Modeling and Simulation of DiffServ Scenarios with the NSDL Framework

Eduardo M.D. Marques, José J. F. Sousa and Paulo N.M. Sampaio
Madeira Interactive Technologies Institute (M-ITI)
University of Madeira (UMa)
9000-390 Funchal, Madeira, Portugal
Email: emarques@uma.pt, jose.jorge.sousa@gmail.com, psampaio@uma.pt

***Abstract*—To provide an integrated and optimized management of network modeling and simulation tools is a complex task. Indeed, the underlying information structures used by these tools are, most of the times, very distinct and it is not simple to relate them. Moreover, the re-utilization of network descriptions and other network related data among tools is not current. The use of a standard data structure, or language, to describe data networks would promote the interoperability among network tools, providing the users with the possibility of applying new platforms and tools to validate their network scenarios. This work proposes the modeling and simulation of Differentiated Services networks based on the Network Scenario Description Language Framework.**

I. Introduction

Currently there are different tools to provide the management of networks. These tools are developed in general as proprietary or open-source applications. The proprietary tools, usually are not free, and offer a broad possibility of options and functionalities, being supported by a more complete and accurate documentation. The open-source tools are free, normally being developed in order to provide the solution of some specific issues and problems, limited to some specific domains, and, most of the times, having a scarce and incomplete documentation. It is also important to consider the functionalities provided by these network tools. Some tools are very limited in terms of number of functionalities available and have a very specific application. Others are based on a large set of models and provide a more complex analysis over a network.

If all these tools were applied jointly in a coordinated way, they could provide a solid and helpful environment for optimizing the management of a network. Nevertheless, the data formats used in general by the tools are very distinct and, most of the times, incompatible. Thus, in order to provide a generic solution for promoting interoperability among different network management tools, this paper presents a framework which relies on the *Network Scenarios Description Language* (NSDL), which has been proposed as a common solution that can be applied to assist network managers with the optimization of the network during its life cycle.

Figure 1 illustrates the layered organization of the several components of the NSDL framework. The top and bottom layers represent the existing networks tools. The top layer represents the management tools to provide the modeling, monitoring and visualization of networks (e.g., GUIs such as, topology generators, operation and failures monitoring, statistics and results, etc). The bottom layer represents the network analysis tools, such as network simulators, management tools, security evaluation, etc. Actually, some of the existing management platforms support both bottom and upper layers; however, most of them are purpose-oriented and are present only in one of the layers. The NSDL represents a middleware layer to connect either layers or different networks tools. The GUIs may read and write the created network scenarios and, through specialized *Application Programming Interfaces*, the bottom layer tools are invoked to execute some operations over the network. In this paper we illustrate the utilization of the NSDL framework to provide the network modeling and simulation of a Quality of Service network.

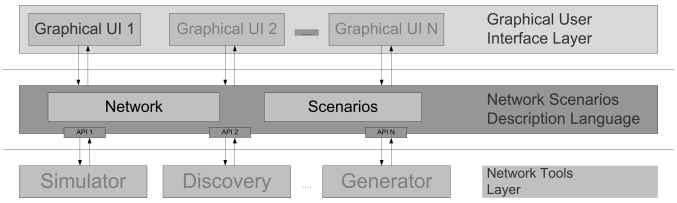

Fig. 1. NSDL Framework layered architecture

The remaining of the paper is organized as follows: Section II presents the *Network Scenario Description Language*. Section III illustrates a case study with the integration of some tools for the modeling, simulation and analysis through the utilization of NSDL. The last section presents some conclusions and future works.

II. Network Scenarios Description Language

The purpose of the *Network Scenario Description Language* (NSDL) is to provide a vocabulary and a set of rules, both able to support the description of wired and wireless data networks and the information of the contexts, or domains, where those networks are used or evaluated.

The principles applied to the design of NSDL were (1) simplicity, which means the language has to be simple and clear not only to be manipulated by an application or tool, but also possible to be edited by a human user with a simple text editor; (2) definition of multiple abstraction levels, allowing to specify not only simple high level descriptions of a network

scenario, but also, if needed, to have the possibility to create a very detailed description of all network scenario objects and its parameters; and (3) extensibility, implying that new objects and parameters can always be incorporated in future descriptions.

The language which has been chosen to support NSDL is XML [1] due its richness and flexibility. An NSDL representation is presented in figure 2 and its structure is composed of two basic elements: *Network* and *Scenarios*.

The *Network* element is composed of *Templates*, *Objects* and *Views*. Since the *Objects* element contains the description of the network topology, it is the main component of the language. This element is composed of nodes, links and domains. Some other important, but not mandatory elements are *Templates* and *Views*. The *Templates* element is important to simplify the description of similar objects. The *Views* element is a mechanism applied to group network objects to be used in the scenarios.

In the *Scenarios* element, two elements are introduced: *Visualization* and *Simulation*. The *Visualization* element provides additional information to enrich the network description, such as objects positioning in the GUI's and graphical data. All the parameters needed to implement a particular simulation over the network using a generic or particular simulation tool is defined in the *Simulation* element. The use of a single language description in several moments of a network life cycle could be very advantageous. If the users responsible for the network are familiar with that language, they can easily understand the current state of the network, thus management of the network is optimized.

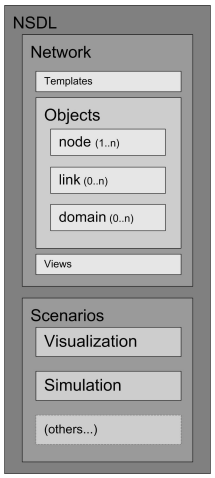

Fig. 2. NSDL file structure

Interoperability is an important NSDL's advantage. If a user deploys several NSDL compliant tools, he can easily analyse a network using each one of them, thus obtaining integrated results.

NSDL can be extended by adding new parameters and creating new objects over the already defined basic objects. For instance, NSDL can be extended to support the description of wireless sensor networks.

In the next section we present a case study to illustrate the utilization of NSDL framework.

III. Case Study

To illustrate the utilization of the NSDL framework for providing the interoperability of different network tools, we propose the integration of some network tools according to the life cycle for a network scenario (modeling, NSDL mapping and simulation/analysis), as presented in the following sections.

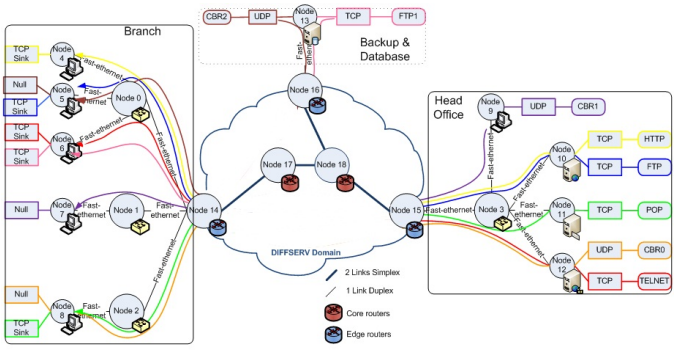

Fig. 3. Network scenario used as example

A. Network Modeling

For the modeling of the network scenarios, we have adopted the network modeling tool called *Visual Network Simulator* (VNS) [2], which was developed at University of Madeira, and provides the broad representation of all the components of a network (nodes, links, protocol, traffic agents, etc.). Besides representing all the components of a scenario, VNS also provides the modeling of unicast/multicast scenarios, *Differentiated Services Architecture* (DiffServ) [3] Scenarios, and further advanced object configuration. After having created and configured the scenario, VNS provides the automatic generation of NDSL based on the internal representation of the network scenario and its components.

The network scenario presented in Figure 3 is related to the backbone of a company which interconnects three different *Local Area Networks* (LAN's) associated, respectively, with the head office, a remote branch and Backup & Database resources (redundancy).

In this implementation the core network is supposed to provide QoS guarantees through the utilization of DiffServ. For this purpose, three main service classes were adopted: *Best Effort* (BE), *Assured Forward* (AF) Gold and AF Platinum.

In the next section, we present how NSDL plays an important role in this case study, in order to describe these set of configurations introduced previously.

B. The NSDL representation

Once the adopted network scenario is defined, we can illustrate the expressiveness of NSDL to declare and specify the network scenario's components as well as its respective attributes. In order to do so, we need to establish a correspondence between our scenario's main components and the elements of the NSDL structure.

The NSDL language provides a base object (called *<node>*) to identify generically all the nodes in a network. It can represent several abstractions of specific objects such as *<computer>*, *<switch>*, *<router>*, etc., which in our scenario is used to declare servers, routers (edge and core) and generic terminals. The *<link>* object can be applied to connect several nodes. As expected, some of its parameters are *bandwidth* and *delay*. A possible specialization for the object

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<nsdl>
   <network>
      <objects>
         ...
         <computer id="node_5"><!-- Client-->
            <ftp id="appFTPclt">
               <role>client</role>
               <src.app>appFTPsrv</src.app>
            </ftp>
            <cbr id="appCBR2clt">
               <role>client</role>
               <src.app>appCBR2srv</src.app>
            </cbr>
         </computer>
         ...
         <link id="RoutersC0CPrd" template="sLink">
            <connection source="rCore0_17"
                  destination="rCProducao_14" />
            <bandwidth>10Mb</bandwidth>
         </link>
         ...
      </objects>
   </network>
   <scenarios>
      ...
   </scenarios>
</nsdl>
```

Fig. 4. Node and Link declaration example

```
...
<dsedgenode id="node_16" template="simpleRouter" >
   <entry>
      <marker>
         <src.node>servBackup_BD_13</src.node>
         <dst.node>wks3Redaccao_6</dst.node>
         <phb.code>0</phb.code>
      </marker>
      <policer>
         <TSW2CM>
            <new.phb.code>2</new.phb.code>
            <cir>0.1Mb</cir>
         </TSW2CM>
      </policer>
   </entry>
   <entry>
      ...
   </entry>
</dsedgenode>
...
```

Fig. 5. Router policy configuration example

<link> is, e.g., the object *<fastethernet>*. Both *Node* and *Link* are illustrated in Figure 4.

NSDL also supports the specification of DiffServ elements and their attributes in order to describe and simulate QoS scenarios. For instance, in our scenario some nodes that represent edge routers, and their behaviour can be described through the utilization of elements such as *<dsedgenode>*, *<marker>*, *<policer>*, among others (Figure 5). The NSDL excerpt presented in Figure 5 defines an edge router's configuration (node_16) to manage the incoming/out coming traffics. Each entry includes the classification of a traffic flow and its policing. The *<marker>* tag is used to define a traffic flow and includes the reference of the source and destination nodes (*<src.node>* and *<dst.node>*) and its correspondent traffic class (defined by *<phb.code>*). NSDL will allow the marker to use the 5 tuples to define a particular packet flow, but, because of the ns-2 libraries limitations, we must, in this scenario, use only the source and destination nodes to specify the traffic. The *<policer>* tag is responsible to define which policer will be used to monitor and re-classify, when needed, each flow (exemplified by *<TSW2CM>* in Figure 6 and with the reference to the new DiffServ Code Point, *<new.phb.code>*). Each policer entry will include a CIR (*committed information rate* - tag *<cir>*) value which once surpassed, increases significantly the probability of packet loss, and, depending on the policer, some further parameters such as *cbs*, *ebs*, among others. Another node object from the DiffServ architecture is the *dscorenode*, needed to implement the per hop behaviours' inside the domain. Along with the phb's there are also the schedulers and the more common ones are already included in NSDL, such as *Round Robin*, *Priority Queuing*, and, *Weighted Round Robin*.

One advantage of NSDL over the use of a particular simulation language (*OTcl* [4] in *Network Simulator 2* (ns-2) [5]) is the more close relation between description and network objects. For example, in NSDL the Diffserv modules, classification or marker, policing and per hop behaviors, are described in the architecture related objects, that are edge and core nodes. In the case of the ns-2 simulator, the same properties are implemented in two simplex links (linking an edge and core node or two edges or two core nodes), mixing edge and core DiffServ functions. In our opinion, this leads users to some difficulties in the execution of DiffServ simulation in ns-2.

A significant feature of NSDL is that it provides the integration of different network management tools. This integration can be achieved using the element Scenarios of this language. The Scenarios is useful to describe specific information related to a particular network tool, which offers a particular perspective over the described network. For the moment, two scenarios have been defined for the modeling and simulation tools, which are respectively the visualization and simulation.

C. Simulation procedure

The NSDL framework allows the integration of a network modeling tool with a network simulation tool for the verification of the correctness of the network topologies and their associate traffic distribution, eventually considering possible Quality of Service constraints as described previously.

Once VNS has generated automatically the NSDL representation, it is possible to apply XSL transformations [6] in order to generate automatically compatible descriptions with any other tool. In this case, we generated *OTcl*, which is the format used by ns-2 for the simulation of the network scenario and further utilization of its associated tools, such as Network Animator (NAM) [7] and xGraph [8], for the visualization and analysis. In Figure 6 we present a short description of the generated *OTcl* to be used with ns-2, more particularly the part related to the construction of a DiffServ object configuration in *OTcl*.

The first experiments with the implementation of the NSDL framework allowed us to validate the expressiveness of this language to describe completely all the components of a network, including their QoS features. For enabling the simulation of the network scenarios, the NSDL scenarios element

```
...
$qEdgeCPCore0 meanPktSize 1000
$qEdgeCPCore0 set numQueues_ 4
$qEdgeCPCore0 setNumPrec 3
$qEdgeCPCore0 addPolicyEntry [$n4 id] [$n10 id] TSW2CM 0 $cir2
$qEdgeCPCore0 addPolicyEntry [$n5 id] [$n10 id] TSW2CM 36 $cir2
...
$qEdgeCPCore0 addPolicerEntry TSW2CM 0 2
$qEdgeCPCore0 addPolicerEntry TSW2CM 36 38
...
$qEdgeCPCore0 configQ 0 0 0 10 1.00
$qEdgeCPCore0 configQ 0 1 2 10 0.80
$qEdgeCPCore0 configQ 1 0 10 40 0.10
...
$qEdgeCPCore0 addPHBEntry 0 0 0
$qEdgeCPCore0 addPHBEntry 2 0 1
$qEdgeCPCore0 addPHBEntry 36 3 0
                                ...
```

Fig. 6. Extract of *OTcl* generated automatically

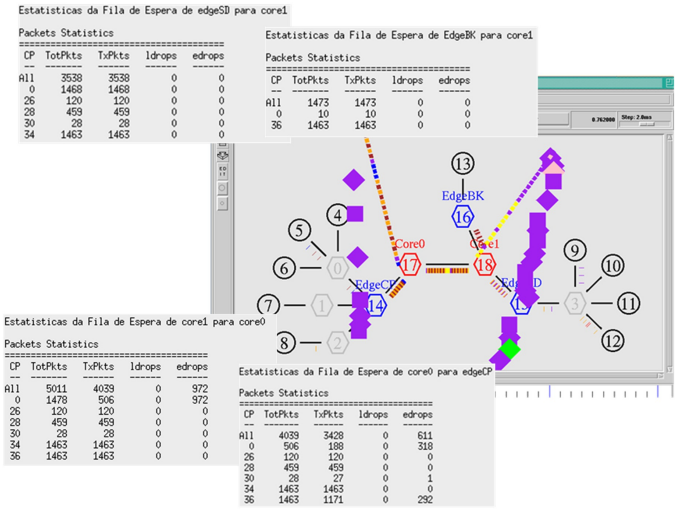

Fig. 7. Results of a DiffServ environment simulation scenario

was also improved to allow the complete description of the simulation characteristics related to ns-2. This experiment was relevant to realize the potential of NSDL as an integration medium able to provide the interoperability among different network management tools.

D. Simulation Analysis

The network scenario to simulate includes an AF-Gold class for the three TCP traffics and an AF-Platinum class for two CBR traffics (CBR0 and CBR2), which have higher priority compared to the other traffics. Meanwhile, traffic CBR1 and TCP traffic (POP and FTP) are holders of the best-effort class.

Once the *OTcl* code is generated automatically from the produced NSDL, we have simulated the network scenario with ns-2 and the results are depicted in Figure 7. By the analysis of the obtained results, we were able to verify that since TCP traffic has an AF-Gold class, it presented minimal packet losses, which allows us to attest the degree of elasticity of this type of traffic compared to the others. Also, we could confirm that applications with AF-Platinum CBR traffic are a good solution to transfer video, since results for the average delay and jitter parameters were conclusively inferior relatively to best-effort traffic under CBR1 application.

Overall, none of CBR applications had exceeded the stipulated value for CIR in their respective sessions, so it was not necessary to ”re-schedule” the packets.

Also, it was possible to conclude that best-effort traffic for CBR applications (in this case to CBR1) may not be a good combination due to large amounts of losses verified, and due the interference caused to other traffics, which cannot be accepted in order to guarantee the liability of CBR applications.

In the last section we present some conclusions and future perspectives of this work.

IV. Conclusion

The analysis of several languages to describe networks provided us a view of distinct options to the characterization of data networks. An important conclusion is that each language offers a particular structure to define a network and none of them, so far, emerged as standard to describe data networks and to provide the integration among different network tools. According to different authors, it would be important the existence of a common language to describe networks in order to promote interoperability among tools.

The first contribution of this paper is a proposal of a common language for the description of network scenarios, the NSDL. The most relevant difference of NSDL, compared to the others languages, is in the principle of separation of the network topology and objects from the several scenarios that may exist over a network.

The second contribution, the NSDL framework was validated by integration of network modeling and simulation tools. Although, these are still some initial results, we intend to improve our framework with the development of new libraries to provide the integration of other tools in different domains.

As for future perspectives, we intend to improve NSDL to support different QoS solutions, including Multi-protocol Label Switching architecture and Traffic Engineering, and also other types of networks such as Wireless Sensor Networks.

Acknowledgment

We would like to thank the M.Sc. students Fernando Rodrigues and Nélio Sousa for carrying out the case study, a valuable contribution to this work.

References

[1] *Extensible Markup Language (XML).* [Online]. http://www.w3.org/TR/xml
[2] Marques, E.M.D.; Placido R.A.S.A.; and Sampaio, P.N.M.; *Visual Network Simulator (VNS): A GUI to QoS simulation for the ns-2 simulator*, AICCSA, pp.342-349, 2009 IEEE/ACS International Conference on Computer Systems and Applications, 2009.
[3] Blake, S.; Black, D.; Carlson, M.; Davies, E.; Wang, Z.; and Weiss, W.; *An Architecture for Differentiated Services*, RFC 2475. 1998.
[4] *Object Tcl (OTcl).* [Online]. http://otcl-tclcl.sourceforge.net
[5] *Network Simulator 2 (ns-2).* [Online]. http://nsnam.isi.edu/nsnam
[6] *XSL Transformations (XSLT).* [Online]. http://www.w3.org/TR/xml
[7] Estrin, D.; Handley, M.; Heidemann, J.; McCanne, S.; Ya Xu; Yu, H.; , *Network visualization with Nam, the VINT network animator*, Computer , vol.33, no.11, pp.63-68, Nov 2000
[8] *xGraph.* [Online]. http://www.xgraph.org/