Chapter 3

WSN Simulators

This chapter depicts Simulation platform for Wireless Sensor Network (WSN) and Network Simulation using OPNET. The aim of Simulation platforms for WSN is to describe the characteristics of WSN simulation, briefly describe the current popular WSN simulation tools, detail the OPNET and simulate a comparative simulation study regarding nature of wired and wireless network.

3.1 Simulation Platforms for WSN

WSN is one of the most worthwhile communication network and simulation tools for WSN are increasingly being used to study sensor webs and to test new applications and protocols in this evolving research field. There is always an overriding concern when using simulation that the results may not reflect accurate behavior. However due to the associated cost, time and complexity involved in implementation of such networks, developers prefer to have first-hand information on feasibility and reflectivity crucial to the implementation of the system prior to the hardware implementation [1]. This is especially true in sensor networks, where hardware may have to be purchased in large quantities and at high cost. Even with readily available sensor nodes, testing the network in the desired environment (running real experiments on a test-bed) can be a time consuming and difficult task.

Besides, repeatability is largely compromised since many factors affect the experimental results at the same time. It is hard to isolate a single aspect. Simulation-based testing can help to indicate whether or not the time and monetary investments are worthwhile. Simulation is, therefore, the most common approach to developing and testing new protocol and applications for sensor networks. This leads to the recent boom of simulator development [2]. There are a number of advantages to this approach including lower cost, ease of implementation, and practicality of testing large-scale networks.

3.2 The Characteristics of WSN Simulation

Wireless sensor network is a highly application-oriented network type. The characteristics of the simulation in the following areas described in Table 3.1 are different from the existing wired and wireless network simulation [3].

In order to effectively develop any protocol based on simulations, it is important to know the different tools available and their benefits and drawbacks. Given the facts that simulation is not perfect and that there are a number of popular sensor network simulators available, thus making different simulators accurate and most effective for different situations/applications. It is crucial for a developer to choose a simulator that

best fits the application [4]. However, without a working knowledge of the available simulators, this can be a challenging task. Additionally, knowing the weaknesses of available simulators could help developers to identify drawbacks of their own models, when compared with these simulators, thus providing an opportunity for improvement. It is thus imperative to have a detailed description of a number of the more prominent simulators available. In this chapter, several main-stream WSNs simulators are described in more detail.

Sr. No.	Characteristics	Comments
1.	Simulation Scale	For the traditional wired network, the entire network performance can be greatly simulated by the use of the limited and represented node topology. However, due to the disadvantages of large redundancy wireless sensor network and high-density node topological structure types, the overall performance of WSN cannot be analyzed by the limited number of nodes [5]. So, in the WSN simulation scale, a large number of nodes in parallel computing must be considered [6, 7].
2.	Simulation Target	The traditional wired and wireless network simulation analysis mainly focus on quality of service (QoS), such as network throughput, end-to-end delay, and packet loss rate, which is not the main target of the analysis in most applications of wireless sensor networks [6]. The life and energy consumption analysis of the node are the most important objectives of the analysis [6, 8].
3.	Node Characteristics	The WSN nodes are influenced not only by the noise, interference, and known destruction factors, but also affect the instability of the nodes [5]. This is because of the limited capability of the node itself, coupled with the easy to fail feature (e.g. node energy exhausted), which have exacerbated the uncertainty of the network, and they are rarely seen in the previous network system [6, 7].

Table 3.1 Characteristics of WSN Simulations

3.2.1 NS-2

NS-2 (Network Simulator-2) [9, 10, 11] is a well-known network simulator for discrete event simulation. Simulations are based on a combination of C++ and OTcl [12]. NS-2 includes a large number of simulated network protocols and tools used for simulating transport control protocol (TCP), routing algorithm, multicast protocol over the wired or wireless (local connection or via satellite connection) networks [9]. NS-2 is committed to OSI model simulation, including the behaviour of physical layer and it is a free open source software and available for free download [11]. NS-2 has a number of limitations [13]:

1) It puts some restrictions on the customisation of packet formats, energy models, MAC protocols, and the sensing hardware models, which limits its flexibility.
2) The lack of an application model makes it ineffective in environments that require interaction between applications and the network protocols.
3) It does not run real hardware code.
4) It has been built by many developers and contains several inherent known and unknown bugs.
5) It does not scale well for WSNs due to its object-oriented design.
6) Using C++ code and OTcl scripts make it difficult to use.

Actually, NS-2 was not initially designed to simulate wireless sensor network, but a few research groups had extended NS-2 in order to enable it to support wireless sensor network simulation, including sensor model, battery model, a small stack, and hybrid simulation tools [14]. It is extensible, but not very scalable because of the split-programming model and object-oriented structure. In addition, because NS-2 can simulate very detailed data packet close to the exact number of running packets, it is unable to carry out large-scale network simulation [9].

To overcome the above drawbacks the improved NS-3 simulator [15] was developed. NS-3 supports simulation and emulation. It is totally written in C++, while users can use python scripts to define simulations. Hence, transferring NS-2 implementation to NS-3 require manual intervention. Besides the scalability and performance improvements, simulation nodes have the ability to support multiple radio interfaces and multiple channels. Furthermore, NS-3 supports a real-time schedule that makes it possible to interact with real systems [15]. For example, a real network device can emit and receive NS-3 generated packets.

3.2.2 GloMoSim

Global Mobile Information System Simulator (GloMoSim) [16-18] is a scalable simulation environment for large wireless and wired communication networks. GloMoSim follows the idea of the OSI reference model by using a layered approach. For the communication between the different simulation layers a standard API is used so that new models and layers can be rapidly exchanged and integrated. The node aggregation technique is introduced into GloMoSim to give significant benefits to the simulation performance. In GloMoSim, each node represents a geographical area of the simulation. Hence the network nodes which a particular entity represents are determined by the physical position of the nodes.

GloMoSim uses the parallel discrete-event simulation capability provided by Parsec (Parallel Simulation Environment for Complex Systems) [19], a c-based simulation language for sequential and parallel execution of discrete-event simulation models. Both, GloMoSim as well as Parsec were developed by the Parallel Computing Lab at UCLA. GloMoSim can be run on Windows as well as Unix derivates.

As in NS-2, GloMoSim uses an object-oriented approach, however for scalability purposes; each object is responsible for running one layer in the protocol stack of every node. This design strategy helps to divide the overhead management of a large-scale network. Though it is a general network simulator, GloMoSim currently supports protocols designed purely for wireless networks. GloMoSim is effectively limited to IP networks because of low level design assumptions and it is not capable of simulating sensor networks accurately [20]. Therefore, it suffers the same problems as Ns-2, the packet formats, energy models, and MAC protocols are not representative of those used in wireless sensor networks.

Moreover, GloMoSim does not support phenomena occurring outside of the simulation environment, all events must be gathered from neighbouring nodes in the network. Finally, GloMoSim stopped releasing updates in 2000 and released a commercial product called QualNet.

3.2.3 QualNet

QualNet (Quality Networks) work from Scalable Network Technologies (a spin out company from UCLA) is a commercial network simulator tool [21] that is derived from GloMoSim. QualNet significantly extends the set of models and protocols supported by GloMoSim. It has a dedicated and fully implemented protocols and modules for both the wired and wireless scenarios including ad hoc, cellular and satellite

models. QualNet is a discrete-event simulator, as such, it is event driven and time aware. It uses a layered architecture that is run by each node. When a protocol resides in a particular layer at one node, the packets are passed down crossing the remaining layers at the sending node, across the network, and then up to the protocol stack at the receiving node. The basic version of QualNet software comes with the standard library which offers the most common models and protocols necessary for both wired and wireless network modelling for research and industrial purposes. Many other libraries can be purchased separately including the MANET library which provides specific components for ad hoc networks, energy and mobility models other than those already present in the standard library; and a QoS library which includes specialised protocols for implementing quality of service. The authors of QualNet claim it to be the most complete network simulator, in terms of available protocols, models and tools for mobile ad hoc networks. Another key advantage is that the authors provide the C source code for all the components, modules, models and protocols; this allows the customers to fully modify or tweak the models as well as to better understand the working of models.

QualNet has a modular design and an intuitive GUI that make it easy to use to learn and modify. The QualNet Developer software suite consist of five different components put together to form a complete solution for any type of network analysis.

3.2.4 TOSSIM

TOSSIM (TinyOS Mote Simulator) [22] is an open-source operating system specially developed for the wireless embedded sensor networks. There are few hardware platforms available for TinyOS, some commercial and some non-commercial. TinyOS release includes a simulator called TOSSIM. It is built especially for Berkeley Mica Mote platform. TOSSIM is an emulator rather than a simulator, as it runs actual application code. Simulated application code can be transferred directly to the platform, but it might not run in a mote as it runs in a simulation due to the simplifying assumptions in TOSSIM.

Figure 3.1 shows the working flow of TOSSIM. The TOSSIM architecture is consisted of 5 segments: Frames, Components, Models, Services and Events.

TOSSIM is a very simple but powerful emulator for WSN. Each node can be evaluated under perfect transmission conditions, and using this emulator can capture the hidden terminal problems. As a specific network emulator, TOSSIM can support thousands of nodes simulation. This is a very good feature, because it can more accurately simulate the real world situation. Besides network, TOSSIM can emulate radio

models and code executions. This emulator may be provided more precise simulation result at component levels because of compiling directly to native codes.

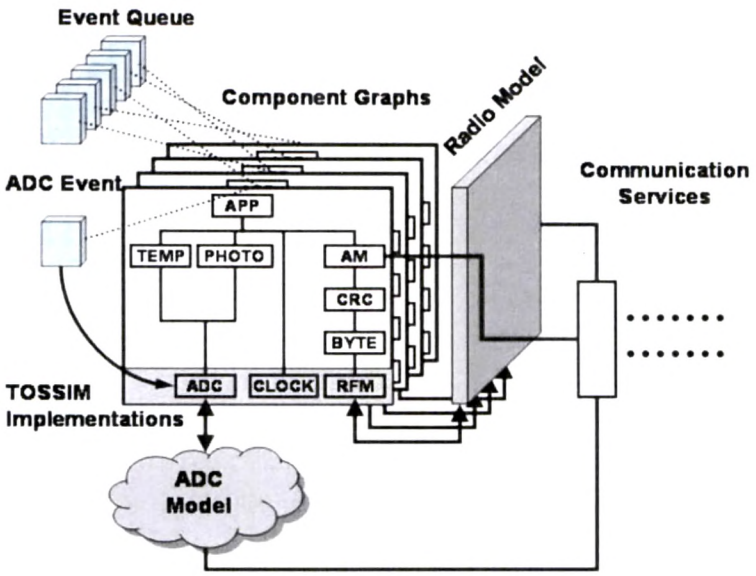

Figure 3.1: TOSSIM Architecture [23]

TOSSIM is a bit-level discrete event network emulator built in Python [24], a high-level programming language emphasizing code readability, and C++. People can run TOSSIM on Linux Operating Systems or on Cygwin on Windows. TOSSIM also provides open sources and online documents.

Developers had set four requirements for TOSSIM: scalability, completeness, fidelity and bridging. To be scalable, a simulator should manage networks of thousands of nodes in a wide variety of configurations. To achieve this, each node in TOSSIM is connected in a directed graph where each edge has a probabilistic bit error. For completeness, a simulator must capture behavior and interactions of a system at a wide variety of levels. And for fidelity, a simulator must capture behavior of a network with a subtle timing of interactions on a mote and between motes. Requirement for bridging is met as the simulated code runs directly in a real mote. [23]

The goal of TOSSIM is to study the behavior of TinyOS and its applications rather than performance metrics of some new protocol. Hence, it has some limitations, for instance, it does not capture energy consumption. Another drawback of this framework is that every node must run the same code. Therefore, TOSSIM cannot be used to evaluate some types of heterogenous applications.

3.2.5 Avrora

Avrora is a command-line framework capable of simulating and analysing programs developed for MEMSIC Mica2 and MicaZ sensor platforms. In the simulation each node has its own separated thread [25, 26].

Avrora, a research project of the UCLA Compilers Group, is an open-source cycle accurate simulator for embedded sensing programs. Unlike other simulators, that are able to simulate only specific platforms, Avrora has language and operating system independence. It provides a framework for program analysis, allowing static checking of embedded software and an infrastructure for future program analysis research. Avrora simulates a network of motes, runs the actual microcontroller programs (rather than models of the software), and runs accurate simulations of the devices and the radio communication [27].

Avrora [28-30] is an instruction-level simulator instead of just simulating software models. This approach provides an accurate simulation of devices and radio communication so that, for example, the energy usage can be predicted according to the number of clock-cycles needed for the used instructions, which removes the gap between TOSSIM and ATEMU. The codes in Avrora run instruction by instruction, which provides faster speed and better scalability. Avrora can support thousands of nodes simulation, and can save much more execution time with similar accuracy. Avrora is implemented in Java which offers great flexibility and portability because the simulation of machine code is operating system independent.

Avrora lacks an integrated graphical user interface so that everything has to be done manual on the command line.

3.2.6 OMNET++

The Objective Module Network Test-bed in C++ (OMNeT++) [31] is a component-based, modular and open-architecture simulation environment with strong GUI support and an embeddable simulation kernel. OMNeT++ is a public source component-based discrete event network simulator [32]. The simulator mainly supports standard wired and wireless IP communication networks, but some extensions for WSN exist. Like NS-2, OMNeT++ is popular, extensible and actively maintained by its user community in the Academia who has also produced extensions for WSN simulation.

OMNET++ is becoming a popular tool and its lack of models is being cut down by recent contributions. For instance, a mobility framework has recently been released for OMNET++ [33], and it can be used as a starting point for WSN modelling. OMNeT++

provides a hierarchical nested architecture. The modules are programmed in C++; the GUI of OMNeT++ is created using the Tk library. The modules are assembled into components and models by using a high-level language (NED). Modules communicate by sending messages. The simulation configuration is managed by .ini files. There are several OMNeT++ based WSNs simulation frameworks. e.g., Castalia, MiXiM, NesCT, PAWiS, SENSIM (SensorSimulator).

Additionally, several new proposals for localization and MAC protocols for WSN have been developed with OMNET++, under the Consensus project [34], and the software is publicly available. Nevertheless, most of the available models have been developed by independent research groups and do not share a common interface, what makes difficult to combine them. As an example, not even the localization and MAC protocols developed in the Consensus project are compatible.

3.2.7 SENS

SENS [35] was developed to solve the deficiencies of the traditional network simulators, which has been used in the field of wireless sensor network. It came after NS-2 providing some necessary changes [4]. SENS uses the system structure in the module which can be reused, as long as the interfaces between modules meet the requirements. Then both modules can be reused and replaced, even the new simulation programs can be fully developed on the basis of SENS. The differences between SENS and other simulators are that SENS uses parallel simulation and serial simulation optional modes. The system default is the serial simulation. This is to consider parallel simulation in many cases caused low efficiency, thus giving users opportunity to choose according to their needs.

SENS is a customizable sensor network simulator for WSN applications [1]. Multiple component implementations in SENS offer varying degrees of realism. Users can assemble application-specific environments; such environments are modelled in SENS by their different signal propagation characteristics. The same source code that is executed on simulated sensor nodes in SENS may also be deployed on actual sensor nodes, this enables application portability.

Simulation/Programming language is written in C++. There exists a thin compatibility layer to enable direct portability between SENS and real sensor nodes.

3.2.8 MATLAB: TrueTime toolbox

TrueTime [36] is a freeware Matlab/Simulink-based simulator for networked and embedded real-time control systems. This toolbox facilitates co-simulation of controller

task execution in real-time kernels, network transmissions and continuous plant dynamics.

This toolbox provides possibility to write tasks as M-files, C++ functions or call Simulink block diagrams from within the code functions. TrueTime blocks include generally used networks as Ethernet, CAN, TDMA, FDMA, Round Robin or Switched Ethernet. It supports simulation of Wireless networks (802.11b/g WLAN and 802.15.4 ZigBee) and battery-powered devices. In a brief description, TrueTime is a small library of simulation blocks which extends usability of Matlab/Simulink to simulate discrete network process control.

Every TrueTime toolbox simulation scheme should contain three crucial parts: TrueTime kernel (computer, I/O device or some embedded system), TrueTime network (network model) and a controlled process. There is also an optional part TrueTime battery (all blocks are shown on Figure 3.2).

TrueTime kernel is responsible for I/O and network data acquisition or data processing and calculations. It can realize a control algorithm/logic and it is the "brain" of every device. It uses several simple M-files (modified by us to satisfy our needs) which handle all mentioned operations. In the kernel can be executed several independent tasks (periodic, non-periodic) which can cooperate on the same goal.

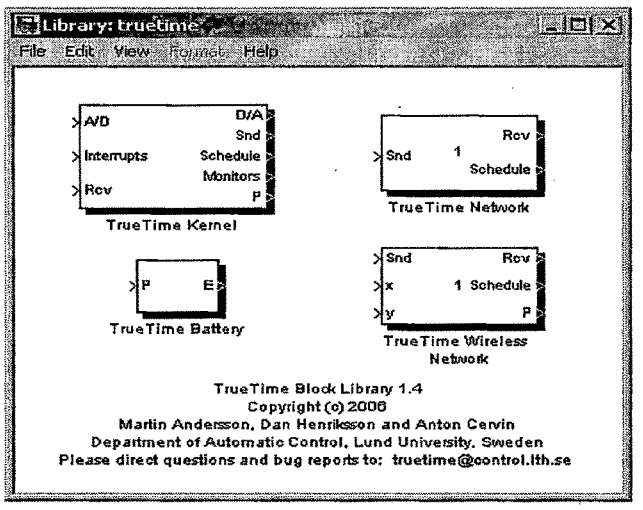

Figure 3.2: The TrueTime toolbox

One of its main features involves the possibility of co-simulation of the interaction between the real-world continuous dynamics and the computer architecture in the form of task execution and network communication. Some fundamental features in the context of this work include [37]: *i*) emulation of a wireless network and transmission

models; *ii*) representation of the kernel of a mobile node communicating through the implemented network; *iii*) representation of battery-powered devices and some basic consumption models.

In contrast to other co-simulation tools such as State flow/Simulink or Ptolemy II [38], TrueTime is not based on a mathematical modelling formalism. Rather, a TrueTime simulation is programmed in much the same way as a real embedded system. The application is written in Matlab code or in C++. The main difference from real programming is that the execution/transmission times must be specified by the developer. This approach makes TrueTime a very flexible co-simulation tool. Also, the step from simulation code to production code is not that large. The main drawback is that the simulation models are not amenable to analysis.

3.2.9 OPNET

Optimised Network Engineering Tools (OPNET) [39] is an event-driven, network simulation tool, which allows an easy implementation of the all model elements. A Graphical User Interface supports the configuration of the scenarios and the development of network models. Three hierarchical levels for configuration are differentiated: network, node and process. The network domain consists of nodes, links and subnets. A node represents a network device and groups of devices, i.e. servers, workstations etc. and WLAN nodes, IP clouds etc. Links represent point-to-point and bus-links between the nodes. The node domain covers the building of blocks, also referred to as modules, including processors, queues and transceivers as well as the specification of interfaces between the modules. The process model defines the underlying protocols, is represented by finite state machines (FSMs) and is created with states and the state transitions of the node model elements. It abstracts the behavior of the network element. The packet format generator allows building any packet consisting of a real byte oriented packets on named unsorted fields. The packets definition can follow exact protocol specifications. It is easy to deploy network elements in the project editor. All parameters can be configured easily. OPNET includes available tools for link setup and mobility profiling. Simulation results can be processed and analyzed with advanced functions. The analysis of simulated data is supported by a variety of built-in functions. The source code is based on C/C++.

To build a network model the workflow centres on the Project Editor. This is used to create network models, collect statistics directly from each network object or from the network as a hole, execute a simulation and view results. See Figure 3.3.

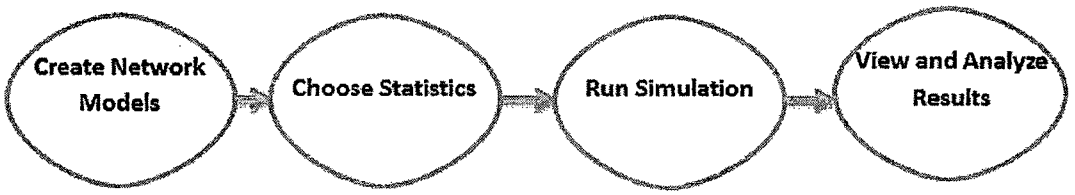

Figure 3.3: Workflow

3.2.9.1 Specification Editors

OPNET Modeler supports [39] model specification with a number of tools, called editors, which capture the characteristics of a modeled system's behavior. Because it is based on a suite of editors that address different aspects of a model, OPNET Modeler is able to offer specific capabilities to address the diverse issues encountered in networks and distributed systems. To present the model developer with an intuitive interface, these editors handle the required modeling information in a manner that is parallel to the structure of real network systems. Therefore, the model-specification editors are organized hierarchically. Models built in the Project Editor rely on elements specified in the Node Editor; in turn, when working in the Node Editor, you use models defined in the Process Editor and External System Editor. The remaining editors are used to define various data models, typically tables of values that are later referenced by process- or node-level models. This organization is reflected in the following list.

- Project Editor—Develop network models. Network models are made up of subnets and node models. This editor also includes basic simulation and analysis capabilities.
- Node Editor—Develop node models. Node models are objects in a network model. Node models are made up of modules with process models. Modules may also include parameter models.
- Process Editor—Develop process models. Process models control module behavior and may reference parameter models.
- External System Editor—Develop external system definitions. External system definitions are necessary for cosimulation.
- Link Model Editor—Create, edit, and view link models.
- Packet Format Editor—Develop packet formats models. Packet formats dictate the structure and order of information stored in a packet.

- ICI Editor—Create, edit, and view interface control information (ICI) formats. ICIs are used to communicate control information between processes.
- PDF Editor—Create, edit, and view probability density functions (PDFs). PDFs can be used to control certain events, such as the frequency of packet generation in a source module.

3.3 Network Simulation using OPNET

As networks are being upgraded from scratch all over the word, network planning is becoming most important. Computing the viability and performance of networks in real can be very expensive and painstaking task. To ease and comfort the process of estimating and predicting a network techniques are widely used and put into practice. A variety of simulation tools (discussed above) like QualNet, NS-2, Matlab and OPNET are available for the purpose of modelling and simulation but the choice of a simulator depends upon the features available and requirements of network application. Among the various network simulators, OPNET provides the industry's leading environment for network modelling and simulation. It allows to design and study communication networks, devices, protocols, and applications with flexibility and scalability. It provides object oriented modelling approach and graphical editors that mirror the structure of actual networks and network components.

The analysis in [40] helped to estimate and optimize the performance of wired and wireless networks using proposed optimization techniques. In [41], performance of wireless and wired networks as well as comparison is evaluated using OPNET simulation tool.

3.4 Simulation Setup and Results

Simulation was carried out for wired, wireless and hybrid network. For wired network, collision count, traffic received, delay, throughput is studied while for wireless network, data dropped, traffic received, media access delay, and throughput is studied. For comparison of both wired and wireless networks, the performance parameter, throughput is investigated. All these performance is carried out by varying number of users. For hybrid network, delay and throughput are investigated for both wired and wireless network part by varying traffic in terms of packet bytes.

3.4.1 Wired and Wireless Network Comparison[1]

The Ethernet is a multi-access network, meaning that a set of nodes sends and receives frames over a shared link. It implements the capability of transmitting and monitoring a connected bus link at a same time. It has full duplex capability. Here Ethernet network model with star topology and data rate of 10Mbps using OPNET [39] with 25, 50,100 users is setup. Scenario for wired network, created for Ethernet using 50 nodes, is shown in Figure 3.4(a). The Wireless LAN model suite includes the features of the IEEE 802.11 operating at a data rate of 10Mbps in a star topology using OPNET Modeler 14.5 is as shown in Figure 3.4(b). The network also expanded for the 25 and 100 users.

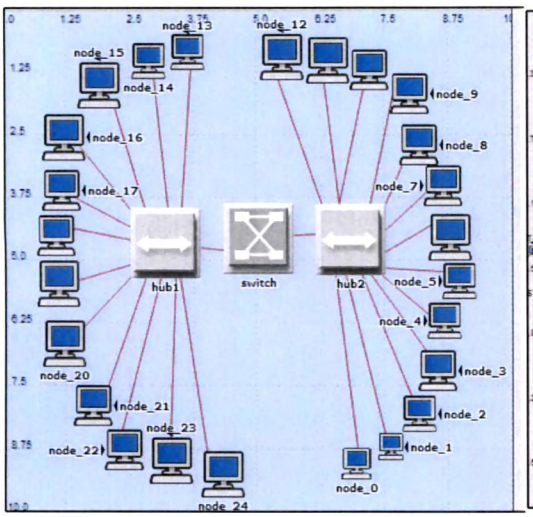

Figure 3.4(a) : Ethernet network model for 50 Ethernet stations

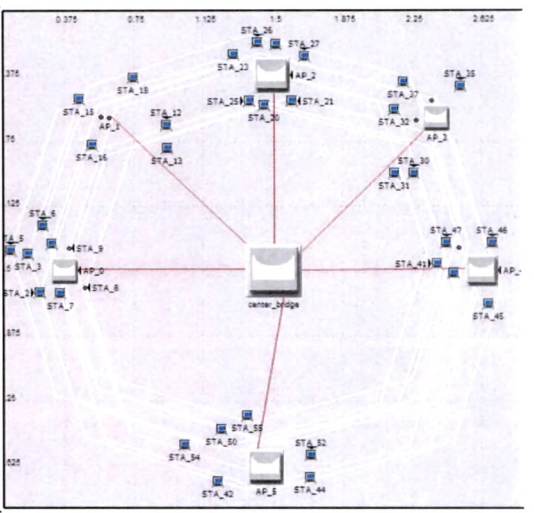

Figure 3.4(b) : Wireless network model for 50 users

The performance metrics evaluated are throughput, delay, data dropped, traffic received, collision count, retransmission attempts [39].

- **Throughput:** Network throughput is the average rate of successful message delivery over a communication channel. It is measured in bits per second (bit/s or bps) or in data packets per second or data packets per time slot.

1

Published a paper, Ms. Sonal J. Rane, Prof. Satish K. Shah, Ms. Dharmistha D Vishwakarma "Performance Evaluation of Wired and Wireless Local Area Networks", *International Journal of Engineering Research and Development ISSN: 2278-067X, Volume 1, Issue 11 PP.43-48, July 2012. www.ijerd.com.*

- **Retransmission attempts:** Total number of retransmission attempts by all WLAN MAC in the network until either packet is successfully transmitted or it is discarded as a result of reaching short or long retry limit.
- **Collision Count**: Total number of collisions encountered by this station during packet transmissions.
- **Data Dropped:** Total higher layer data traffic (in bits/sec) dropped by the all the WLAN MACs in the network as a result of consistently failing retransmissions. This statistic reports the number of the higher layer packets that are dropped because the MAC couldn't receive any ACKs for the (re)transmissions of those packets or their fragments, and the packets' short or long retry counts reached the MAC's short retry limit or long retry limit, respectively .

From the Figure 3.5 it is observed that the received bit rate for wired network is approximately equal to the sent bit rate for small number of users. As the number of user increases, more traffic was sent and received. As the number of users increases the hub switch becomes overloaded and cannot deliver all the traffic that it received.

Figure 3.5(a) : Traffic sent (bits/sec) of different scenarios having 25,50,100 users

Figure 3.5(b) : Traffic received (bits/sec) of different scenarios having 25, 50, 100 users

Discrepancies between send and receive rates can be accounted for by inspecting the collision count statistic as shown in Figure 3.6. In Figure 3.7, it is observed that maximum throughput is achieved in the case when less number of users is deployed. As the number of users increases, more number of the higher layer packets that are dropped because the MAC couldn't receive any ACK for the (re) transmissions of those packets or

their fragments, and the packets' short retry limit or long retry limit, respectively. With less number of users, the overall performance of the system increases as data transmission will be faster.

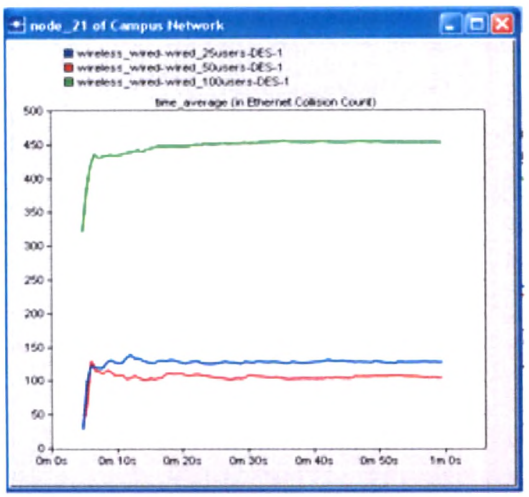

Figure 3.6 : Collision count in Wired Network

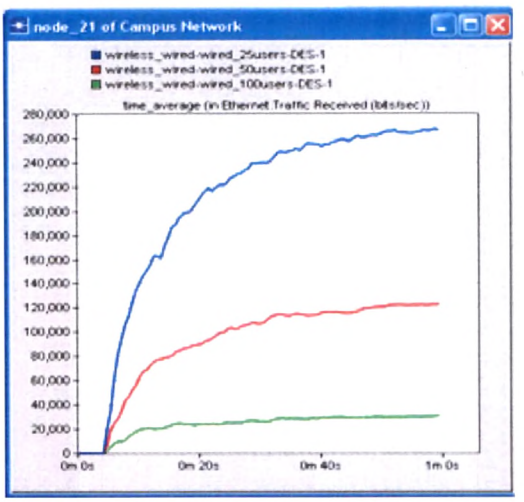

Figure 3.7 : Throughput (bits/sec) of Different scenarios on node 21 for Ethernet

As the number of users increases in WLAN, data dropped in wireless LAN increases. Some of the packets that were sent collided and require retransmissions. So as the users increases, retransmission rate increases for the more number of users which is shown in Figure 3.8(a) and Figure 3.8(b) respectively.

Figure 3.8 (a) : Data dropped for different scenarios for W LAN

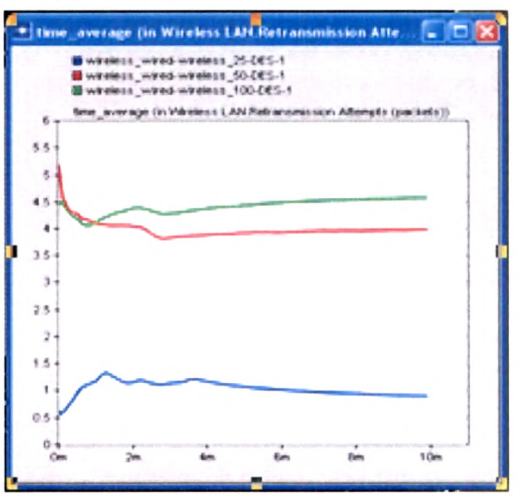

Figure 3.8(b) : Retransmission attempts of different scenarios for W LAN

As the number of user increases, collision between the user data increases and the retransmissions of the user increases which cause for the degradation in the performance

of wireless network. It can be observed from Figure 3.9 that as the number of users increases from 25 to 100 the performance of WLAN decreases.

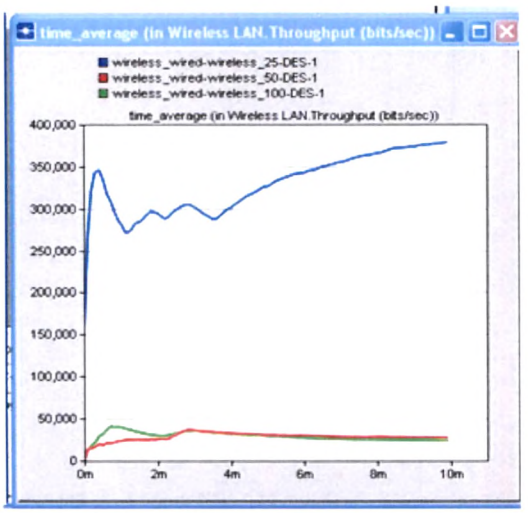

Figure 3.9 : Throughput (bits/sec) of different scenarios for WLAN

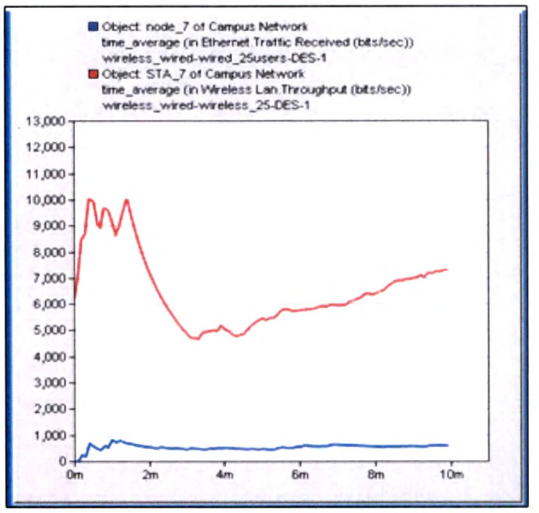

Figure 3.10(a) : Throughput of Ethernet and WLAN on node 07

Throughput comparison of wired and wireless network is performed for the number of users of 25, 50, and 100 with the data rate of 10 MBPS for both the network. The performance of the networks measured using performance of the random node present in the network. While comparing throughput of Ethernet (wired) in blue color and WLAN (wireless) in red color in Figure 3.10(a), (b), (c) for 25 users on node_07, for node 37 in 50 users scenario and node 97 for the scenario with 100 users respectively, with less number of users, the overall performance of the system increases as data transmission will be faster.

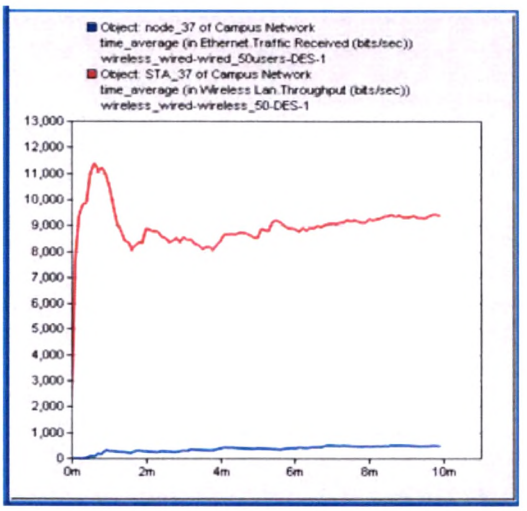

Figure 3.10(b) : Throughput of Ethernet and WLAN on node 37

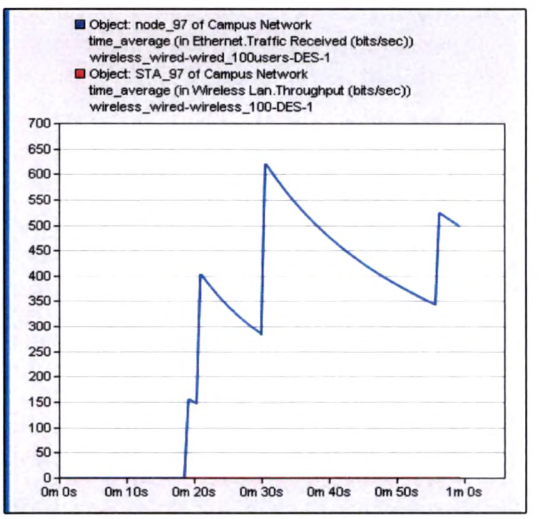

Figure 3.10(c) : Throughput of Ethernet and WLAN on node 97

Also it is observed that the throughput of Wireless LAN is greater than throughput of Ethernet for less number of users. As the number of user increases, throughput of WLAN becomes poor because of slow transmission speed and data dropped. When the number of users is increased, the throughput decreases for wireless systems, these effects associated with wireless transmission limit the SNR (Signal to Noise Ratio) and bandwidth of the received signal, and therefore the maximum number of bits that can be sent.

3.4.2 Hybrid Network

As discussed in chapter 2, the interoperability in heterogeneous networks with hybrid structure is in doubtfully a major requirement, when implementing communication scenarios for home and industrial applications. As IEEE 802.3 and IEEE 802.11 are becoming the most common and widely used LAN/WLAN standards, an interoperable architecture is required in order for communication to be treated transparently at the higher-levels. However, it has to be pointed out that from the user point of view, the entire system is seen as a black box and is expected to function equally well, independently from network heterogeneity.

It is already discussed that Hybrid networks implementation uses both ad hoc connectivity and access points. Access points receive or transmit data between both wired and wireless networks. In short, wireless networks don't replace wired networks because some wiring is still required to deploy even a simple model so an implementation of a simple hybrid network is done to evaluate the performance of global network performance and QOS tenability.

3.4.2.1 Model Outline

The proposed scenario consists of a wireless and a wired network (hybrid network). Figure 3.11 shows the structure of the model or the deployment. The purpose of the scenario is to demonstrate the inter-communication between the wireless and wired network.

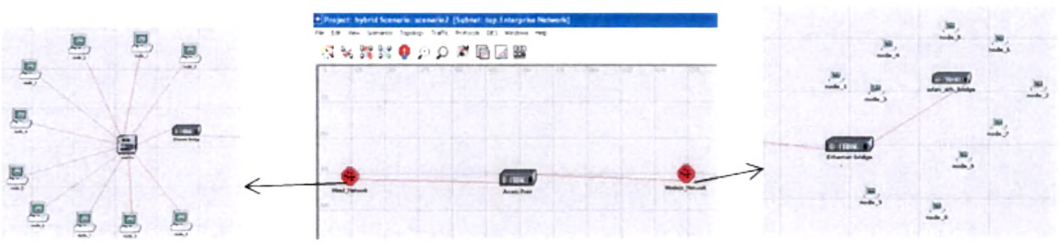

Figure 3.11: Hybrid Network

Two data traffic patterns were simulated; heavy traffic and light traffic (by adjusting packet size in bytes) as shown in Table 3.2.

Traffic Type	Packet Size (in bytes)
Light Traffic	1000
Heavy Traffic	10000

Table 3.2 Traffic Specification

As shown in Figure 3.12(a) and (b), it is obvious that the throughput for light traffic in wired and wireless network performance same but for heavy traffic, throughput in wired network is somewhat greater than wireless network part by considering segmentation.

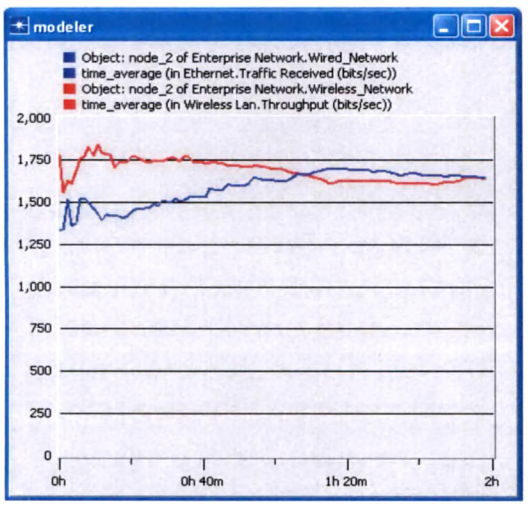

Figure 3.12(a) : Light Traffic

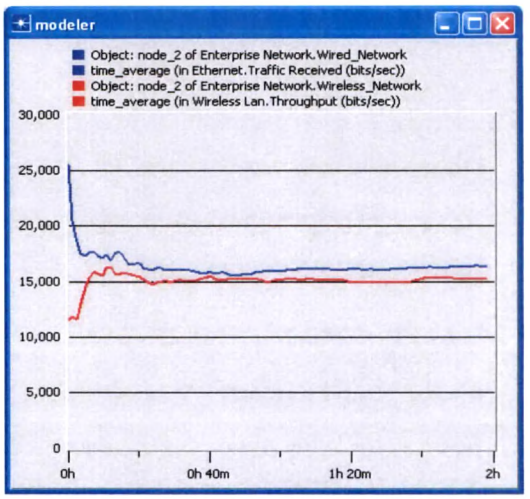

Figure 3.12(b) : Heavy Traffic

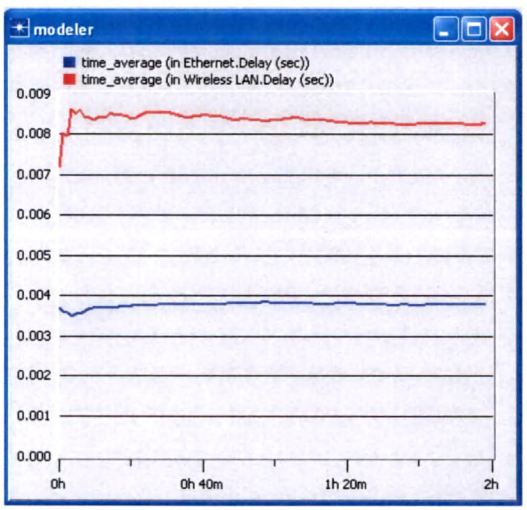

Figure 3.13 (a) : Delay for Light Traffic

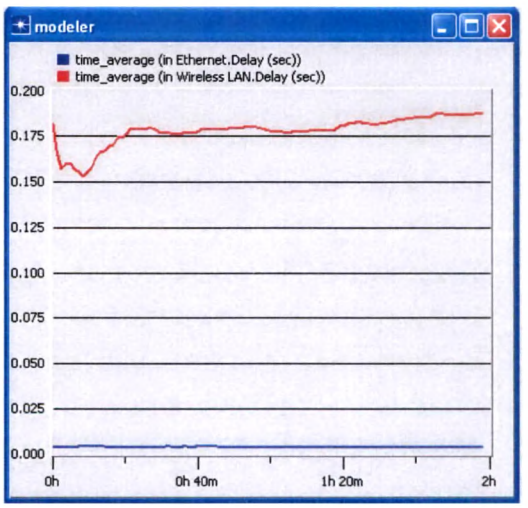

Figure 3.13(b) : Delay for Heavy Traffic

Same thing can be observed in below Figure 3.13(a) and (b), analysing delay for wired and wireless network part. For light as well as heavy traffic there is less delay in wired network.

Summary:

In this chapter, different network simulators like NETWORK SIMULATOR NS-2, OPNET, OMNET++, GLOMOSIM, and QUALNET are surveyed. OPNET simulation tool was used to evaluate the performance of the Wireless and Wired Network in terms of different number of users, traffics. For high traffic and users wired network outperforms than Wireless network due to the transmission limit, SNR (signal to noise) and bandwidth of the received signal. So to improve the overall performance of the system it is better to use hybrid network which is the combination of both wired and wireless network.