

A Comparative Study of NS-2 and NS-3

Pramod Sharma¹ Dr. Ravindra Gupta² Dr. Satyendra Sharma³

¹Research Scholar ^{2,3} Professor

^{1,3} University Of Pacific, India.

² MIT, Bikaner, India.

Abstract--Network Simulators are used by the research community to evaluate new theories and hypotheses. It allows research questions and prototypes to be explored at relatively lesser cost and time than that required to experiment with real implementations and networks. It provides a virtual environment for an assortment of desirable features such as modelling a network based on a specific criteria and analyzing its performance under different scenarios. The primary purpose of this paper is to review NS-3 simulators, as well as find its advantages in the field of research and how it is different from NS-2.

Keywords: NS-2, NS-3, Simulation, Network Simulator,

I. INTRODUCTION

To simulate and get accurate results we require powerful simulation tools, these tools are known as Network simulators. A network simulator is a piece of software or hardware that predicts the behaviour of a network, without an actual network being present. Since a network simulator imitates the working of a computer network, it is a well suited tool to test all the newly proposed mechanisms. One can define the various types of devices used in the network and also model the network links, capacity of the links, the traffic load capacity, the type of traffic, behaviour of each device and also various attacks can be simulated to observe the behaviour in the presence of an attacker. Today there exist various simulators; some are NS-2, NS-3, OPNET, SENSE and Net Sim. These network simulators are mostly GUI driven and can be easily installed on a desktop pc or a laptop. Some network simulators require input scripts, some other require command and still other are fully GUI based where the user can click and drag network components into a simulated topology example of one such simulator is OPNET. In a simulator one can define the placement of various network components like nodes, servers, routers, gateways and links. Also various events can be defined like packets to be sent, packets to be dropped, Time intervals and various attacks. Cryptographic operations can also be defined into the protocols and thus one can observe the outcome on the packet size, delivery ratio, number of packets dropped and traffic load in the network. The results of the simulation may be in the form of graphs, animations of a trace file. A trace file can record every event that has occurred in the simulation. The graphs are used as a comparison technique between various protocols. Graphs can show the changes in the packet delivery ratio, throughput of the network, delay of the network and many other parameters used to measure network performance. Most network simulators are discrete event simulation, which means that before the execution of the protocol the user defines a list of events to be triggered at particular time periods or some conditional events, which are stored. These

stored events are then processed in order when the command to begin the simulation is given.

II. NETWORK SIMULATOR 2 (NS-2)

NS-2 is the open source network simulators. The original NS is a discrete event simulator targeted at networking research.

A. Overview

NS-2 is the second version of NS (Network Simulator). NS is originally based on REAL network simulator The first version of NS was developed in 1989 and evolved a lot over the past few years. The current NS project is supported through DARPA. The second version NS-2 is widely used in academic research and it has a lot of packages contributed by different non-benefit groups.

B. Main features

1) Programming languages:

NS-2 is implemented using a combination of oTCL (for scripts describing the network topology) and C++ (The core of the simulator). This system was chosen in the early 1990s to avoid the recompilation of C++ as it was very time consuming using the hardware available at that time, oTCL recompilation takes less time than C++. oTCL disadvantage: there is overhead introduced with large simulations. oTCL is the only available scripting language.

2) Memory Management :

NS-2 requires basic manual C++ memory management functions.

C. Packets :

A packet consists of 2 distinct regions; one for headers, and the second stores payload data. NS-2 never frees memory used to store packets until the simulation terminates, it just reuses the allocated packets repeatedly, as a result, the header region of any packet includes all headers defined as part of the used protocol even if that particular packet won't use that particular header, but just to be available when this packet allocation is reused.

1) Performance:

The total computation time required to run a simulation scales better in NS-3 than NS-2. This is due to the removal of the overhead associated with interfacing oTcl with C++, and the overhead associated with the oTcl interpreter.

2) Simulation output:

NS-2 comes with a package called NAM (Network Animator), it's a Tcl based animation system that produces a visual representation of the network described.

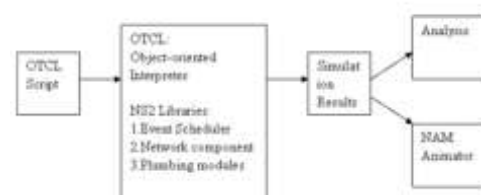


Fig. 1: Simplified User's View of NS-2

D. Recent Developments and its Future

The most recent version of NS-2 is NS-2.33 version which was released on Mar 31, 2008. Compared with the previous version, this newest version [NS-2] has integrated the most recent extension on new 802.11 models which include the Ilango Purushothaman's infrastructure mode extensions, the 802.11Ext models from a Mercedes-Benz R&D, NA and University of Karlsruhe team, and the dynamic libraries patch and multirate 802.11 library from Nicola Baldo and Federico Maguolo of the SIGNET group, University of Padova. NS is now developed in collaboration between some different researchers and institutions, including SAMAN (supported by DARPA), CONSER (through the NSF), and ICIR (formerly ACIRI). Contributions have also come from Sun Microsystems and the UCB and Carnegie Mellon Monarch projects. Generation 3 of NS (NS-3) has begun development as of July 1, 2006 and is projected to take four years.

III. NETWORK SIMULATOR 3 (NS-3)

NS-3 is also an open sourced discrete-event network simulator which targets primarily for research and educational use. NS-3 is licensed under the GNU GPLv2 license, and is available for research and development.

A. Overview

NS-3 is designed to replace the current popular NS-2. However, NS-3 is not an updated version of NS-2 since that NS-3 is a new simulator and it is not backward-compatible with NS-2.

B. Main features

The basic idea of NS-3 comes from several different network simulators including NS-2, YANS, and GTNetS. The major difference lying between NS-3 and NS-2 includes:

1) Different Software Core :

The core of NS-3 is written in C++ and with Python scripting interface (compared with OTcl in NS-2). Several advanced C++ design patterns are also used. The component diagram of NS-3 is given in Figure 2.

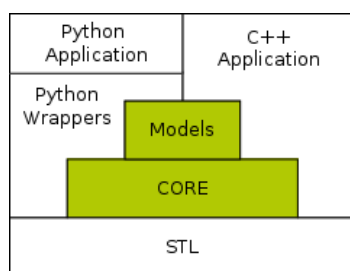


Fig. 2: Architecture of NS-3

2) Attention to Realism:

Protocol entities are designed to be closer to real computers.

3) Software Integration :

Support the incorporation of more open-source networking software and reduce the need to rewrite models for simulation.

4) Support for Virtualization:

Lightweight virtual machines are used. Figure 3 gives an example virtualization testbed of NS-3.

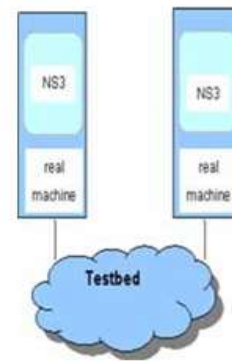


Fig. 3: Testbeds interconnect NS-3 stacks

5) Tracing Architecture:

NS-3 is developing a tracing and statistics gathering framework trying to enable customization of the output without rebuilding the simulation core.

6) Performance:

NS-3 performs better than NS-2 in terms of memory management. The aggregation system prevents unneeded parameters from being stored, and packets don't contain unused reserved header space.

7) Simulation output :

NS-3 employs a package known as PyViz, which is a python based real-time visualization package

8) Ns-3 Models:

The simulators must be updated for the rapid growth in wireless networking, including the many variants of IEEE 802.11 networking, emerging IEEE standards such as WiMax (802.16), and cellular data services (GPRS, CDMA). Table 1 summarizes the models used in the current ns-2, as well as models planned for ns-3. Many of the planned models may already exist in some form as contributed code; for a new model to be incorporated into the main branch of ns-3, it will need to be validated, conform as appropriate to the coding style, be licensed in a compatible way, and be maintained going forward. Table 1 lists the Models built so far for NS-3 project.

Layer	Existing core ns-2 capability	Planned additions for ns-3
Application Layer	Ping, vat, telnet, FTP, multicast FTP, HTTP, probabilistic and trace driven traffic generators, web cache.	Sockets-like API, peer to-peer (e.g. Bit Torrent)
Transport Layer	TCP (many variants), UDP, SCTP, XCP, TFRC, RAP, RTP Multicast: PGM, SRM, RLM, PLM	TCP stack emulation (Linux, BSD), DCCP, additional high-speed TCP variants
Network Layer	Unicast: IP, Mobile IP, generic dist. vector and link state, IPinIP, source routing, Nixvector Multicast: SRM, generic centralized, MANET:AODV, DSR, DSDV, TORA, IMEP	full IPv4 support, full IPv6 support, NAT ORP/Click Routing support: BGP, OSPF, RIP, IS-IS, PIM-SM, IGMP/MLD

Link Layer	Queueing: Diffserv, DropTail, RED, RIO, WFQ SRR, Semantic Packet Queue, REM, Priority, VQ ping, vat, telnet, FTP, multicast FTP, HTTP, Probabilistic and tracedriven traffic generators, webcache, ARP, HDLC, GAF, satellite Aloha	new 802.11 model, 802.11 variants (mesh, QoS), 802.16(WiMax), TDMA, CDMA, GPRS
Physical Layer	Two Way, Shadowing, Omni Antennas, Energy Model, Satellite Repeater	IEEE 802 physical layers, Rayleigh and Rician fading channels, GSM

Table. 1: Models Of Ns-3 Project

Through the comparison between NS-2 and NS-3, we can summarize the NS 3' s features as follows:

1. Modular, documented core
2. C++ programs and Python scripting
3. Alignment with real systems
4. Software integration
5. Virtualization and testbed integration
6. Attribute system
7. Updated models

C. Recent Developments and its Future

The NS-3 Project started around mid-2006 which is still under heavy development NS-3[2, 3, 5 7] is a discrete event network simulator written in C++ with an optional Python scripting API. It allows researchers to study Internet protocols and large-scale systems in a Controlled environment. In 2009, ns-3 releases were downloaded around 14,000 times. In 2011, ns-3 releases were downloaded 86,014 times. As of ns-3.13 release, 93 people are listed as authors. However, NS-3 is still in the process and some major challenges still remain for NS-3 to solve. Generally there are four points that are important for NS-3 to solve this problem. They are:

A Survey of Network Simulation Tools: Current Status and Future Development

- a) Hosting NS-3 code and scripts for published work
- b) Tutorials on how to do things right
- c) Flexible means to configure and record values
- d) Support for ported code should make model validation easier and more credible

Secondly, NS-3 is intended to replicate the successful mode of NS 2 in which a lot of different organizations contributed to the models and components based on the framework of NS-2.

Thirdly, NS-3 needs a lot of specialized maintainers in order to let the NS-3 have the advantages as the commercial OPNET network simulators which is documented well. Specialized maintainers can play a key part in the system. In NS-3, the active maintainers are required to respond to the user questions and bug reports, and help to test and validating the system.

All in all, NS-3 is an active open-source project and it is still under development. It has several simulator features designed to aid current Internet research. It is also a community-based development and maintenance model, which needs more people and organizations to participate to

contribute before it become good enough for the Internet research community.

IV. EXAMPLE OF SIMULATORS: NS-2 AND NS-3

NS-2: The code of NS-2 should be written in tcl. And the output fle is shown in nam window. Example:

```

Set node (s1) [$ns node]
Set node (s2) [$ns node]
Set node (r1) [$ns node]
Set node (r2) [$ns node]
Set node (s3) [$ns node]
Set node (s4) [$ns node]
$ns duplex-link $node (s1) $node (r1) 10Mb 2ms DropTail
$ns duplex-link $node (s2) $node (r1) 10Mb 3ms DropTail
$ns duplex-link $node (r1) $node (r2) 1.5Mb 20ms RED
$ns queue-limit $node (r1) $node (r2) 25
$ns queue-limit $node (r2) $node (r1) 25
$ns duplex-link $node (s3) $node (r2) 10Mb 4ms DropTail
$ns duplex-link $node (s4) $node (r2) 10Mb 5ms DropTail
$ns duplex-link-op $node (s1) $node (r1) orient right-down
$ns duplex-link-op $node (s2) $node (r1) orient right-up
$ns duplex-link-op $node (r1) $node (r2) orient right
$ns duplex-link-op $node (r1) $node (r2) queues 0
$ns duplex-link-op $node (r2) $node (r1) queues 0
$ns duplex-link-op $node (s3) $node (r2) orient left-down
$ns duplex-link-op $node (s4) $node (r2) orient left-up
Set tcp1 [$ns create-connection TCP/Reno $node (s1)
TCPSink
$node (s3) 0]
$tcp1 set window_ 15
Set tcp2 [$ns create-connection TCP/Reno $node (s2)
TCPSink
$node (s3) 1]
$tcp2 set window_ 15
Set ftp1 [$tcp1 attach-source FTP]
Set ftp2 [$tcp2 attach-source FTP]
$ns at 0.0 "$ftp1 start"
$ns at 3.0 "$ftp2 start"
$ns at 10 "finish"
$ns run
Output:

```

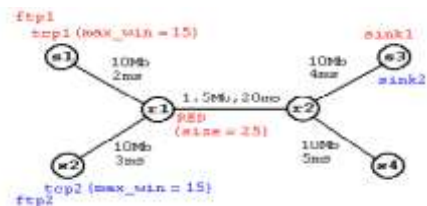


Fig. 4: Example NS-2

NS-3: the NS-3 code is written in .cc file.

Example of NS-3 is

```

Int main (int argc, char *argv [])
{
LogComponentEnable ("UdpEchoClientApplication",
LOG_LEVEL_INFO);
LogComponentEnable ("UdpEchoServerApplication",
LOG_LEVEL_INFO);
RandomVariable::UseGlobalSeed (1, 1, 2, 3, 5, 8);
NodeContainer nodes; nodes.Create (2);
PointToPointHelper pointToPoint;

```

```

pointToPoint.SetDeviceAttribute("DataRate",StringValue("
5Mbps"));
pointToPoint.SetChannelAttribute ("Delay", StringValue
("2ms"));
NetDeviceContainer devices;
devices = pointToPoint.Install (nodes);
InternetStackHelper stack; stack.Install (nodes);
Ipv4AddressHelper address;
address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer interfaces = address.Assign
(devices);
UdpEchoServerHelper echoServer (9);
ApplicationContainer serverApps = echoServer.Install
(nodes.Get(1));
serverApps.Start (Seconds (1.0));
serverApps.Stop (Seconds (10.0));
UdpEchoClientHelper echoClient (interfaces.GetAddress
(1), 9);
echoClient.SetAttribute ("MaxPackets", UintegerValue (1));
echoClient.SetAttribute ("Interval", TimeValue (Seconds
(1.)));
echoClient.SetAttribute ("PacketSize", UintegerValue
(1024));
ApplicationContainer clientApps = echoClient.Install
(nodes.Get (0));
clientApps.Start (Seconds (2.0)); clientApps.Stop (Seconds
(10.0));
Simulator::Run (); Simulator::Destroy (); return 0;}

```

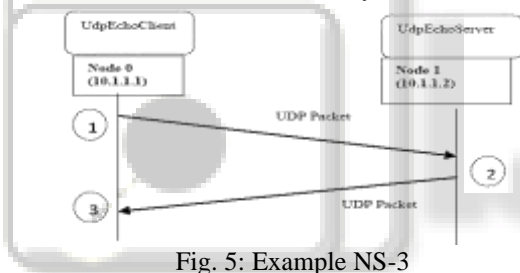


Fig. 5: Example NS-3

V. FUTURE WORK

There are so many challenges faced by simulator field. Not a single simulator satisfies current user's need. Here we have compared two simulators which are open source, where ns-3 is in development phase and needed more support from its users and researchers. Ns-3 overcomes certain problems but there is need of some improvement like, network animator tool for wireless scenarios, user friendliness and ease of use, so that a new user can easily get comfortable with it.

VI. CONCLUSION

Despite ns-2's popularity, there is a critical need for a new simulator to perform core refactoring, integration, software maintenance, and extension. Despite all these NS-3 is an active open-source development model, several simulator features designed to aid current Internet research, community-based development and maintenance model, trying to avoid some problems with ns-2, such as interoperability and coupling between models, lack of memory management, debugging of split language objects. An emerging question now-a-days is to still use Ns-2 or move to ns-3. The answer is that it depends. NS-3 does not have all of the models that NS-2 currently has, on the other hand, NS-3 does have new capabilities (such as handling

multiple interfaces on nodes correctly, use of IP addressing and more alignment with Internet protocols and designs, more detailed 802.11 models etc). Some of the Ns-2 models can usually be ported to Ns-3.

REFERENCES

- [1] E. Weingarten, H. V. Lehn, K. Wehrle, "A performance comparison of recent network simulators," IEEE International Conference on Communications, 2009.
- [2] D. M. Nicol, "Comparison of Network Simulators Revisited," Dartmouth College, <http://www.ssfnet.org/Exchange/gallery/dumbbell/dumbbellperformance-May02.pdf>, May 2002, accessed Jan 25th, 2012.
- [3] Jianli Pan, Prof. Raj Jain, Project, "A Survey of Network Simulation Tools: Current Status and Future Developments", report.
- [4] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification, IEEE Std. 802.11, 1997.
- [5] NS-3 Simulator ns-3 Tutorial Release ns-3.15 ns-3 project November 13, 2012.