monitoring and management purpose. The principal advantage of the use of the NetView is that it may be simply integrated in complex applications running over different Operating Systems. In fact, it accepts standard GraphML files as input and it is implemented using Java technology. Moreover, it can be easily customized by means of XML-based configuration files. In addition, the Netview may be used for the visualization of the topology of an ad-hoc network (e.g., an experimental sensor network) when that network supports a database containing the updated topology information and when that database is accessible via an XML-based interface.

The NetView was applied in a testbed in which an application needs to access and to visualize the information relevant to a VPN established among a set of Hosts and across an MPLS-based transport network. Since current networks do not allow applications to request on-demand network service, we used the Service Plane (SP) of the SO-ASTN architecture for performing the mapping between a VPN topology request issued by an application and the relevant set of directives to the transport network. The testbed demonstrated that the use of the NetView combined with the SP allows applications to visualize on-demand the network topology relevant to their VPN without the knowledge of the network implementation details.

Future evolution of this work will be the integration of the NetView with the DSE in order to provide the SP with the capability to visualize the physical topology of the controlled network.

Acknowledgement

This work has been supported by Italy-Tunisia FIRB project "Software and Communication Platforms for High-Performance Collaborative Grid" (RBIN043TKY).

References

[1] ITU-T, “Recommendation G.807/Y.1302, Requirements for Automatically Switched Transport Networks (ASTN),” July 2001.

[2] B.Martini, F.Baroncelli, and P.Castoldi, “A novel service oriented framework for automatically switched transport network,” in *9th IFIP-IEEE International Symposium on Integrated Network Management (IM)*, Nice, France, 15-19 May 2005.

[3] E. Rosen and Y. Rekhter, “RFC 4364 - BGP/MPLS IP Virtual Private Networks (VPNs),” IETF, Tech. Rep., February 2006.

[4] “RFC 2547 - BGP/MPLS VPNs,” IETF, Tech. Rep., March 1999.

[5] “Next Generation Optical Networks for Broadband European Leadership (NOBEL) project phase 2: http://www.ist-nobel.org.”

[6] “Extensible Markup Language (XML) 1.0 (Third Edition). http://www.w3.org/XML/, 4 February 2004.”

[7] “GraphML http://graphml.graphdrawing.org.”

[8] “Java Universal Network/Graph Framework (JUNG) http://jung.sourceforge.net/.”

[9] “XML Schema Part 1: Structures Second Edition,” http://www.w3.org/TR/xmlschema-1, 28 October 2004.

[10] B.Martini, F.Baroncelli, P. Castoldi, and A. Aprigliano, “Experimental validation of a service oriented network architecture applied to global grid computing,” in *AXMEDIS 2005*, Florence, Italy, 30 November - 2 December 2005.

[11] “XSL Transformations (XSLT) Version 1.0,” http://www.w3.org/TR/xslt, 16 November 1999