International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 17 (2018) pp. 13136-13140

LAN-WAN-LAN end-to-end Network Simulation with NS2

Andrés Felipe Hernández Leon[1], Octavio José Salcedo Parra[1, 2], Giovanny Mauricio Tarazona Bermudez[1]

[1]*Faculty of Engineering - Universidad Distrital Francisco José de Caldas, Bogotá D.C., Colombia.*

[2]*Faculty of Engineering - Universidad Nacional de Colombia, Bogotá D.C., Colombia.*

Abstract

This article shows the main features of the NS2 open source simulator specifically for the simulation of LAN-WAN-LAN end-to-end networks. This simulator was designed for the simulation of discrete events in networks and is used in multiple institutions and universities around the world. The step-by-step procedure for performing the simulation of an end-to-end LAN-WAN-LAN network is described.

Keywords: TCP. NS2, LAN, WAN, Simulation, Host, P2P.

INTRODUCTIÓN

ISPs, as well as universities and research groups, are finding it difficult to conduct research in order to observe and improve end-to-end networks (LAN-WAN-LAN) because these networks present a complex and, above all, expensive infrastructure. In addition to the difficult access that the part corresponding to the WAN can have, for this arise simulation tools that allow to make such investigations and tests with results very close to reality.

This reduces time, effort and costs which allows a faster and more significant advance in the investigative process and implementation of new technologies. Within the world of communications networks end-to-end networks present unpublished and unexplained simulations, a situation that this article seeks to solve with the help of the NS2 simulation tool.

BACKGROUND

As part of the research process we investigated several software options for the simulation, then proceed to show a selection of the tools found:

NETWORK SIMULATOR 3 (NS3)

It is the third version of a discrete event network simulator, it is a free distribution software under the GNU GPLV2 license. The core of its operation is in C ++ language and instructions are generated in Python (Azana Hafizah Mohd Aman, 2016).

It is widely used to simulate as previously mentioned internet networks with the advantage of supporting simulations of Wi-Fi, Wimax and LTE (nsman, 2017).

It allows the emulation of the different equipment that is used in a network.

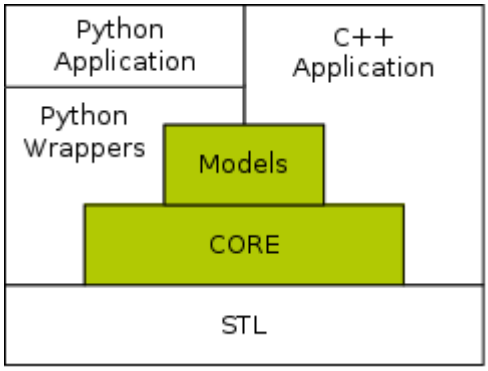

Figure 1. Structure in block diagram simulator ns3. (Source: https://www.nsnam.org/overview/key-technologies/)

QUALNET

This is a protocol simulation software strongly used in wireless simulations but also presents the libraries corresponding to wired networks. The instructions are entered using PARSEC which determines the number of discrete events (Azana Hafizah Mohd Aman, 2016).

It allows a three-dimensional graphical interface where you can plan, test and monitor activities and different networks according to the need (scalable-networks, 2017).

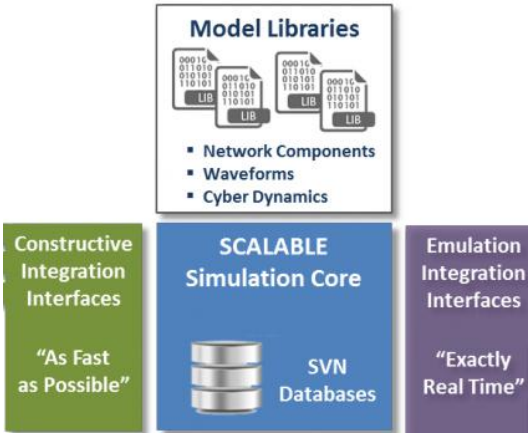

Figure 2. Structure in block diagram QUALNET simulator. (Source: http://web.scalable-networks.com/simulation-platform-architecture)

JiST

Java: "Simulation Time" is a software tool that allows the implementation of simulations of networks in Java, it is used mostly with SWANS1 at the same time..

In this software we can find different types of objects that represent network elements such as nodes, the simulation is formed by method invocations between these objects (Elias Weingartner, 2009).

In order to execute the simulations, JiST loads the classes in a custom way that dynamically rewrites the code (jist, 2017).

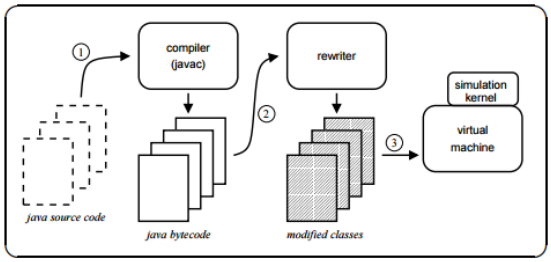

Fig. 3. Structure in JiST simulator block diagram.
(Source: http://jist.ece.cornell.edu/docs/040319-jist-user.pdf)

OMNET++

It is a module-based and object-based simulator, written in C ++ and allows real-time simulations of telecommunication networks, allows integration with databases, and supports other programming languages such as C # or Java (Shivangi Surati , 2016).

It also presents utilities in simulation of IT systems, applied queuing theory, various hardware architectures and business processes.

It also presents a series of models that present P2P protocols. It also supports Java but in this case it is necessary to use the JSimpleModule extension (omnetpp, 2017).

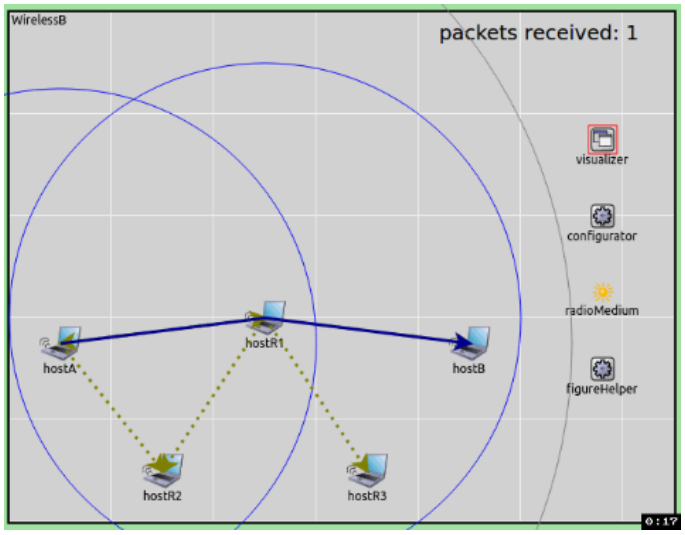

Figure 4. Simulation example in OMNET ++
(source: https://omnetpp.org/)

NETWORK SIMULATOR 2 (NS2)

Once the model has been developed we proceed to devise the way to test it in order to determine if it meets the expectations, for this stress tests are made of the networks in simulators developed for this, in this case we chose the simulator ns2 (Network Simulator 2), which is a simulator of discrete events (isi.edu, 2017) on LAN, WAN, MAN, Wi-Fi, which allows the use and implementation of different protocols, taking into account the approach of this work Research on LAN-WAN-LAN networks using TCP protocol (Shivangi Surati, 2016).

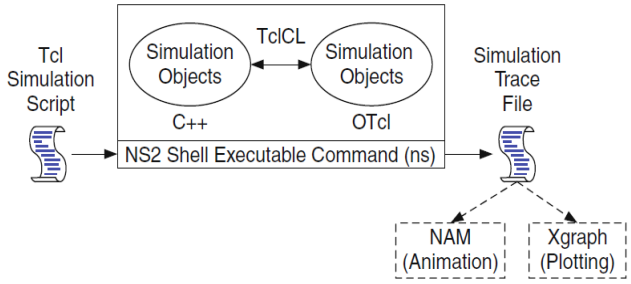

Figure 5. Structure in block diagram simulator ns2.
(Source: https://ns2projects.org/)

Simulación de la red LAN-WAN-LAN en NS2

As part of the simulation work, a network with the following characteristics will be simulated:

- 3 nodes on WAN, 1 LAN node on each end
- LAN Routers with 16MB buffer
- WAN Routers with 500MB buffer
- Bandwidth per user of 6MB / s for the LAN.
- Bandwidth per 80MB / s link for the WAN network.
- The most used ports are 7:
  - ✓ 20, 21 FTP
  - ✓ 23 TelNet
  - ✓ 25 SMTP
  - ✓ 110 POP3
  - ✓ 53 DNS
  - ✓ 80,8080 HTTP
  - ✓ 69, 161 UDP
- Package size 50Kb
- 24 concurrent LAN users
- 86 concurrent users on WAN network
- 10 concurrent LAN packages
- 90 concurrent WAN packets

Introductory part of the script

The script has a first introductory part which its implementation is mandatory in this type of simulation as in any other, now proceed to explain it.

The first step is to create the simulator object:

set ns [new Simulator]

A file is opened for "out.nam" format writing. This to send the plot of the simulation. It is recommended to create it as $ nf object.

```
set nf [open out.nam w]
```

You must send the entire path to the file, this is achieved with the command:

```
$ns namtrace-all $nf
```

The previous path is not very understandable since it is closer to the machine language, but must be created for the program to interpret it. However, there is another type of layout that is more readable for users. To do this it is necessary to add the following lines

```
set tf [open out.tr w]
$ns trace-all $tf
```

The definition of the "finish" procedure is done, this is the one that is called when the simulation is finished.

```
proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0
}
```

Customizing the script

Creation of the 31 nodes that will contain the simulation called n0, n1, ..., n31 always starts at zero and so is built the programming language:

```
set n0 [$ns node]
set n1 [$ns node]
.
.
.
set n30 [$ns node]
```

Definition of the links between each node. For example, the first one says that in the object $ ns, the nodes $ n0 and $ n7 will have a bidirectional link of 1.15Mbps, with a delay of 1.5ms and the type of queue will be DropTail, in this way the procedure is performed with All connections:

Connection between host and node on LAN:

```
$ns duplex-link $n0 $n7 1.15Mb 1.5ms DropTail
```

Connection between nodes in WAN:

```
$ns duplex-link $n7 $n8 5.75Mb 5ms DropTail
```

The size of the queue is defined in 10 packets between the nodes and the LAN (n0 and n7) and of 90 packets between the nodes of the WAN (n7 and n8), the rest will be discarde:

```
$ns queue-limit $n0 $n7 10
$ns queue-limit $n7 $n8 90
```

It is necessary to make the configuration of the agent in each node, which is in charge of generating the traffic. Taking into account that the TCP protocol by its nature handles acknowledgments (ACK) it is necessary to handle two agents one for the transmitter and one for the receiver in this way proceed to configure the transmitter as the first measure.

The configuration consists of 3 steps, Configure, add and limit the size of the packages; For the network to simulate we will call the agent "tcp0" we assign it to the node "n0" and we indicate that the maximum size of the package will be of 64Kb:

```
# Configuration of the TCP agent to the node $n0
set tcp0 [new Agent/TCP]
# Add the agent to the node $ n0
$ns attach-agent $n0 $tcp0
# maximum package size
$tcp0 set packetSize_ 64
```

The agent responsible for sending the ACKs in the simulator is called "SINK" agent, for the network to be simulated it will be called "ack0", it is assigned to the node "n7", in this case no packet sizes are specified because it is an ACK:

```
# Configuring the SINK Agent to the $n7 Node
set ack0 [new Agent/TCPSink]
# Add the agent to the node $ n7
$ns attach-agent $n7 $ack0
```

Once you have the two agents created and configured in each node, proceed to connect them together:

```
# Connection for tcp and sink agents
$ns connect $tcp0 $ack0
```

In the case of UDP services it is only necessary to configure and add the agent to the transmitter node since, as is known, UDP does not generate ACK; For the network to simulate the agent will be called "cbr12" with packets of 64Kb in size:

```
# Configuration of the UDP application (CBR)
set cbr6 [new Application/Traffic/CBR]
# Add the application to the agent
$cbr6 attach-agent $udp6
$cbr6 set type_ CBR
$cbr6 set packet_size_ 64
$cbr6 set rate_ 1mb
$cbr6 set random_false
```

Once the configuration of the nodes is done we proceed to program the start of the events, for each of the agents it is necessary to indicate the start time, as well as the end, in the

sample network all start at 0.1 seconds "Start" and stop at 4.9 seconds "stop":

```
$ns at 0.1 "$tcp0 start"
$ns at 0.1 "$tcp1 start"
.
.
.
$ns at 0.1 "$cbr6 start"
$ns at 4.9 "$tcp0 stop"
$ns at 4.9 "$tcp1 stop"
.
.
.
$ns at 4.9 "$cbr6 stop"
```

After stopping all the agents it is necessary to stop the complete simulation, this is done at 5 seconds with the function "finish" for the case of the example of this simulation:

```
$ns at 5.0 "finish"
```

Finally it is necessary that at the end of the script the command "run" is indicated so that once the software goes through the whole script and stores the values of the nodes and their main characteristics, it knows when to start:

```
$ns run
```

Run Simulation

Once the script is run onscreen the following model appears:

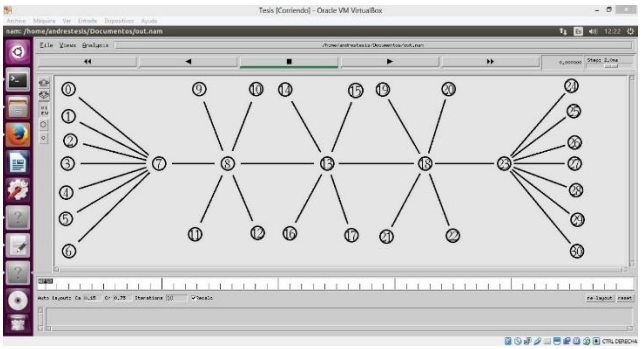

Figure 6. Scheme of the LAN-WAN-LAN network to be simulated. (Source: Authors)

As one can see in one end is a LAN with 7 services, node 0 to 5 are TCP services and node 7 corresponds to the UDP service, the same schema can be seen at the other end where node 23 Represents the other LAN that supports the 7 previously mentioned services, being from 24 to 29 the 6 TCP services and the 30 node the UDP service.

3 nodes were placed inside the WAN network, (8, 13 and 18) as indicated previously each link has 80MB of bandwidth and each node uses 20MB to include this in the simulation 4 extra nodes, each of 20MB To each WAN node.

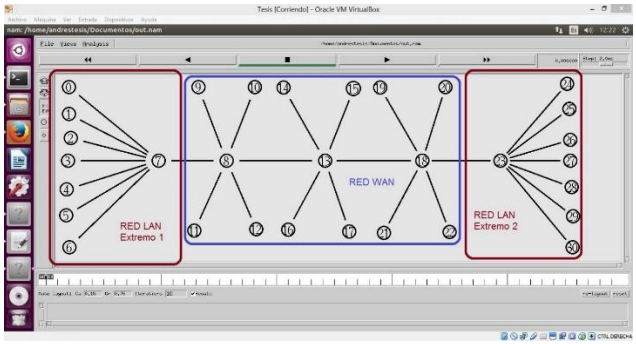

Figure 7. Identification of the LAN endpoints, and the totality of the WAN nodes of the network to be simulated. (Source: Authors)

Once the model was made and this was implemented in the scripts for the simulator were entered the data obtained from the real network explained in the previous section obtained the following simulation:

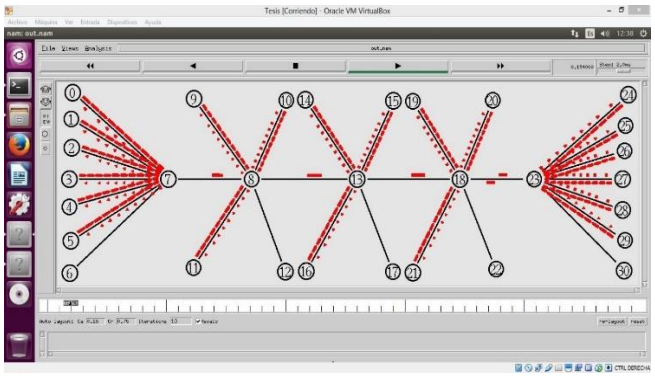

Figure 8. LAN-WAN-LAN network during the execution of the simulation. (Source: Authors)

Reports

Once the simulation is finished proceed to review the output file which as mentioned above is extension .tr the reported that is shown is the following:

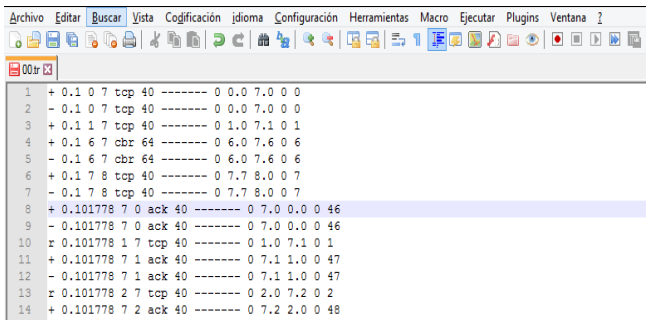

Figure 9. Example of output file ".tr" in a text editor. (Source: Authors)

Which is interpreted as follows:

Table 1. Fields that compose the output file ".tr" of the simulation

Event	*Time*	*Node Source*	*Destination Node*	*Package Type*	*Package Size (Kb)*	*Various Flags*
+	0.1	0	7	tcp	40	0 0.0 7.0 0 0
-	0.1	6	7	cbr	64	0 6.0 7.6 0 6
+	0.101778	7	0	ack	40	0 7.0 0.0 0 46

In the field called "event" you can present 4 possible values:

Table 2: Possible values that the Event field of output file ".tr" can present

r	Received
+	In queue
-	Out of the queue
d	Discarded

With this report the respective analysis of results of the network on which the simulation was performed will be carried out.

CONCLUSIONS

In general, network simulation tools help companies, ISPs or universities and their respective fields of research to analyze the results and behaviors of new networks and new protocols or methods that are used for their improvements without the need to perform real infrastructure tests Which on several occasions represent a problem because of its cost or administration.

NS2 presents a variety of options and configurations of scripts that allow to perform a very good simulation, analysis and sizing that allow later to this to realize good optimizations of networks having evaluated the performance of the simulated designs.

NS2 presents the enormous venjata of being free, aspect that do not have other simulation tools, its installation is done on Linux environments and presents high degrees of reliability what makes it one of the best tools available for these needs besides that the final report It can be exported as text files to other operating systems, making it even easier to analyze.

REFERENCES

[1] Azana Hafizah Mohd Aman, A.-H. A. (2016). Network Simulators Parametric Comprasion for Network Mobility Management. International Journal Of Future Generation Communication and Networking, 17-28.

[2] Elias Weingartner, H. v. (2009). A performance comparison of recent network simulators. IEEE ICC .

[3] isi.edu. (02 de 03 de 2017). NSNAM. Obtained from http://nsnam.sourceforge.net/wiki/index.php/User_Information#The_Network_Simulator_-_ns-2

[4] JIST. (04 de 04 de 2017). jist.ece.cornell.edu. Obtained from http://jist.ece.cornell.edu/index.html

[5] NSMAN. (04 de 04 de 2017). www.nsnam.org. Obtained from https://www.nsnam.org/overview/what-is-ns-3/

[6] Scalable-networks. (04 de 04 de 2017). http://web.scalable-networks.com. Obtained from http://web.scalable-networks.com/simulation-platform-architecture

[7] Shivangi Surati, D. C. (2016). A survey of simulators for P2P overlay networks with a case study of the P2P tree overlay using an event-driven simulator. Engineering Science and Technology, an International Journal.

[8] OMNETPP. (04 de 04 de 2017). omnetpp.org. Obtained from https://omnetpp.org/