ISSN:0975-9646

A study of comparison of Network Simulator -3 and Network Simulator -2

Rachna Chaudhary[#1], Shweta Sethi[*2], Rita Keshari[#3], Sakshi Goel[#4]

[#1,2,4] *B.Tech.(CS), IIMT Engineering College, Meerut*

[*3] *M.Tech(cs), ABES Engineering College*

ABSTRACT **- Network simulation is undoubtedly one of the most prevalent evaluation methodologies in the area of computer networks. While simulation is not the only tool used for data networking research, it is extremely useful because it often allows research questions and prototypes to be explored at relatively lesser cost and time than that required to experiment with real implementations and networks. The network simulators allow one to model an arbitrary computer network by specifying both the behavior of the network nodes and the communication channels. It provides a virtual environment for an assortment of desirable features such as modeling a network based on a specific criteria and analyzing its performance under different scenarios. The newly proposed network simulator NS-3 supports coupling, interoperability, good memory management, debugging of split language objects, coding in C++ and object oriented concepts, as well as supports models supported by NS-2 and most suitable for wireless networks. The primary purpose of this paper is to review this new simulator, as well as find its advantages in the field of research and how it is different from others mainly Ns2.**

KEYWORDS **- Ns-2, TCl, Simulation, Network Simulator,**

I. INTRODUCTION

Simulation is a key component of network research which requires debuggability, reproducibility, parameter exploration and no dependency on existing hardware or software [3]. Simulation is the imitation of some real thing, state of affairs, or process. It is widely used for the development of new communication architectures and network protocols. Due to growth of computer networks and complex scenarios the role of Network simulators in research field cannot be ignored. Simulators are useful tools when one wants to consider time and resources, implementation of new security solutions, performance estimation etc. Key issues in simulation include acquisition of valid source information about the relevant selection of key characteristics and behaviors, the use of simplifying approximations and assumptions within the simulation, and fidelity and validity of the simulation outcomes. To test any network there is a need of real system or tools or simulators, but installing a real network is unfeasible due to several reasons such as – high implementation cost, expensive field tests etc. Moreover experiments (especially wireless) can be hard to reproduce. A typical network simulator can provide the programmer with the abstraction of multiple threads of control and inter-thread communication. Functions and protocols are described either by finite-state machine, native programming code, or a combination of the two. A simulator typically comes with a set of predefined modules and user-friendly GUI. Some network simulators even provide extensive support for visualization and animations. There are lots of good things about simulation:

- Reproducibility
- Easier to setup, deploy, instrument
- Investigate non-existent systems
- Scalability

Most available network simulation toolkits are based on the paradigm of discrete event-based simulation [5] (DES). Here, the simulated network nodes trigger events, for instance, when a packet is sent to another node. The simulator maintains an event queue sorted by the scheduled event execution time. The simulation itself is performed by successively processing the events in the queue. Section 2 will discuss about network simulators and their roll and will compare ns-2 and ns-3. The overview of NS-3 is given in section-3 with its features and advantages. Section 4 reviews the basic models, fundamental Objects and Code architecture. Emulation support and the Tracing Model are also given in this section. Conclusion and future work is discussed next.

II. NETWORK SIMULATOR

NS or the network simulator is a discrete event network simulator. It is popular in academia for its extensibility (due to its open source model) and plentiful online documentation. Ns is popularly used in the simulation of routing and multicast protocols, among others, and is heavily used in ad-hoc networking research. Ns supports an array of popular network protocols, offering simulation results for wired and wireless networks alike. It can be also used as limited-functionality network emulator.

NS began development in 1989 as a variant of the REAL network simulator. By 1995, ns had gained support from DARPA, the VINT (Virtual Inter Network Testbed) project. at LBL, Xerox PARC, UCB, and USC/ISI.

A. Ns 2

The ns-2 simulator [1] has long been a widely used simulator for research and education on Internet and other network Systems. Network simulations for ns-2 are composed of C++ code, which is used to model the behavior of the simulation nodes, and OTcl scripts that control the simulation and specify further aspects, for instance the network topology.

Problems with Ns 2

There are a lot many problems found with NS2, such as: it has Split object model (OTcl and C++) and use of Tcl. There is a large amount of abstraction at the network layer and below leads to big discontinuities when transitioning from simulation to experiment .It has Lack of support for creating methodologically sound simulations. Furthermore, documentation is outdated. One of the biggest problem is that the Tracing system is difficult to use and there is a need to parse trace files to extract results.

B. Ns 3

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. NS-3 is a new simulator (not backwards-compatible with NS-2). It is a free, open source software project organized around research community development and maintenance. The target user community is networking researchers and educators [3]. NS-3 is not an extension of ns-2. It is a new simulator, written from scratch. The project will continue to maintain ns-2 while NS-3 is being built, and will study transition and integration mechanisms. NS-3 is a free software simulation platform which aims at network technology and whose source code is open. Researchers can use it easily to develop network technology.NS-3 contains an abundance of modules, almost relating to all the aspects of network technology.

C. Difference between Ns 2 and Ns 3

The most visible difference between NS-3 and NS-3 is the choice of scripting language. Ns-2 is scripted in OTcl and results of simulations can be visualized using the Network Animator nam[10]. It is not possible to run a simulation in ns-2 purely from C++ (i.e., as a main () program without any OTcl). Moreover, some components of ns-2 are written in C++ and others in OTcl. In NS-3, the simulator is written entirely in C++, with optional Python bindings. User code protocols and scenarios also in C++ . Simulation scripts can therefore be written in C++ or in Python. Furthermore, NS-3 generates pcap packet trace files; other utilities such as Wireshark can be used to analyze traces as well. NS-3 does not have all of the models that ns-2 currently has, but on the other hand, NS-3 does have new capabilities. Some models from ns-2 have already been ported from ns-2 to ns-3.

TABLE 1: NS2 VS. NS

	NS-2	**NS-3**
First Release	1996	2008
Based on	NS-1 & REAL simulators	NS-2, GTNets, YANS
Architecture	OTcl & C++	C++ & optional Python Scripting
Funded by	DARPA VINT SAMAN & NSF CONSER	NSF CISE & INRIA
Current Support	Volunteers, USC ISI & Sourceforge	NSF, INRIA, GT, WashU & Volunteers
Scripting	OTcl	Python
Visualization	NAM	NS-3-viz, pyviz, nam, iNSpect (all under development)
Scalability	Sequential Simulation	Distributed Simulation

D. Ns-3 Project Goals[12]

The main goals for developing Ns-3 Project are as follows :

1) To develop a preferred, open simulation environment for networking research

2) A tool aligned with the simulation needs of modern networking research

3) An open-source project that encourages community contribution, peer review, and validation of the software

III. NS-3 OVERVIEW

The brief overview of simulator is discussed in this section with its features [4]. Ns 3 I is basically a synthesis of:

- YANS(Yet Another Network Simulator),
- Ns-2,
- GTNetS simulators, and
- New software.

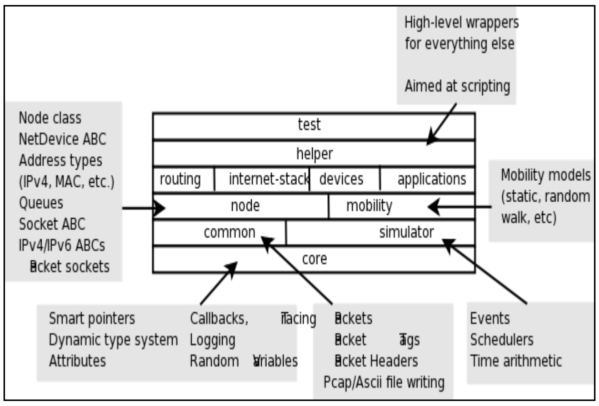

Fig 1: Software organization of NS-3[2]

NS-3 is a discrete-event network simulator in which the simulation core and models are implemented in C++. NS-3 is built as a library which may be statically or dynamically linked to a C++ main program that defines the simulation topology and starts the simulator. NS-3 also exports nearly its entire API to Python, allowing Python programs to import an "ns3" module in much the same way as in C++. The NS-3 Project started around mid 2006 which is still under heavy development. The official funded partners are: University of Washington (Tom Henderson, Craig Dowell), INRIA, Sophia Antipolis (Mathieu Lacage), and Georgia Tech University (Atlanta) George Riley (main author of GTNetS) and Raj Bhattacharjea[5].

A. NS-3 version release[6]:

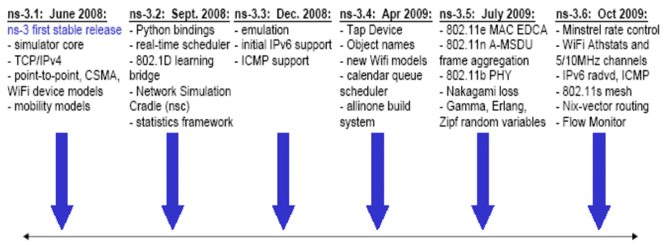

Fig 2: An open source project building a new network simulator [4]

B. Ns-3 Features[7,8,9]

The ns-2 simulator has long been a widely used simulator for research and education on Internet and other network systems. However, work is progressing on a replacement for ns-2. The features of NS3 are as under:

1. New software core: Designed to improve scalability, modularity, coding style, and documentation, the core is written in C++ but with an optional Python scripting interface (instead of OTcl). Several C++ design patterns such as smart pointers, templates, callbacks, and copy-on-write are leveraged. Object aggregation capabilities enable easier model and packet extensions.

2. Attention to realism: The Internet nodes are designed to be a more faithful representation of real computers, including the support for key interfaces such as sockets and network devices, multiple interfaces per nodes, use of IP addresses, and other similarities.

3. Software integration: Architecture to support the incorporation of more open-source networking software such as kernel protocol stacks, routing daemons, and packet trace analyzers, reducing the need to port or rewrite models and tools for simulation.

Support for virtualization: Lightweight virtual machines running over a (possibly wireless) simulation network are an attractive combination for current research; ns-3 plans to support a few modes of such operation including a native “process” environment where Posix-compliant applications can be easily ported to run in simulation space with their own private stack, and including support for tying together virtual machines of various types.

1. Testbed integration: Ns-3 will enable the testbed-based researcher to experiment with novel protocol stacks and emit/consume network packets over real device drivers or VLANs. The internal representation of packets is network-byte order to facilitate serialization.

2. Attribute system: Researchers require a means to identify and possibly reassign all values used to configure parameters in the simulator. Ns-3 provides an attribute system that integrates the handling and documentation of default and configured values.

3. Tracing architecture: Ns-3 is building a tracing and statistics gathering framework using a callback-based design that decouples trace sources from trace sinks, enabling customization of the tracing or statistics output without rebuilding the simulation core

4. Topology: For ease of use, a number of stock topology objects should be predefined. These stock objects can be instantiated by a single line of C++ code constructing the object, with configurable arguments. Stock objects should include trees, meshes, stars, and random topologies of arbitrary size.

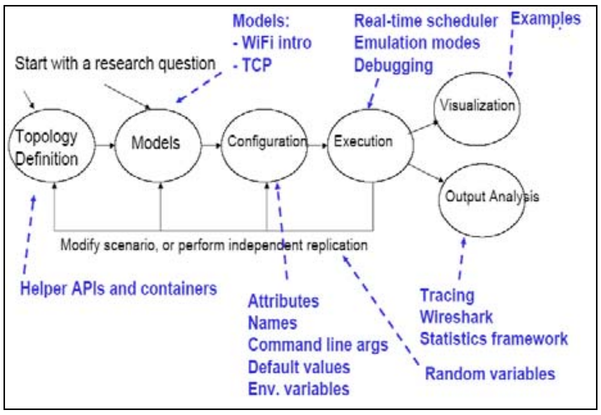

Fig 3: Overview of Ns3 features[8]

C. The Advantages of NS-3 as a simulation means:

Network simulation-3 technology has the following characteristics [6]:

a) The entirely new simulation experiment mechanism makes it have the characteristics of gaining high reliability results under a network environment of high degree of complexity.

b) Forecast function of network simulation is unmatched by any other method.

c) Wide range of use, both optimization and expansion of the existing networks and design of the new network can be used, and particularly applicable for design and optimization of Medium and large network.

d) Low initial application cost, only very few funds will be able to provide practical network design and operating environment for large number of students, furthermore, the constructed network model can continue to use so that the latter investment will still decline continuously.

e) It is flexible, vivid and visual to use simulator for teaching. To teach via NS-3, students can visually see the dealing of the network protocol and understand the effect of various environmental or other factors on the network, can also demonstrate the advantages and disadvantages of various strategies through comparison.

f) The simulation results can be reproduced and easily analysis. In this platform, the experimenter can obtain "ideal" network environment via configuring environmental parameters and can real-time track and record important information of key node so as to gain the first-hand information about network performance evaluation. Moreover, certain special circumstances can be reproduced at any time, which is difficult to do in the real network.

IV. NS 3 Core Concepts

A. NS 3 Basic Model[8]

Key objects in the simulator are Nodes, Packets, and Channels. Nodes contain Applications, "stacks", and NetDevices.

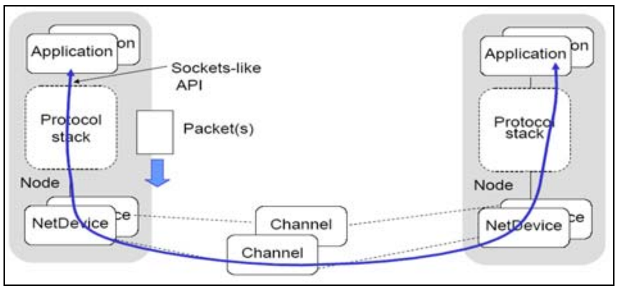

Fig 4: NS-3 Basic Architecture [8]

B. The Fundamental Objects[5]

1. Node: the motherboard of a computer with RAM, CPU, and, IO interfaces.In NS-3 the basic computing device abstraction is called the node. This abstraction is represented in C++ by the class Node. The Node class provides methods for managing the representations of computing devices in simulations.

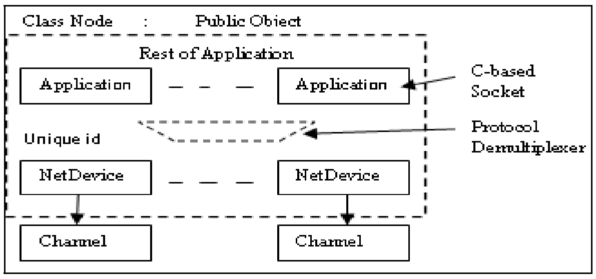

Fig 5: High-level Node Architecture [7]

2. Application: a packet generator and consumer which can run on a Node and talk to a set of network stacks. In NS-3 the basic abstraction for a user program that generates some activity to be simulated is the application. This abstraction is represented in C++ by the class Application. The Application class provides methods for managing the representations of our version of user-level applications in simulations. For example, some specializations of class Application called "UdpEchoClientApplication "& "UdpEchoServer Application". These applications compose a client/server application set used to generate and echo simulated network packets.

3. NetDevice: a network card which can be plugged in an IO interface of a Node. In NS-3 the net device abstraction covers both the software driver and the simulated hardware. A net device is "installed" in a Node in order to enable the Node to communicate with other Nodes in the simulation via Channels. Just as in a real computer, a Node may be connected to more than one Channel via multiple NetDevices. The net device abstraction is represented in C++ by the class NetDevice. The NetDevice class provides methods for managing connections to Node and Channel objects. NetDevices are strongly bound to Channels of a matching type.

4. Channel: a physical connector between a set of NetDevice Objects. In the simulated world of NS-3, one connects a Node to an object representing a communication channel. Here the basic communication subnetwork abstraction is called the channel and is represented in C++ by the class Channel. The Channel class provides methods for managing communication subnetwork objects and connecting nodes to them.

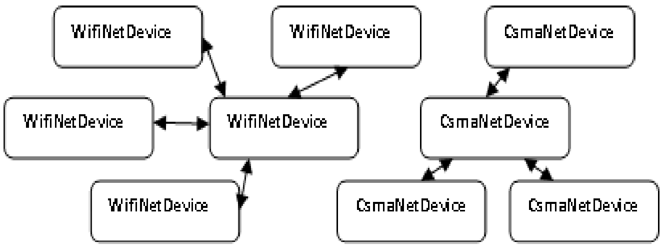

Fig 6 NetDevices connected to a channel [9]

5. Packet: each network packet contains a byte buffer, a list of tags, and metadata

– buffer: bit-by-bit (serialized) representation of headers and trailers

– tags: set of arbitrary, user-provided data structures (e.g., per-packet cross-layer messages, or flow identifiers)

–metadata: describes types of headers and trailers that have been serialized.

6. Socket: the interface between an application and a network stack. Ns-3 provides two types of sockets APIs, and it is important to understand the differences between them. The first is a native ns-3 API, while the second uses the services of the native API to provide a POSIX-like API as part of an overall application process.

7. Typical containers and helpers: There are different container and helper classes in ns-3. NodeContainer, NetDeviceContainer, Ipv4AddressContainer are some of the container classes and InternetStackHelper, WifiHelper, MobilityHelper, OlsrHelper are some of the helper classes in ns3.

C. Ns-3 Code Architecture

NS-3 code is divided into different parts. Here we start with topology definition and then we define models to use, after that we configure over model by giving them some addresses and setting other parameters, and finally we executed the code. The output which is generated in trace format will be analyzed with some tools like Wireshark. The procedure of code creation is shown in the fig. :

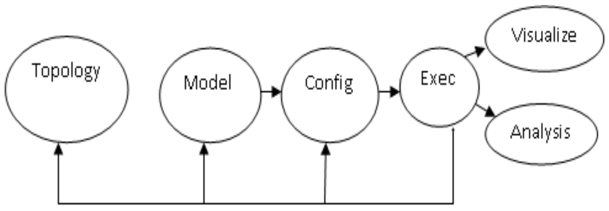

Fig 7 The NS-3 source code

D. The WAF Build System[4]

The build system used on the ns-3 project is Waf. It is one of the new generation of Python-based build systems.Ns-3 uses the waf build system i.e., instead of "./configure; make" ," ./waf" is used .

Waf is a Python-based framework for configuring, compiling and installing applications. It is a replacement for other tools such as Autotools, Scons, CMake or Ant.

Example[6]:

```
configure -> ./waf -d [optimized|debug] configure

make -> ./waf

make test -> ./waf check (run unit tests)
```

Programs can run through a special waf shell; e.g.

```
./waf --run simple-point-to-point

./waf—shell
```

E. Emulation support in Ns 3[5]

Ns-3 has been designed for integration into testbed and virtual machine environments. This need has been addressed by providing two kinds of net devices. The first kind, which is called an Emu NetDevice, allows ns-3 simulations to send data on a "real" network. The second kind, called a Tap NetDevice allows a "real" host to participate in an ns-3 simulation as if it were one of the simulated nodes. An ns-3 simulation may be constructed with any combination of simulated, Emu, or Tap devices.

Real-Time Scheduler[5]

Ns-3 has been designed for integration into testbed and virtual machine environments. To integrate with real network stacks and emit/consume packets, a real-time scheduler is needed to try to lock the simulation clock with the hardware clock. A new component is the Real-time scheduler. The purpose of the Real-time scheduler is to cause the progression of the simulation clock to occur synchronously with respect to some external time base. Without the presence of an external time base (wall clock), simulation time jumps instantly from one simulated time to the next.

F. 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 2 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 3 lists the Models built so far for ns3 project.

TABLE 2: MODELS PLANNED FOR NS-3 PROJECT.[12]

	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, webcache	Sockets-like API (to allow porting of existing applications to ns environment), peer-to-peer (e.g. BitTorrent)
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 XORP/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 trace-driven traffic generators, webcache,	new 802.11 model, 802.11 variants (mesh, QoS), 802.16 (WiMax), TDMA, CDMA, GPRS

	ARP, HDLC, GAF, satellite Aloha	
Physical Layer	TwoWay, Shadowing, OmniAntennas, EnergyModel, Satellite Repeater	IEEE 802 physical layers, Rayleigh and Rician fading channels, GSM

G. Tracing Model in NS 3[5,8,10]

The ns-3 tracing system is built on the concepts of independent tracing sources and tracing sinks; along with a uniform mechanism for connecting sources to sinks. The Ns3 Simulator provides a set of pre-configured trace sources. Users may edit the core to add their own trace sources and sinks. Users provide trace sinks and attach to the trace source. Multiple trace sources can connect to a trace sink.

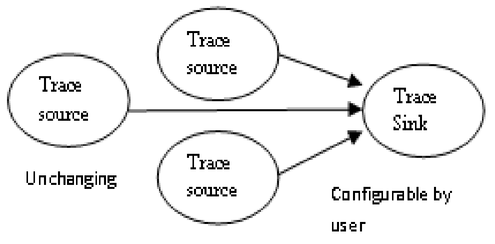

Fig 8:The NS-3 tracing model[10]

Trace sources are entities that can signal events that happen in a simulation and provide access to interesting underlying data. For example, a trace source could indicate when a packet is received by a net device and provide access to the packet contents for interested trace sinks. Trace sources are not useful by themselves; they must be connected to other pieces of code that actually do something useful with the information provided by the source. The entities that consume trace information are called trace sinks. Trace sources are generators of events and trace sinks are consumers.

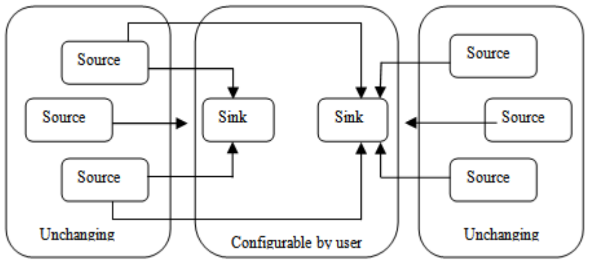

Figure 9: The Configurable Trace Sinks [4,5]

Multiple levels of tracing [5]

NS-3 provides multiple tracing levels- High, Mid and Low

• High-level: use a helper to hook a predefined trace sink to a trace source and generate simple tracing output (ascii, pcap). Use built-in trace sources and sinks and hook a trace file to them

• Mid-level: hook a special trace sink to an existing trace source to generate ad hoc tracing. Customize trace source/sink behavior using the tracing namespace.

• Low-level: add a new trace source and connect it to a special trace sink. Add trace sources to the tracing namespace or expose trace source explicitly.

V. Future Work

There are so many challenges faced by simulator field. Not a single simulator satisfies current user's need. There are so many researches, comparisons and surveys are needed before designing any simulator. 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 as well as good tutorial and wider community support, so that a naive user can easily get comfortable with it. And most important that before release the stable version it should be well tested, so that it is free of any bugs and errors.

VI. Conclusion

There are many simulators like Opnet, QualNet, but because of terms of use and high cost for industrial partners or publicly-funded research these cannot get education licenses. Despite ns-2's popularity, there is a critical need for a new project to perform core refactoring, integration, software maintenance, and extension of the simulator.

Despite all these NS-3 is an active open-source project and 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 [9]. 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.] S. M. Metev and V. P. Veiko, *Laser Assisted Microtechnology*, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

[2.] J. Breckling, Ed., *The Analysis of Directional Time Series: Applications to Wind Speed and Direction*, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

[3.] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, "A novel ultrathin elevated channel low-temperature poly-Si TFT," *IEEE Electron Device Lett.*, vol. 20, pp. 569–571, Nov. 1999.

[4.] M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, "High resolution fiber distributed measurements with coherent OFDR," in *Proc. ECOC'00*, 2000, paper 11.3.4, p. 109.

[5.] R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, "High-speed digital-to-RF converter," U.S. Patent 5 668 842, Sept. 16, 1997.

[6.] (2002) The IEEE website. [Online]. Available: http://www.ieee.org/

[7.] M. Shell. (2002) IEEEtran homepage on CTAN. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/supported/IEEEtran/

[8.] *FLEXChip Signal Processor (MC68175/D)*, Motorola, 1996.

[9.] "PDCA12-70 data sheet," Opto Speed SA, Mezzovico, Switzerland.

[10.] A. Karnik, "Performance of TCP congestion control with rate feedback: TCP/ABR and rate adaptive TCP/IP," M. Eng. thesis, Indian Institute of Science, Bangalore, India, Jan. 1999.

[11.] J. Padhye, V. Firoiu, and D. Towsley, "A stochastic model of TCP Reno congestion avoidance and control," Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.

[12.] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1997.