Using a OMNET++ network based simulator as test-bed for network design algorithms

JAVIER DIAZ-ESTEBARANZ, J. ANTONIO PORTILLA-FIGUERAS, SANCHO SALCEDO-SANZ, MIGUEL FARO-RIVAS, GUILLERMO ESTEVE-ASENSIO
Departamento de Teoría de la Señal y Comunicaciones
Universidad de Alcalá de Henares
Carretera Madrid-Barcelona, Km 33.600
28871, Alcalá de Henares, Madrid, SPAIN

.

Abstract: The creation of algorithms for optimal network design based on some concrete parameters (traffic, link occupation, QoS) is usually bounded to some mathematical models that does not takes into account many characteristics of the real world. This simplification causes that a algorithm that mathematically may offer good performance, when is applied to a real network, it does not behaves as it was expected. The objective of this paper is the evaluation of the suitability of using OMNeT++ to create testbeds to analyse the quality of network design algorithms before applying them to a real network.

Key-Words: network simulation, network analysis, algorithm testing, IP based networks

1 Introduction

Current network architecture and its underlying technology is constantly evolving towards more distributed environments [1] [2]. In these new frameworks, several network entities such as Internet Service Providers (ISP), Internet Access Transport Provider (IATP), and Trunk Network Internet Transport Provider (ITP) appear, requiring the distribution of the network functions among these elements and user terminals, and the definition of open interfaces between them and even between network elements of different networks. All these new situation requires new ideas for IP future generation network planning and optimization, that must be based on advanced optimization algorithms that have to be implemented in new software planning tools. These software tools, due to the high amount of different entities involved in the world of service and network provisioning, can not be limited to a specific entity, and must be able to consider the relations between entities when necessary.

There are several projects where these new algorithms mentioned above are being developed. After this development, they must be tested in an emulation environment and then in a real network model with commercial equipment to assess their validity. Finally, they should be integrated in a software planning tool which would be use both for academic and industrial works.

In this paper, we have evaluated the suitability of using OMNeT++ as testbed for the aforementioned algorithms, firstly analysing this tool in chapter 2. Then, we have developed a simulator prototype to test OMNeT++, that is described in chapter 3. Afterwards, in chapter 4, an example of application of this prototype is described and analysed, while in the last chapter the conclusions and future work of this paper will be detailed.

2 OMNeT++

In this chapter, OMNeT++ will be introduced. Initially we will demonstrate the suitability of OMNeT++ as test-bed for network design algorithms. Then, the main features of OMNeT++ will be outlined, and afterwards, the INET framework will be presented.

2.1 Suitability of OMNeT++

Before starting to work with OMNeT++, a study of several network simulators available [3] was done in order to work with the most suitable tool.

Analysing the existing information and the literature about several simulation tools, it was considered that the most suitable tools for creating the test-bed are OMNeT++ [4] y ns2 [5], and in a second place, JiST/SWANs [6] and Cnet [7]. However, these last tools were discarded because their lack of activity and its small support by the academic world.

Regarding ns2 and OMNeT++, in [8] can be found a deep comparison between them. The main difference can be found is that while ns2 is focused in the simulation of TCP/IP networks, OMNeT++ is more generic and may be adapted to very different network types. This makes this tool more suitable for using it as a testbed for network designing algorithms, because of the great variety of networks that be necessary to simulate.

This advantage, however, makes OMNeT++ a more complex tool and difficult to master it. Fortunately, there is a large compilation of documentation about OMNeT++ and a strong community supporting it, that compensates this complexity.

2.2 Main Features

OMNeT++ [9] is an public source, component-based, modular and open-architecture event discrete simulation environment.

OMNeT++ runs different operating systems on Linux, other Unix-like systems and on Windows (XP, Win2K). OMNeT++ was developed at the Technical University of Budapest, Department of Telecommunications (BME-HIT).

OMNeT++ provides a component architecture for models. Components (modules) are programmed in C++, and then are assembled into larger components and models using a high-level language named NED. Files with NED source describe the structure of the module (parameters and connections to other modules) whereas C++ files implement their behaviour (initialization, message management and store data for possible statistics). Due to its generic and flexible modular architecture, the simulation kernel and models can be embedded easily into new applications.

Its primary application area is the simulation of communication networks and because of its generic and flexible architecture, it has been successfully used in other areas like the simulation of IT systems, queuing networks, hardware architectures and business processes as well, [10].

OMNeT++ is rapidly becoming a popular simulation platform in the scientific community as well as in industrial settings. There is a commercial license for OMNeT++ and an academic public license for non-profit use. The last one is widespread in university locations.

2.3 INET Framework

The INET Framework [11] is a open-source communication networks simulation package for OMNeT++ that contains IPv4, IPv6, TCP, UDP protocol implementations, and several application models. The framework also includes an Multiprotocol Label Switching (MPLS) model with RSVP-TE and LDP signalling [12]. Among link-layer models, INET provides PPP, Ethernet and 802.11. It also enables using static routing or several routing protocol implementations.

Using INET Framework eases and speeds up the work with OMNeT++, because it provides a lot of pre-built modules that can be used in simulation only connecting them and making some small adjusts in their configuration parameters, that avoids to re-implement all the protocols and elements already defined in INET.

Taking advantage of this situation, the prototype of simulator analysed in this paper is fully based in this framework.

3 Description of the simulator

As we mentioned previously, one of the most interesting features of OMNeT++ is its modularity. Taking benefit of it, we have created the prototype of the simulator divided in several modules linked among them, all of them based in simpler modules already defined in INET framework.

The modules that take part of this simulator are the Core Network, the Client Module, the Server Module, the Channels and the Test Network, that includes the rest of them. In the following sections, we are going to describe in detail all of these modules.

3.1 Core Network

The core network module is a component that acts as a core network that routes traffic among all the elements that are connected to it.

This module is parametrizable, being possible to create in it as many connections as desired. Each connection is linked to a type of router named “EdgeRouter” that will forward the traffic received to other type of routers named “CoreRouter” [13].

Note that using this scheme we can simulate hierarchical divisions in the network as it happens in the real core IP networks. The *Edge Routers* are placed logically in the bounds of the network, connecting the access networks and depends hierarchically of two *Core Routers* which are the elements that form the trunk network

These connections among all the routers of the module, enables to route traffic between two points of the network using at least two different paths.

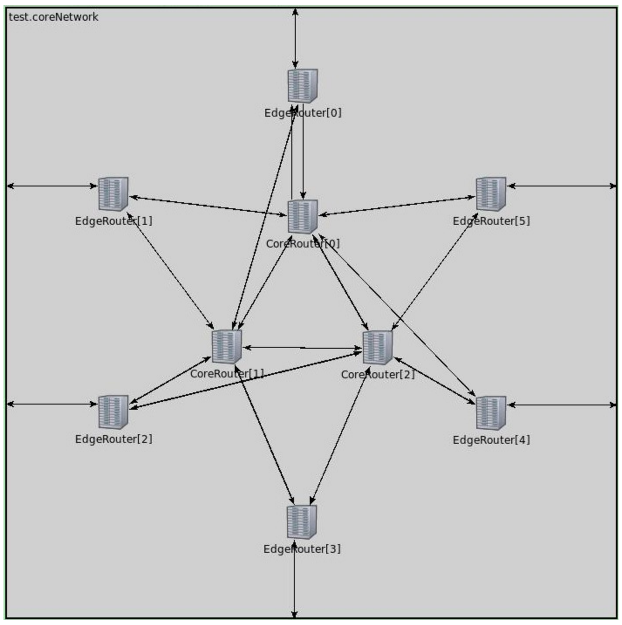

Fig.1 – Core Network

All links existing among the routers of this module are connected using Point to Point Protocol (PPP) interfaces, using a static routing mechanism. In figure 1 can be found a core network with six connections and their corresponding six edge routers and three core routers, that have been used in this simulation.

3.2 Client Module

The client module is an element that acts as a client in a network. It is composed by two types of hosts that are connected to the outside by a router.

The first type of host is a termination node for IP traffic. There is only one element of this type in the module and its objective is to act as sink of all the traffic that has as destination this client.

The other type of host is a IP traffic generator. The amount of hosts of this type is provided as an input to simulate as many types of IP traffic as desired with different values of IP traffic parameters, packet interarrival, packet size, number of transmitted packets and the time of start of the transmission.

This module has some special features that must be remarked. The first of all, all the hosts have the same IP. This is possible because inside the module there is only one host that receives all the traffic, and because outside the module all the traffic of the different hosts is mixed, resembling outside as only one host receiving and sending different types of traffic (e.g. TV, voice). Another special feature is that all the links between the router and the hosts are nearly ideal, to avoid an undesired behaviour, because these links are used only to connect all the hosts to the router, but not to analyse its behaviour.

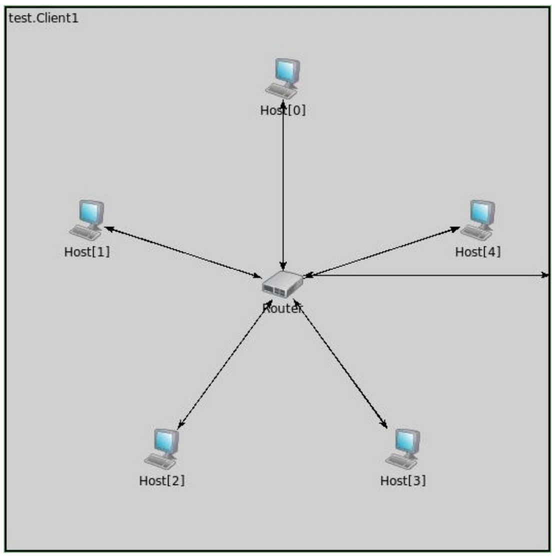

Fig.2 – Client Module

3.3 Server Module

The server module is similar to the client module, differing only that instead host there are servers in it, but these servers behave exactly as the hosts mentioned in the previous section.

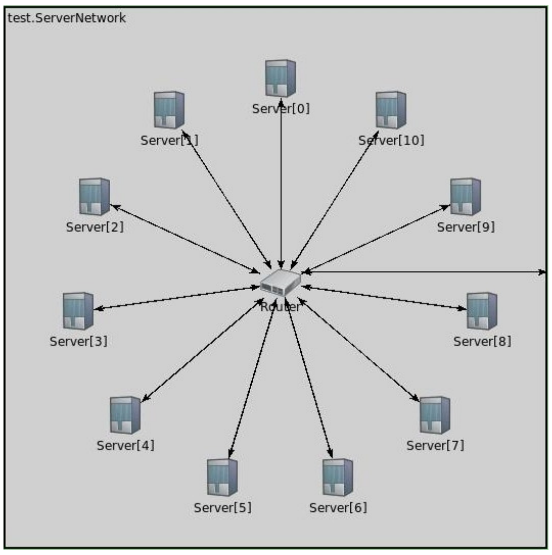

Fig.3 – Server Module

3.4 Channels

The channels are not exactly a OMNeT++ module, but can be considered as another element of the prototype. In it, all the existing links are characterized, each one with their own values of delay, data rate and bit error.

In this case, it has been defined the following channel types, that are used by inside all the modules mentioned previously or to interconnect them: "coreLink", used in the core network module to connect each module in it, "clientDownloadAccess-Link", and "clientUploadAccessLink", used in the connection between the core network and the client modules, "serverAccessLink", used in the connection between the core network and the server modules, and "idealLink", used internally in the server and client modules. This last channel type differ from the rest that its delay and bit error are zero, and its bitrate is very high (and this is the reason why is named "ideal")

3.5 Test Network

Finally, there is the test network, that using the rest of the modules, creates a complete network to simulate. This network is composed by a core network that connects five clients with a server that interchange different types of traffic between them.

In the figure 4 it is shown this network, having only a special element, the "throughputMeter" that is used to analyse the bandwidth occupation of the link where it is connected, in this case the connection of the server to the core network.

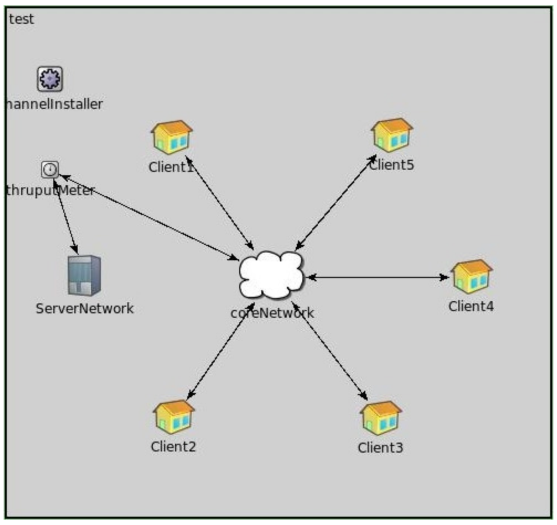

Fig.4 – Test Network

4. Applications

The simulator described previously can be used to simulate simple examples of networks that interchange IP packets.

In this section, firstly will be introduced the traffic models of the IP packets that have been used in the simulator to test OMNeT++ suitability as testbed. Then, the results of the simulation where the traffic models were used will be explained.

4.1 Traffic models used

To simulate a realistic example of a network with IP traffic, different types of traffic were needed. In this case, four types were chosen, that differ in their traffic profiles: "Real Time", "Streaming", "Guaranteed Data" and "Best Effort".

These different traffic services were defined from several parameters that were chosen from [14], and are detailed in the following table, where λ is the average packets per second, C(Ta) is the variance ot the interarrival between packets, L is the average packet length in bytes, and C(L) its variance.

Type	λ [p/s]	C(Ta)	L (Bytes)	C(L)
Real Time	21.33	1.5	384	0
Streaming	300	2	512	2
Guaranteed Data	20	2.5	512	1.5
Best Effort	20	3	512	3

Table 1 – Traffic Profiles

4.2 Simulation Results

Finally, a simulation with OMNeT++ was realised. The network used was the described in figure 4, creating in each Client four IP traffic profiles, one of each type all of them with destination to the Server. In the server, five traffic IP profiles of each type were generated, each one with destination to one of the Clients.

The purpose of the simulation was to analyse the bandwidth occupation of the link from the Server to the Core Network (as we commented before, the "throughputMeter" objective is to analyse this occupation).

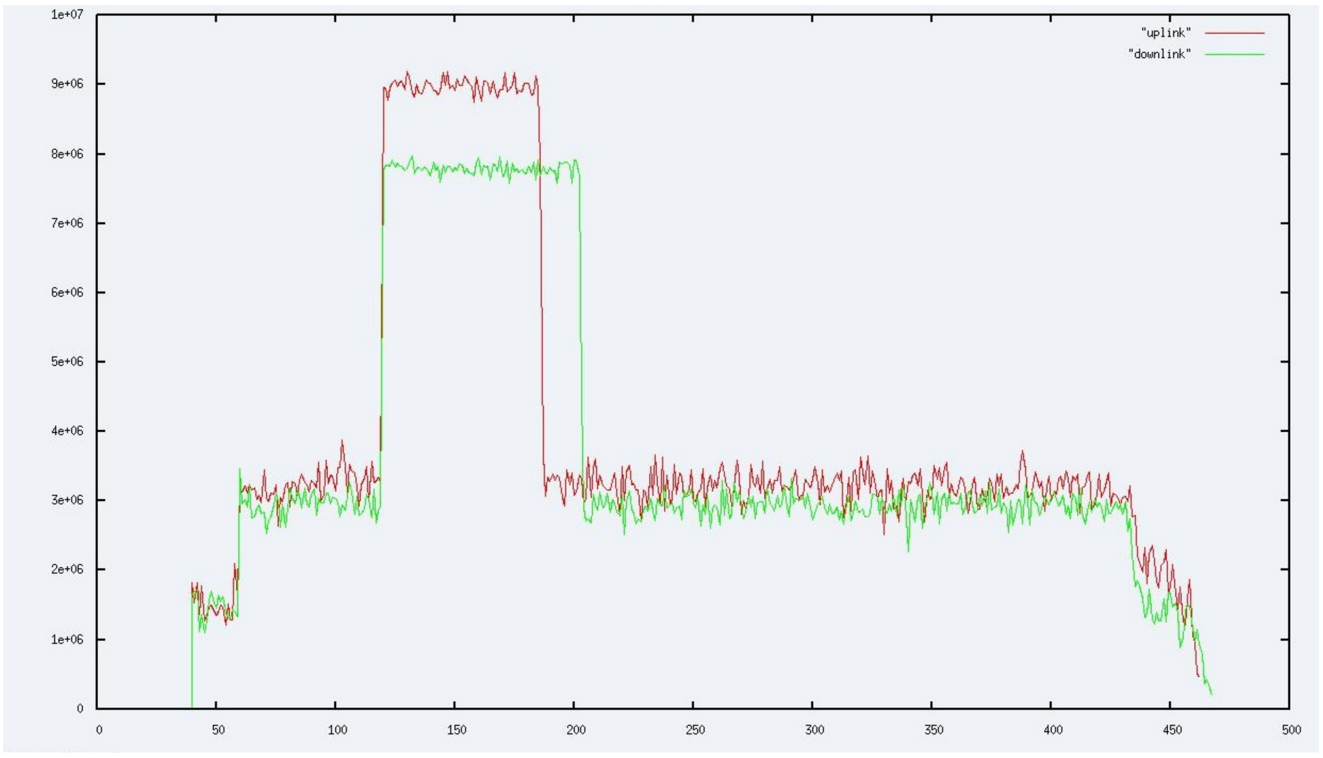

Fig.5 - Simulation Results

To check better the behaviour of the network and OMNeT++, the start of transmission of each type of traffic is different. The “Real Time” traffic starts at second 40, while the “Guaranteed Data” and “Best Effort” at second 60, and finally the “Streaming” one at second 120. So, at these instants, an increase of the bandwidth occupation should be produced.

The results of the simulation can be seen at figure 5. The red line the bandwidth (bits/second) occupation of the traffic that goes from the Server to the core Network, and the green one the opposite way in each second of the simulation. The occupation follows the pattern expected, at the time it starts the transmission of a traffic type, the occupation increases until this transmission ends.

There are two points which require further explanation. The first one is that the downlink, at second 120 does not reach the same occupation than in the uplink. The reason for this difference is that in client, the “clientDownloadAccessLink” has 3 Mbit of bandwith, while the “clientUploadAccessLink” has only 1.28 Mbit, so some IP packets are retarded, and in the downlink, the traffic that arrives while streaming traffic is transmitted, is smaller than in the uplink but as compensation its duration is some seconds bigger.

The second point is is that the downlink traffic is always slightly smaller that the uplink one. The reason for this small difference is due to all the links of the simulation (with the exception of the ideal ones) have bit error, so some IP packets can be discarded, and while the packets that go through the uplink are not discarded because it is the first non ideal link that they cross, the packets of the Client-Server way go through a lot of non-ideal links, where some packets are discarded. However, the amount of discarded packets is very small, while in the uplink the total amount of processed packets are 220000, in the downlink are received 219902 packets.

5. Conclusions and future work

This paper has presented an example to test the suitability of OMNeT++ as testbed for network design algorithms. Firstly, the network to be used in the test was described, and then this network was used to do a simulation to test the operation of OMNeT++. We have to conclude that OMNET++ is very appropriate to perform network testing.

There are several improvements that we consider as future work. Firstly, changing the routing policy used in the simulator from static to dynamic, to make it more realistic, using the OSPF routing protocol. Afterwards, another important feature to test in OMNeT++ is to enable the modification of network elements during simulation, to test for example the reaction of a network when a router or a link falls down [15]. Finally, some improvements should be done in traffic generation, to enable simulations with more complex traffic profiles towards the rest of the network.

The final objective will be to develop all the elements required to design a complete MPLS network and to test the algorithms to design it.

Acknowledgments

This work has been partially supported by Comunidad de Madrid, Universidad de Alcalá and Ministerio de Educacion of Spain, through Projects CCG06-UAH/TIC-0460, CCG07-UAH/TIC-1894 and TEC2006-07010.

References:

[1] K. Knightsotn, N. Morita, T. Towle, *NGN Architecture: Genetic Principles, Functional Architecture and Implementation,* IEEE Communications Magazine, pp 49-56, October 2005

[2] E. Gelenbe, *Users and services in intelligent network*, 1st EuroNGI Conference on Next Generation Networks, Rome, Italy 2005.

[3] Andrea Emilio Rizzoli, *A Collection of Modelling and Simulation Resources on the Internet*, www.idsia.ch/~andrea/simtools.html#network

[4] A. Varga, *Omnet ++,* IEEE Network Interactive. July 2002, Vol.16 No.4.

[5] *ns2 Home Page.* http://www.isi.edu/nsnam/ns/

[6] *JiST/SWANs Home Page.* http://jist.ece.cornell.edu/index.html

[7] *cnet network simulator Home Page.* http://www.csse.uwa.edu.au/cnet/

[9] *OMNeT++ - ns2 comparison.* http://ctieware.eng.monash.edu.au/twiki/bin/view/Simulation/OMNeTppComparison

[9] András Varga, The OMNeT++ discrete event simulation system, *Proceedings of the European Simulation Multiconference (ESM'2001).* June 6-9, 2001. Prague, Czech Republic, 2001.

[10] P. García-Díaz, S. Salcedo-Sanz, Antonio Portilla-Figueras, David Núñez-Clemente, GSMSIM: *An Educational Simulation Tool for Teaching GSM-based Mobile Communications in Laboratory Lectures*, International Journal of Electrical Engineering Education, In Press

[10] *INET Framework Documentation.* http://www.omnetpp.org/doc/INET

[12] U. Black, *MPLS and Label Switching Netwirks,* Prentice Hall, 2002.

[13] El-Sayed M. El-Alfy, *Applications of genetic algorithms to optimal multilevel design of MPLS-based networks*, Computer and Communications, Volume 30 , Issue 9, 2007.

[14] L. Rodríguez de Lope , K. Hackbarth, , *Cost Models for BitStream Access With QoS parameters,* Journal of Universe Computer Science, vol 14, nº 5, pp 653-672.

[15] A. Bahri, S. Chamberland *A global approach for designing reliable WDM networks and grooming the traffic,* Computers & Operations Research, Volume 35, Issue 12, December 2008, Pages 3822-3833.