UNIVERSITY OF VAASA
FACULTY OF TECHNOLOGY
TELECOMMUNICATION ENGINEERING

Waleed Ahmed

SIMULATION AND EVALUATION OF WIRED AND WIRELESS NETWORKS WITH NS2, NS3 AND OMNET++

Master's thesis for the degree of Master of Science in Technology submitted for inspection, Vaasa, 23th of March, 2015.

Supervisor Mohammed Elmusrati
Instructor Tobias Glocker

ACKNOWLEDGEMENTS

First of all, praises to Allah Almighty whose blessings are countless to me in all shadows of my life.

Secondly, I would like to express my deepest gratitude towards my master thesis supervisor, Mohammed Elmusrati for his knowledgeable guidance during the whole course of my studies. Nevertheless, special thanks to my instructor Tobias Glocker for guiding and motivating me for completing the master thesis with high quality and productivity.

Then, I would like to convey special thanks to my parents, spouse Farrah Farooq and daughter Wafaa Fatima Ahmed who supported me all the days and nights with all the necessities and time I needed to complete my master thesis. Special thanks to my brothers Sheraz Ahmed, Fraz Ahmed, Jawad Ahmed and sister Mishal Waqas for their uncountable and admirable support during this time.

Furthermore, I am really grateful to University of Vaasa and Government of Finland for giving me an opportunity to study in one of the best University in a region for free and allowing me to complete my studies in extended period of time.

At the end, I would like to thanks my university teachers, classmates, all the friends in Finland, Pakistan and around the globe for giving me respectable support and guidance throughout my study period and filling my life with pleasant experiences and delights.

Waleed Ahmed

Vaasa, Finland, 12 March 2014

TABLE OF CONTENTS **PAGE**

LIST OF FIGURES **PAGE**

LIST OF TABLES **PAGE**

ABBREVIATIONS

ACK	Acknowledgement
AODV	Ad-hoc On-Demand Distance Vector
AOMDV	Ad-hoc On-demand Multipath Distance Vector
AQM	Active Queue Management
ATM	Asynchronous Transfer Mode
BGP	Border Gateway Protocol
CDMA	Code Division Multiple Access
CIMS	Columbia IP Micro-Mobility Suite
CISE	Computer and Information Science and Engineering
CPU	Central Processing Unit
CRCN	Cognitive Radio Cognitive Network
CSMA	Carrier Sense Multiple Access
CWND	Congestion Window
DDCP	Datagram Congestion Control Protocol
DES	Discrete-event Simulation
DHCP	Dynamic Host Configuration Protocol
DLSR	Dynamic Link State Routing Protocol
DMCR	Distributed Multiple Criteria Routing
DSDV	Destination-Sequenced Distance Vector
DSR	Dynamic Source Routing
DYMO	Dynamic Manet On-demand
FCC	Federal Communication Commission
FIFO	First In First Out
EURANE	Enhanced UMTS radio access network
GloMoSim	Global Mobile Information System
GPRS	General Packet Radio Service
GSM	Global System for Mobile

GT	Grid Topology
GTNetS	Georgia Tech Network Simulator
HWMP	Hybrid Wireless Mesh Protocol
IEEE	Institute of Electrical and Electronics Engineers
IGMP	Internet Group Management Protocol
ISI	Inter Symbol Interference
IP	Internet Protocol
IR-UWB	Impulse-Radio Ultra-Wide Band
IS-IS	Intermediate System to Intermediate System
ITS	Intelligent Transportation Systems
IVC	Inter-vehicle Communications
JiST	Java in Simulation Time
J-Sim	JavaSim
LAN	Local Area Network
LIFO	Last in First Out
LT	Linear Topology
LTI	Linear Time Invariant
LTV	Linear Time Variant
MAC	Media Access Control Protocol
MANET	Mobile Ad-hoc Networks
Mbps	Megabits per Seconds
MPLS	Multi Protocol Label Switching
NAT	Network Address Translation
NCI	Network Interface Card
NS	Network Simulators
NSF	National Science Foundation
OFDM	Orthogonal Frequency Division Multiplexing
OLSR	Optimized Link State Routing Protocol
OMNET++	Objective Modular Network Test-bed in C++

OSFP	Open Shortest Patch First
PI	Proportional Integral
PIM-SM	Protocol Independent Multicast Sparse Mode
PPP	Point to Point Protocol
QoS	Quality of Services
RCDS	Reactive Connected Dominating Set
RED	Random Early Detection
REM	Random Exponential Marking
RIP	Routing Information Protocol
RSTP	Rapid Spanning Tree Protocol
RSU	Road Side Units
RTO	Retransmission Timeout
RTT	Round Trip Time
RWP	Random Way Point
SFQ	Stochastic Fair Queuing
STP	Spanning Tree Protocol
SWANS	Scalable Wireless Ad-hoc Network Simulator
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TORA	Temporally-Ordered Routing Algorithm
TOSSIM	TinyOS Simulator
UDP	User Datagram Protocol
V2I	Vehicle-to-infrastructure Communications
V2V	Vehicle-to-Vehicle Communication
VANET	Vehicular Ad-hoc Area Network
VINT	Virtual InterNetwork Testbed
WAN	Wireless Ad-hoc Networks
YANS	Yet Another Network Simulator
ZRP	Zone Routing Protocol

UNIVERSITY OF VAASA
Faculty of Technology

Author:	Waleed Ahmed
Topic of the Thesis:	Simulation and Evaluation of Wired and Wireless Networks with NS2, NS3 and OMNET++
Supervisor:	Mohammed Elmusrati
Instructor:	Tobias Glocker
Degree:	Master of Science in Technology
Department:	Department of Computer Science
Degree Programme:	Degree Programme in Information Technology
Major of Subject:	Telecommunication Engineering
Year of Entering the University:	2009
Year of Completing the Thesis:	2015 **Pages:** 131

ABSTRACT

Communication systems are emerging rapidly with the revolutionary growth in terms of networking protocols, wired and wireless technologies, user applications and other IEEE standards. Numbers of industrial as well as academic organizations around the globe are bringing in light new innovations and ideas in the field of communication systems. These innovations and ideas require intense evaluation at initial phases of development with the use of real systems in place. Usually the real systems are expensive and not affordable for the evaluation. In this case, network simulators provide a complete cost-effective testbed for the simulation and evaluation of the underlined innovations and ideas. In past, numerous studies were conducted for the performance evaluation of network simulators based on CPU and memory utilization. However, performance evaluation based on other metrics such as congestion window, throughput, delay, packet delivery ratio and packet loss ratio was not conducted intensively. In this thesis, network simulators such as NS2, NS3 and OMNET++ will be evaluated and compared for wired and wireless networks based on congestion window, throughput, delay, packet delivery and packet loss ratio. In the theoretical part, information will be provided about the wired and wireless networks and mathematical interpretation of various components used for these networks. Furthermore, technical details about the network simulators will be presented including architectural design, programming languages and platform libraries. Advantages and disadvantages of these network simulators will also be highlighted. In the last part, the details about the experiments and analysis conducted for wired and wireless networks will be provided. At the end, findings will be concluded and future prospects of the study will be advised.

KEYWORDS: communication systems, networking protocols, wired and wireless networks, network simulators, performance evaluation

1. INTRODUCTION

Simulation is an advance concept used to model and analyze various scenarios in real world. It applies to different engineering, mathematics, science and other application areas for achieving different purposes. With the use of simulation, one can easily model hypothetical and a real life object, simulate and analyze the results to study the behavior of the system based on different parameters in various contexts (Pan & Jain 2008). Computer simulations are used in various areas for modeling and analysis of natural systems where the real modeling becomes really expensive and even hard to implement.

With the advance revolution in computer technologies, computer networks became a predominant area for the researchers and industrial specialists considering study and experiment objectives. In this case, the complete understanding of the computer networks is really important especially for network researcher so that they can deeply evaluate different technologies, component and protocol used in computer networks and also work to enhance the technologies more appropriately for end users. Considering all this, network simulation became an important tool to understand the systems in depth. Network simulations are used during the development of new communication architectures and protocols (Weingartner, Lehn, Wehrle 2009: 1287-1291).

The implementation of new networking architectures and protocols is a progressive method and requires continues changes and evaluation during the whole process. It also requires a proof of concept prototypes for experiments and understanding of the whole system. Furthermore, large real networks are usually used for evaluation which requires various resources with high cost. In this case, network simulators (NS) provide a test bed for simulating and evaluating new architectures and protocols. It helps researchers to understand end to end behavior of the underline technologies, also changes in the system definition, attributes and prototypes, even rewriting of the whole system is really trivial.

Network simulators are emerging everyday and mostly provides infrastructure for all kind of network technologies including the current running IEEE standards and also the future prospect technologies. For example, it covers application of simulation technology into network area such as network traffic simulation which is relatively a new technology. MANET and VANET are other emerging network applications which can be simulated in network simulators using different ad hoc networking protocols including DSDV, TORA, DSR and AODV. As Network simulators mainly work for computer networks, therefore understanding of computer networks in depth is very important and must be considered as a prerequisite for network simulation.

Network simulators provide mechanism for modeling wired and wireless networks using different kind of network nodes, routers switches, bridges, routing protocols, channel models, packet types, queue types, channel propagation models, signal generators, sink devices and network and physical layer protocols including TCP, UDP and MAC. Different network simulators uses different programming languages for network design, protocol implementation and flow control handling within the network. Mostly network simulators come with editor for development and animator tool for witnessing the graphical view of network simulation. For statistical data collection and analysis, network simulators provide various tracing methods which produce data. Researcher can analyze the data by plotting into any graph application and compare the results.

Different types of network simulators are available for researchers and industrialists. Open source network simulators are mainly used for academic work where students and researchers execute different simulations using free and open source tools. On the other side, commercial tools are expensive and used by companies for commercial purposes. The choice of network simulators depends on various factors, for example performance and memory consumption of network simulators for large scale networks. Other factors include scalability, reliability and troubleshooting of different network models and protocols,

programming languages, available support for different network components and documentation.

The main objective of the thesis is to simulate and evaluate wired and wireless networks in different network simulators. For experiment part, NS2, NS3 and OMNET++ network simulators will be used for modeling different models, and computing throughput, end to end delay, packet loss, jitter and TCP congestion control. At the end, results will be combined and compared. This study will be an extension to existing studies where the results of existing studies will be reviewed and outcomes will be combined to generate and evaluate new models for NS2, NS3 and OMNET++.

In the thesis, following questions will be answered:

- What are the benefits and drawbacks of NS2, NS3 and OMNET++ network simulators?
- What models can be simulated and how can be simulated?
- What programming languages are used in network simulators?
- How accurate are the results in comparison with the theory?
- What are the benefits and drawbacks in terms of usability?
- What are the suitable simulation parameters and how these affect the simulation results?

2. THEORY AND BACKGROUND INFORMATION

In communication and computer network research, network simulation is widely used to observe the behavior of a small scale to large scale networks. The idea of a network simulation came from computer simulation which has been developed hand in hand with the rapid growth of the computers since the early days of digital world. The first large scale computer simulation deployment was Manhattan project following a World War II to model the process of nuclear detonation (Winsberg 2010). With the rapid growth in computer simulation methodologies, it has become indispensable in a wide variety of scientific disciplines including astrophysics, high-energy physics, climate science, engineering, ecology and economics. (Winsberg 2010). **Figure 1** illustrates design of wired and wireless networks.

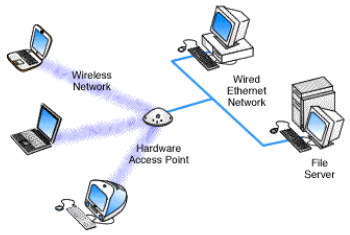

Figure 1. *Wired and Wireless Networks.*

In the following section, various studies will be reviewed conducted to evaluate different available network simulators with respect to performance, memory consumption, scalability, reliability and accuracy compared to theory.

2.1. Literature Review

Most of the Network Simulators are based on discrete-event simulation for evaluating protocols and architecture of wired and wireless networks. Due to differences in various Network Simulators, selecting appropriate network simulators is a crucial task.

According to the authors (Fernandes & Ferreira 2012), the existing network simulators differ with each other when it comes to realism, accuracy, performance and scalability of the simulation. Some of the network simulators perform accurately for small scale networks while performance in large scale network simulation decreases potentially. Based on the previous studies about performance comparisons of different network simulators, authors performed his experiments in NS3 for scalable VANET and evaluate the possible enhancements to the physical layer and mobility models of NS3. For the solution, authors proposed a spatial indexing data structure which helps in efficiently storing and updating nodes and finding a neighbor within a given range from source node.

The authors (Weingartner, Lehn & Wehrle 2009) evaluated five different Network Simulators namely NS2, NS3, OMNET++, SimPy and JiST/SWANS in his paper. Authors mainly emphasized on recent developments in the field of network simulation and conducted performance analysis by implementing identical simulation setup from scratch in all five simulators. The results of the simulation show notable difference among all the simulators in run time performance and memory uses. In the experimental part, authors implemented a sample network topology with 16 nodes and analyzed the outcome for end-to-end packet loss, computational time and memory consumption based on network size and drop probability. At the end authors concluded that NS3 demonstrated best overall performance while JiST proved to be a fastest simulator among all. However, NS3 development is still in early stage and only few of the simulation models from NS2 are ported and available. OMNET++ is also another good choice when it comes to graphical user interface and scalability of the networks. Authors gave an impression that out of five simulators, three including NS3, JiST and OMNET++ would be a smart choice when scalability is a main concern.

Authors (Gupta, Mangesh, Ghonge, Parag, Thakare & Jawandhiya 2013) presented a comprehensive survey on comparison of different Open Source Network Simulators namely NS2, NS3, OMNET++ and JiST. In the study, authors highlighted the key

components and features of Network Simulators and provided a detailed overview of advantages and disadvantages. The purpose of the study was to provide a clear picture of the Network Simulators to the researchers in order to help them in selection of an appropriate Network Simulator for their research. According to the authors, the use of network simulators is inexpensive, helps in finding bugs in advance and provides generality over analytic and numerical techniques. On the other hand, there is no guarantee that the model reflects the reality and for large scale networks, one has to simulate lots of resources which can affect the performance of the simulation, also there is a possibility of statistical uncertainty in results. At the end, authors concluded that NS2 is the best option among all as it covers almost all models. NS3, OMNET++ and JiST are still in development phase. However, NS2 lacks of a GUI which other three provide for end users. For large scale networks, OMNET++ is more effective than others.

Network Simulators have essential utilization in the analysis of wireless sensor networks (WSN). Choosing an appropriate Network Simulator for wireless sensor networks analysis is a challenging task for researchers. The authors (Khan, Hasbullah &Nazir 2014) explored different Network Simulators including NS2, NS3, OMNET++ Castalia, TOSSIM and J-sim for wireless sensor networks and examined them together based on parameters, CPU usage, memory usage, computational time period and scalability of a wireless sensor network. For the experiment part, the authors assessed the execution of the state craftsmanship test system using a LEACH routing protocol and results revealed that NS-2, NS3 and OMNET++ Castalia are more suitable for conveying out broad ascend mesh simulations. In all simulators, NS3 proved to be a fastest simulator in computation time and CPU utilizations.

In communication and computer network research, wireless networks are the key areas of interests for researcher where examining a behavior of the systems in real world and evaluating new protocols are challenging tasks for everyone. In this case, network simulators help researchers to analyze wireless networks in a virtual world using a testbed

having all the required ingredients of wireless networks to cook the experiment in an efficient and cost effective way. The research conducted by authors (Khan, Bilal & Othman. 2012) is based on above phenomena where they rehearsed experiments for wireless networks using Mobile Ad hoc Networks (MANET) protocols. For the experiment, the authors selected NS2, NS3, OMNET++ and GloMoSim network simulators and compared them on the basis of CPU utilization, memory usage, computational time and scalability of the networks. In the study, authors used Ad hoc on demand distance vector (AODV) routing algorithm due to pre-availability in selected Network Simulators. The results shows that NS3 uses the lowest memory and NS2 uses the most memory compared to other two simulators which proved that NS3 is most efficient in memory usage among others. In CPU utilization, NS2 and NS3 approved to be more effective than OMNET++ and GloMoSim, however with the parallel execution of other application, NS2 and NS3 CPU utilization decreased to certain level. In comparison for a large scale network, NS3, OMNET++ and GloMoSim are effective compared to NS2. In overall comparison, NS3 demonstrates the best performance among all despite of being quite new in race.

In addition to performance analysis based on computation time and memory usage, another objective of network simulators is to meet the results of a simulation with theory metrics. In this case, the comparison of different network simulators based on throughput, packet loss, jitter and end to end delay is indispensable and the expectations are always to reach near the theoretical values. To accomplish the goal, anthers (Ikeda, Kulla, Barolli & M Takizawa) compared throughput results of wireless ad hoc network simulations using NS2 and NS3.

The simulation experiment was conducted for two different models including Linear Topology (LT) and Grid Topology (GT) where authors used TwoRayGround radio model and OLSR protocol for performance evaluation. The results shoes that NS3 throughput values are more close to theoretical values than NS2, also the memory consumption in NS3 is much better compared to NS2.

2.2. Wired Networks

Wired networks are collections of physically connected nodes using wires for exchanging information to and from different hosts within a network. Wired networks are also known as Ethernet networks which is a known type of local area network (LAN) (Pandey & tyagi 2013). The nodes in wired networks can be any computers, printers or other devices connected through Ethernet cables. Ethernet is proven to be the fastest wired networks protocol which provides connection speeds of 10 megabits per seconds (Mbps) to 100 megabits per seconds or higher (Pandey et al. 2013). In wired network, all nodes in a network require an Ethernet adapter, commonly known as Network Interface Card (NIC) to connect with other devices. The Network Interface Card (NIC) can be internally installed through Ethernet adapter port in the computer or attached externally to the nodes. In the following section, the most commonly used wired networks will be reviewed.

2.2.1. Star Network

In a Star Network, three or more nodes are connected with each other through central devices usually called hub or switch where nodes can be different computers or printers. Star Network topologies are mainly used in small business or even as a house network. Nodes in a network use separate cables and failure of one node or cable does not harm other computers and network keeps on functioning all the time. However, failure of central hub or switch affects the networks and nodes cannot continue in a network. In this case, the trouble shooting or replacement of a central node is essential. Due to high usage and less use of cables, Star Network is a most commonly used types of wired LAN. A TCP congestion control mechanism is used in a network to avoid collisions as all nodes can send data at a same time which ensures high throughput in a network with minimum packet loss. **Figure 2** shows a basic architecture of Star Network.

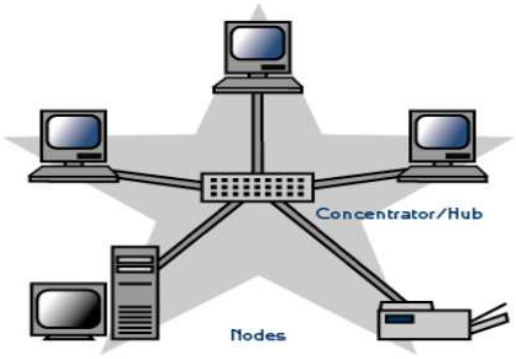

***Figure 2**. Star Network (Pandey et al. 2013).*

2.2.2. Bus Network

Bus Network is initial type of wired networks which uses common circuit instead of a central hub to connect different nodes within a network. In Bus Network, only one node can transmit data at a time and if there are two nodes trying to send data then collision will occur and result in data loss. All the nodes in a network broadcast messages in a network with a destination address and the message travels to all nodes until a destination is reached. **Figure 3** shows a basic architecture of Bus Network.

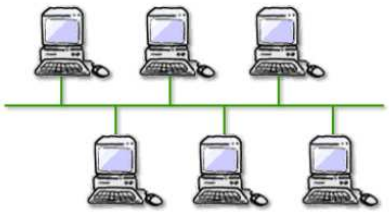

***Figure 3**. Bus Network (Pandey et al. 2013).*

Special software is used in a network which decides data transmission in a network. As there is no central hub used in a network, so failing of one node is robust and doesn't affect the network to work. Special terminators are used at the end of common cable to ensure that packets are not re-bounced.

2.2.3. Ring Network

In Ring Network, nodes are connected in a closed loop or ring where every node connects to the next node and the last node connects to the first node. A special ring token is used in a network and any node which has ring token can transmit data in a network. A message in a network always travels in one direction. Ring Network has no central control hub, thus network remains functioning even any of the nodes is broken. **Figure 4** shows a basic architecture of Ring Network.

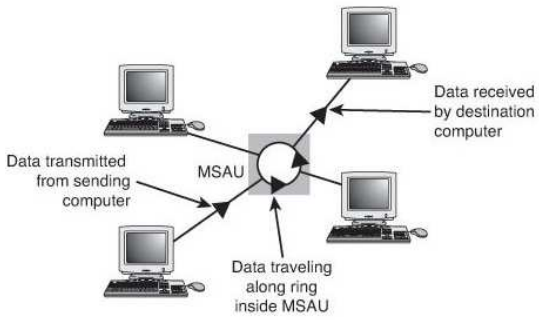

***Figure 4**. Ring Network (Bestofmedia Team 2012).*

2.3. Wireless Networks

Wireless networks are the fastest growing mean of the communication in a modern world. In the presence of wired communication systems, it has captured the attention of the media and the imagination of the public remarkably due to its efficient, reliable and long distance coverage services. The exponential growth in the use of cellular systems over the last decade turns the number of mobile users into almost two billion figures worldwide. Furthermore, many home users, businesses, and campuses are using wireless local area networks in place of ancient wired networks. Many new applications including wireless sensor networks (WSN), mobile ad hoc networks (MANET), vehicular ad hoc networks

(VANET), automated highways and factories, smart homes and appliances, and remote telemedicine are emerging from research ideas to concrete systems (Goldsmith 2004).

In brief, wireless communication is a transfer of information from source to destination over a distance without the use of electrical conductors or wires. The distances involved may be short (a few meters as in television remote control) or long (thousands or millions of kilometers for radio communications). There are many wireless communication systems available used for multi purposes. Some of these systems are: mobile, portable two-way radios, cellular telephones, personal digital assistants (PDAs), GPS units, garage door openers and or garage doors, wireless computer mouse, keyboards and headsets, satellite television and cordless telephones. Different background technologies used in wireless systems are bluetooth, infrared, wifi, gsm, gprs, 3G, 4G, LTE and satellite.

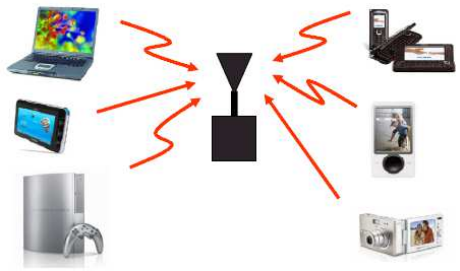

***Figure 5**. Wireless Network.*

Wireless networks (see **Figure 5**) support communications using radio or light waves propagating through an air medium. Many wireless networks are used depending upon the need of use and distance requirements. Following is an overview of different wireless networks used in communication systems.

2.3.1. Wireless Sensor Network

Wireless Sensor Network (WSN) shown in **Figure 6** consists of wireless nodes connected through access point in a relatively long distance area equipped with sensors which measures quantities in surroundings to monitor physical and environmental changes. Sensor

nodes in a network are tiny devices having small memory and power storage which require an analytical design for the devices to work in long durations. Due to tiny nature and small infrastructure, these networks faces quite many challenges ranges from reliability, power consumption, reduced node sizes with high utilization, mobility to privacy and security. Some of the applications for WSN consist of military surveillance and monitoring, medical diagnosis and monitoring, environmental monitoring, industrial sensing and diagnostics for appliances, factory, supply chains, infrastructure protection monitoring including power grids and water distribution monitoring and other miscellaneous applications (K Sharma, M K Chose. 2010).

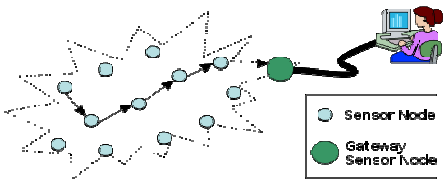

***Figure 6**. Wireless Sensor Networks.*

2.3.2. Wireless Ad-hoc Network

Wireless Ad-hoc Network (WAN) shown in **Figure 7** is a collection of wireless nodes connected with each other in a temporary network without any infrastructure and control of a central administration. These nodes connect on random basis and communicate with the neighboring nodes for information sharing and notifications. These networks are not robust in nature and always under certain security threads due to the independent nature of connected nodes; therefore making the communication in Wireless Ad-hoc networks is a critical challenge.

***Figure 7**. Wireless Ad-hoc Network (Pandey et al. 2013).*

2.3.3. Mobile Ad-hoc Network

Mobile Ad-hoc Network (MANET) shown in **Figure 8** is a sub type of wireless ad-hoc networks and a self-organizing and self configuring multihop wireless network where structure of the network changes dynamically all the time. The dynamic nature of the network is based on mobility of the nodes. In a network, nodes connect with each other in friendly manner and become of part of multihop forwarding mechanism. These nodes behave in a network as a host as well as router to forward information from one node to another. The routing in MANET is a challenge due to the unavailability of the infrastructure; therefore all nodes in a network are responsible to forward data for other nodes using proper routing mechanism. Without routing, out of range destination nodes become unreachable as other nodes cannot find a proper route to reach, hence resulting in packet loss. In MANET base stations access all the network nodes by sending broadcast messages instead of following routing flow. Applications of MANETs are used in classrooms, battlefields and vehicle-to-vehicle communications in certain scenarios.

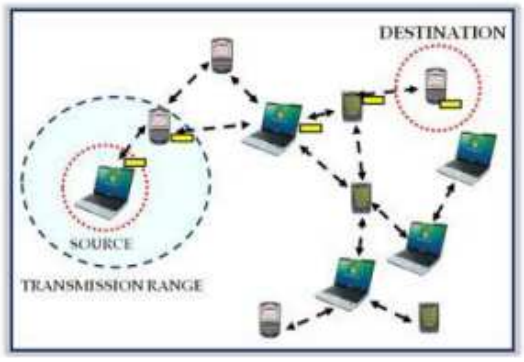

***Figure 8**. Mobile Ad-hoc Network (MANET).*

2.3.4. Vehicular Ad-hoc Network

Vehicular Ad-hoc Network (VANET) shown in **Figure 9** is an emerging technology which integrates modern wireless networking capabilities to vehicles. It is based on Mobile Ad Hoc Networks (MANETs) which is a collection of wireless mobile nodes connected with

each other for exchanging information in the absence of any infrastructure. In principle, Vehicular Ad Hoc Network (VANETs) is a subclass of MANETs; however it behaves in fundamentally different ways than MANETs due to fragile nature of vehicles connectivity, fast movement, varying driver behaviors and high mobility in a network (Ho & Leung 2007).

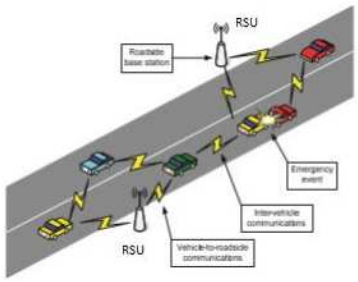

***Figure 9**. Vehicular Ad-hoc Network (VANET).*

In Vehicular Ad Hoc Network, the idea is to provide ubiquitous connectivity among different mobile users on the road with the efficient vehicle-to-vehicle communication which enables the Intelligent Transportation Systems (ITS). Due to the connectivity paradigm, Vehicular Ad Hoc Networks (VANETs) (Li & Wang 2007) are also called Inter-vehicle Communications (IVC) or Vehicle-to-Vehicle Communication (V2V) and Vehicle-to-infrastructure Communications (V2I) where vehicles communicate with the nearby vehicles and road side units (RSU) through dynamic wireless links. The applications of ITS in Vehicular Ad Hoc Networks (VANETs) includes (Li et al. 2007) co-operative traffic monitoring control of traffic flows, blind crossing, prevention of collisions, nearby information services and real time detour routes computation. Due to variety of safety critical applications, Vehicular Ad Hoc Networks (VANETs) are gaining intention from researchers and engineers in academic and automobile industries for road safety and pleasure applications. Vehicle Ad Hoc Networks (VANETs) require automobile cars to be equipped with computing technologies and internet connectivity through wireless networks and major car manufacturer companies have already announced to add computing and connectivity powers to their vehicles.

2.3.5. Wireless Local Area Network

Wireless Local Area Network (WLAN) (see **Figure 10**) is a short range network which connects two or more devices in a closed circuit through access point. These networks are mainly used for providing internet access to the connected devices using the IEEE 802.11 WLAN standard called WiFi. In wireless area network, spread-spectrum or OFDM technologies allow mobility to the devices in local area without disconnecting from the network. The applications of wireless area networks are in houses, offices, airports, shopping markets, universities and others for providing free and on-demand internet services to the end users.

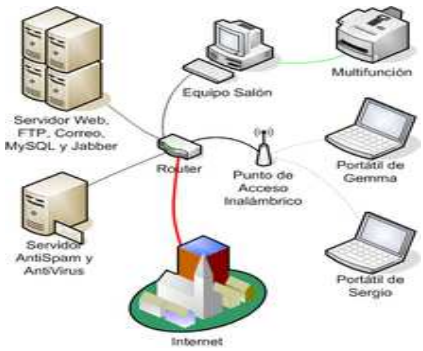

***Figure 10**. Wireless Local Area Network (WLAN).*

2.4. Wireless Channel

Wireless channel is a path in a spectrum being used for transmission of electromagnetic signals. A defining characteristic of the mobile wireless channel is the variations of the channel strength over time and over frequency (Tse & Viswanath 2005). The variations can be roughly divided into two types:

Large-scale fading– Large-scale fading is caused by path loss of signal due to distance and shadowing by large objects such as buildings and hills. This occurs as the mobile moves

through a distance of the order of the cell size, and is typically frequency independent (Tse et al. 2005).

Small-scale fading– Small-scale fading occurs due to the constructive and destructive interference of the multiple signal paths between the transmitter and receiver. This occurs at the spatial scale of the order of the carrier wavelength, and is frequency dependent (Tse et al. 2005). Due to the variation in signal power, receiver tries to adopt the changing characteristics of the channel e.g. changing the equalizer parameter. However if changes are too fast, such as driving on a highway, receiver can't adopt fast enough and hence the transmission error probability can be dramatically increased.

Large-scale fading is more relevant to issues such as cell-site planning whereas small-scale multipath fading is more relevant to the design of reliable and efficient communication systems (Tse et al. 2005).

Figure 11 compares short term and long term fading.

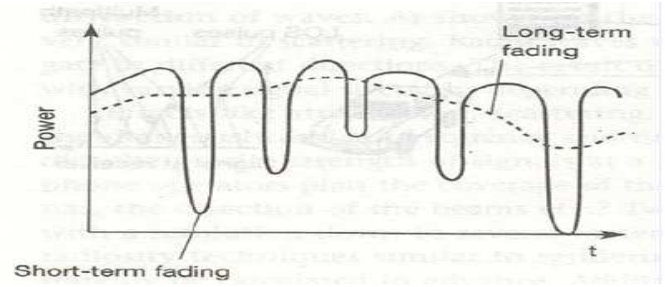

Figure 11*. Short term and long term fading (Schiller 2003).*

2.4.1. Wireless Channel Physical Modeling

The radio propagation of electromagnetic waves between transmitter and receiver is characterized by the presence of multipath due to various phenomena such as reflection, refraction, scattering, and diffraction. The study of wave propagation appears as an important task when developing a wireless system (El Zein 1993). The performance of

communication systems depends on the propagation medium and the physical modeling of antenna. For broadband systems, the analysis is usually made in the frequency domain and the time domain which allows measuring the coherence bandwidth, the coherence time, the respective delay spread, and Doppler spread values. Coherence distance and wave direction spread are also used to highlight the link between propagation and system, in the space domain. **Table 1** lists details for radio channel parameters.

Table 1. *Radio Channel Parameters (R H Katz. 1994).*

Coherence Domain	Dispersion Domain
Frequency, f	Delay, τ
Time, t	Doppler, ν
Space, r	Spatial pulsation, k

In USA, the Federal Communication Commission (FCC) has limited the cellular communications in one of three frequency bands, one around 0.9 GHz, one around 1.9 GHz, and one around 5.8 GHz. The wavelength λ of electromagnetic radiation at any given frequency f is given by $\lambda = c / f$, where $c = 3 * 10^8 m / s$ is the speed of light (Tse et al. 2005). The wavelength in these cellular bands is a fraction of a meter, so to calculate the electromagnetic field at a receiver, the locations of the receiver and the obstructions would have to be known within sub-meter accuracies. Thus, the spatial and temporal properties of the channel with accurate measurements are necessary for the design of broadband multi-antenna systems with a choice of suitable network topology.

2.4.1.1. Free space, fixed transmit and receive antenna

Fixed transmit and receive antennas model works in a similar fashion as wired communication model works, where signal is viewed as simply a voltage or current waveform. In this case, in a far field, the electric field and magnetic field at any given location are usually perpendicular and proportional to each other and to the direction of propagation from the antenna. Therefore, it is sufficient to know only one of them.

In response to a transmitted sinusoid $\cos 2\pi ft$, the received waveform at fixed antenna point $u = (r, \theta, \varphi)$ is then:

$$E_r(f, t, u) = \frac{\alpha(\theta, \varphi, f)\cos 2\pi f(t - r/c)}{r} \quad (1)$$

Where r represents the distance from transmitter antenna to receive point u, (θ, φ) represents the vertical and horizontal angles from the antenna to u respectively. The constant c is the speed of light, and $\alpha(\theta, \varphi, f)$ is the product of the antenna patterns of transmitter and receiver antennas in the given direction (Tse et al. 2005). Here, (1) is linear in input and forms linear time invariant (LTI) channel. That is, the received waveform at u in response to a weighted sum of transmitted waveforms is simply the weighted sum of responses to those individual waveforms which doesn't change frequency.

System function at point u is given by:

$$H(f) = \frac{\alpha(\theta, \varphi, f)e^{-j2\pi fr/c}}{r} \quad (2)$$

Where

$$E_r(f, t, u) = \Re\left[H(f)e^{j2\pi ft}\right] \quad (3)$$

And inverse Fourier transforms of $H(f)$ is an impulse response of a channel.

2.4.1.2. Free space, moving antenna

A model, where receiver antenna becomes non-stationary with speed v in the direction of increasing distance of transmitter antenna and changes its position with respect to time t, a destination point is represented as $u(t) = (r(t), \theta, \varphi)$ and relative distance becomes

$r(t) = r_0 + vt$. Thus, the received electric field using (1) at the moving point $u(t)$ is given by:

$$E_r(f, t, (r_0 + vt, \theta, \varphi)) = \frac{\alpha(\theta, \varphi, f)\cos 2\pi f\left[(1 - v/c)t - r_0/c\right]}{r_0 + vt} \qquad (4)$$

Where the sinusoid at frequency f has been converted to a sinusoid of frequency $f(1 - v/c)$ and $-fv/c$ is Doppler shift due to the motion of the observation point.

The channel is represented as linear time variant (LTV) that changes the frequency with respect to times. The channel can be represented in terms of a system function followed by translating the frequency f by the Doppler shift $-fv/c$, if the time varying attenuation in the denominator of (4) is ignored. It is important to observe that the amount of shift depends on the frequency f. Here it is not important that either transmitter or receiver or both are moving. However, channel characteristics depend on a relative distance between two antennas caused by the movement.

2.4.1.3. Reflecting walls, fixed antenna

In this model, the characteristics of channel and signal propagation depends on the multi signal interference caused due to reflecting walls in the surroundings of fixed transmitter and receiver antennas.

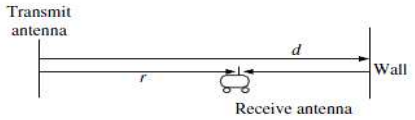

***Figure 12**. Illustration of the direct path and reflective path (Tse et al. 2005).*

In **Figure 12**, transmitter antenna sends a sinusoidal signal $\cos 2\pi ft$ towards a fixed receiver antenna where transmitted signal has two paths; direct and indirect. The indirect signal path

comes from the fixed reflecting wall that adds up to the direct path signal at receiver. The electromagnetic field at the receive antenna is the sum of the free space field coming from the transmit antenna plus a reflected wave coming from the wall. Assume that if the receive antenna is absent; the perturbation of the field due to the antenna is represented by the antenna pattern. An additional assumption here is that the presence of the receive antenna does not appreciably affect the plane wave impinging on the wall (Tse et al. 2005). In this situation, the intensity of the reflecting signal is same as a free space wave that would exist on the opposite side of a large wall in case wall doesn't exist (see **Figure 13**) which means that the total intensity of the reflective signals is of length equal to the sum of distance from transmit antenna to wall and then back to the receive antenna from wall, i.e., $2d - r$.

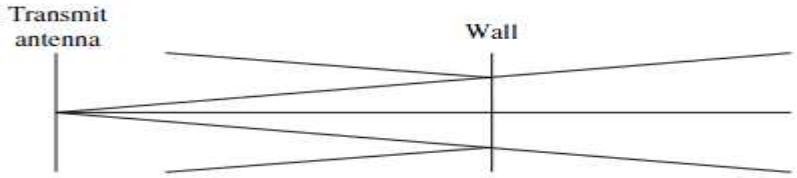

Figure 13*. Relation of the reflected wave to the wave without wall (Tse et al. 2005).*

Considering both direct and reflected wave with the antenna gain α , equation (1) becomes:

$$E_r(f,t) = \frac{\alpha \cos 2\pi f(t - r/c)}{r} - \frac{\alpha \cos 2\pi f(t - (2d - r)/c)}{2d - r} \qquad (5)$$

Where, phase difference between two waves is:

$$\Delta\theta = \left(\frac{2\pi f(2d - r)}{c} + \pi\right) - \left(\frac{2\pi fr}{c}\right) = \frac{4\pi f}{c}(d - r) + \pi \qquad (6)$$

On the basis of above equations, the interference of both signals can be either constructive or destructive depending upon the phase value. If the phase value is integer multiple of 2π , the interference will be constructive that makes the signal strong at the receive antenna. On

the other hand, if the phase value is odd integer multiple of π then the received signal becomes week due to destruction. The difference of the peak to low intensity of the signal is referred as *coherence distance* and is denoted by:

$$\Delta x_c = \frac{\lambda}{4} \tag{7}$$

Where $\lambda = c / f$ is a wavelength of transmitted sinusoid.

2.4.1.4. Reflecting walls, moving antenna

Consider the above model with the assumption that the receive antenna starts moving towards the large reflective walls at speed v. **Figure 14** illustrates direct and reflected path.

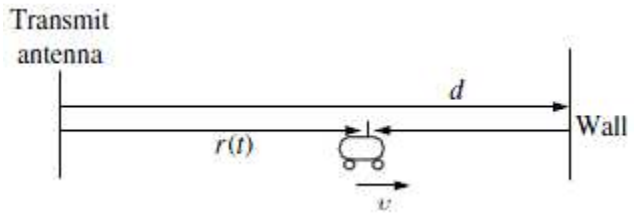

Figure 14. *Illustration of a direct and reflected path (Tse et al. 2005).*

The movement of a receive antenna causes interferences between two signal waves and the intensity of the receive signals starts either decreasing or increasing. The construction and destruction of signals occurs due to the phase change and the phenomenon of variation in a signal quality is called *multipath fading*. The time taken to travel from a peak to a valley is $c / 4fv$: this is the time-scale at which the fading occurs, and it is called the *coherence time* of the channel (Tse et al. 2005).

Consider if the starting location of a receiver at time 0 is r_0 and total distance is $r = r_0 + vt$ at time t, the received signal equation using (5) becomes:

$$E_r(f,t) = \frac{\alpha \cos 2\pi f\left[(1-v/c)t-(r_0/c)\right]}{r_0+vt} - \frac{\alpha \cos 2\pi f\left[(1+v/c)t+((r_0-2d)/c)\right]}{2d-r_0-vt} \qquad (8)$$

From the equation, Doppler shift effects can be easily seen from the direct and reflected sinusoid signals where in the direct wave term at frequency $f(1-v/c)$, Doppler shift is $D_1 = -fv/c$ and from the indirect wave at frequency $f(1+v/c)$, Doppler shift is $D_2 = fv/c$. The difference between D_1 and D_2 is called *Doppler spread.*

2.4.1.5. Reflection from a ground plane

In physical modeling of mobile systems, if transmit and receive antennas lies on a ground plane in a way that the horizontal distance between the antennas is larger than the height of both antennas from a ground plane then at certain distance r two waves; direct and reflected from the plane starts to cancel each other (see **Figure 15**). In these kinds of models, the difference in length of direct and reflected wave is directly proportional to r^{-1}.

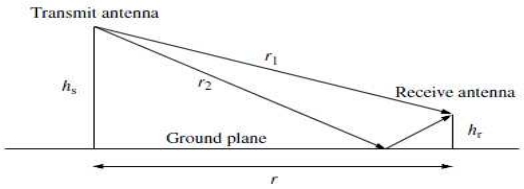

Figure 15*. Illustration of direct and reflected path of ground plane (Tse et al. 2005).*

In case, if receive antenna is moving in the opposite direction of transmit antenna, than situation happens when the length difference r^{-1} between both the antennas becomes 0 due to the increase in total distance r. As a result both direct and indirect signals become equal and start canceling each other due to opposite phase. The electric wave at the receiver is then attenuated as r^{-2}, and the received power decreases as r^{-4} (Tse et al. 2005). The impact of this situation is important to consider in those areas where base-station transceivers are placed on roads.

2.4.2. Input/output models of the wireless channel

In wireless communication, the behavior of input signal throughout the channel path is usually affected by the involvement of different obstacles; specially buildings or hills. On the basis of these multipath effects caused by reflection, diffraction or scattering, channel can be modeled as linear-time varying system. In the following section, the characteristics of linear-time varying model and discrete-time varying model derived from continuous-time channels will be reviewed.

2.4.2.1. Wireless channel as a linear-time varying system

Wireless channels behavior changes depending on the spatial attributes of the transmitter, receiver and environmental factors. If all elements of the communication are stationary then the channel becomes linear-time invariant channel while on the other hand, if any of the element of wireless communication starts moving with respect to times then channel becomes linear-time variant channels in which attenuation factors of transmitter and receiver and propagation delay changes as time passes.

In linear-time variant channel, if the input signal is $x(t) = \cos 2\pi ft$ then the received signal can be written as:

$$y(t) = \sum_i \acute{a}_i(f,t)x(t-\tau_i(f,t)) \qquad (9)$$

Where $\acute{a}_i(f,t)$ and $\tau_i(f,t)$ are attenuation and propagation delay respectively from transmitter to receive antenna at path i from n available paths. In practical scenario, attenuation and propagation delays are slow varying function of frequency f . These variations follow from the time- varying path lengths and also from frequency-dependent antenna gains (D Tse & P Viswanath. 2005), therefore if the frequency from linear-time variant system model is omitted then (9) becomes:

$$y(t) = \sum_i \acute{a}_i(t)x(t-\tau_i(t)) \tag{10}$$

Due to the linear nature of the input/output system, the complete system equation in terms of impulse response of a channel can be defined in following manner:

$$y(t) = \int_{-\infty}^{\infty} h(\tau,t)x(t-\tau)d\tau \tag{11}$$

Where $h(\tau,t)$ is an impulse response at time t for the input signal transmitted at $t-\tau$. Alternatively, the expression for the impulse response from (11) can be given as:

$$h(\tau,t) = \sum_i \alpha_i(t)\delta(\tau-\tau_i(t)) \tag{12}$$

In special case when transmitter, receiver and all environmental reflectors becomes stationary where attenuation $\acute{a}_i(t)$ and propagation delay $\tau_i(t)$ don't depend on time than the equation of impulse response $h(\tau,t)$ for the resulting linear-time invariant systems for the input signal $x(t) = \cos 2\pi ft$ can be written as:

$$h(\tau) = \sum_i \alpha_i \delta(\tau-\tau_i) \tag{13}$$

Considering the time-varying impulse response $h(\tau,t)$, the time-varying frequency response can be derived as:

$$H(f;t) = \int_{-\infty}^{\infty} h(\tau,t)e^{-j2\pi f\tau}d\tau = \sum_i \alpha_i(t)e^{-j2\pi f\tau_i(t)} \tag{14}$$

Usually linear-time invariant channels reduce frequency response in a system. One can overcome this issue by considering the channel as a slow varying function of time t with

frequency response $H(f;t)$ at each fixed time t. In this case, time t at which channel varies becomes much longer than the delay spread, thus these types of channel can be called as underspread channels.

2.4.2.2. Baseband equivalent model

Normally, in mobile communication data is transmitted using passband signal from sender to receiver within a frequency range $[f_c - W/2, f_c + W/c]$. But other processing including coding/decoding, modulation/demodulation, compression/decompression, synchronization etc at each terminal is happened in a baseband signal. Usually, transmission signal is up-converted from baseband to passband at receiver before sending on the transmission medium and at receiver down-converted from passband to baseband before processing. Therefore understanding the conversion process of passband and baseband signal is crucial in telecommunication arena.

If the passband signal is $s(t)$ with Fourier transform $S(f)$ and frequency band limited to $[f_c - W/2, f_c + W/c]$, then baseband equivalent signal $s_b(t)$ can be represented as:

$$s_b(f) = \begin{cases} \sqrt{2}S(f + f_c) & f + f_c > 0 \\ 0 & f + f_c \le 0 \end{cases} \qquad (15)$$

The factor of $\sqrt{2}$ is quite arbitrary but chosen to normalize the energies of $s(t)$ and $s_b(t)$ to be the same. $s_b(t)$ is band-limited in $[-W/2, W/c]$ (Tse et al. 2005). **Figure 16** shows relationship between passband spectrum and baseband equivalent.

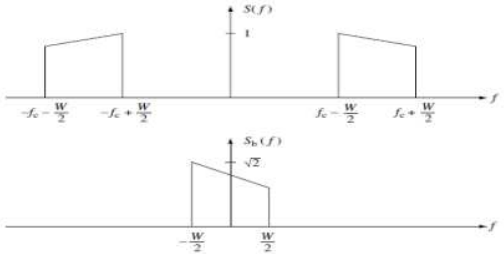

Figure 16*. Passband spectrum and baseband equivalent Relationship (Tse et al. 2005).*

Given baseband signal $s_b(t)$, passband signal can be constructed with the following observation:

$$\sqrt{2}S(f) = S_b(f - f_c) + S_b^*(-f - f_c) \tag{16}$$

When taking the Fourier transform:

$$s(t) = \frac{1}{\sqrt{2}}\left[s_b(t)e^{j2\pi f_c t} + s_b^*(t)e^{-j2\pi f_c t}\right] = \sqrt{2}\Re\left[s_b(t)e^{j2\pi f_c t}\right] \tag{17}$$

The complete realization of conversions (up and down) for passband and baseband is illustrated in **Figure 17**.

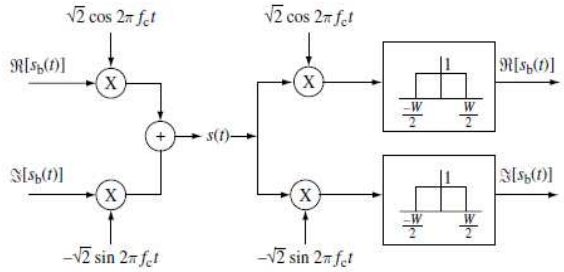

Figure 17*. Illustration of up-conversion followed by down-conversion (Tse et al. 2005).*

Figure 18 relates baseband transmitted signal to baseband received signal.

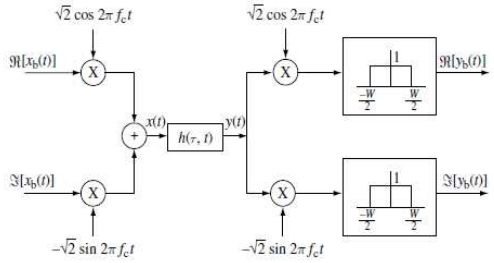

Figure 18. *Baseband transmitted signal to baseband received signal(Tse et al. 2005).*

2.4.2.3. A discrete-time baseband model

Finally, a discrete-time channel can be derived from continuous-time channel using sampling theorem technique. For the input waveform with a band limited to W , the baseband equivalent has the following form with the band limited to $W/2$:

$$x_b(t) = \sum_n x[n] \sin c(Wt - n) \tag{18}$$

Where

$$x[n] = x_b(n/W) \text{ and } \sin c(t) = \frac{\sin(\pi t)}{\pi t}$$

From the above baseband input signal, baseband output signal can be represented as:

$$y_b(t) = \sum_n x[n] \sum_i a_i^b(t) \sin c(Wt - W\tau_i(t) - n) \tag{19}$$

Discrete-time channel equation then can be obtained by sampling the output baseband signal at multiple of $1/W$ as:

$$y[m] = \sum_n x[n] \sum_i a_i^b(m/W) \sin c(m - n - \tau_i(m/W)W) \tag{20}$$

Where

$$y[m] = y_b(m/W)$$

Equation (20) can be simplified by considering $l = m - n$ as:

$$y[m] = \sum_n x[m-l] \sum_i a_i^b (m/W) \sin c(l - \tau_i (m/W) W) \qquad (21)$$

All equations used are taken from (Tse et al. 2005).

2.5. Congestion Control and Queue Management

In last decade, communication systems have brought revolutionary changes in everyone's life where the benefits of wired and wireless networks are utilized for impressive countless quality of services in different areas. With communication revolution, the speed and capacity of various components in versatile networks such as transmission media, switches and routers have been drastically increased. Also the number of users and traffic flow has reached to a new sky level which makes communication system more complicated and diversified (Koo, Ahn &Chung. 2004).

In this case, the Quality-of-Services (QoS) is obligatory in wired and wireless Networks for ensuring the integrity, reliability and security of the information from source to destination in a fast and efficient way resulting in best-effort services for end users. Also performance requirements in terms of throughput, delay, jitter and packet loss are essential for good quality services. QoS enabled network provides various functions for improving packet delivery performance such as rate controller, classifier, scheduler and admission control (Koo et al. 2004). A sequential order in which packets are required to be processed is handled through congestion control and queue management policies namely called as

Active Queue Management (AQM).

TCP and AQM are designed to work for both wired and wireless networks. In wired networks, main reason for packet loss is network congestion while in wireless networks, small bandwidth, mobile nature of the nodes and pure wireless links cause packet loss (Dhadse & Chandavarkar 2014). In a network, congestion occurs when source message gets time out or source receives three duplicate ACK messages. In this case, router is flooded with messages and its queue gets over flown resulting in dropping last messages from queue tail. This method is synchronized with all the connected nodes. Router informs nodes about the congestion resulting nodes to reduce the data sending rate which may lead in low link utilization. In this case, AQM helps in utilizing the underline network by properly managing the queue and keeping the average queue size small which results in fewer number of packet loss. In coming sections, different queue mechanisms used for congestion control will be reviewed along with TCP design to handle the congestion control in wired and wireless networks.

2.5.1. TCP Congestion Control

Transmission Control Protocol (TCP) is a standard set of rules used with Internet Protocol (IP) to send data in form of packets from source computer to destination computer over the internet. TCP is a controller which keeps track of the data divided into small junks called packets and ensures end-to-end transmission of data, using certain routing algorithms, format of the data, ordering and retransmission in case of failure (Bhargava, Bhargava, Mathuria, Gupta & Jyotiyana 2013).

TCP Congestion Control mechanism is used to ensure the uninterrupted communication and better utilization of network resources from source to destination. It provides window based end-to-end flow control where on the receiving a message, receiver sends ACK message back to sender to notify about correctly receiving a packet and sender updates the congestion window size. A window size at the source is always proportional to the allowed

transmission rate. In case of congestions, TCP congestion control dissolves congestion by asking distributed nodes to reduce window sizes. Sources then update the window sizes to avoid more congestion, also congestion measures are updated by channel links. These updated measures are feedback to sources using the link.

TCP Congestion Control Algorithms

First TCP congestion control algorithm was introduced to the TCP protocol stack in 1988 by VAN Jacobson followed by three other introduced until 1990 (Sun & Xiaoling. 2012) including the slow start algorithm, congestion avoidance algorithm, fast retransmission and fast recovery algorithms. These algorithms provide basic architecture of TCP flow control and congestion control. Due to the emerging requirements in network transmission area, researchers proposed and implemented various improvements including TCP Tohoe, TCP Reno, TCP new Reno, TCP SACK (Selective Acknowledgement) and TCP Vegas. **Table 2** shows different congestion avoidance methods:

Table 2*. Congestion Avoidance Methods.*

Variants of TCP	Name of Algorithms
TCP Tahoe	Slow start + Fast retransmission
TCP Reno	Fast retransmission + Fast recovery (in case of single packet loss)
TCP New Reno	Fast retransmission + Fast recovery (in case of multiple packets lost)
TCP SACK	Fast Retransmission + Fast Recovery (in case of retransmission of more than one lost packet)
TCP Vegas	New retransmission mechanism + Modified slow start + New congestion avoidance mechanism

Four TCP core congestion control algorithms are executed at source end and they belong to source algorithms in terms of realization (Sun et al. 2012). These algorithms are using

adjustable parameters including congestion window (CWND), window slow start threshold (ssthresh), delay (RTT) and overtime counter (RTO). In the following section, the core algorithms used for TCP Congestion Control and set of rules with the parameters used by these algorithms will be reviewed.

Slow Start and Congestion Avoidance

In slow start algorithm (see **Figure 19**), TCP uses ACK received from receiver to adjust congestion window (CWND) size. At the start of TCP connection, it enters into a slow start-up phase by setting CWND value to 1 for all the connected nodes in a network and gradually increases CWND on getting successful ACK from sender. A threshold value is set to 65535 bytes. On a certain condition when CWND reaches and overflow to threshold value, TCP enters in a congestion avoidance phase.

Slow start and congestion avoidance algorithms (Bhargava et al. 2013):

- Declare congestion window (CWND) and bandwidth threshold (ssthresh) variables.
    - *Declare CWND, ssthresh*
- Initialize variables.
    - *Initialize CWND = 1, ssthresh = 65535*
- Increment CWND value on getting every successful ACK from receiver.
    - *CWND = CWND + 1*
- On each RTT, increase CWND value exponentially.
    - *CWND = 2 x CWND*
- Entering congestion avoidance phase when
    - *CWND >= ssthresh*
- In congestion avoidance phase, reduce the ssthresh to half of CWND and initialize CWND to 1
    - *ssthresh = CWND / 2*

- *initiaze CWND = 1*
- *Start*

- *Increases the CWND value linearly infractions until congestion avoidance phase is over.*
  - *CWND = CWND + 1 / CWND*

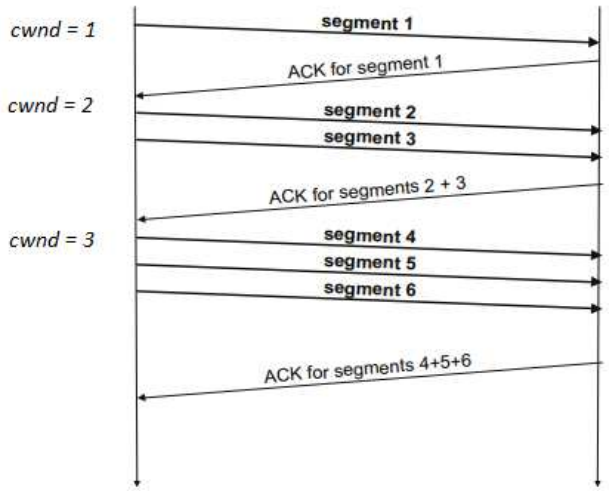

***Figure 19**. Slow start congestion control.*

Rapid Retransmission and Recovery

In TCP, retransmission timeout counter (RTO) has impact on TCP performance such as it has much bigger value then each packet round trip time (RTT) which results in long idle time in case of real packet lost. In order to avoid such situation, a new retransmission protocol is introduced which helps in avoiding the wait time instead retransmitting the lost packet in quick and efficient manner. In general, when sender receives three same ACK from receiver then it decides to retransmit a packet by considering it a lost packet. In this case, the threshold value is reduced to current congestion window, and CWND is reduced to half of the original value. When lost packet retransmission is in progress, TCP does not wait for RTT and start recovering data before getting RTT as it is a fast retransmission and recovery algorithm (see **Figure 20**). It allows high throughput under moderate congestion especially for large window.

Fast retransmission and recovery algorithms (Bhargava et al. 2013):

- On receiving three duplicate ACK in a row, set threshold value to current congestion window.
    - *ssthresh = CWND*
- Retransmit a missing packet
- Set congestion window using threshold value.
    - *CWND = ssthresh + 3*
- On getting same duplicate ACK each time, increase congestion window by 1.
    - *CWND = CWND + 1*
- In case of non-duplicate ACK arrives, set congestion window value using threshold and continue with a linear increase such as congestion avoidance.
    - *CWND = ssthresh*

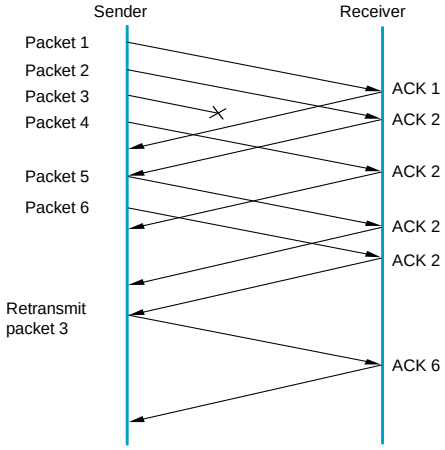

***Figure 20**. Fast retransmission and recovery algorithm.*

2.5.2. Queue Management

In communication networks, hosts understand the behavior of congested router either by time-out occurs and sender receives series of same ACK from receiver. In this case, the network hosts adjust the data sending rate to reduce the network congestion. At the same time, a better queue management is required at router to ensure a free and fare handling of

network packets coming from all network nodes. A traditional queue management process works in a way that when there is network congestion and queue is filled, router drops packet from tail of a queue to overcome the congestion. Router also notifies all the connected nodes about the congestion and in a result all nodes reduces the data sending rate. However, this approach causes Lock out and full Queue problem (Dhadse &Chandavarkar 2014). To avoid these problems, new queue management algorithms known as Active Queue Management have been introduced to ensure better and fair queue management for all nodes in a network.

The main goal of a queue management in network systems is to increase the throughput as well as decrease the average packet loss and end-to-end delay. In the following section, the definitions of different key indicators for a good network system will be briefly described.

Queue is a place in a system, where packets arrive for a service, wait for a service if not an urgent packet, and leave the system on getting served. In principle, queuing system is defined by four basic characteristics in networks such as arrival pattern of packets, service pattern of schedules, queue discipline and system capacity.

Delay is a time elapsed between start and end point of a communication system. Total delay is calculated from source to destination including all in between nodes and normally is called end to end delay. Delay in a network can happen due to over congestion situation or transmission delay of a network. In communication systems, delay insensitive applications can be effected badly by a network delay such as audio or video applications which requires minimum delay for better quality propagation at receiver.

Packet loss occurs when receiver does not receive intended packet from sender and hence start recovering process. Packet loss affects the throughput of a network. Packets can be lost when the queue in a network gets overflow, hence affecting the loss insensitive applications. Some applications cannot perform well if end-to-end packet loss between nodes is large relative to some threshold value, causing excessive packet loss effecting the

certain real time applications. (Koo et al. 2004.)

End-to-end delay is a sum of delays encountered at every point in a network from source to destination. Each such delay consists of two constant components including transmission delay at the node and propagation delay on the link to next node. (Koo et al. 2004.)

Active Queue Management

Active Queue Management (AQM) ensures reduce number of packets dropped in router by keeping the average queue size small. Without AQM, more packets can be dropped when queue overflows which results in bad quality of a system. In the following section, the traditional TailDrop queue management along with few AQM policies will be reviewed which helps in improving the overall performance of a network.

DropTail

DropTail simply revolves around first in first out (FIFO) queues without using any additional parameters. In case of queue outburst, packets are always dropped from tail which results in unfairness of a network. In DropTail, most of the queue shares are used by few of the network nodes while others suffer from less utilization of a queue. This results in Lock Out and Full Queue problem. The Lock Out is often resulted due to the synchronization in a network. As the tail drop only informs connected nodes about the congestion when queue is full and packet drop occurs, which results in sudden drop for whole of a network and there is no intelligence sharing during the time when queue is not full. In this case, handling the full queue size affects all the nodes casing full queue delays.

Random Early Detection

Random Early Detection (RED) is a mechanism of AQM which ensures better queue management by deducting the congestion in advance. The objective of a RED is to

minimize the packet loss and queuing delay as well as avoid global synchronization of sources to maintain high link utilization and remove biases against high data rate sources (Koo et al. 2004). RED is not specifically designed for certain protocol; however it works impressively for the protocols which perceive drops as indication of congestion. TCP is one of those where RED performs well.

In RED, user is required to specify five parameters including maximum buffer size or queue limit, minimum threshold, maximum threshold, maximum dropping probability and wait factor used to calculate the average queue size (Dhadse et al. 2014). RED works in three different zones.

- Normal operation zone - when average queue size (Q_{avg}) is below minimum threshold (min_{th}), there is no packet drop.
- Congestion avoidance zone - when average queue size (Q_{avg}) is between minimum threshold (min_{th}) and maximum threshold (max_{th}), packets are discarded with certain probability P_a.
- Congestion control region - when average queue size (Q_{avg}) is above maximum threshold (max_{th}), all packets are dropped.

Here are formulas (Koo et al. 2004) for calculating the average queue size (Q_{avg}) and probability (P_a).

Average queue size:

$$Q_{avg} = (1 - W_q) \cdot Q_{avg} + W_q \cdot Q \qquad (22)$$

where Q is the current queue length and W_q is a weight parameter, $\leq W_q \leq 1$

Probability:

$$P_a = P_b \ / (1 - count \cdot P_b) \qquad (23)$$

And

$$P_b = max_p / (max_{th} - min_{th}) . (Q_{avg} - min_{th}) \quad (24)$$

where max_p is a maximum value of P_b when average queue length Q_{avg} is equal to max_{th}.

Other AQM policies include BLUE, REM and PI which ensures better congestion control and fair queue management for QoS-enabled systems.

SFQ

SFQ uses large number of separate FIFO queues to provide fairness in queue management.

3. NETWORK SIMULATORS

Network simulation is an important technique in model era, in which researchers can model hypothetical and real life network objects on computer and observe the behavior of the underline networks by executing different experiments based on combination of various parameters. A simulation can consist of different network entities including hosts, interconnections between nodes, connecting devices such as router, switches and bridges, configuration systems, mobility models and system level networking protocols.

In Network research area, implementation and deployment of complete test bed containing various network nodes, connecting devices including routers, bridges and switches and data links to validate and verify certain networking protocols is very expensive. In this case, network simulators are cost and time effective solution to achieve the tasks through different network simulations. Network simulators help researchers to test the design of new networking protocols or change the existing networking protocols in a controlled and reproducible manner (Pan & Jian 2008).

Network simulators are used by different researchers, industrial scientists and Quality Assurance (QA) engineers for testing the performance and validity of different networking protocols where visibility of a simulation is irrelevant. In other words, the main objective of network simulation is to observe the characteristics and behavior of a network where one can simulate, emulate and analyze the end results of network simulations. The network research area is very wide where new revolutions as well as innovations are seen every day. Different organizations work in parallel in different technologies and contribute in building of the network communication platforms where new technologies are evolving every single moment. In this case, the progressive growth in network simulators is very important so that all the new technologies are immediately available and can be evaluated as soon as these become standards for everyone. The growth in network simulators cannot be handled by a single organization and contributions from different organizations is essential. It can

only be achieved by open platforms of all network simulators such that everyone can work and contribute to the development of network simulators and make them up to date with the recent technologies.

Authors (Gupta, Mangesh, Ghonge, Thakare & Jawandhiya 2013) listed some of the advantages and disadvantages of network simulators given in **Table 3**:

Table 3*. Network simulators advantages and disadvantages.*

Advantages	Disadvantages
• Inexpensive • Help to gain the knowledge about the improvements of a system • Help to understand the working principle of a system • Testbed for evaluating new communication protocols • Finding bugs and troubleshooting in advance • Use of analytic and numerical techniques for observing the behavior of a system • Partial simulation for observing parts of full model	• Unclear model reflection to reality • High simulation complexity in large scale networks • Slow compare to reality (1 minute of real time can take hours in simulations) • Hard to determine right level of model complexity • Uncertain statistical results

In the following sections, the basic concepts, main features, languages and recent as well as future developments of different open source network simulators will be assessed. At the end, a comprehensive comparison between different network simulators will be provided based on merits and demerits, platforms, cost, license, API, user interface and supported network types.

3.1. Basic Concepts in Network Simulators

Understanding of basic concepts in network simulators is essential for everyone especially for new researchers so that they can utilize maximum benefits of network simulators and produce results approached near to reality. Here, some of the basic concepts to help readers are listed.

3.1.1. Network Simulator and Simulation

Network simulator is a tool which provides user interface to the users for defining a model using diverse network components. User interface can be a command line or graphical user interface (GUI). Command line interface requires strong programming skills while GUI requires basic knowledge for novel users. In principle, network simulators allow users to model any real world model where users can tweak different network properties and analyze the result. But in reality, network simulators are not perfect and models rarely matches the real world models due to diversity, unpredictable and random nature involved in real models. However, network simulators can provide relatively close results which gives user a meaningful insight into the network under test and how the parameter changes can affect its operation (Pan et al. 2008).

3.1.2. Network Simulation and Emulation

Network simulation is a process in which a researcher models range of real world models using various network components including nodes, routers, switches, physical links or packets and applies mathematical formulas for evaluation. The simulation experiments conducted either online or offline in the controlled environments can be observed using various combinations of parameters and configuration settings.

On the other hand, emulation is an extension to simulations where end systems such as computers can be attached to simulation models through emulators and act as they are

connected to real network. A famous NS2 simulator can be used as a limited-functionality emulator whereas WANsim is the typical bridge WAN network emulator.

3.1.3. Discrete Event Simulation

In network simulation, discrete-event simulation (DES) is a process of modeling the operation of a system as discrete sequence of events in time where each event occurs at a particular time causing a change in the state of a system (Borboruah & Nandi 2014). In this process, continuous events are not possible between two consecutive events; hence state of a system can jump between states on specific time intervals. Currently, most of the available network simulators are based on discrete-event simulation.

3.2. Type of Network Simulators

Network simulators can be classified based on certain criteria such as they are free or commercial, open source or proprietary and simple or complex.

3.2.1. Free and Commercial

Some of the network simulators are free and provide open source code for researchers. The advantage of such simulators is that the source code is available to everyone for contributions and researchers can analyze different parts of the software as well as improve the functionality or introduce new features based on their requirements. On the other hand, different people contribute and make amendments in a source code and there is no single organization which governs the development, resulting in diversity and lack of systematic and complete documentation which can lead to serious problems. Free and open source network simulators include NS2, NS3, OMNET++, SSFNet and J-Sim.

Commercial network simulators are proprietary software and source code is not open to anyone. Only organization which owns network simulator can manage the source code and implement new features. All the users have to pay for a license to use their software as a whole or as a partial system where user pays for specific packages. The advantage of commercial network simulators is that they come with a well organized, systematic and up-to-date documentation for end user which is consistently maintained by specialized staff in a company. The famous commercial network simulators are OPNET and QualNet.

3.2.2. Simple and Complex

Currently, there are penalty of different network simulators available in market ranges from simple to complex in nature. The basic functionality in simple simulators allows end users to define simple topologies consisting of scenarios, specifying the nodes and links between these nodes and generating traffic in a network. GUI enabled network simulators also make life easy for end users and they can view underline simulations clearly and can define network models using drag and drop features.

Contrary, complex network simulators provide more room to end users to play with core networking protocols by providing them programmatic platform where only skilled researchers can effectively work. These simulators are usually text based and provide less interactive interface but allow advance customization to the source code.

3.3. Network Simulator 2

Network simulator 2 (NS2) is a most eminent object oriented, open source discrete-event simulator used in communication research and development. NS2 and extension of NS which was developed in 1989 based on REAL network simulator and evolved revolutionary over the past couple of years (Pan et al. 2008). NS2 was originally developed at University of California, Berkeley for focusing the simulation of IP networks on packet levels. NS2

project is now part of the Virtual InterNetwork Testbed (VINT) project which is responsible for developing tools for network simulation research (Karl 2005). Currently, NS2 is used by large group of organizations in academic research and non-profit groups all around the world contributed various packages in a core base. In recent years, NS2 development is discontinued and NS3 has taken over the place.

NS2 covers a large number of networking applications, protocols, networking types, network elements and traffic models. Researchers use NS2 for the development and analysis of various protocols such as TCP and UDP, router queuing policies such as RED, ECN and CBQ, unicast and multicast transmissions, multimedia applications and other networking resources.

A platform for NS2 is based on two languages consisting of C++ and OTcl (Tcl script language with object oriented extension developed at MIT). NS2 core is based on C++ and NS2 frontend is based on OTcl. C++ is an efficient language for writing device drivers and low-level applications, the purpose to use C++ in NS2 is to have an efficient mechanism to execute simulations which increases the overall performance and reliability of the simulations. On the other hand, C++ is not good and easy to use for graphical user applications and here OTcl language helps researchers to cover this area.

Figure 21 illustrates a simplified user view for NS2.

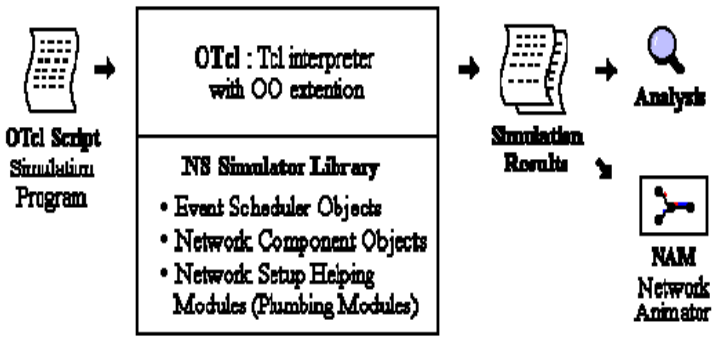

***Figure 21**. Simplified NS2 user view (Pan et al. 2008).*

In NS2, OTcl language is used to model networks by defining various network nodes, data link connections, routers and network configurations such as queue management, routing and congestion control. OTcl provides easy approach to modify and assemble different components and change different parameters on the fly to the end users. Considering the main principle of these languages, C++ covers the control part as well as OTcl covers data part of the simulation implementation. NS2 uses C++ for the implementation and compilation of the event scheduler and basic network component objects to reduce packet and event processing time. Moreover, C++ is used to implement detailed network protocols whereas OTcl is used for providing a controlled way to define different simulation scenarios and researchers can schedule different events using the provided C++ event scheduler classes.

In NS2, user defines simulation scenario using network nodes, protocols, network topology, specific application and form of required output in OTcl script. OTcl interpreter links the written script to compiled C++ components through OTcl linkage that creates one to one match of OTcl object for each of C++ object. After the execution of the simulation, the simulation results are captured in different ways such as in tracing files which are used to analyze the results using different statistical and analytical techniques. NS2 comes with network animator (NAM) shown in **Figure 22** which displays the visual simulation to the end users.

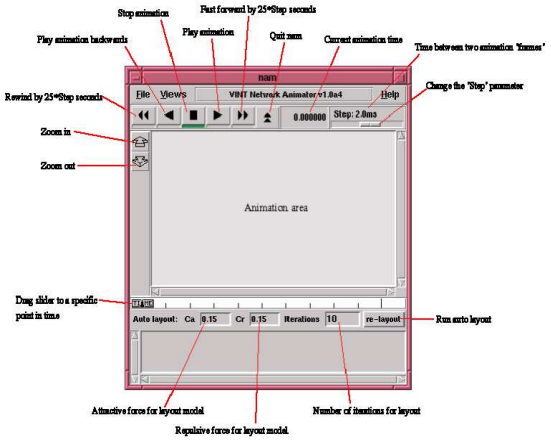

***Figure 22**. NAM (Karl 2005).*

3.3.1. Architectural Overview

NS2 architecture (see **Figure 23**) consists of the following five parts:

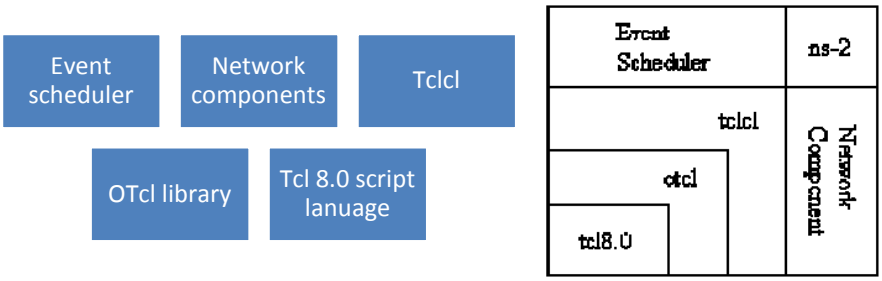

***Figure 23**. NS2 architecture (Karl 2005).*

Event scheduler is used to schedule simulation events on discrete time intervals, therefore is known as discrete-event scheduler (DES). Event scheduling can be any type such as packet-handling delay events or specific timers use for scheduling certain actions. **Figure 24** shows important component if discrete event scheduler.

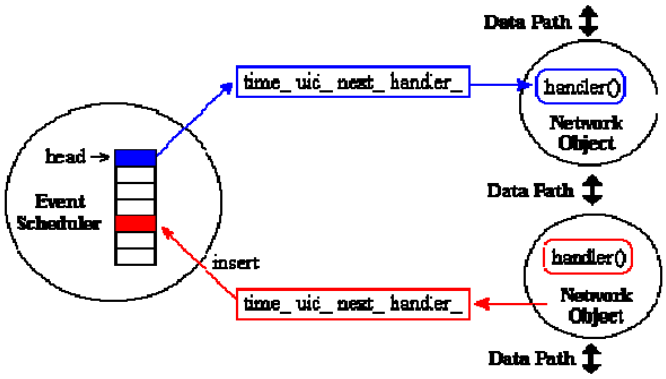

***Figure 24**. Discrete event scheduler (Karl 2005).*

Network components are network elements defined as C++ class hierarchy in NS2. The example of OTcl class hierarchy is given in the following figure:

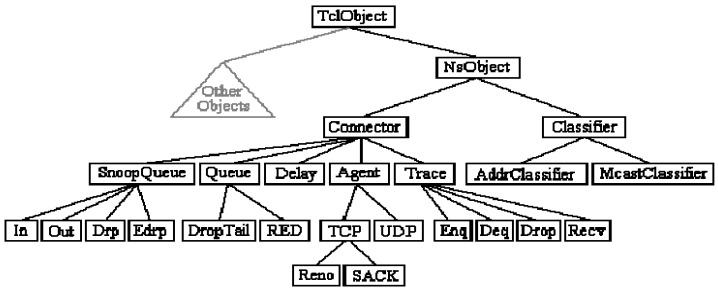

***Figure 25**. OTcl class hierarchy (Karl 2005).*

In the OTcl class hierarchy (see **Figure 25**), TclObject is a super class of all the objects whereas NsObject is a super class of Connector and Classifier. Connector class is a parent of all the basic network components which have only one output data path whereas Classifier class is a super class of advance network components which has multiple output data paths.

Tclcl is used to implement OTcl linkage.

OTcl is an extension to Tcl/Tk scripting language (Karl2005) for object oriented programming. It is used to define the prototype, configuration and control model of the simulation. User can define event scheduler, the network topology and data links, traffic, errors configurations and tracing options in NS2 OTcl script.

Tcl 8.0 is a scripting language used for writing OTcl script in NS2.

3.3.2. NS2 Models and Technologies

NS2 has a long list of supported models and communication technologies contributed by various non-profit organizations around the globe. Though the development of NS2 is clogged due to future NS3 but researches are still utilizing existing NS2 models for the network research area.

Table 4 briefly illustrates the available models and technologies in NS2.

***Table 4**. NS2 models and technologies.*

Category	Model and technologies
Routing	AODV, AODV-UV, ZRP, AOMDV, IS-IS, RCDS, DLSR, DMCR, DYMO, UM-OLSR, ATM, HWMP
Wired, Wireless and Mobility	ARP, HDLC, GAF, MPLS, LDP MAC: CSMA, Satellite Aloha, Queuing: Drop Tail, RED, RIO, SRR, WFQ, REM, IEEE 802.11b, IEEE802.15.4, IEEE 802.11 support, IEEE 802.11 PHY-MAC design and implementation, IEEE 802.11 PCF, IEEE 802.11 PSM, IEEE 802.11e EDSA and CFB simulation model, IEEE 802.11e HCCA module, IEEE 802.15.4, IEEE 802.16 model, IEEE 802.16 model MIRACLE framework, IEEE 802.16 wireless mesh networks, NS2-emulation extension (optimized for wireless networks, IR-UWB, TDMA DAMA satellite support, WiMAX, CRCN, UCBT Bluetooth, SUNSET underwater networking, VANAT, CanuMobiSim, EURANE extensions, BonnMotion, a java mobility scenario generator and analyzer, GPRS, BlueHoc, a bluetooth extension, CIMS
Transport	TCP Pacing, UDP, DCCP for wired and wireless networks, Linux TCP Congestion Control for 12 different congestion control algorithms (BIC, CUBIC, HighSpeed TCP, H-TCP, TCP-HYBLA, NewReno, Scalable TCP, Vegas, Westwood, TCP Veno, TCP Compound and TCP Low-Priority), Network Simulation Cradle, TCP Westwood, Extensions to RTP code, Freeze-TCP, Multipath TCP, Data Center TCP (DCTCP), TCP ex Machina, SCTP, TCP Rate-Halving Algorithm, MFTP, SNACK
Other models and technologies	Satellite networks, Topology and traffic generation, Differentiated services, Integrated services, Scheduling and queue management, Multicast, DTN, Application layer

NS2 development for new models is almost terminated and no notable enhancements have been made during last one decade. However, NS2 is lightly maintained through its active mailing list.

3.4. Network Simulator 3

Network simulator 3 (NS3) is a next generation simulator which aims to improve existing system functionalities and network models of NS2 with the improved software core and execution methodologies. NS3 is based on NS2, GTNets and YANS. NS3 development work started in 2006 by NSF CISE and INRIA and the first official release was in 2008. The main objectives behind the development of NS3 were to provide a different software core written in C++ and python scripting interface in order to enhance the simulation performance. Furthermore, other focused areas included the intention to realism and software integration with more open source networking software.

Similar to NS2, NS3 is an open source discrete-event simulator and various organizations and researchers are continuously striving to contribute new telecommunication models and network protocols implementations, data link layers functionalities and tracing as well as analytical methodologies. NS3 is not backward compatible and uses completely different programming languages and platform for writing core libraries and network simulations compared to NS2. However, a few of the existing NS2 models have already been transformed to NS3 and are currently being used. NS3 allows researchers to study and evaluate various internet protocols and large scale systems in a controlled environment.

In **Figure 26**, NS3 simulation architecture components are listed.

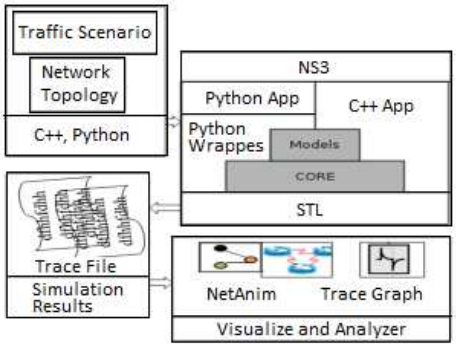

***Figure 26**. NS3 Simulation Architecture (Rajankumar, Nimisha & Kamboj 2014).*

NS3 core is designed as C++ libraries which can be statistically as well as dynamically linked to C++ main program for various network simulations. Python scripting interface is used by users as a wrapper to encapsulate C++ modules which are easy to use compared to programming in C++. For a simulation, user creates a traffic scenario by defining network topologies consisting of various network components in either C++ or Python. A traffic scenario is attached to C++ core where different NS3 models and libraries are linked to the program. The execution of a simulation can be viewed using an animator, for example NetAnim. For the analysis, NS3 provides a tracing mechanism which produces trace files. These trace files can be plotted to various graphs using different graph tools.

Authors (Chaudhary et al. 2012) categorized NS3 features in different sections such as testbed integration, attribute system, tracing architecture and topology generation. According to them, NS3 testbed integration enables researchers to experiment various novel protocol stacks and emit network packets over real device drivers. NS3 attribute system allows researchers to identify and configure values to the parameters in a simulator. These values can be handled as default values, hard coded in a simulator script or configured values provided at run time from console. Furthermore, NS3 tracing system uses call back functions to separate tracing data completely from trace sink and enable customization of the tracing or statistics output without rebuilding the simulation program. Authors also emphasizes on a topology building feature of NS3 which allows users to design and model simulation scenario using number of stock topology objects implemented

in C++ libraries. NS3 Stock objects include trees, meshes, stars and other random topologies.

Figure 28 illustrates NS3 features in end to end simulation starting from reason question to simulation visualization and analysis.

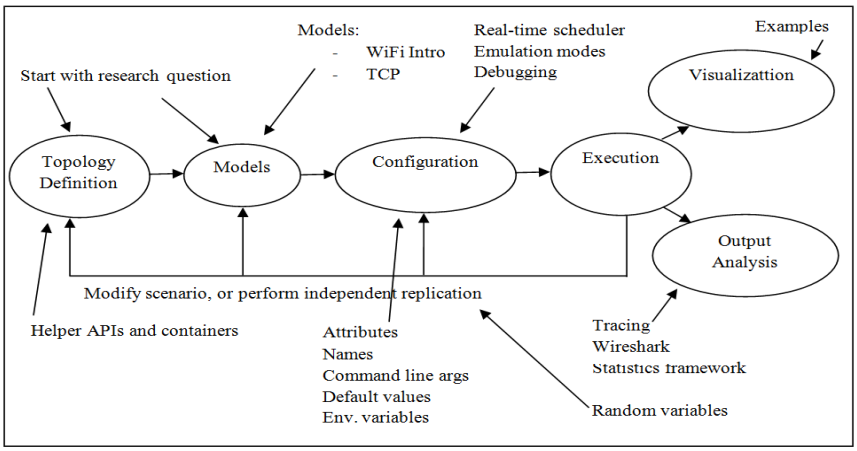

***Figure 27**. NS3 Features (Chaudhary et al 2012).*

3.4.1. Architectural Overview

NS3 Internal architecture consists of various components and **Figure 28** illustrates these components in detail:

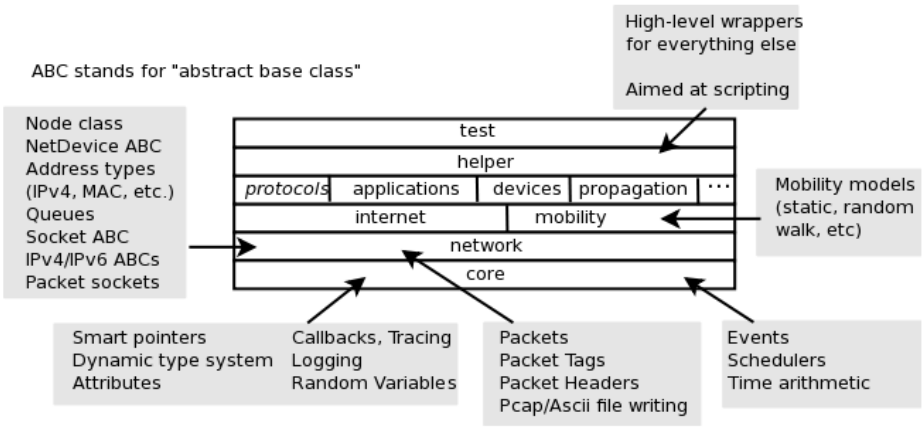

***Figure 28**. NS3 Internal Architecture.*

The NS3 core contains all the common components used across all protocols, hardware and environmental models. NS3 network consists of fundamental network objects including packets and nodes. In addition, other network layer components including address types, queues and sockets also belong to NS3 network. NS3 core and network are two basic platform components which are used not only in all network simulations but also in other simulations as well. Other components such as internet, mobility, protocols, applications, devices and propagations are subclasses of core components. Helper classes are wrappers which encapsulate low level complex API calls for easy use. These classes provide convenient ways for python scripts where NS3 core libraries can be imported using these helper classes.

Figure 29 exemplifies NS3 IP stack architecture showing various components working in end-to-end communication:

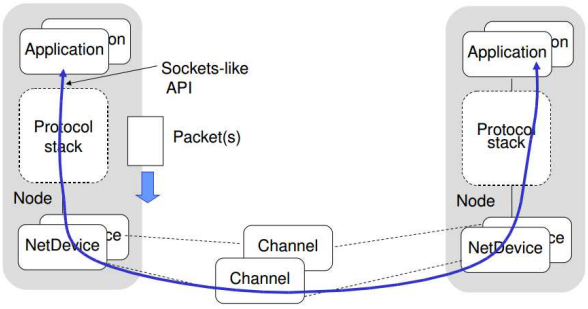

***Figure 29**. NS3 IP Stack Architecture.*

In NS3 IP stack architecture, nodes are entities which can be static or have mobility nature. These nodes contain network devices which transfer packets over a channel over physical layer and data link layer phase. Network protocols such as IP and ARP are managed at network layer whereas transport protocols such as TCP and UDP are supported at transport layer. Network simulation applications are written by end users at application layer. From sender to receiver, data travels on certain channels and all application layers are traversed in reverse order at receiver.

NS3 testbed supports real system integration with the network simulation and **Figure 30** shows brief details for NS3 testbed:

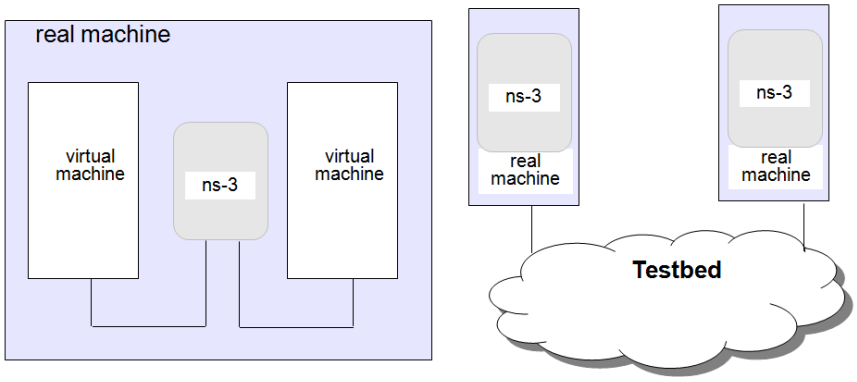

***Figure 30**. NS3 Testbed.*

NS3 interconnects virtual machines on real machines and testbed interconnects NS3 stacks.

For simulation visualization, NS3 uses various tools such as NetAnim, ns-3-viz, pyviz and iNSpect. Most of these tools are still under development. **Figure 31** shows NetAnim interface which is most commonly used tool in NS3 community:

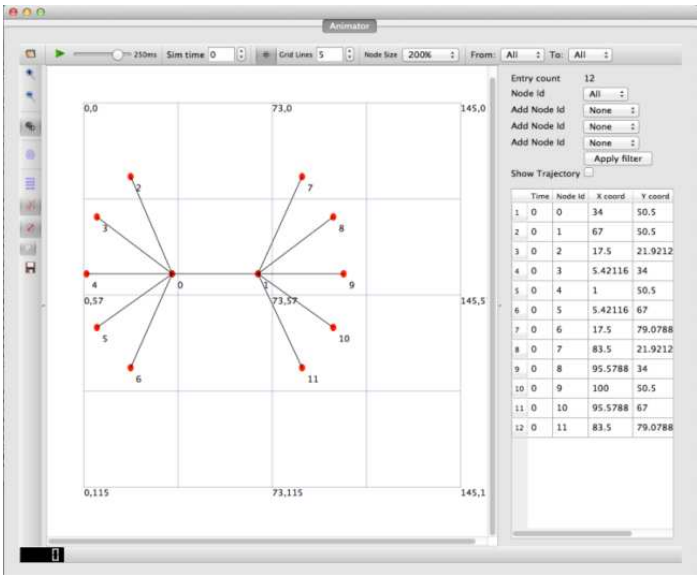

***Figure 31**. NetAnim (NetAnim from ns-3 wiki).*

NetAnim is based on QT 4 toolkit developed by George F Riley which uses XML trace files collected from simulation and animates the results in offline mode.

3.4.2. NS3 Models and Technologies

Table 5 illustrates the available model and technologies in NS3.

Table 5*. NS3 Models and Technologies.*

Category	Model and technologies
Routing	NAT, BGP, OSPF, RIP, IS-IS, PIM-SM, IGMP, Static (Dijkstra) unicast, Static multicast, DSDV, Global (link state), Nix-vector, DSR, MANET, OLSR, AODV, VANET
Wired, Wireless and Mobility	IEEE 802 physical layers, New 802.11 model, Wifi 802.11 links, Mesh 802.11s, IEEE 802.11 variants (mesh, QoS), WiMAX 802.16, TDMA, CDMA, , GPRS, CSMA, Bridge (802.1D Learning), PPP, Zigbee, MPLS, Rayleigh and Rician fading channels, GSM, Jakes composite loss model, Friis, Hierarchical, Random direction, RWP, ns-2 Scen-Gen
Transport	TCP stack emulation (Linux, BSD), UDP, Additional high speed TCP variants, DDCP
Other models and technologies	Sockets-like API, Traffic generator, Ping, Echo, Packet sink, Topology input reader, Random number generator, Tracing, Unit test, Logging, Callback, Error models

Many of the models and technologies for NS3 are under development at the moment and major challenges for NS3 are open to solve. Experts from different research organizations are voluntarily contributing to NS3 core stack on daily basis which means that new components will be added to the NS3 stack in coming years.

3.5. OMNET++

Objective Modular Network Test-bed in C++ (OMNET++) is an open source, discrete-event and component based network simulator with GUI support. It is available as free software for academic and research use as well as it has commercial license for the industry. OMNET++ has a better documentation and support for commercial license as dedicated group of experts are paid and work in different areas such as research and development, testing, support and documentation for end users. OMNET++ was developed by András Varga at the Department of Telecommunications, Technical University of Budapest (Erdei Márk et al) in 1997. The primary area of OMNET++ simulations is communication networks; however its generic and a flexible architecture allows working with other areas such as IT systems, queuing networks, hardware architectures and even business processes models as well.

OMNET++ is based on component architecture where all components, also called as modules, are written in C++. High level language (NED) is used to assemble these components in simulation scenarios same as OTcl in NS2 and Python in NS3. The modular architecture of OMNET++ supports simulation kernel to be embedded into various kinds of different end user applications. **Figure 32** illustrates OMNET++ graphical user interface.

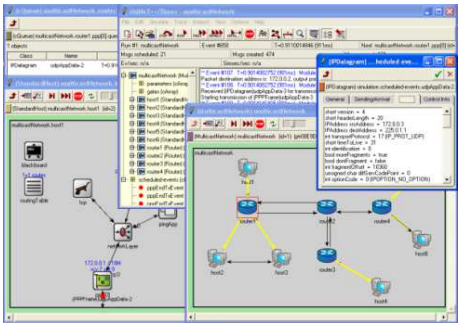

***Figure 32**. OMNET++ GUI.*

OMNET++ works for both wired and wireless networks and offers an Eclipse-based IDE, a graphical runtime environment and a host of other tools. It also comes with the extensions

for real time simulation, network emulation, database integration, SystemC integration and several other functions.

Here is a list of OMNET++ components in brief.

- *Simulation kernel library*
- *NED topology description language*
- *OMNET++ IDE based on Eclipse platform*
- *GUI for simulation execution, links into simulation executable (Tkenv)*
- *Command line user interface for simulation execution (Cmdenv)*
- *Utilities (makefile creation tool, etc*

Simulation kernel library is written in C++ and consists of utility classes for random number generation, statistics collection, topology discovery etc. These classes are used to define simulation components including simple modules and channels which are assembled and configured for simulation models in NED. Simulation programs using NED and C++ are written in Eclipse IDE for designing, running and evaluating simulations. Runtime user interface environments such as Tkenv and Cmdenv are used for the simulation execution. Users can use utilities such as *makefile* to build and run the simulation from command line. OMNET++ has organized documentation and sample programs for learning and support researchers.

3.5.1. Architectural Overview

Figure 33 represents OMNET++ architecture of a simulation program:

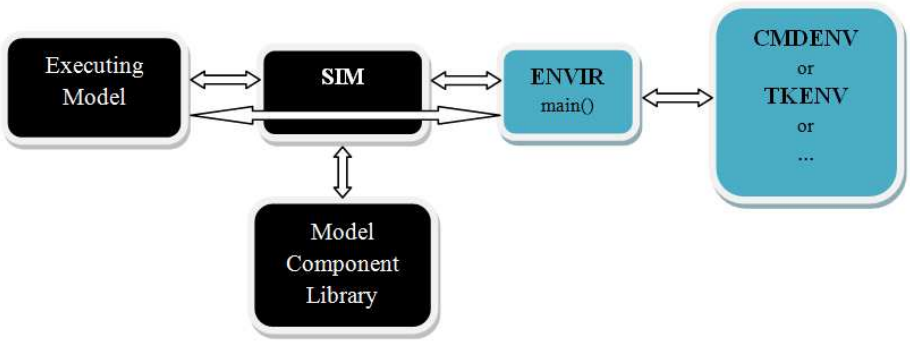

***Figure 33**. OMNET++ Architecture.*

In order to simulate any communication network in OMNET++, one has to follow certain steps from a start to the end of the simulation experiment. The steps include defining a model structure and network topology in NED file, writing active components in C++, building a *MakeFile* and creating a simulation executable using *make* command. Furthermore, writing simulation configurations and parameters in *OMNetpp.ini* file, running a simulation executable and processing the results are the important steps in OMNET++. User can also modify configuration parameters, build and run simulation executable for any number of times.

In **Figure 34**, summary of all the OMNET++ simulation steps using a flowchart is given.

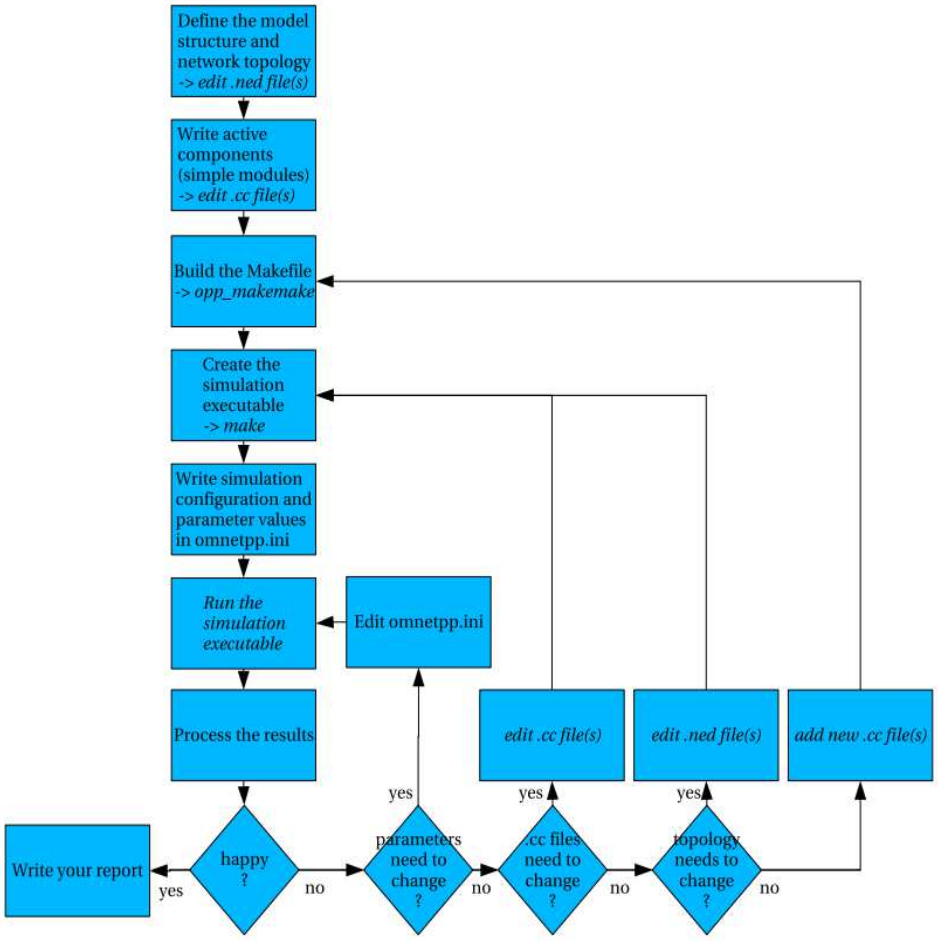

Figure 34*. OMNET++ Simulation Process.*

For simulations, OMNET++ includes INET framework which is an open source communication networks simulation package. It contains models for various wired and wireless networking protocols including TCP, UDP, SCTP, IP, IPv6, Ethernet, PPP, 802.11, MPLS, OSPF and others. The INET framework uses the same concepts as OMNET++ such as modules communicating by message passing. In INET, protocols are designed as simple modules with external interface defined in a NED file whereas implementation is provided in a C++ class with the same name. The INET framework is helpful for the beginners and provides various example models for the simulations.

3.5.2. OMNET++ Models and Technologies

Table 6 provides a list of models and technologies available in OMNET++ INET framework.

Table 6*. OMNET++ INET Models and Technologies.*

Category	Model and technologies
Routing	Link-state routing, OSPF (INET), OSPF (Quagga), BGP (INET), RIP (INET), BGP (Quagga), RIP (Quagga), STP, RSTP, MANET: AODV, DYMO-UM, DYMO-FAU, DSDV, DSR, DSR, OLSR
Wired, Wireless and Mobility	PPP, Ethernet, IEEE 802.11 (INET), IEEE 802.1e, IEEE 802.11 (MF), IEEE 802.16e (WiMAX), IEEE 802.16 (WiMAX), IEEE 802.15.4 (LR-WPAN), MPLS, LDP, RSVP-TE, ARP, HIP, DHCP
Transport	TCP (INET), TCP (lwIP), TCP (NSC), UDP, SCTP, RTP, RTCP
Other models and technologies	Traffic generators (CBR/VBR), HTTP traffic generators, File transfer, Basic and advance video and voice streaming models, Sensor networks, Vehicular networks, Cellular networks, Satellite networks, Optical networks, Interconnection networks, NoCs, Cloud Computing, HPC clusters and SANs

Due to GUI support and well organized libraries, OMNET++ is popular in academic and industrial research for its extensibility and open source code. Online documentation is also a good resource for beginners to start working with OMNET++.

3.6. Comparison of Network Simulators

In this section, NS2, NS3 and OMNET++ will be compared on the basis of programming languages, platforms, memory management, performance, network models, and simulation

output. Merits and demerits of using these network simulators in communication network research will also be explained.

All three network simulators core libraries are written in C++ language. In addition, different scripting languages are used by each network simulator for network typology design and implementation. **Figure 35** highlights the common as well as uncommon programming languages used by these simulators.

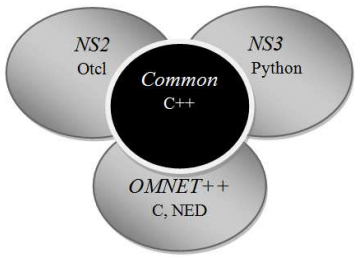

***Figure 35**. Network Simulators and Programming Languages.*

In addition to programming languages, NS2 and NS3 provides command line interface to the user whereas OMNET++ has a graphical user interface where user can drag and drop different network elements to design the network topology on the fly.

Network simulators are supported by different operating system platforms and are not available for all operating systems. **Figure 36** provides details for platforms support in NS2, NS3 and OMNET++.

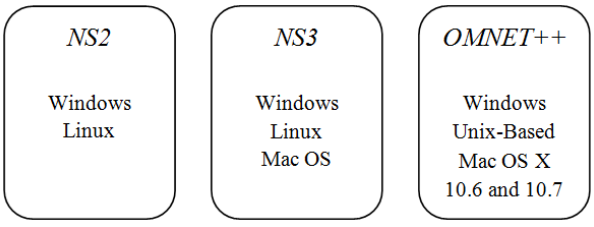

***Figure 36**. Network Simulators and Platforms.*

Memory management is critical in network simulations, therefore special considerations are given in the design of NS2, NS3 and OMNET++. NS2 uses basic manual C++ memory management functions to utilize memory in best possible way. However, these are old methods ad NS2 cannot compete with newly developed network simulations when it comes to memory management and utilization. NS3 uses basic manual C++ memory management functions such as *new*, *delete*, *malloc* and *free*. It also uses new techniques such as automatic de-allocation of objects using reference counting (track number of pointers to an object) to deal with unused packets.

OMNET++ and NS3 are proved to be better in performance compared to NS2. In NS2, the computation time consumed by interfacing Otcl interpreter with C++ is an actual overhead which takes time and affects the performance of a simulation. In NS3, the aggregation system helps to avoid storage of the unused parameters and reserved header spaces for packets. NS3 and OMNET++ use a garbage collection which enhances the memory utilization in the simulation of large scale networks without having effects on performance.

NS2 is the most widely used network simulator as it supports almost all network models. Other simulators including NS3 and OMNET++ are in development phase and still lack support for numerous number of network models. **Table 7** lists all the supported models for NS2, NS3 and OMNET++.

Table 7*. Network Simulators and Supported Network Types.*

Network Simulator	Supported Network Types
NS2	• Wired networks • Wireless Ad-Hoc networks • Wireless managed networks • Wired cum wireless networks • Wireless sensor networks with the exception that it cannot simulate problems of bandwidth or power consumption in these

	networks
NS3	• Wired networks • Wireless networks • Wireless sensor networks
OMNET++	• Wired networks • Wireless managed networks

For simulation visualization and analysis, NS2 includes a utility called Network Animator (NAM) whereas NS3 comes with visualization utility programs known as ns3-viz, pyviz, NetAnim. OMNET++ IDE provides a good support for the simulation visualization and analysis within a tool. Other tools such as OMVis are also available for OMNET++ simulation analyses which focus on the intuitive and spatio-temporal visualization of simulation data.

Merits and demerits of network simulators

NS2 merits

- NS2 is most used network simulator in communication network research due to a rich collection of network models and technologies.
- NS2 supports parallel and distributed simulations.
- Over 50% of ACM and IEEE network simulation research papers cite the use of NS2.

NS2 demerits

- Development is almost stopped and unmaintained for a long period of time.
- Lacking the adaptation of modern programming techniques such as smart pointers and design patterns therefore has outdated code design.

- Network scalability is not well supported raising memory management and performance issues for large scale networks.
- Simulation analysis is difficult due to use of complex tracing system where one needs to parse the trace files to extract required results. Trace files contain unnecessary information and sometimes miss the required information.
- Hard to find centralized documentation and tutorials, information is really dispersed.
- Hard to debug the code due to use of bi-languages, especially for Otcl scripting language.

NS3 merits

- Fast compared to NS2 as everything is designed in C++ with optional python scripting support.
- NS3 is active open source project and is continuously under development for enhancements and improvements under the supervision of experts working for different organizations voluntarily.
- Integrations support for external tools such as random mobility generators, traffic generators and others.
- Emulation mode supports integration with real systems which makes NS3 preferable to use.
- Attribute system allows end users to either configure simulation parameters within a code or provide as command line arguments during run time which makes NS3 really flexible which helps to change the experiment outputs without building the simulation.
- Good network scalability and suitable for large scale network simulations, resulting in improved memory management and performance for network simulations.
- Use of modern programming techniques such as smart pointers and design patterns.
- Tutorials and well organized documentation is available.

- Easy debugging of a code due to full use of C++.

NS3 demerits

- NS3 is still underdevelopment and lacks a lot of models already available in NS2, therefore is not recommended for all kind of network simulations.
- Limited GUI which makes network modeling very complex and time consuming task.
- Even though NS3 is considered to be a next generation of NS2, but it is written from scratch and therefore lacks backward compatibility with NS2.

OMNET++ merits

- Support simulations for large scale networks.
- Use modern programming techniques such as design patterns.
- Modular structure where modules are used as components and the definition is separated from the implementation.
- Modules are reusable and can be used as combinations in various scenarios.
- Due to use of generic and flexible architecture, OMNET++ makes a successful use in IT systems, queuing networks and hardware architectures along with communication networks.
- GUI interface and design of NED is easy to use.
- Ready to use simulation library known as INET framework.
- Provide parallel simulation support to the end users.
- Good animation and tracing support for data visualization and analysis.
- Well maintained documentation, tutorials and support from a group of dedicated experts hired for a single organization.

OMNET++ demerits

- OMNET++ is relatively new in growth and does not contain all the network models with certain features, therefore in some cases not recommended for use.
- No contributions from external organizations therefore speed of new features implementation and enhancement is slow compared to other open source network simulators.
- Free academic license has fewer features compared to commercial license.

4. EXPERIMENTS AND ANALYSIS

This chapter contains details about the experiments and analysis conducted for NS2 and NS3 network simulators. In the first part of the chapter, the performance evaluation metrics is described which is used for number of versatile experiments. Second part illustrates the details about the underlined system and software used for experiments. Third part lists all the chosen models for experiments, whereas fourth part provides details about all the experiments and analysis performed for NS2 and NS3 network simulators.

4.1. Performance Evaluation Metrics

Before proceeding further to experiments, the details about the performance evaluation matrix used in experiments are described which provides information about the different key indicators used to evaluate networking models in chosen network simulators. For the evaluation, we have considered the productivity, responsiveness, utilization, packet loss, congestion window utilization and queue management characteristics of a network. **Table 8** provides details about the performance evaluation metrics used in the experiments.

Table 8*. Performance Evaluation Metrics.*

Category	Metric	Units
Productivity	Node throughput Network throughput Packet delivery ratio	Mega bits per second (Mbps) / Kilo bytes per second (KBps) Packet delivery percentage
Responsiveness	Average end-to-end delay	Milliseconds (ms)
Utilization	Congestion Control	CWND
Losses	Packet loss	Packet lost percentage
Queue Management	Queue drop (DropTail, RED, SFQ)	Packet lost percentage

Node throughput is a measure of packets received at specific node.

Network throughput is a measure of packets received by all receiving nodes in a network.

Packet delivery ratio is a measure of a packets received at receiver compared to packets sent from source node.

$$Packet\ delivery\ ratio = \frac{No.\ of\ packet(s)\ recieved\ at\ y}{No\ .of\ packet(s)\ sent\ at\ y} \times 100 \qquad (25)$$

$Where\ x\ is\ a\ source\ node\ and\ y\ is\ a\ destination$

Average end-to-end delay is a measure of an average time difference between sending and receiving time of all the received packets at destination nodes. It is calculated as follow:

$$Average\ end-to-end\ delay = \frac{\sum_{i=0}^{n} di}{N} \times 100 \qquad (26)$$

$Where\ d\ is\ a\ delay\ for\ packet\ i\ and\ N\ is\ total\ received\ packets$

Congestion control is used by TCP protocol to utilize the channel more appropriately for all the connected nodes. It ensures the maximum utilization of a channel by gradually increasing the packet sending rates until congestion is occurred in a network which results in congestion avoidance phase. Congestion control window size is managed by source nodes which increases or decreases the window size based on channel availability.

Packet loss is a measure of finding the difference between no. of packets sent by source node and received by a destination node.

4.2. System and Software

All the experiments are performed on Linux system and latest software versions are used for NS2 and NS3. The underlying system and software details are given in **Table 9**.

***Table 9**. System and Software.*

System and Software	Details
Operating system	Linux Ubuntu 14.04 LTS Intel Core i5-4310U CPU @ 2.00GHz x 4 64-bit
NS2	ns-allinone-2.35
NS3	ns-allinone-3.21

4.3. Network models

Four different networking models are chosen including two wired and two wireless networking models. In the experiments, different scenarios are made using these models and applied various algorithms and mathematical formulas in order to get various values fulfilling the performance evaluation metrics. **Table 10** provides the details about different scenarios used in this thesis.

***Table 10**. Network Scenarios.*

Network	Scenario
Wired	• S1 Star Network • S2 Star Network and Large Simulation Time • S3 Star Network and Queue Types
Wireless	• S1 Simple Ad-hoc Network • S2 Complex Ad-hoc Network

4.4. Simulations

In the following sections, the results about all the simulation experiments with appropriate analysis are provided and executed for wired and wireless networks using NS2 and NS3 network simulators. These results are based on comparative, qualitative and mathematical analysis conducted using different mathematical formulas, communication standards and programming algorithms.

4.4.1. Wired Network

4.4.1.1. S1 Star Network

Here a local area office network (see **Figure 37**) is selected where all the client computers connect to server systems through router in a wired network.

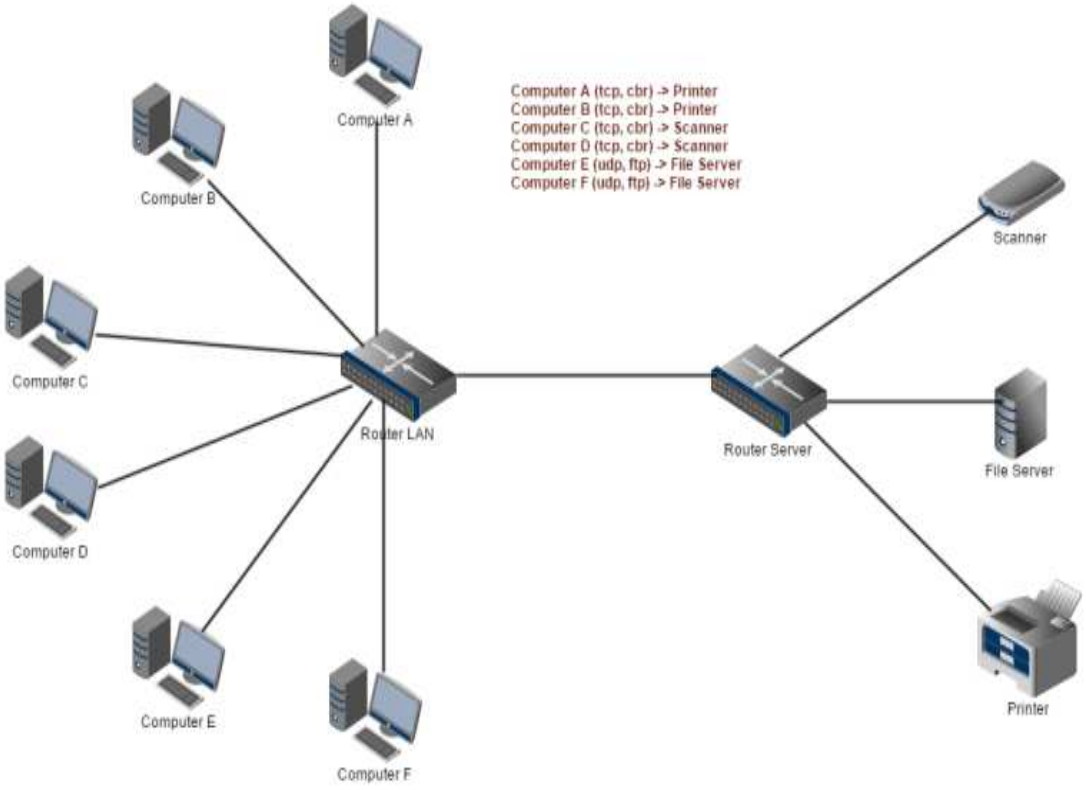

***Figure 37**. Star Office Network (point to point).*

Configurations

Star network: six computers, two routers, printer, fileserver and scanner ***Simulation time:*** 100 second ***Transport protocol:*** UDP, TCP ***Applications:*** FTP, CBR ***TCP agent:*** Newreno (e.g. Agent/TCP/Newreno etc) ***UDP agent:*** UDP (e.g. Agent/UDP etc) ***Sink:*** DelAck, Null (e.g. Agent/TCPSink/DelAck) ***TCP packet size:*** 552 ***UDP packet size:*** 512 ***Queue:*** DropTail ***Queue size:*** default 50 ***Computer (1-6):*** 1Mb 10ms ***Router (1-2):*** 5Mb 50ms ***Router2-printer:*** 1Mb 10ms ***Router2-fileserver:*** 3Mb 10ms ***Router2-scanner:*** 1Mb 10ms

Experiment execution

In this scenario, six network computers are connected to server systems including printer, scanner and file server. These computers connect to LAN router which further connects to server router. Server router connects all the network hosts to the printer, scanner and file server. Communication in a network by passes through LAN and server routers. All the computers in a network have 1Mb channel bandwidth with 10ms delay. LAN and server routers have 5Mb bandwidth and 10ms delay. Channel bandwidth for printer and scanner is 1Mb with 10ms delay whereas fileserver shares 3Mb channel bandwidth with 10ms delay.

Computer A and Computer B use TCP transmission with FTP packet generation application and connects to printer. Computer C and Computer D connects to scanner using the same TCP transmission with FTP packet generation application. Computer E and Computer F uses UDP with CBR application and connects to file server.

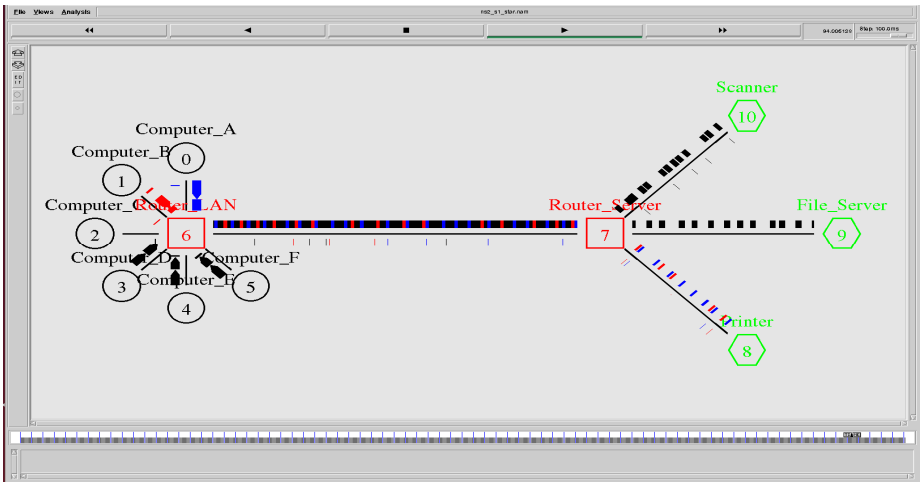

***Figure 38**. S1 Star simulation in NAM.*

In **Figure 38**, computer A, B, C and D uses TCP transmission, therefore the congestion window configurations are configured for these nodes. The congestion window threshold is 8000 and packet size is 552 bits. Packet size for computer E and F is set to 512 bits with 1.0 mb date rate.

After implementing a NS2 TCL script and NS3 C++ application program, the simulation was executed for 100 seconds.

Analysis

After the execution, we have calculated congestion window, node and network throughout, average end-to-end delay, packet delivery ratio and average packet loss by running different perl scripts for NS2 and NS3 output trace files.

Congestion Window (CWND)

First, NS2 and NS3 congestion window information for all the TCP sources in separated graphs (see **Figure 39** & **Figure 40**) is presented. These graphs show that the congestion

window for NS3 has higher values than NS2 congestion window ensuring a better utilization of channel in NS3 compared to NS2.

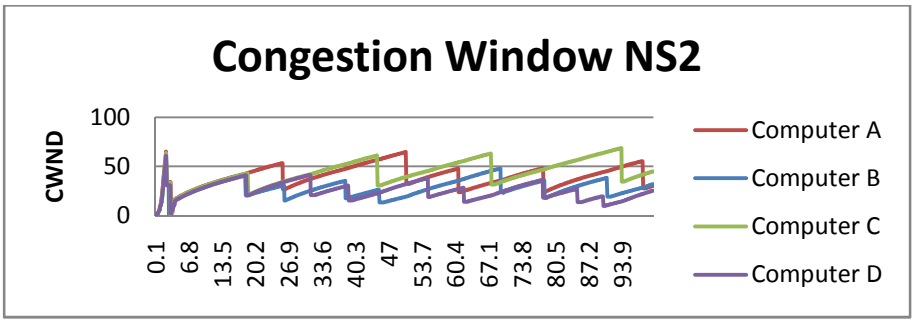

***Figure 39**. S1 Star Network (Congestion Window NS2).*

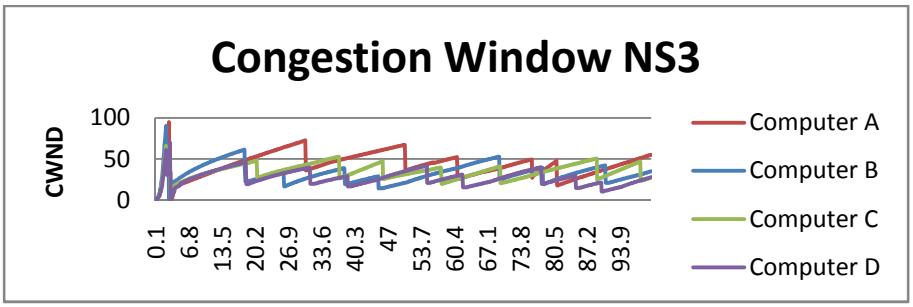

***Figure 40**. S1 Star Network (Congestion Window NS3).*

For node by node analysis, the information for each node is combined into separate graphs for NS2 and NS3 and results are shown in **Figure 41**.

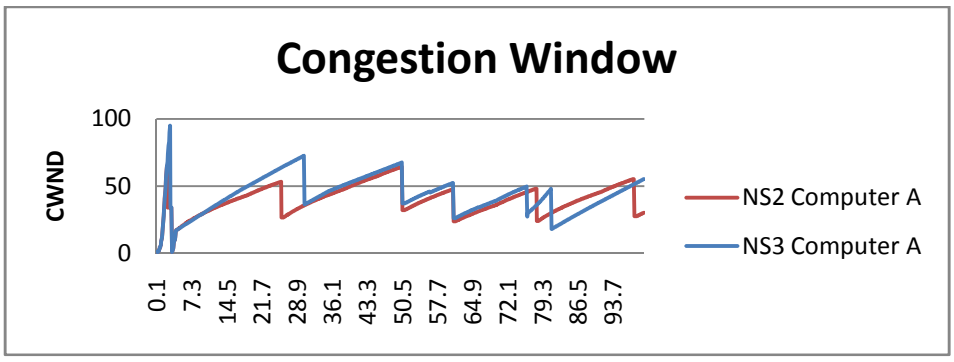

***Figure 41**. S1 Star Network (Congestion Window NS2, NS3) – Computer A.*

Comparison for Computer A for NS2 and NS3 shows that NS3 congestion window increases more efficiently than NS2 which is an indication of better channel utilization in NS3 compared to NS2.

Comparison for other computers shows similar results as Computer A in the following graphs (see **Figure 42**, **Figure 43** and **Figure 44**).

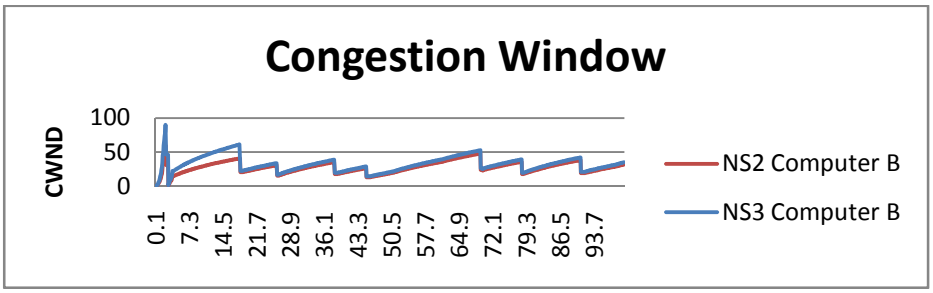

***Figure 42**. S1 Star Network (Congestion Window NS2, NS3) – Computer B.*

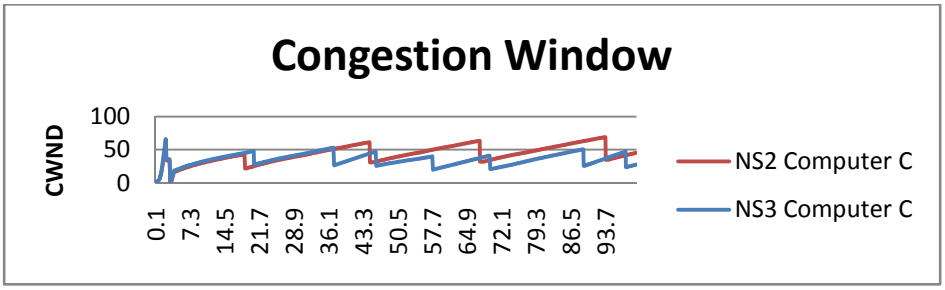

***Figure 43**. S1 Star Network (Congestion Window NS2, NS3) – Computer C.*

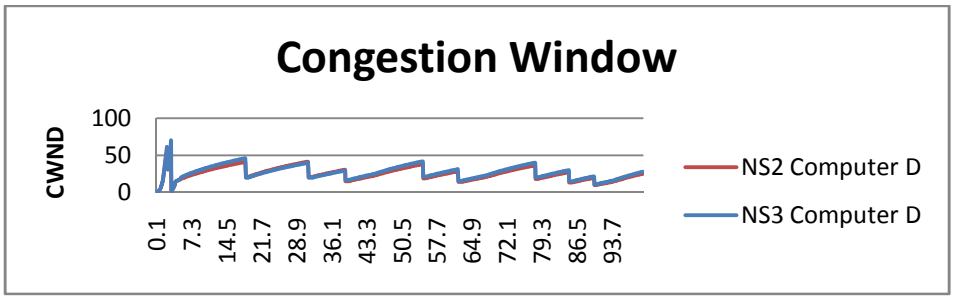

***Figure 44**. S1 Star Network (Congestion Window NS2, NS3) – Computer D.*

Throughput

In this experiment, node and network throughput is calculated for all the receiving nodes in NS2 and NS3 simulation. Node throughput is observed for Router LAN, Router Server, Printer, File Server and Scanner, whereas network throughput is calculated for the whole network and compared for NS2 and NS3. Mean and standard deviation for network throughout are also calculated.

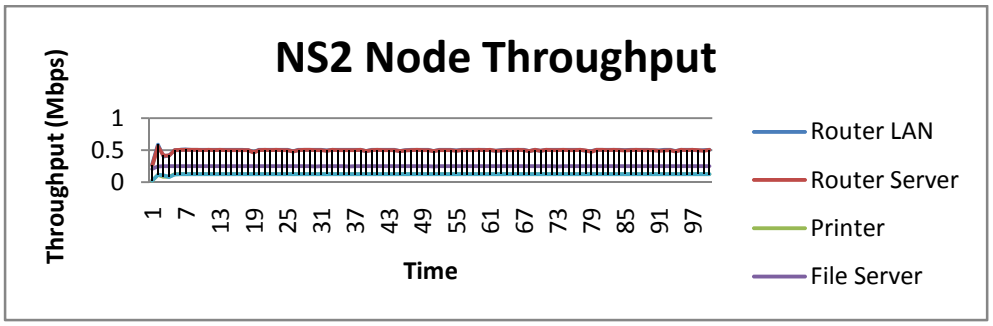

***Figure 45**. S1 Star NS2 Node throughput.*

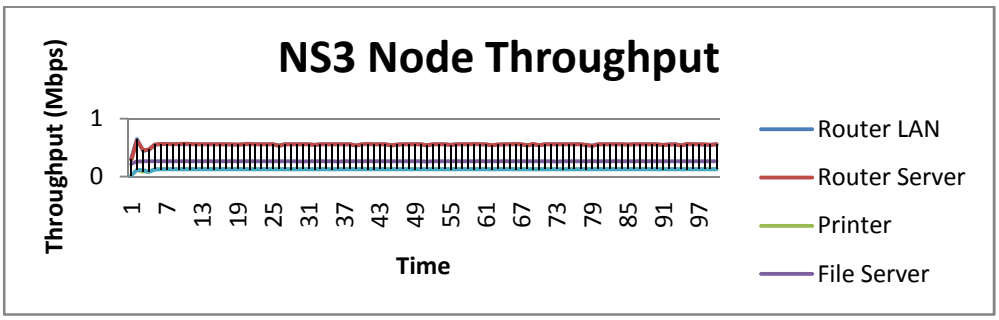

***Figure 46**. S1 Star NS3 Node Throughput.*

NS2 and NS3 node throughput shows symmetry in both graphs given in **Figure 45** and **Figure 46** for each node. However, NS3 node throughput has high figures compared to NS3. In the graph, the maximum throughput can be seen at both routers. The maximum throughput is further divided among all server nodes. In NS2, the throughput for routers is around .050 Mbps whereas in NS3, it is approximately 0.56 Mbps. In the simulation, UDP with CBR applications are used for File Server connections and TCP with FTP application

are used for Printer and Scanner connections. CBR rate was high around 1.0 Mbps and data packet size was 512 which resulted in high throughput at File Server. On the other hand FTP data rate was based on TCP congestion window with packet size 552 which resulted in low throughput at Printer and Scanner.

For comparison between NS2 and NS3, the network throughput with mean and standard deviation is calculated and results are shown in the **Figure 47**, **Figure 48** and **Table 11**.

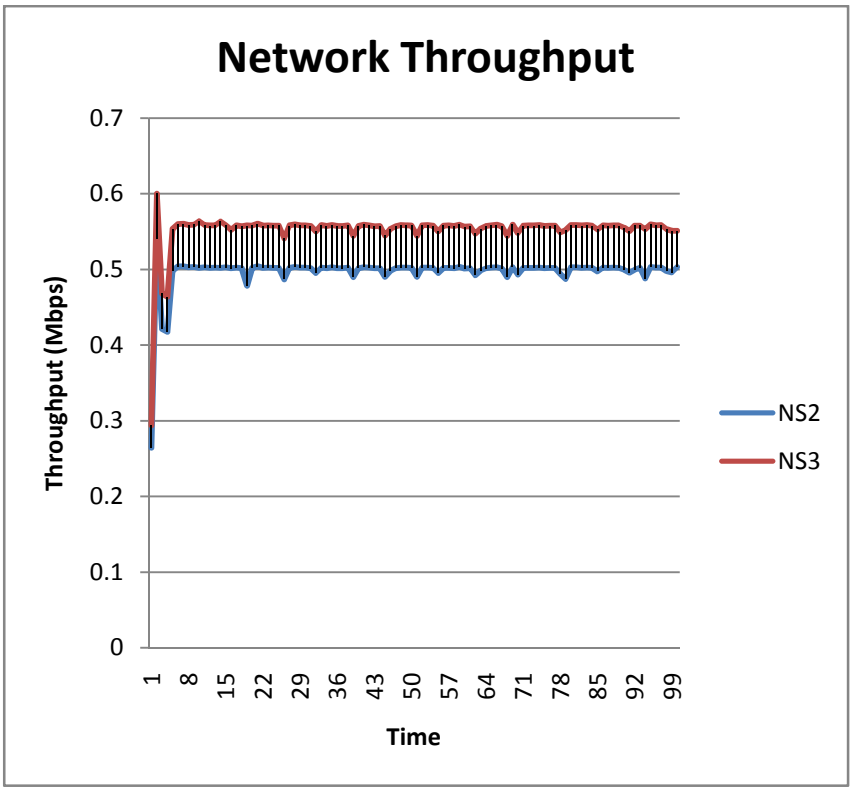

Figure 47. *S1 Star Network Throughput Comparison.*

Table 11. *S1 Star Network Throughput Mean and Standard Deviation.*

	Time count	Throughput μ (Mbps)	Throughput σ(Mbps)
NS2	100	0.50675051	0.10414388
NS3	100	0.56155041	0.08715246

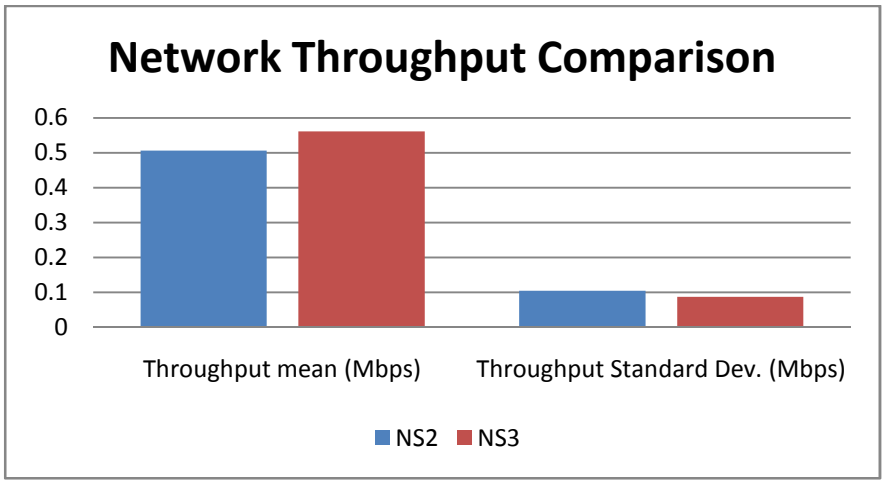

***Figure 48**. S1 Star Network Throughput Mean and Standard Deviation.*

Network throughput mean is higher for NS3 compared to NS2 which shows that NS3 provides more efficiency in transmitting data and allow better throughput for the underlying network. At the same time, standard deviation for NS3 is lower than the NS2 and confirms that NS3 network throughput has fewer variations and provides smooth communications as compared to NS2.

Average End-to-End Delay

In a simulation, the average time elapsed between data sending and receiving for all the TCP and UDP packets is calculated and compared the results for NS2 and NS3 (see **Figure 49**). Results show that TCP packets consume more time compared to UDP packets in order to reach to the destination. In NS2, average end-to-end delay for TCP packets is 245.23 milliseconds and for UDP packets, it is 77.29 milliseconds. If the average end-to-end delay is combined for TCP and UDP then the average end-to-end delay for whole network is 161.26 milliseconds. In NS3, the average end-to-end delay for TCP packets is 168.25 and for UDP, it is 35.56. When combining TCP and UDP then average end-to-end delay for an underlined network is 101.91. In comparison, NS2 has higher figures compare to NS3 for

average end-to-end delay which makes NS3 better in terms of network delay compared to NS2.

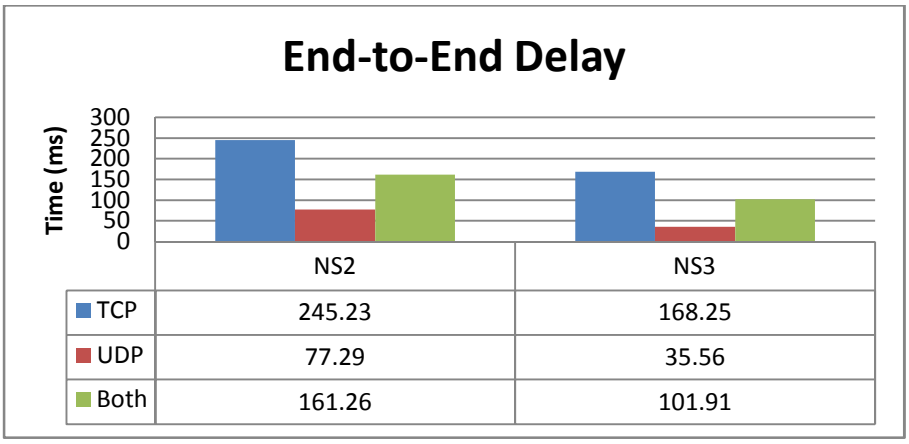

***Figure 49**. S1 Star End-to-End Delay for NS2, NS3.*

Packet Delivery Ratio

In the experiments, the packet delivery ratio is measured for NS2 and NS3 and **Figure 50** shows the results:

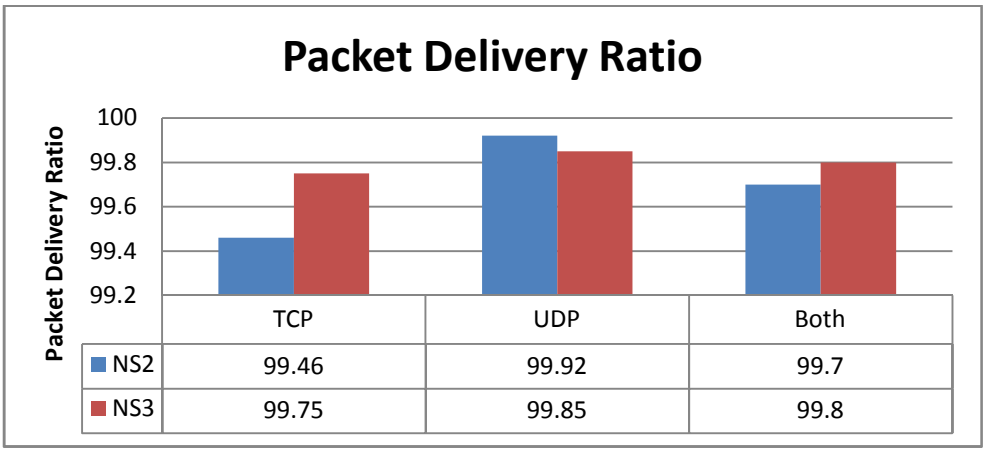

***Figure 50**. S1 Star Packet Deliver Ratio for NS2, NS3.*

Packet delivery ratio for TCP packets in NS3 is enhanced compared to NS2 while packet delivery ratio for UDP packets in NS3 lack behind NS2 UDP packets. But as a whole network, NS3 provides high packet delivery ratio 99.8% compared to NS2 99.7%. However the difference is very small and findings are concluded that both NS2 and NS3 performance in terms of packet delivery ratio is at same level.

Packet Loss

In comparison for packet loss (see **Figure 51**) in TCP and UDP packets, NS2 looses more TCP packets than UDP packets compared to NS3. However, as a whole packet loss in NS2 is noticeable compared to NS3 which provides minimum packet loss percentage.

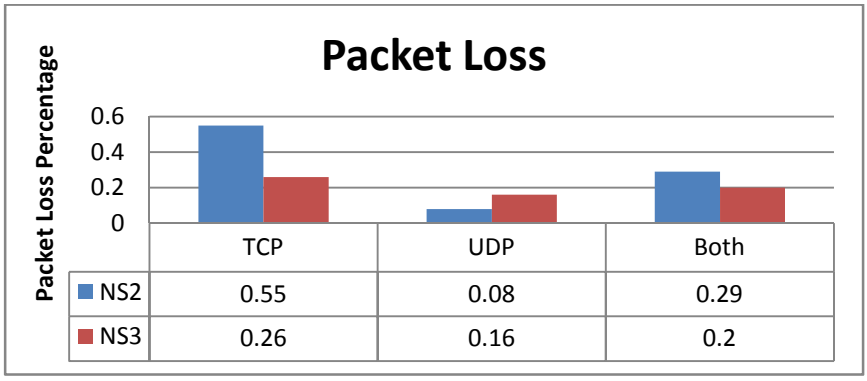

***Figure 51**. S1 Star Packet Loss for NS2, NS3.*

In S1 Star experiment, NS3 produced enhanced and efficient results for all the performance evaluation metrics compared to NS2.

4.4.1.2. S2 Star Network and Large Simulation Time

In this experiment, S1 Star office LAN network is simulated for large simulation time and observed various performance evaluation metrics in details. Here, the most important details are presented about the outcome for NS2 and NS3.

Configurations

Star network: six computers, two routers, printer, fileserver and scanner ***Simulation time:*** 500 second ***Transport protocol:*** UDP, TCP ***Applications:*** FTP, CBR ***TCP agent:*** Newreno (e.g. Agent/TCP/Newreno etc) ***UDP agent:*** UDP (e.g. Agent/UDP etc) ***Sink:*** DelAck, Null (e.g. Agent/TCPSink/DelAck) ***TCP packet size:*** 552 ***UDP packet size:*** 512 ***Queue:*** DropTail ***Queue size:*** default 50 ***Computer (1-6):*** 1Mb 10ms ***Router (1-2):*** 5Mb 50ms ***Router2-printer:*** 1Mb 10ms ***Router2-fileserver:*** 3Mb 10ms ***Router2-scanner:*** 1Mb 10ms

Experiment execution

In this scenario, six network computers are connected to server systems including printer, scanner and file server. These computers connect to LAN router which further connects to server router. Server router connects all the network hosts to the printer, scanner and file server. Communication in a network by passes through LAN and server routers. All the computers in a network have 1Mb channel bandwidth with 10ms delay. LAN and server routers have 5Mb bandwidth and 10ms delay. Channel bandwidth for printer and scanner is 1Mb with 10ms delay whereas fileserver shares 3Mb channel bandwidth with 10ms delay.

Computer A and Computer B use TCP transmission with FTP packet generation application and connects to printer. Computer C and Computer D connects to scanner using same TCP transmission with FTP packet generation application. Computer E and Computer F uses UDP with CBR application and connects to file server.

Computer A, B, C and D uses TCP transmission, therefore the congestion window configurations are implemented for these nodes where congestion window threshold is

8000 with packet size 552 bits. Packet size for Computer E and F is set to 512 bits with 1.0 mb date rate.

After implementing a NS2 TCL script and NS3 C++ application program, the simulation was executed for 500 seconds.

Analysis

After the execution, the congestion window, node and network throughout, average end-to-end delay, packet delivery ratio and average packet loss are calculated by running different perl scripts over NS2 and NS3 output trace files. As the network model is same as of S1 Star, few of the important results are presented.

Congestion Window (CWND)

In **Figure 52** and **Figure 53**, congestion window is compared for Computer A and B in NS2 and NS3 which shows that congestion window for nodes in NS3 sometimes touches 70 while the maximum boundary observed for NS2 nodes is approximately close to 60.

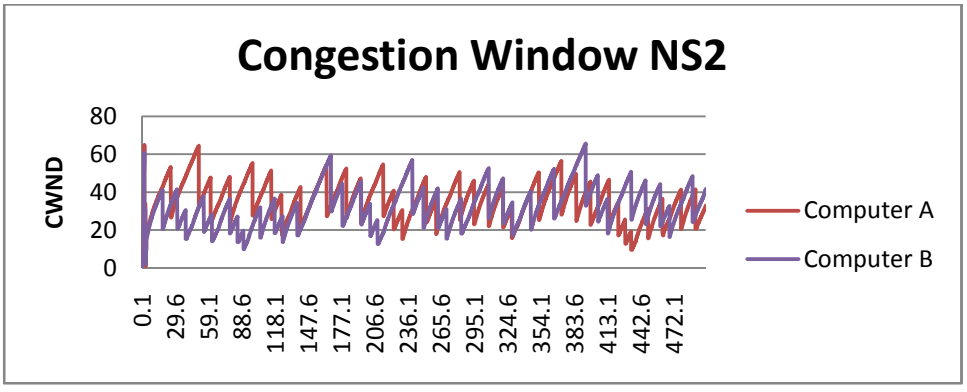

***Figure 52**. S2 Star Network (Congestion Window NS2).*

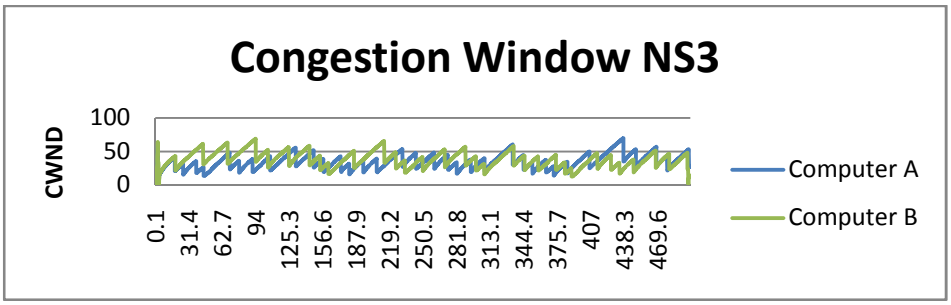

***Figure 53**. S2 Star Network (Congestion Window NS3).*

Comparison for Computer A between NS2 and NS3 shows that the NS3 congestion window increases more efficiently than NS2 which is an indication of a better channel utilization in NS3 compared to NS2. **Figure 54** provides node by node comparison.

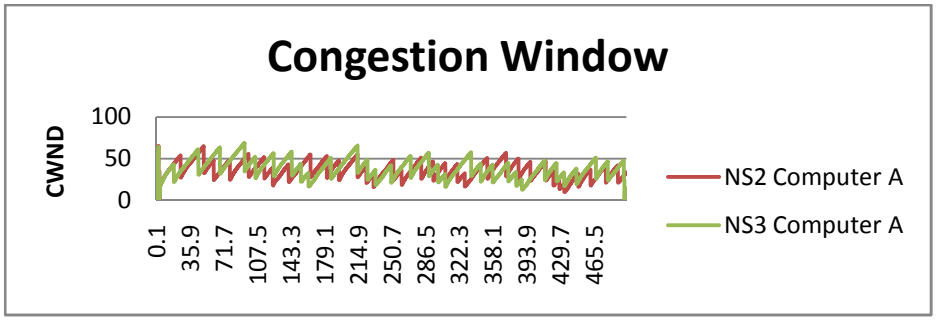

***Figure 54**. S2 Star Network (Congestion Window NS2, NS3) – Computer A.*

Comparison for other computers (see **Figure 55**) shows similar results as of Computer A in the following graphs.

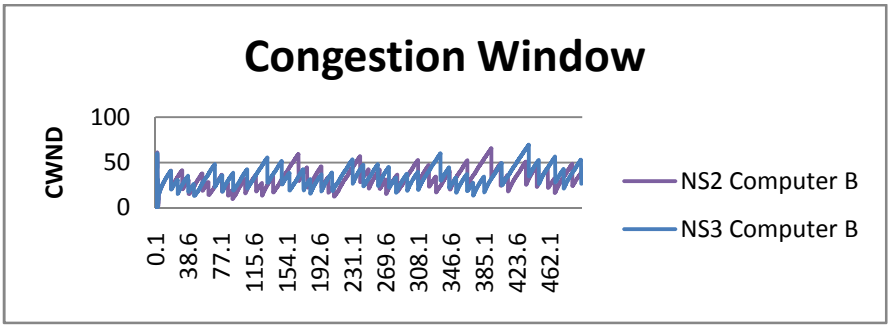

***Figure 55**. S2 Star Network (Congestion Window NS2, NS3) – Computer B.*

Considering the congestion window graphs, it is concluded that the congestion window in NS3 provides better utilization of the channel compared to NS2 even with the large simulation time experiments.

Throughput

In this experiment, the network throughput with mean and standard deviation is compared for NS2 and NS3. The network throughput comparison shows that NS3 performs well compared to NS2 for a large simulation time. Here the network throughput for NS2 is approximately 0.2 times higher than NS2. Mean and standard deviation of the network throughput are also measured.

Figure 56 illustrates results of network throughput in NS2 and NS3.

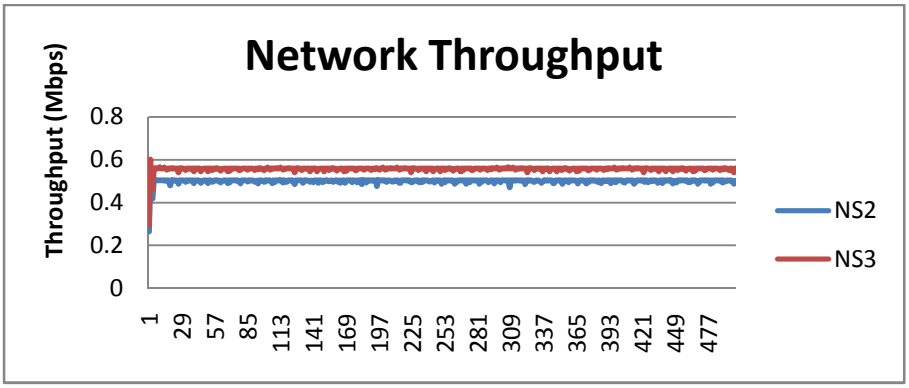

***Figure 56**. S2 Star Network Throughput Comparison.*

In **Table 12** and **Figure 57**, Network throughput mean is higher for NS3 compared to NS2 which shows that NS3 provides more efficiency in transmitting data for large simulation time and allow better throughput for underlined network. At the same time, the standard deviation for NS3 is lower than the NS2 and confirms that NS3 network throughput has fewer variations and provides smooth communications compared to NS2.

***Table 12**. S2 Star Network Throughput Mean and Standard Deviation.*

	Time count	Throughput μ (Mbps)	Throughput σ(Mbps)
NS2	500	0.501366	0.096775
NS3	500	0.561552	0.062542

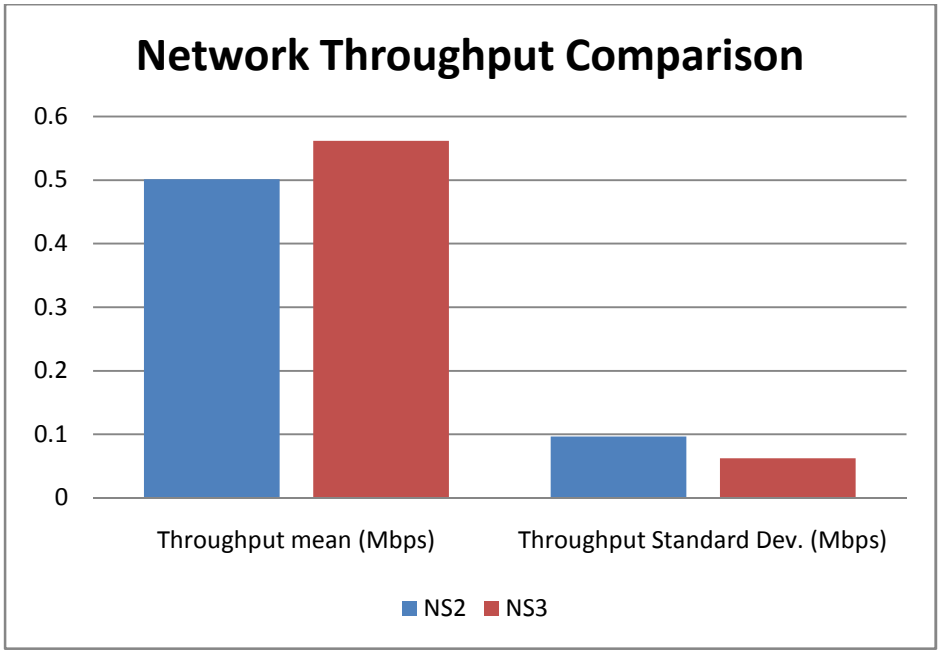

***Figure 57**. S1 Star Network Throughput Mean and Standard Deviation.*

Average End-to-End Delay

In the experiment, the average time elapsed for TCP and UDP data transmission is calculated and compared for NS2 and NS3 (see **Figure 58**). The results show that with the large simulation time, average end-to-end delay 100.92 *ms* is improved for NS3 compared to small simulation time 101.91 *ms* in the previous experiment. On the other hand, the average end-to-end delay for large simulation time has been increased to 163.18 *ms* compared to the small simulation time which was 161.26 *ms* previous experiment.

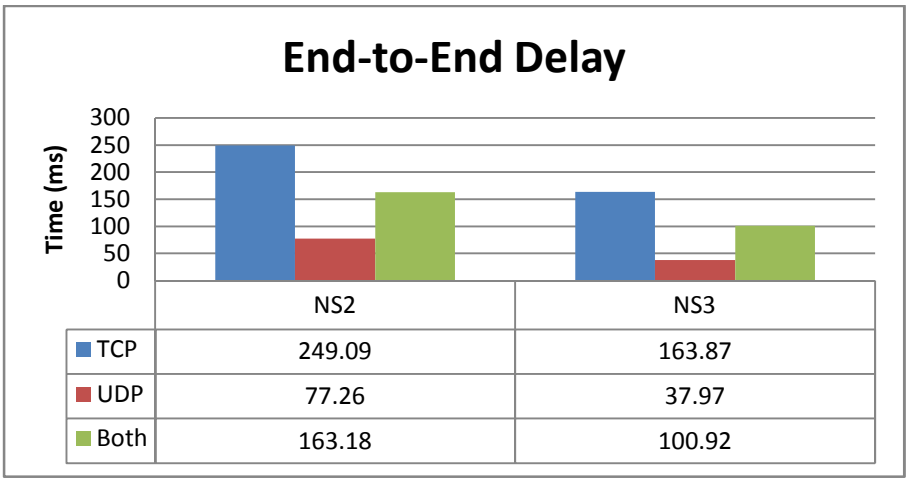

***Figure 58**. S2 Star End-to-End Delay for NS2, NS3.*

Packet Delivery Ratio

In the experiments, the packet delivery ratio is calculated for NS2 and NS3. With a large simulation time, the packet delivery ratio for TCP packets in NS3 is enhanced compared to NS2 while the packet delivery ratio for UDP packets in NS3 lack behind the UDP packets in NS2. But as a whole network, NS3 provides high packet delivery ratio 99.9% compared to NS2 99.83%. However the difference is very small and it is concluded that both NS2 and NS3 performance in terms of the packet delivery ratio is at same level. **Figure 59** shows the results for the packet delivery ratio.

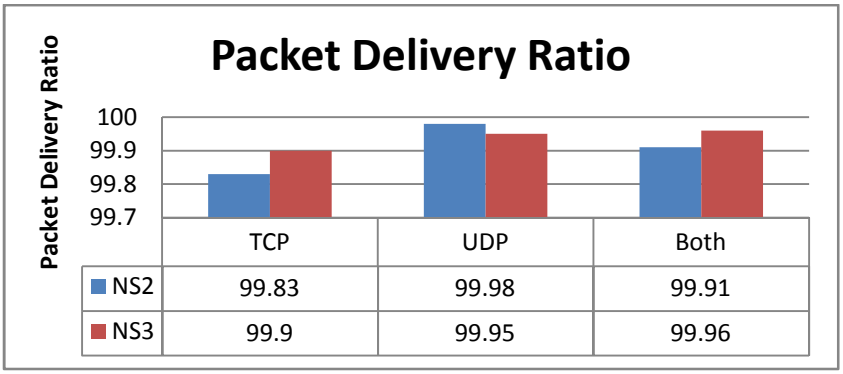

***Figure 59**. S2 Star Packet Deliver Ratio for NS2, NS3.*

Packet Loss

In comparison for the packet loss in TCP and UDP packets, NS2 looses more TCP packets than UDP packets compared to NS3. However, as a whole, the packet loss in NS2 is noticeable compared to NS3 which provides a minimum packet loss percentage. The comparison for a packet loss is given in **Figure 60**.

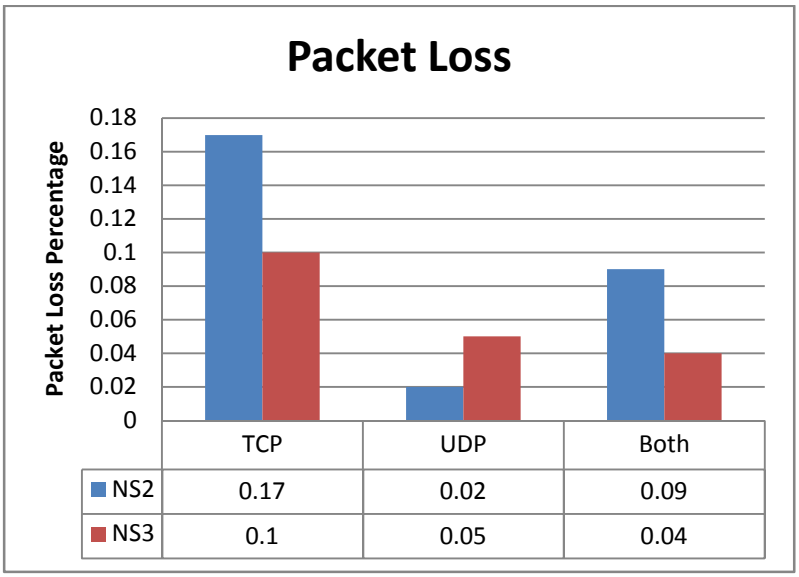

***Figure 60**. S2 Star Packet Loss for NS2, NS3.*

With a large simulation time in S2 Star Network experiment, NS3 produced enhanced and efficient results for all the performance evaluation metrics compared to NS2.

4.4.1.3. S3 Star Network and Queue Types

In this experiment, S1 Star office LAN network is simulated for different queue types available in NS2 and NS3. These queue types include DropTail, RED and SFQ. For the analysis, the congestion control window and packet loss are compared for all source nodes in a network based on each queue type.

Configurations

Star network: six computers, two routers, printer, fileserver and scanner ***Simulation time:*** 100 second ***Transport protocol:*** UDP, TCP ***Applications:*** FTP, CBR ***TCP agent:*** Newreno (e.g. Agent/TCP/Newreno etc) ***UDP agent:*** UDP (e.g. Agent/UDP etc) ***Sink:*** DelAck, Null (e.g. Agent/TCPSink/DelAck) ***TCP packet size:*** 552 ***UDP packet size:*** 512 ***Queue:*** DropTail | RED | SFQ ***Queue size:*** default 50 ***Computer (1-6):*** 1Mb 10ms ***Router (1-2):*** 5Mb 50ms ***Router2-printer:*** 1Mb 10ms ***Router2-fileserver:*** 3Mb 10ms ***Router2-scanner:*** 1Mb 10ms

Experiment execution

In this scenario, six network computers are connected to server systems including printer, scanner and file server. These computers connect to LAN router which further connects to server router. Server router connects all the network hosts to the printer, scanner and file server. Communication in a network travels through LAN and server routers. All the computers in a network have 1Mb channel bandwidth with 10ms delay. LAN and server routers have 5Mb bandwidth and 10ms delay. Channel bandwidth for printer and scanner is 1Mb with 10ms delay whereas fileserver shares 3Mb channel bandwidth with 10ms delay.

Computer A and Computer B use TCP transmission with a FTP packet generation application and connects to printer. Computer C and Computer D connects to a scanner using same TCP transmission with a FTP packet generation application. Computer E and Computer F uses UDP with a CBR application and connects to the file server.

Computer A, B, C and D uses TCP transmission, therefore the congestion window is configured for these nodes. The congestion window threshold is 8000 and the packet size is 552 bits. The packet size for Computer E and F is set to 512 bits with 1.0 mb date rate.

For the experiment, three different queue types such as DropTail, RED and SFQ are used to observe behavior of underlying network.

After implementing a NS2 TCL script and NS3 C++ application program, the simulation was executed for 100 seconds.

Analysis

After the execution, the congestion window and packet loss are calculated for each TCP source node. As the network model is same as S1 Star, results are provided (see **Figure 61** to **Figure 66**) only for Computer A and Computer B using DropTail, RED and SFQ in NS2 and NS3.

Congestion Window (CWND)

DropTail

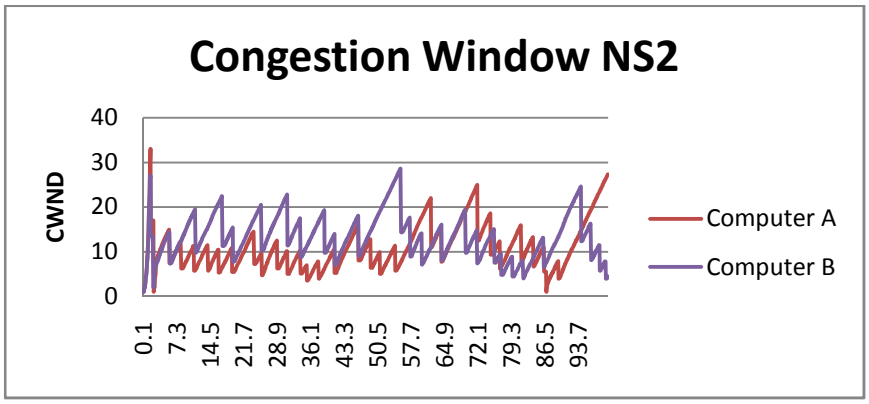

Figure 61. *S3 Star Network (DropTail Congestion Window NS2).*

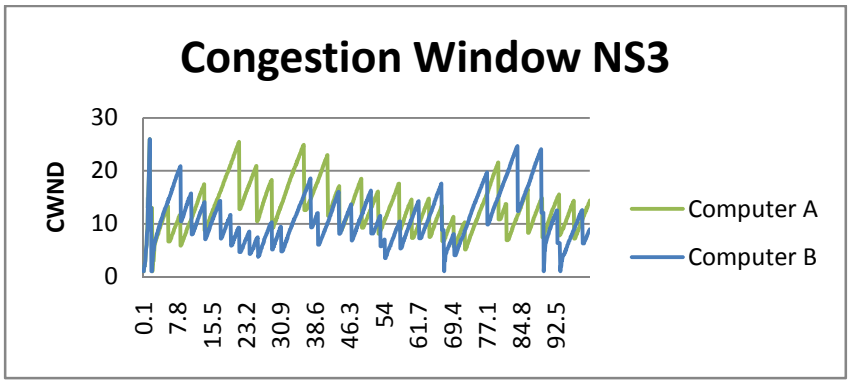

Figure 62. *S3 Star Network (DropTail Congestion Window NS3).*

RED

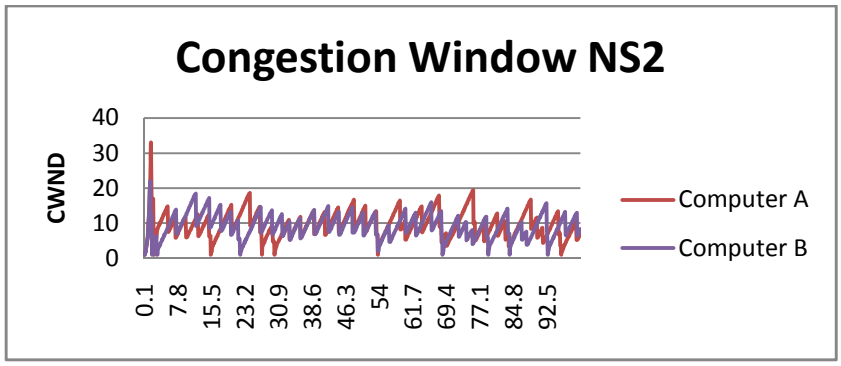

Figure 63. *S3 Star Network (RED Congestion Window NS2).*

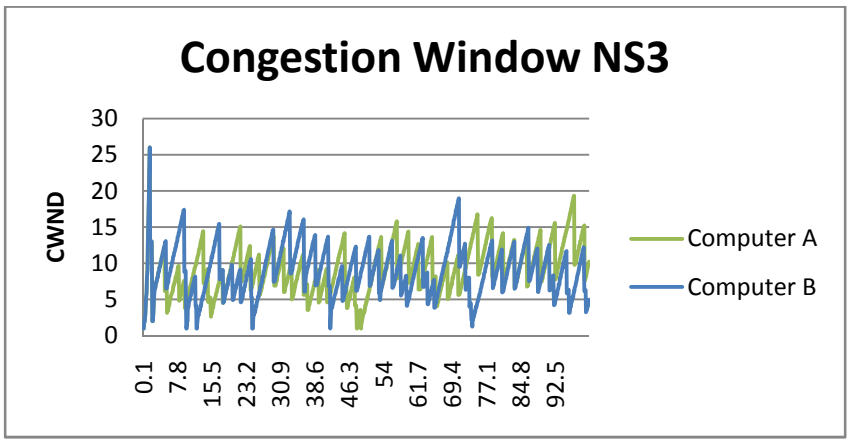

Figure 64. *S3 Star Network (RED Congestion Window NS2).*

SFQ

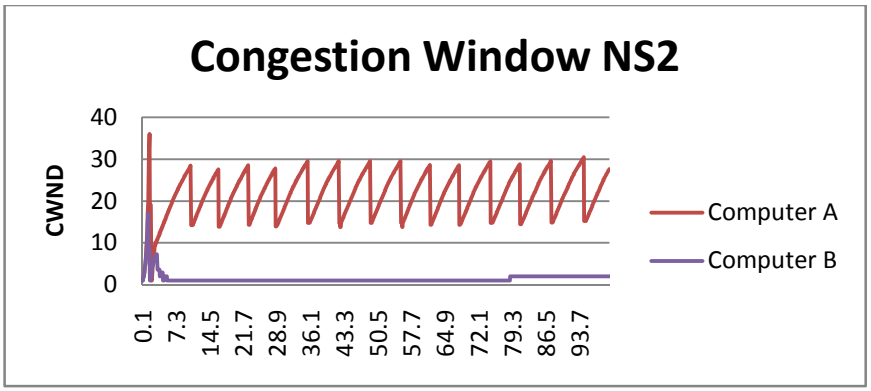

***Figure 65**. S3 Star Network (SFQ Congestion Window NS2).*

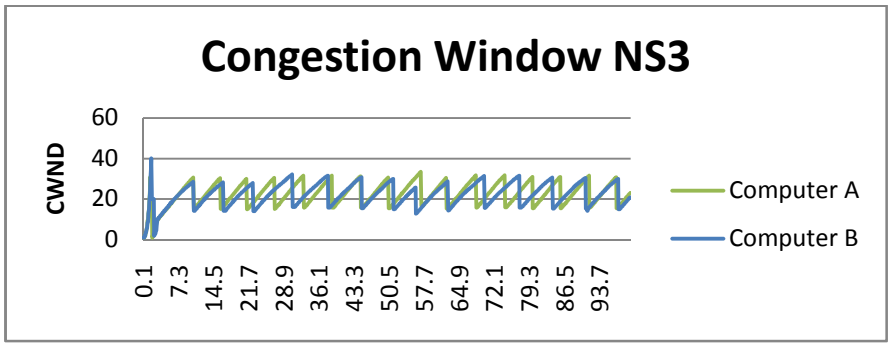

***Figure 66**. S3 Star Network (SFQ Congestion Window NS3).*

In this experiment, the congestion control window is compared for DropTail, RED and SFQ and the results show that SQF is a fair queue which handles queue fairly among all the connected TCP source nodes. DropTail is proved to be an unfair queue, which handles packets from different source nodes in different ways. In case of congestion occurrence, it always drops the packet from a tail of the queue causing blockage for a specific node data for a long duration. The results shows that congestion window with RED queue is slightly fair than DropTail where congestion window for different nodes have symmetry among all the nodes.

In comparison with NS2 and NS3, SFQ is a best queue for handling network traffic. However, sometimes it behaves strangely with one or more nodes where the congestion window remains under a minimum congestion control threshold. On the other side, SFQ increases the congestion window fairly and efficiently for all the other connected nodes which results in an improved network performance by all means.

Packet Loss

For the queue analysis, packet loss is computed for each source node in a network using DropTail, RED and SFQ queue types. **Table 13** shows the computed results for NS2 and NS3:

Table 13*. S3 Star Packet Loss Results for DropTail, RED & SFQ.*

DropTail	NS2						
		Comp. A	Comp. B	Comp. C	Comp. D	Comp. E	Comp. F
	Packets send	14445	14673	17390	17882	73120	72727
	Packets drop	46	46	39	44	7964	16331
	Packet drop %	0.32	0.31	0.22	0.25	10.89	22.45
	NS3						
		Comp. A	Comp. B	Comp. C	Comp. D	Comp. E	Comp. F
	Packets send	15252	17252	14952	15247	83524	83495
	Packets drop	42	50	37	27	6055	16073
	Packet drop %	0.28	0.29	0.25	0.18	7.25	19.25
RED	NS2						
		Computer A	Computer B	Computer C	Computer D	Computer E	Computer F
	Packets send	15640	14660	14973	14982	72939	72917
	Packets drop	56	53	62	62	10784	13560
	Packet drop %	0.36	0.36	0.41	0.42	14.79	18.6
	NS3						
		Comp. A	Comp. B	Comp. C	Comp. D	Comp. E	Comp. F
	Packets send	16242	16298	16324	15834	83524	83495
	Packets drop	50	47	57	67	11426	13810
	Packet drop %	0.31	0.29	0.35	0.42	13.68	16.54

SFQ	NS2						
		Comp. A	Comp. B	Comp. C	Comp. D	Comp. E	Comp. F
	Packets send	30530	324	31152	14982	65880	65915
	Packets drop	31	30	30	62	12172	12173
	Packet drop %	0.1	9.26	0.1	0.41	18.48	18.47
	NS3						
		Comp. A	Comp. B	Comp. C	Comp. D	Comp. E	Comp. F
	Packets send	32621	32542	32492	31985	67524	67622
	Packets drop	33	33	32	67	11168	11043
	Packet drop %	0.1	0.1	0.1	0.21	16.54	16.33

In the following section, the results are visualized with respect to different queue types and network simulators:

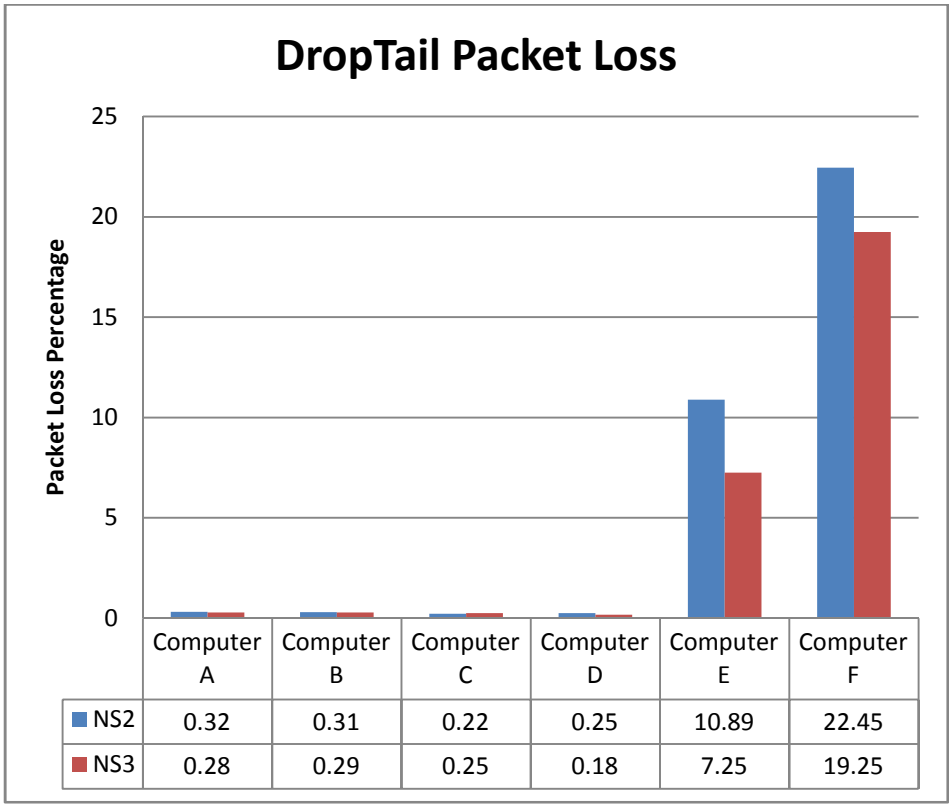

***Figure 67**. S3 Star DropTail Packet loss for NS2, NS3.*

In a DropTail (see **Figure 67**), Computer F is impacted mostly and encountered higher packet loss in NS2 and NS3. Packet loss for Computer E is relatively less than Computer A. The difference in the packet loss between Computer E and F is approximately 12% in NS2 and NS3.

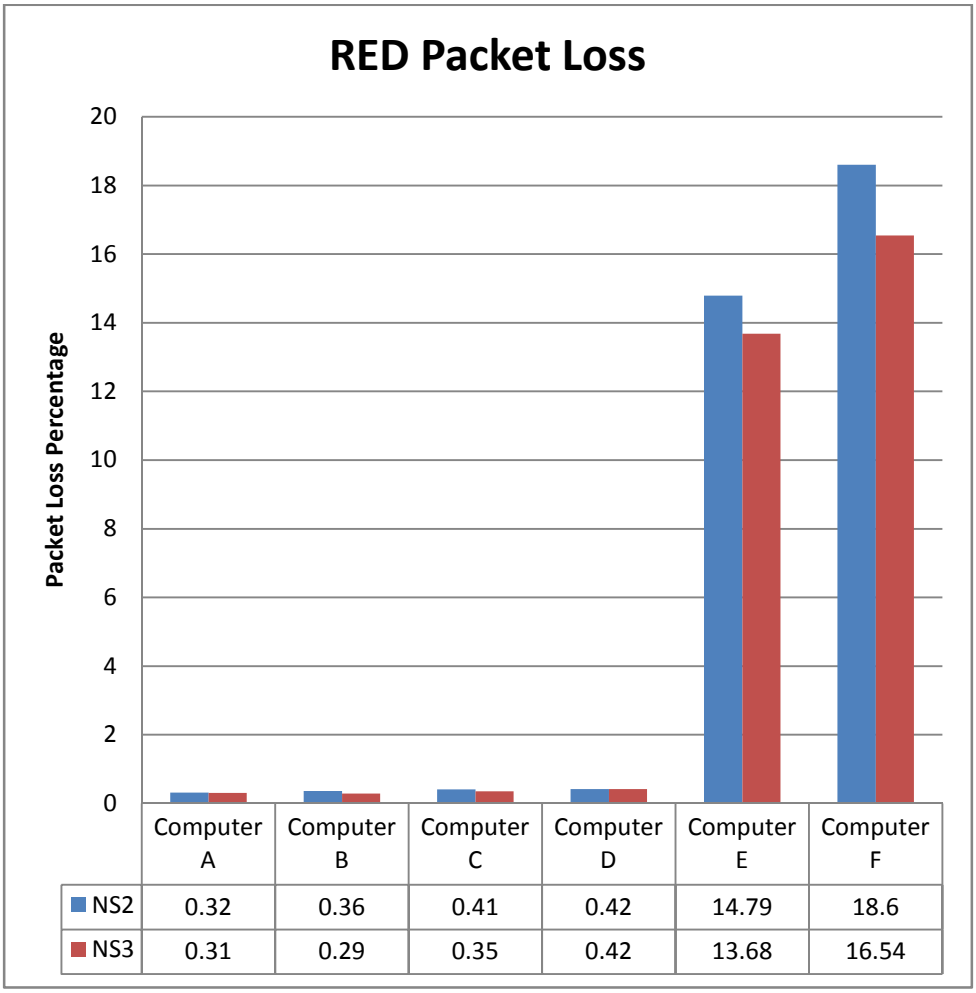

Figure 68. *S3 Star RED Packet loss for NS2, NS3.*

Using a RED queue (see **Figure 68**), the packet loss is relatively small compared to a DropTail. The packet loss difference for Computer E and F is reduced to 4% in NS2 and NS3 but still it is still not reasonable.

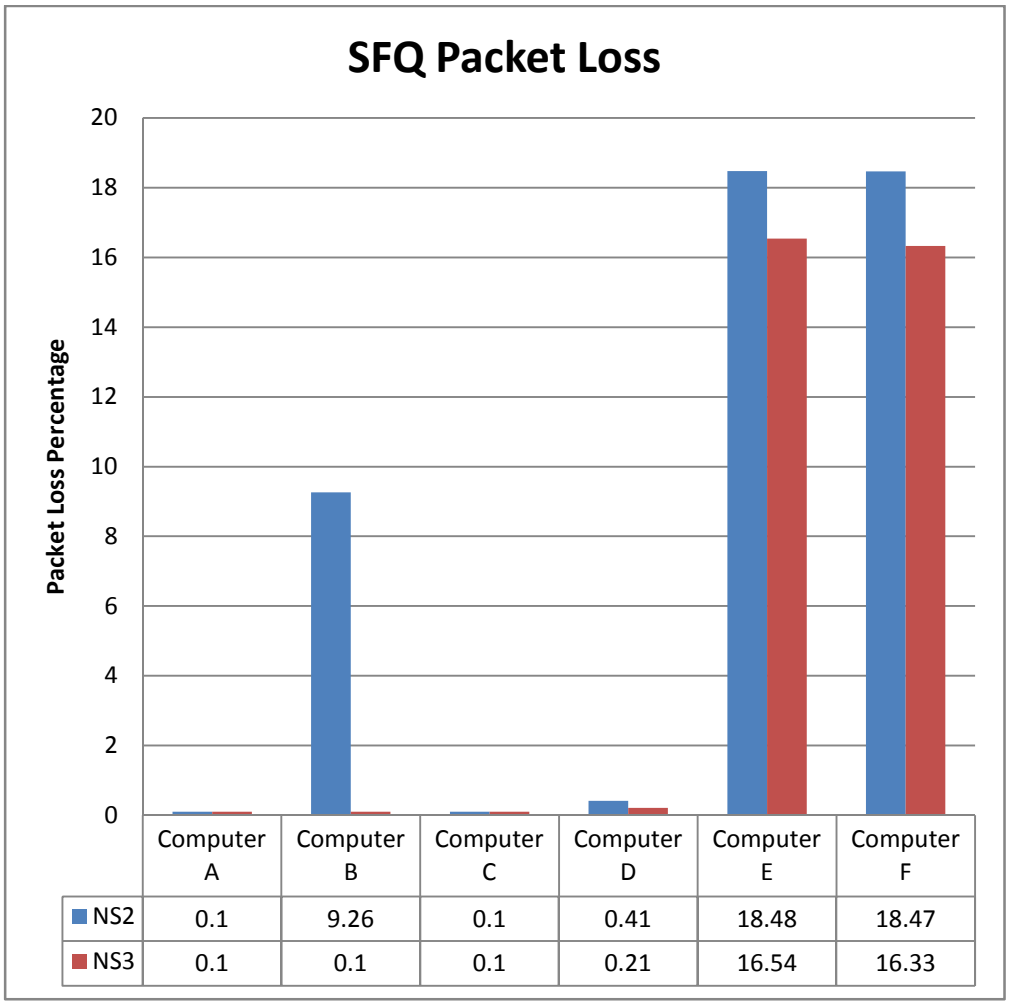

Figure 69. *S3 Star SFQ Packet loss for NS2, NS3.*

In **Figure 69**, SFQ has high improvements in terms of reduced differences in the packet loss, especially for Computer E and F in NS2 and NS3.

Considering the figures and graphs, it is concluded that SFQ is proved to be an impressive queue type and provides fairness in network simulation. However, sometimes it behaves strangely in NS2 and drops packets for a certain node all the time. Example of such node is Computer B in the NS2 simulation which has a very high packet loss percentage compared to the other nodes in a network.

4.4.2. Wireless Networks

4.4.2.1. S1 Simple Office Ad-hoc Network

In this experiment, a simple wireless office ad-hoc network consisting of four nodes connected on ad hoc basis is selected and implemented. These nodes are either laptops or mobile devices as shown in **Figure 70**.

***Figure 70**. S1 Simple Office Ad-hoc Network.*

Configurations

a) *AODV with TwoRayGround*

```
# Define options
set val(chan)           Channel/WirelessChannel;        # channel type
set val(prop)           Propagation/TwoRayGround;   # radio-propagation model
set val(netif)          Phy/WirelessPhy;                # network interface type
set val(mac)            Mac/802_11;                     # MAC type
set val(ifq)            Queue/DropTail/PriQueue;        # interface queue type
set val(ll)             LL;                             # link layer type
```

```
set val(ant)          Antenna/OmniAntenna;          # antenna model
set val(ifqlen)       50;                           # max packet in ifq
set val(nodeCount)    4;                            # number of mobile nodes
set val(rp)           AODV;                         # routing protocol
set val(x)            500;                          # X dimension of topography
set val(y)            400;                          # Y dimension of topography
set val(stop)         20;                           # time of simulation end
```

b) AODV with FreeSpace

```
# Define options
set val(chan)         Channel/WirelessChannel;      # channel type
set val(prop)         Propagation/FreeSpace;        # radio-propagation model
set val(netif)        Phy/WirelessPhy;              # network interface type
set val(mac)          Mac/802_11;                   # MAC type
set val(ifq)          Queue/DropTail/PriQueue;      # interface queue type
set val(ll)           LL;                           # link layer type
set val(ant)          Antenna/OmniAntenna;          # antenna model
set val(ifqlen)       50;                           # max packet in ifq
set val(nodeCount)    4;                            # number of mobile nodes
set val(rp)           AODV;                         # routing protocol
set val(x)            500;                          # X dimension of topography
set val(y)            400;                          # Y dimension of topography
set val(stop)         20;                           # time of simulation end
```

c) DSDV with TwoRayGround

```
# Define options
set val(chan)         Channel/WirelessChannel;      # channel type
set val(prop)         Propagation/TwoRayGround;     # radio-propagation model
set val(netif)        Phy/WirelessPhy;              # network interface type
set val(mac)          Mac/802_11;                   # MAC type
set val(ifq)          Queue/DropTail/PriQueue;      # interface queue type
set val(ll)           LL;                           # link layer type
set val(ant)          Antenna/OmniAntenna;          # antenna model
set val(ifqlen)       50;                           # max packet in ifq
set val(nodeCount)    4;                            # number of mobile nodes
set val(rp)           DSDV;                         # routing protocol
set val(x)            500;                          # X dimension of topography
set val(y)            400;                          # Y dimension of topography
set val(stop)         20;                           # time of simulation end
```

d) DSR with TwoRayGround

```
# Define options
set val(chan)           Channel/WirelessChannel;          # channel type
set val(prop)           Propagation/TwoRayGround;  # radio-propagation model
set val(netif)          Phy/WirelessPhy;                  # network interface type
set val(mac)            Mac/802_11;                       # MAC type
set val(ifq)             CMUPriQueue;                     # interface queue type
set val(ll)             LL;                               # link layer type
set val(ant)            Antenna/OmniAntenna;              # antenna model
set val(ifqlen)         50;                               # max packet in ifq
set val(nodeCount)      4;                                # number of mobilenodes
set val(rp)             DSR;                              # routing protocol
set val(x)              500;                              # X dimension of topography
set val(y)              400;                              # Y dimension of topography
set val(stop)           20;                               # time of simulation end
```

Experiment execution

In the first experiment, network consists of four wireless node including two laptops and two mobile phones. These nodes were connected using TCP protocol with FTP applications. Laptop A was connected to Laptop B and Mobile A was connected to Mobile B through WLAN IEEE 802.11 network. All the nodes in a network have mobility characteristics. The initial positions of the nodes were defined at time 0.0 and all nodes were placed at a distance from each other. At time 0.1, 0.2, 0.3 and 0.4 Laptop A, Mobile A, Mobile B and Laptop B started to move to a new destination point respectively. In the middle of the simulations, both source laptop and mobile nodes became close to receiver nodes and then moved away again gradually. The simulation area was set to 500 X 400 and nodes moved within the range of 20 seconds. **Figure 71** shows S1 simple office ad hoc network in NAM.

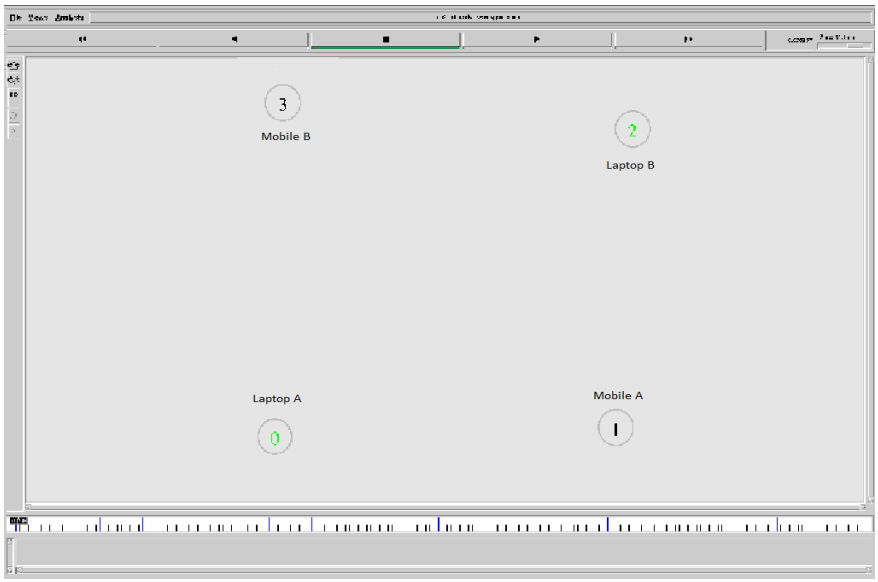

***Figure 71**. S1 Simple Office Ad-hoc Network in NAM.*

Analysis

For the evaluation of performance evaluation metrics, different configurations are used for different propagation models such as TwoRayGround and FreeSpace for comparing the results. A selected network is also analyzed for different routing protocols such as AODV, DSDV and DSR. For the evaluation, the congestion window and network throughput are calculated for NS2 and NS3 simulations.

TwoRayGround and FreeSpace

First, the simulation is executed for TwoRayGround and FreeSpace propagation model using AODV routing protocol in NS2 and NS3, and results are compared.

Congestion Window (CWND) for TwoRayGround

The results show almost symmetrical results for NS2 and NS3 using AODV routing protocol with TwoRayGround propagation model. However, to some extent, NS3 results

are better than NS2 (see **Figure 72**). In the simulation, the congestion window for nodes started to increase as soon as nodes find valid route to the destination. In NS3, the route was found around 5.75 milliseconds when the congestion window started to increase while in NS2, nodes spent more time in the destination discovery and the congestion window started to increase around 6.10 milliseconds.

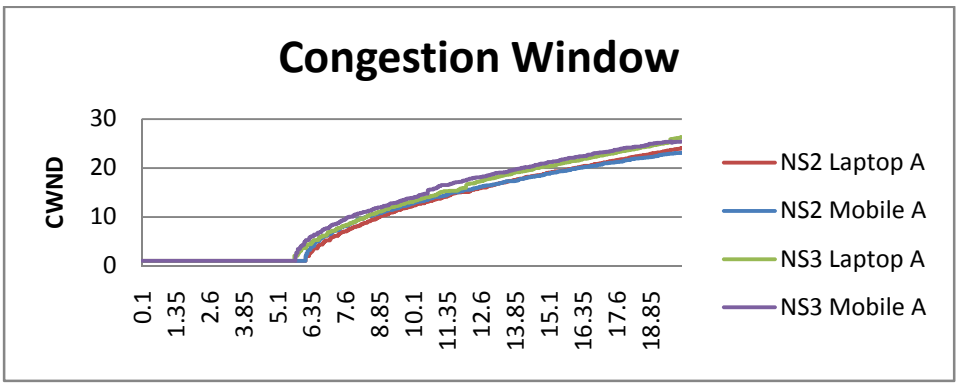

***Figure 72**. S1 Simple Office Ad-hoc Network (CWND for TwoRayGround).*

Congestion Window (CWND) for FreeSpace

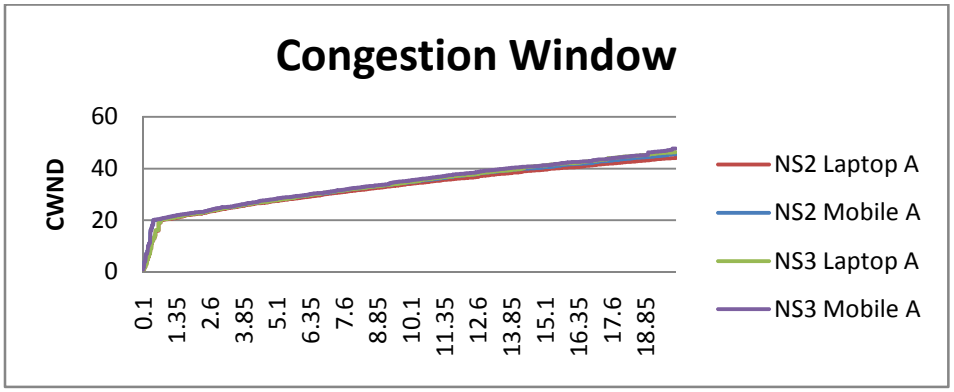

***Figure 73**. S1 Simple Office Ad-hoc Network (CWND for FreeSpace.*

As the definition of FreeSpace propagation model suggests, the congestion window for AODV using a FreeSpace propagation model (see **Figure 73**) started to increase quite earlier than TwoRayGround Model due to less error probability and channel noises. In

FreeSpace, nodes in a network discovered each other at start and the congestion window started to increase in a quick fashion. After reaching 20, the congestion window for both NS2 and NS3 progressed relatively in a slow speed until the simulation ended. In comparison, NS2 and NS3 results are similar to each other. However, the NS3 congestion window size becomes proportionally higher compared to NS2 as long as simulation time is increased.

Throughput

In the experiment, the network throughput with mean and standard deviation is calculated for NS2 and NS3 simulation using TwoRayGround and FreeSpace propagation models (see **Figure 74** and **Figure 75**).

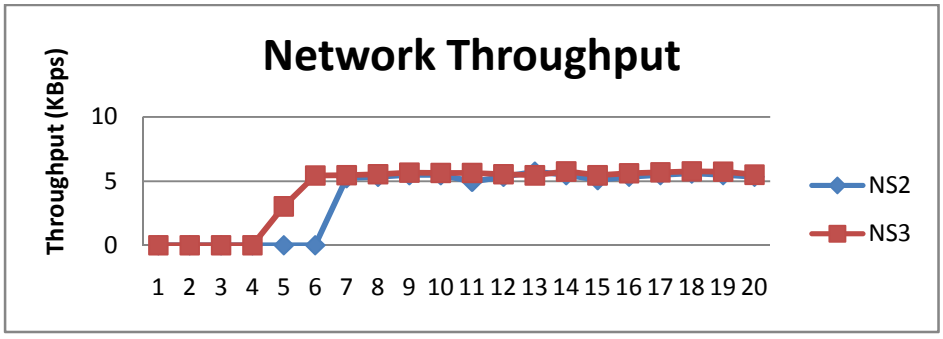

***Figure 74**. S1 Simple Office Ad-hoc Network (Network Throughput TwoRayGround).*

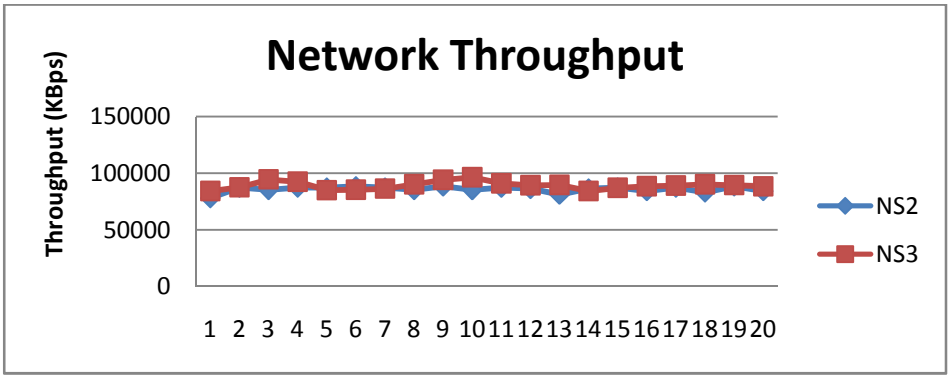

***Figure 75**. S1 Simple Office Ad-hoc Network (Network Throughput FreeSpace).*

Using a TwoRayGround (see **Figure 76**) and FreeSpace (see **Figure 77**) propagation model in NS2 and NS3, the network throughput mean is higher for NS3 compared to NS2 which shows that NS3 provides more efficiency in transmitting data for a large simulation time and allow better throughput for underlying network. However, the standard deviation for NS2 using a FreeSpace model is better compared to NS3 and shows fewer variations in the network throughput. In comparison between TwoRayGround and FreeSpace models, it is evident that the FreeSpace propagation generates better throughput and fewer fluctuations in a data transmission.

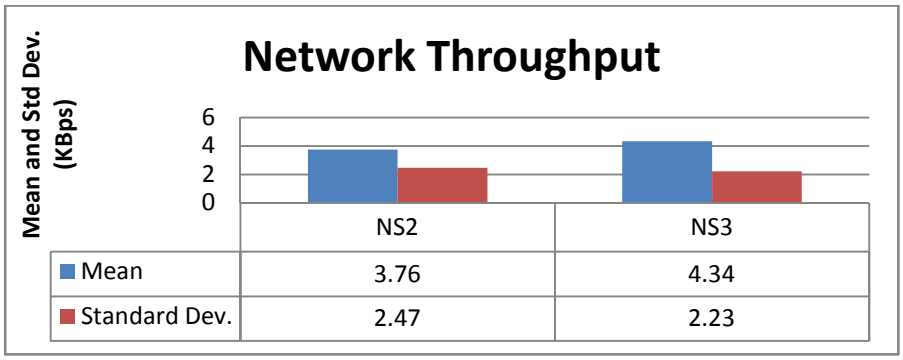

***Figure 76**. S1 Simple Office Ad-hoc Network (Mean & Std Dev. TwoRayGround).*

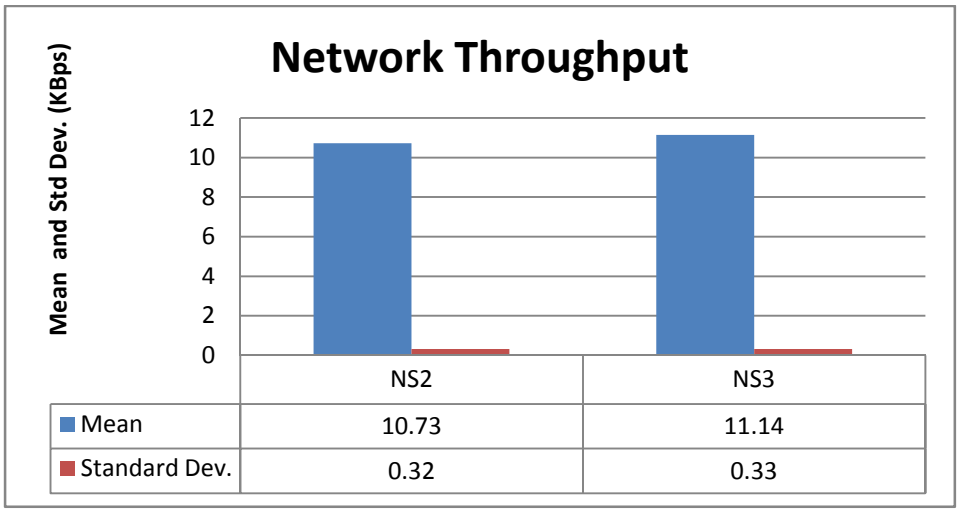

***Figure 77**. S1 Simple Office Ad-hoc Network (Mean & Std Dev. FreeSpace).*

Routing Protocols

Next the behavior of different routing algorithms used in ad hoc networks is analyzed. For the comparison, the simulation for AODV, DSDV and DSR routing protocols in NS2 and NS3 was executed. In the experiments, the congestion control window and network throughput are calculated and visualized.

Congestion Window (CWND) AODV

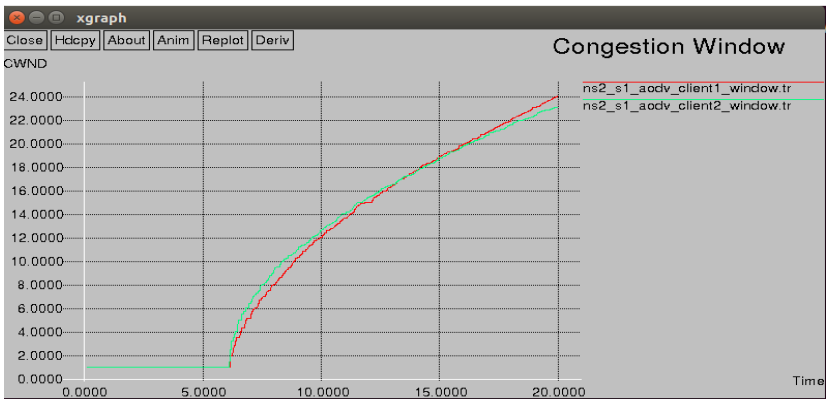

***Figure 78**. S1 Simple Office Ad-hoc Network (Congestion Window AODV).*

Congestion Window (CWND) DSDV

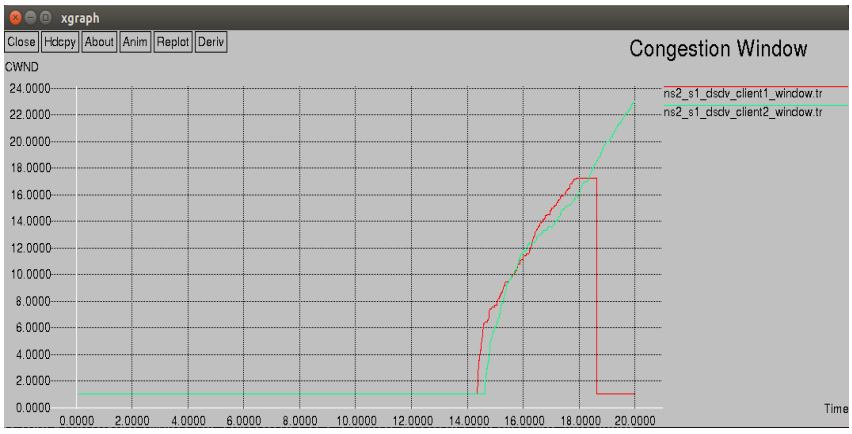

***Figure 79**. S1 Simple Office Ad-hoc Network (Congestion Window DSDV).*

Congestion Window (CWND) DSR

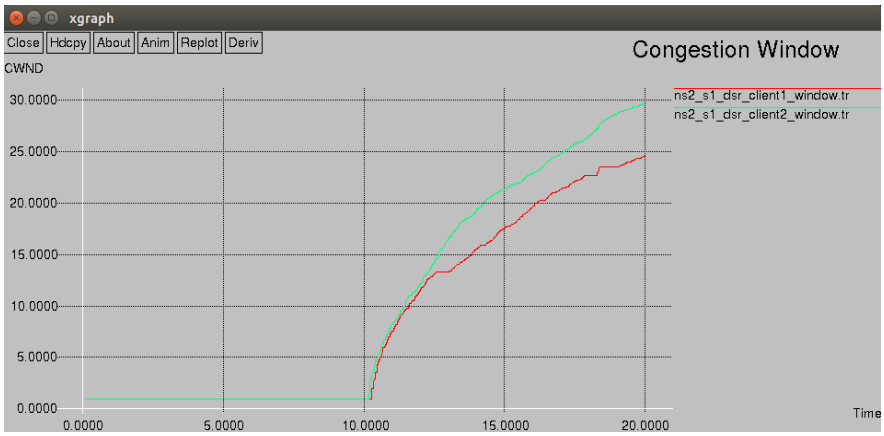

***Figure 80**. S1 Simple Office Ad-hoc Network (Congestion Window DSR).*

When comparing the congestion windows for all three routing protocols, it is proved that AODV performs better in comparison with DSDV and DSR. In AODV (see **Figure 78**), the congestion window started to grow around 6 seconds. DSDV (see **Figure 79**) performed inadequately and the congestion window started to increase around 15 seconds of the simulation which shows it took more time to construct a routing table compared to other two protocols. In terms of performance, DSR (see **Figure 80**) is proved to be a middle level protocol compared to AODV and DSDV.

4.4.2.2. S2 Complex Office Ad-hoc Network

In the experiment, the complex wireless office ad-hoc network (see **Figure 81**) consisting of eight moving nodes connecting with each other on ad-hoc basis is selected and implemented. In addition, a stationary node is introduced in a network which reacts as a central hub and supports fast and efficient routing connectivity among other nodes in a network. All nodes are either laptops or mobile devices.

Figure 81. *S2 Complex Office Ad-hoc Network.*

Configurations

a) AODV with TwoRayGround

```
# Define options
set val(chan)           Channel/WirelessChannel;        # channel type
set val(prop)           Propagation/TwoRayGround;       # radio-propagation model
set val(netif)          Phy/WirelessPhy;                # network interface type
set val(mac)            Mac/802_11;                     # MAC type
set val(ifq)            Queue/DropTail/PriQueue;        # interface queue type
set val(ll)             LL;                             # link layer type
set val(ant)            Antenna/OmniAntenna;            # antenna model
set val(ifqlen)         50;                             # max packet in ifq
set val(nodeCount)      8;                              # number of mobile nodes
set val(rp)             AODV;                           # routing protocol
```

```
set val(x)          700;                    # X dimension of topography
set val(y)          600;                    # Y dimension of topography
set val(stop)       100;                    # time of simulation end
```

b) AODV with FreeSpace

```
# Define options
set val(chan)       Channel/WirelessChannel;    # channel type
set val(prop)       Propagation/FreeSpace;      # radio-propagation model
set val(netif)      Phy/WirelessPhy;            # network interface type
set val(mac)        Mac/802_11;                 # MAC type
set val(ifq)        Queue/DropTail/PriQueue;    # interface queue type
set val(ll)         LL;                         # link layer type
set val(ant)        Antenna/OmniAntenna;        # antenna model
set val(ifqlen)     50;                         # max packet in ifq
set val(nodeCount)  8;                          # number of mobile nodes
set val(rp)         AODV;                       # routing protocol
set val(x)          700;                        # X dimension of topography
set val(y)          600;                        # Y dimension of topography
set val(stop)       100;                        # time of simulation end
```

c) DSDV with TwoRayGround

```
# Define options
set val(chan)       Channel/WirelessChannel;    # channel type
set val(prop)       Propagation/TwoRayGround;   # radio-propagation model
set val(netif)      Phy/WirelessPhy;            # network interface type
set val(mac)        Mac/802_11;                 # MAC type
set val(ifq)        Queue/DropTail/PriQueue;    # interface queue type
set val(ll)         LL;                         # link layer type
set val(ant)        Antenna/OmniAntenna;        # antenna model
set val(ifqlen)     50;                         # max packet in ifq
set val(nodeCount)  8;                          # number of mobile nodes
set val(rp)         DSDV;                       # routing protocol
set val(x)          700;                        # X dimension of topography
set val(y)          600;                        # Y dimension of topography
set val(stop)       100;                        # time of simulation end
```

d) DSR with TwoRayGround

```
# Define options
set val(chan)           Channel/WirelessChannel;        # channel type
set val(prop)           Propagation/TwoRayGround;  # radio-propagation model
set val(netif)          Phy/WirelessPhy;                # network interface type
set val(mac)            Mac/802_11;                     # MAC type
set val(ifq)            CMUPriQueue;                    # interface queue type
set val(ll)             LL;                             # link layer type
set val(ant)            Antenna/OmniAntenna;            # antenna model
set val(ifqlen)         50;                             # max packet in ifq
set val(nodeCount)      8;                              # number of mobilenodes
set val(rp)             DSR;                            # routing protocol
set val(x)              700;                            # X dimension of topography
set val(y)              600;                            # Y dimension of topography
set val(stop)           100;                            # time of simulation end
```

Experiment execution

In the experiment, the network consists of eight wireless nodes including four laptops and four mobile phones. These nodes were connected using TCP protocol with FTP application. In addition, one stationary node is added in a network which reacts as central hub and helps all connected nodes for data transmission. Laptop A and B were connected to Laptop C and D respectively and Mobile A and B were connected to Mobile C and D through WLAN IEEE 802.11 network. All the nodes in a network have mobility characteristics except stationary node in a center. The initial positions of the nodes were defined at time 0.0 and all nodes were placed at a distance from each other. During the simulation, all the laptops and mobile nodes moved from one position to another and tried to send messages to the connected nodes. In the middle of the simulation, all source laptops and mobile nodes reached near to receiver nodes and then stated to move away gradually. The simulation area was set to 700 X 600 and nodes moved within area in the range of 100 seconds. **Figure 82** illustrate S2 complex office ad hoc network in NAM.

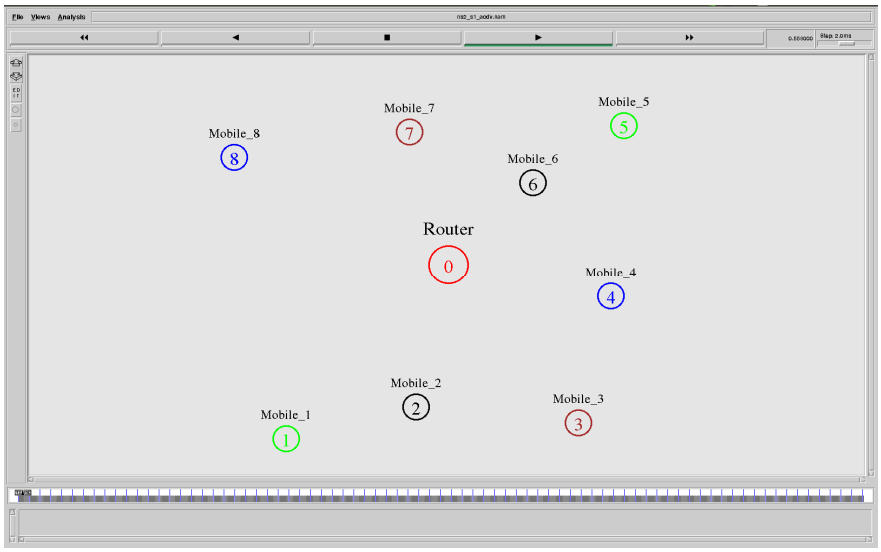

***Figure 82**. S2 Complex Office Ad-hoc Network in NAM.*

Analysis

For the evaluation of performance evaluation metrics, different configurations were used for routing protocols such as AODV, DSDV and DSR with TwoRayGround propagation model. The underlying network is analyzed for different routing protocols including AODV, DSDV and DSR. For the evaluation, the congestion window and network throughput are calculated for NS2 and NS3. However, the most important results conducted in NS2 are provided here including the congestion control and network throughout.

AODV

First the simulation for an AODV routing protocol with TwoRayGround model in NS2 was executed, and results are compared. **Figure 83** shows simulation graph for the congestion window and network throughput using xgraph.

Congestion Window (CWND)

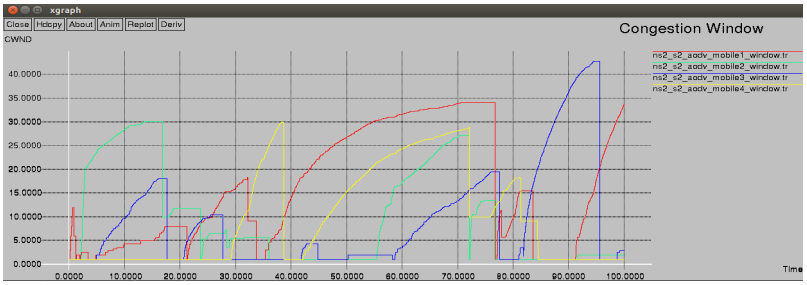

***Figure 83**. S2 Complex Office Ad-hoc Network (Congestion Window AODV).*

In a complex network with TwoRayGround propagation model, the network congestions is observed at various points during the first and the last 20 seconds of the simulation. In the simulation, mobile 3 (Laptop A) had maximum throughput around 95 seconds when all other nodes were distance apart from other nodes and the network was used only by mobile 3 (Laptop A) and mobile 1 (Mobile A) nodes. Between 30 to 80 seconds, the congestion window size for mobile 1 (Mobile A), mobile 2 (Mobile B) and mobile 4 (Laptop B) was high ensuring the maximum network throughput during this time.

Throughput

In the experiment, the network throughput (see **Figure 84**) with NS2 simulation is calculated for the whole network and results are compared.

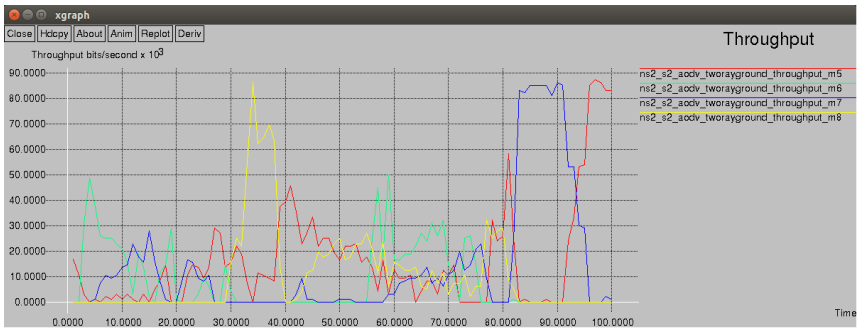

***Figure 84**. S2 Complex Office Ad-hoc Network (Network Throughput AODV).*

At the middle of a simulation, the network throughput using AODV routing protocol shows high throughput for all the connected nodes. During this time, the nodes had minimum distance to the other nodes. Also during the last couple of seconds, two nodes had high throughput as other nodes were not transmitting at this time.

DSDV

Secondly, simulation for a DSDV routing protocol with TwoRayGround model in NS2 was executed, and results are compared. **Figure 85** and **Figure 86** show the simulation graphs for the congestion window and network throughput using xgraph respectively.

Congestion Window (CWND)

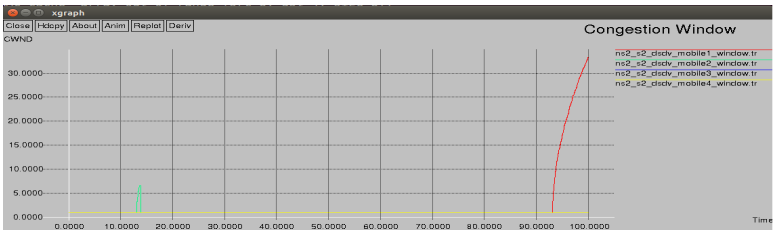

Figure 85. *S2 Complex Office Ad-hoc Network (Congestion Window DSDV).*

As discussed earlier that DSDV is worst among other routing protocols, the graph (see **Figure 85**) shows that the congestion window for only mobile 1 (Mobile A) and mobile 2 (Mobile B) increased during the simulation at certain time. Other nodes could not find the destination node information; hence congestion window for such nodes remained at the initial value all the time.

Throughput

In this experiment, the network throughput in NS2 simulation is calculated for a network and results are compared.

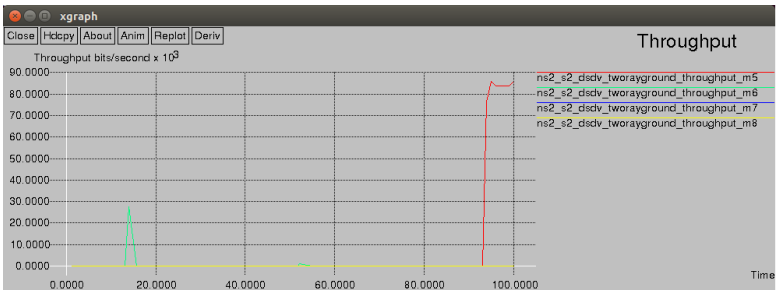

Figure 86. *S2 Complex Office Ad-hoc Network (Network Throughput DSDV).*

The network throughput using DSDV routing protocol reflects the congestion control window size given above and shows that only two nodes were able to send data during the whole simulation.

DSR

Thirdly, the simulation is executed for a DSR routing protocol with TwoRayGround model in NS2, and results are compared. **Figure 87** and **Figure 88** show the simulation graphs for the congestion window and network throughput using xgraph respectively.

Congestion Window (CWND)

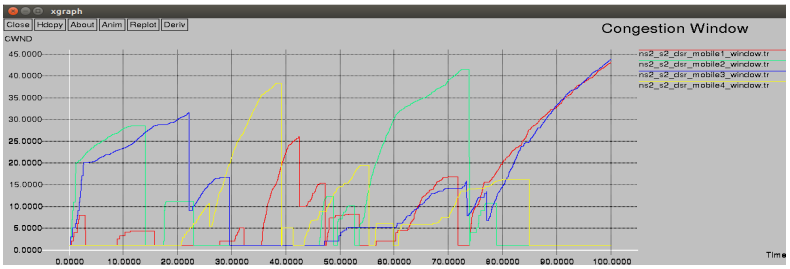

Figure 87. *S2 Complex Office Ad-hoc Network (Congestion Window DSR).*

In a complex network with TwoRayGround propagation model, the network congestion was observed at the middle of the simulation. At start, mobile 2 (Mobile A) and mobile 3

(Laptop A) had a high congestion window size allowing them to send packets at higher speed. As long as other nodes in a network started to send data, the congestion occurred and the congestion window for mobile 2 and mobile 3 were reduced to initial positions. During the middle of the simulation, all nodes started to send data at same time which reflected on the congestion window for all sending nodes. During the last 10 seconds, all the nodes except mobile 1 (Mobile A) and mobile 3 (Laptop A) stopped to send data which resulted to increase the congestion window size for mobile 1 and mobile 2.

Throughput

In this experiment, the network throughput is calculated in NS2 simulation. Network throughput is calculated for a whole network and results are compared.

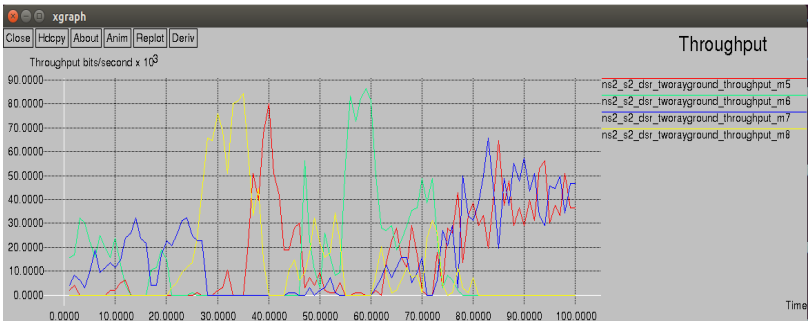

Figure 88*. S2 Complex Office Ad-hoc Network (Network Throughput DSR).*

The network throughput using DSR routing protocol reflects the congestion control window size and shows that all the sending nodes were able to send data during the whole simulation. During the middle of the simulation, the increase in a throughput for all the sending nodes can be seen from the graph, whereas during the end of simulation time, only two of the nodes where able to send data in a network.

Comparison AODV, DSDV and DSR

For comparison among routing protocols, mean and the standard deviation for these simulations calculated and results are presented in **Figure 89**.

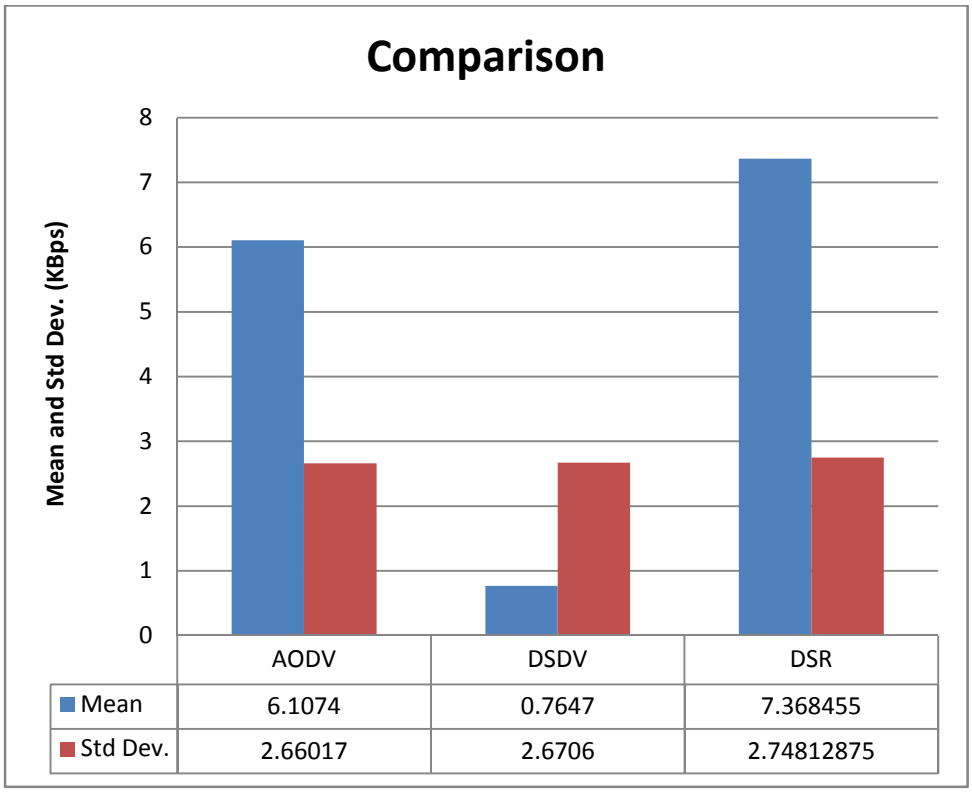

***Figure 89**. S2 Complex Office Ad-hoc Network (Comparison Network Throughput).*

In the complex Office Ad-hoc Network with long simulation time, DSR routing protocol generated the highest average network throughput compared to AODV and DSDV. On the other hand, DSDV is proved to be a really poor routing protocol which gave a very minimum average network throughput. In comparison, AODV is comparable with DSR which performed effectively in the simulations. Considering the variations in throughput, all three routing protocols are almost at the same level with the reasonable standard deviation.

5. CONCLUSION AND FUTURE WORK

After completing the experiment part and considering the theoretical part in this thesis, it is concluded that use of network simulators in the field of communication systems is really valuable and cost-effective. These simulators provide support for various real time models with the possibility of customization which can be used to evaluate new emerging protocols and technologies.

Comparing these simulators with respect to time and space complexity, NS3 is proved to be the fastest due to use of C++ code and also requires less memory compared to NS2 which uses abstract level OTCL programming language, requiring more resources during the building and running process of a simulation. Use of OMNET++ is user friendly due to use of built-in toolbox and drag and drop features which makes it preferable for researchers with limited understanding of programming paradigm. In comparison for usage complexity, NS2 is easy to use due to OTCL programming language compared to NS3 which requires deep knowledge of C++ programming language features such as pointers, pointer functions, inheritance, abstract data types and other programming features.

After running different simulations using performance evaluation metrics, it is concluded that NS3 simulation time, and output values in all the simulations were better in comparison with NS2. In wired network simulations, the experiments are executed for a star network using TCP and UDP protocols with CBR and FTP applications where congestion window, throughput, end-to-end delay, and packet delivery and drop ratios resulted in higher and smooth values for NS3. The increase in the simulation time also shows the improvements in NS3 results whereas NS2 performance decreases as soon as simulation time is increased. In comparison to the transport layer protocol, UDP is fast and consumes less time to reach the destination compared to TCP, resulting in a less end-to-end delay in both NS2 and NS3. Comparing CBR with FTP applications, CBR produces higher throughput due to the constant transmission rate.

In wireless simulation, same improved behavior is observed in NS3 compared to NS2 where the calculation of performance evaluation metrics resulted in better performance for NS3. For the experiments, different propagation models are used such as TwoRayGround and FreeSpace in simple and complex wireless office networks where FreeSpace resulted in a better network throughput and less packet loss in a network ensuring a better overall performance. In addition to propagation models, different routing protocols are evaluated such as AODV, DSDV and DSR for both simple and complex office wireless networks. From the results, it is concluded that AODV is more consistent in establishing connections in a network and produces high network throughput. DSDV is worst in all which produces minimum network throughput. DSR proved to be as close as AODV and produces similar network throughput compared to AODV but it is not consistent in establishing routing information resulting in a high end-to-end delay.

Considering all above, it is concluded that NS3 is a network simulator for future academic use and in coming years it will fully replace NS2 due to the high performance, available features, network models and documentation. On the other side, OMNET++ is leading and recommended in commercial industries due to better use, support and documentation.

In the future, one can simulate the same experiments with OMNET++ and other network simulators for comparing the results with NS2 and NS3. In this thesis, wired and wireless networks are evaluated with LAN characteristics. In the future, performance evaluation metrics can be executed for advanced communication technologies such as LTE, 3G, 4G, WiMAX, COAP and ZigBee. One enhancement would be to combine the available wired and wireless networks in this thesis to form a wired-wireless network, execute the performance evaluation metrics and observe the behavior of various network simulators.

REFERENCES

Andersen, Rappaport & Yoshida (1995). *Propagation Measurements and Models for Wireless Communications channels.* In: Communications Magazine, IEEE, 33:1, 42–49.

Bestofmedia Team (2012). *Network Hardware & Assembly: LAN 102*. Available from Internet <URL: http://www.tomsitpro.com/articles/local_area_network-gigabit_ethernet-networking-nics-scott_mueller,2-263-4.html>.

Bhargava, Dr. Ritu Bhargava, Manish Mathuria, Shipla Gupta & Kamal Kumar Jyotiyana (2013). *Analysis of Different Congestion Avoidance Algorithms*. In: International Journal of Computer Networks and Wireless Communications (IJCNWC), 3:1, ISSN: 2250–3501.

Borboruah & Gypsy Nandi (2014). *A Study on Large Scale Network Simulators*. In: Internal Journal of Computer Science and Information Technologies, 5:6, ISSN: 0975–9646.

Cox (1995). *Wireless Personal Communications: What is it?*. In: Personal Communications, IEEE, 2:2, 20–35.

Dhadse &B R Chandavarkar (2014). *Comparative Analysis of Queue Mechanisms with Respect to Various Traffic in Wired-Cum-Wireless Network*. In: Fourth International Conference on Advanced Computing & Communication Technologies (ACCT), Feb. 2014, 8–9.

El Zein (2009). *Propagation Channel Modeling for Emerging Wireless Communication Systems*. In: ACTEA 15–17 July.2009, Zouk Mosbeh, Lebanon, 457– 462.

Erdei, Wagner Ambrus, Sója Katalin & Székely Márk (2001). *A Networked Remote Simulation Architecture and its Remote OMNET++ Implementation*. In: Proceedings of the European Simulation Multiconference (ESM 2001), 7–9 June. 2001, Prague. Available From Internet <URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.96.5033&rep=rep1&type=pdf>

Fernandes & Michel Ferreira (2012). *Scalable VANET Simulations with NS-3*. In: Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th May. 2012, 1–9.

Flickenger (2007). *Wireless Networking in the Development World*. 3rd Ed. ISBN-13: 978-1484039359. Available from Internet: < URL: http://wndw.net/pdf/wndw3-en/wndw3-ebook.pdf>.

Goldsmith (2004). *Wireless Communication*. California: Stanford University. Available from Internet: < URL: http://web.cs.ucdavis.edu/~liu/289I/Material/book-goldsmith.pdf>.

Gupta, Mangesh M. Ghonge, Parag D. Thakare & Dr. P. M. Jawandhiya (2013). *Open-Source Network Simulation Tools: An Overview*. In: International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), April. 2013 2:4, 1629–1635.

Ho, King K Leung, John W Polak & Rahul Mangharam (2007). *Node Connectivity in Vehicular Ad Hoc Networks with Structured Mobility*. In: 32nd IEEE Conference on Local Computer Networks (LCN 2007), 15–18 Oct. 2007, 635–642.

Ikeda, Elis Kulla, Leonard Barolli & Makoto Takizawa (2011). *Wireless Ad-hoc Networks Performance Evaluation Using NS-2 and NS-3 Network Simulators*. In: International Conference on Complex, Intelligent and Software Intensive Systems (CISIS), 30 June – 2. July. 2011, 40–45.

Katz (1994). *Adaptation and Mobility in Wireless Information Systems*. In: Personal Communications, IEEE, 1:1, 6–17.

Khan, Muhammad Amir, Hasbullah Halabi & Babar Nazir (2014). *Recent open source wireless sensor network supporting simulators: A performance comparison*. In: International Conference on Computer, Communications and Control Technology (I4CT), 24 Sept. 2014, 324–328.

Khan, Sardar M. Bilal & Mazliza Othman (2012). *A performance comparison of open source network simulators for wireless networks*. In: IEEE International Conference on Control System, Computing and Engineering (ICCSCE), 23–25 Nov. 2012, 34–38.

Koo, Seongjin Ahn & Jinwook Chung (2004). *A Comparative Study of Queue, Delay and Loss Characteristics of AQM Schemes in QoS-Enabled Networks*. In: Computing and Informatics, 22:2003, 317–335.

Li & Yu Wang (2007). *Routing in Vehicular Ad hoc Networks: A Survey*. In: IEEE Vehicular Technology Magazine, June. 2007, 2:2, 12–22.

Micheal (2005). *A Comparison of the Architecture of Network Simulators NS-2 and TOSSIM*. In: im Rahmen des Seminars: Performance Simulation of Algorithms and Protocols,31 Jan. 2005. Available from Internent: <URL:http://wing.nitk.ac.in/resources/Comparison.pdf>.

NS2 Documentation: *The Network Simulator ns-2*. Available from Internet <URL: http://www.isi.edu/nsnam/ns/ns-documentation.html>.

NS3 Documentation: *A Discrete-Event Network Simulator*. Available from Internet <URL: http://www.nsnam.org/documentation>.

OMNET++ Documentation: *Discrete Event Simulator*. Available from Internet <URL: http://www.omnetpp.org/documentation>.

Pan & Raj Jian (2008). *A Survey of Network Simulation Tools: Current Status and Future Developments*. St. Louis : Washington University, Tech. Rep. Available from Internet: <URL: http://www.cse.wustl.edu/~jain/cse567-08/ftp/simtools/>.

Pandey & Vibhore Tyagi (2013). *Performance Analysis of Wired and Wireless Network using NS2 Simulator*. In: International Journal of Computer Applications (0975-8887), June. 2013, 72:21.

Rachna, Sethi Shweta, Keshari Rita & Goel Sakshi (2012). *A Study of Comparison of Network Simulator -3 and Network Simulator -2*.In: International Journal of Computer Science and Information Technologies, 3:1, 3085 – 3092.

Rajankumar, P Nimisha & P Kamboj (2014). *A comparative study and simulation of AODV MANET routing protocol in NS2 & NS3*. In: International Conference on Computing for Sustainable Global Development (INDIACom), 5-7 March 2014, 889–894.

Schiller (2003). *Mobile Communications*.2nd Ed. Pearson Education. ISBN: 978-0321123817.

Sekercioglu. *Experiment II: Introduction to Discrete-Event Simulation and OMNET++ Framework*. Monash University Department of Electrical & Computer Systems Engineering. Available from Internet <URL: http://titania.ctie.monash.edu.au/netperf/netperf-omnetpp-tictoc-01.pdf>.

Sharma & M K Chose (2010). *Wireless Sensor Networks: An Overview on its Security Threats*. In: IJCA Special Issue on Mobile Ad-hoc Nerworks MANETs, 42–45.

Sun & Xiaoling (2012). *TCP Congestion Control Algorithm Research*. In: 8th International Conference on Information Science and Digital Content Technology (ICIDT), -28 June. 2012, vol.3, pp.703, 706, 26.

Tse & Pramod Viswanath (2005). *Fundamentals of Wireless Communication*. Cambridge University Press. Available from Internet: < URL: http://www.eecs.berkeley.edu/~dtse/book.html>.

Verdú (2000). *Wireless Bandwidth in the Making*. In: Communications Magazine, IEEE, 38:7, 53–58.

Weingartner, Hendrik vom Lehn & Klaus Wehrle (2009). *A performance comparison of recent network simulators*. In: IEEE International Conference on Communication (ICC 2009), Dresden, Germany, 14–18 Jun. 2009, 1287–1291.

Winsberg (2010). *Science in the Age of Computer Simulation*. University of Chicago Press, 168 p. ISBN 9780226902043.