

Characterization of Selfish Behavior in Mobile Ad Hoc Networks through Virtual Emulation

Jawwad Nasar Chattha

Thesis submitted
to the Faculty of the Virginia Polytechnic Institute and State University in
partial fulfillment of the requirements for the degree of
Master of Science
In
Electrical Engineering

Dr. Luiz DaSilva

Dr. Scott F. Midkiff

Dr. Mohamed Eltoweissy

September 3rd, 2009

Falls Church, Virginia

Keywords: (Emulation, Virtualization, Generous tit for tat)

Characterization of Selfish Behavior in Mobile Ad Hoc Networks through Virtual Emulation

Jawwad Nasar Chattha

(ABSTRACT)

Unlike infrastructure-based networks, mobile ad hoc networks consist of nodes independent of any infrastructure. Cooperation among these nodes is essential for the sustenance of multi hop communication. However, battery and bandwidth constraints may lead nodes in an ad hoc network to adopt energy- and bandwidth-conserving strategies. As routing and packet forwarding are end results of cooperation, network performance is affected when nodes in the network behave selfishly to conserve their resources.

Our work involves characterizing selfish behavior by nodes in ad hoc networks and assessing the effectiveness of adopting tit for tat based strategies, which are meant to discourage selfish behavior in the network. We show that in an ad hoc network where other nodes act selfishly to conserve their resources, a node can benefit by adopting a generous tit for tat strategy. We also show that a node can gain benefit by avoiding selfish nodes in an ad hoc network, adopting a strategy that we call generous tit for tat with selfish avoidance (GTFT-SA).

To analyze the effectiveness of cooperation strategies in selfish ad hoc networks we create an emulation environment based on virtualization. Such an emulation environment is more flexible to changes and is simpler to replicate than real life testbeds, while providing higher fidelity than simulations.

Acknowledgments

I would like to dedicate this manuscript to my wonderful family. I could not have done this without them.

This is also dedicated to all my teachers throughout my educational career but especially to my current advisor Dr. Luiz DaSilva. I sincerely thank you for your patience and the time you dedicated for making this possible. You always had the time for discussing my queries and guiding me in the right direction.

Table of Contents

1	Introduction.....	1
1.1	Problem Statement.....	1
1.2	Motivation.....	2
1.3	Contribution.....	2
1.4	Organization.....	3
2	MANET Testing Techniques	4
2.1	Analytical Modeling	4
2.2	Network Simulation	5
2.3	Network Emulation	6
2.3.1	Categorization of Emulators.....	7
2.3.1.1	Degree of Emulation	8
2.3.1.2	Control	9
2.3.1.3	Mobility.....	10
2.3.1.4	Wireless Medium Modeling.....	11
2.3.2	Emulator Examples	12
2.4	Real World Testbeds	14
2.4.1	Testbed Examples.....	15
3	Emulation through Virtualization.....	19
3.1	Virtual Emulation	19
3.1.1	Virtual Network User Mode Linux (VNUML)	20
3.1.1.1	Networking under User Mode Linux.....	20
3.1.1.2	Parts of VNUML.....	21
3.2	Challenges of Virtual Emulation of Ad Hoc Networks.....	23

3.2.1	Mobility Emulation	23
3.2.1.1	MobiEmu.....	24
3.2.1.2	Mobility Models	25
3.2.2	Promiscuous Listening	26
3.2.3	Scalability.....	29
4	Characterizing Cooperation Strategies in MANETs.....	32
4.1	Reputation-Based Schemes	33
4.1.1	Detection and Analysis	33
4.1.2	Building Reputation	34
4.1.3	Punishment and Avoidance	34
4.2	Strategies Employed	35
4.2.1	Generous Tit for Tat (GTFT)	35
4.2.2	Generous Tit for Tat with Selfish Avoidance (GTFT_SA).....	36
4.3	Emulation Setup.....	36
4.4	Results of Experimentation.....	37
4.4.1	Overall Percentage of Packets Forwarded	37
4.4.1.1	NS2 Generated Random Waypoint Mobility	38
4.4.1.2	MANIAC Challenge Traces.....	41
4.4.2	Packets Forwarded by Nodes Using GTFT Strategies	44
4.4.2.1	NS2 Generated Random Waypoint Mobility	45
4.4.2.2	MANIAC Challenge Traces.....	47
4.4.3	Percentage Packets Received	50
4.4.3.1	NS2 Generated Random Waypoint Mobility	50
4.4.3.2	MANIAC Challenge Traces.....	55
4.4.4	Percentage Observation Error	60

5	Conclusion and future work	62
5.1	Conclusions	62
5.2	Future Work.....	63
References	64

List of Figures

Figure 3.1: Networking with UML.....	21
Figure 3.2: VNUML Language.....	22
Figure 3.3: MobiEmu Architecture.....	25
Figure 3.4: Packet Traffic from Node A to Node C	27
Figure 3.5: Packet Traffic from Node A to Node E.....	28
Figure 3.6: Emulator Setup.....	30
Figure 4.1: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.	38
Figure 4.2: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.	39
Figure 4.3: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.	40
Figure 4.4: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.	41
Figure 4.5: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces.	42
Figure 4.6: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces.	42
Figure 4.7: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces.	43
Figure 4.8: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces.	44
Figure 4.9: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.	45

Figure 4.10: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.	46
Figure 4.11: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.	46
Figure 4.12: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.	47
Figure 4.13: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces.	48
Figure 4.14: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces.	48
Figure 4.15: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces.	49
Figure 4.16: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces.	49
Figure 4.17: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.	51
Figure 4.18: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.	52
Figure 4.19: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.	52
Figure 4.20: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.	53
Figure 4.21: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.	53

Figure 4.22: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility. 54

Figure 4.23: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility. 54

Figure 4.24: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility. 55

Figure 4.25: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces. 56

Figure 4.26: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces. 56

Figure 4.27: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces. 57

Figure 4.28: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces. 57

Figure 4.29: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces. 58

Figure 4.30: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces. 58

Figure 4.31: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces. 59

Figure 4.32: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces. 59

Figure 4.33: 4 nodes act as traffic generators; remaining nodes adopt GTFT_0.1 strategy; nodes follow the ns2 generated random waypoint mobility model. 61

1 Introduction

Mobile ad hoc networks (MANETs) comprise autonomous nodes that are independent of any infrastructure. Such autonomous nodes require cooperation for routing and packet forwarding to work as a network. Thus cooperation between these independent autonomous entities is the binding force keeping the network functioning. As the nodes in such a network are mobile, they face battery and bandwidth constraints, and thus they may be driven to energy- and bandwidth-conserving selfish or malicious behavior. Any misbehavior in the form of malicious or selfish strategies exhibited by these nodes can result in severe degradation of network performance and can even cause the network to break down.

Characterizing and avoiding such selfish or malicious behavior is thus essential for the network to survive or function properly, but due to the lack of any infrastructure the detection and avoidance from such misbehavior is not easy.

1.1 Problem Statement

In this thesis, we characterize selfish behavior in a MANET and assess the effectiveness of both a generous tit for tat (GTFT) strategy and a generous tit for tat with selfish avoidance (GTFT_SA) strategy operating in such a selfish environment. The GTFT strategy is based on the concept of reciprocating the behavior that others are exhibiting towards you. With GTFT_SA nodes also try to avoid selfish nodes by redirecting traffic to its final destination through more cooperative nodes in the network.

After assessing the level of cooperation among the nodes in the network using a reputation-based mechanism, the generous tit for tat cooperation strategy can be deployed to discourage selfish behavior.

We show that in an ad hoc network where some nodes are exhibiting selfish behavior toward others to conserve their energy and bandwidth resources, a node can benefit by using a GTFT strategy. Further benefits can accrue from adopting a GTFT_SA strategy.

1.2 Motivation

Much of the work on the performance evaluation of MANETs involves either simulation or real life testbeds. In contrast, we create an emulation environment to analyze the effectiveness of cooperation strategies in a MANET. We argue that such an emulation environment is more flexible to changes and is simpler to replicate than real life testbeds, while providing higher fidelity than simulations.

Emulation provides a controllable and repeatable environment to build new cooperation strategies and to characterize existing cooperation strategies in a selfish ad hoc networking environment.

1.3 Contribution

Our main contribution is to build a testing environment for cooperation strategies in a MANET. As the emulator replicates the network stack in every node, it can be used as a general pretest tool for most protocols in the networking stack working above the data link layer. Thus most protocols for transport, networking and even application layer can be ported and tested in this emulator. With this emulation environment for MANETs a compromise between simulation and real life experiments is achieved.

Our work also involves an assessment of generous tit for tat as a cooperation strategy and of the benefits of adding selfish avoidance to generous tit for tat. The network performance is measured by determining the percentage of packets forwarded by every node and the amount of traffic correctly received by the each node.

1.4 Organization

This thesis is divided into five chapters. Chapter 2 presents related work on different MANET testing techniques. Chapter 3 describes our proposed work on emulation through virtualization. It also highlights the challenges of virtualized emulation. Chapter 4 characterizes cooperation strategies in a selfish MANET environment. This chapter explains the different cooperation strategies implemented and presents and discusses results from the experiments. Chapter 5 concludes by discussing the contribution of the work and outlining potential future work.

2 MANET Testing Techniques

Performance analysis of a protocol in a mobile ad hoc network (MANET) involves the re-creation or modeling of the network with realistic conditions of transmission range, contention for medium access, mobility of nodes and network traffic. This chapter briefly describes some of the performance evaluation techniques employed to characterize and evaluate the behavior of MANETs.

Whether it is a new protocol or an improvement to an old protocol in MANETs, it has to undergo performance evaluation based upon one the below mentioned methodologies before it is deployed at large. These methodologies are divided into four groups:

1. Analytical modeling;
2. Network simulation;
3. Network emulation; and
4. Real world experimentation and testbeds.

These are discussed in more detail in the next subsections.

2.1 Analytical Modeling

“Analytical models are a set of equations describing the performance of a system.”[1]

Although comprehensive analytical or mathematical models for entire wireless ad-hoc networks are often intractable, they may provide a first-order abstraction of the system bounded by some specific conditions. Various characteristics of MANETs are

modeled independently in [2], [3] and [4]. For example, de Carvalho in [2] presents a model for capturing the interaction between the physical (PHY) and medium access control (MAC) layers, along with the interference between nodes. Similarly, de Carvalho and Garcia-Luna-Aceves [3] present an analytical model of wireless ad hoc networks that considers the impact of antenna-gain patterns on network performance. Thus it allows the study of ad hoc networks in which nodes are equipped with directional antennas. Cagalj et al. [4] employ game theory to predict the impact of selfish behavior by nodes on the throughput in wireless networks.

Analytical models can also be used to reduce the complexity of problem. Junghoon [5], gives such an example where network formation of large scale mobile ad-hoc networks is modeled using the K-Means Clustering Algorithm. Network formation is carried out by exchanging topology control messages based on link connectivity information. For a large network consisting of thousands of nodes, it is infeasible to simulate network formation due to processing power requirements and the time consumed. However, if aspects of the problem are first modeled analytically, these aspects can then be abstracted, enabling the process to be simulated in reasonable amount of time using a typical simulation tool.

Thus these analytical modeling techniques often become the seed for the simulation process.

2.2 Network Simulation

“Simulation is duplication or reproduction of certain characteristics of a system or physical process by use of analytical models for study or training purposes.” [6]

In a simulation, the influencing factors and the algorithms that are to be investigated are modeled and examined in a software environment with a high degree of abstraction. This allows repeatability, tight control, large scale and cost effective tests, possibly with heterogeneous operating systems and programming languages. Most

network simulators use discrete event simulation, in which a list of events is stored, and those events are processed in order, with some events triggering future events, such as the event of the departure of a packet at one node triggering the event of the arrival of that packet at a downstream node.

Commonly used network simulators include NS3, Qualnet, OMNET++ and OPNET Modeler.

On the downside, simulations lack realism, and inaccurate assumptions may lead to test results that do not reflect the behavior of a real world implementation. Furthermore the simulator software may contain non-standard, simplified protocol implementations. Therefore, most results produced by simulation should be considered as qualitative assessments. Even if a simplified model appears to be a good approximation for evaluating a specific MANET protocol, there are no assurances that the model will be valid for other routing protocols, or even the same protocol under different experimental conditions. This is highlighted in [7] and [8], where accuracy of simulation results is discussed. Here authors compare several popular simulation tools and conclude that simulating a particular protocol using different simulators can produce different results.

Due to the above mentioned problems in simulation, some guidelines are presented in [9] for carefully describing experimental study scenarios and understanding assumptions made by a particular simulator.

2.3 Network Emulation

“Emulation is a combination of software and hardware which simulates properties of an existing, planned or non-ideal network in order to assess performance and predict the impact of change.” [10]

In emulation, hardware and software designed for real-world deployment are modified and combined with simulation components to run under controllable laboratory

conditions. Unlike simulation, which is a closed environment, emulation allows the user to interact with the physical interfaces or physical networks. The advantage over simulation is an increased degree of realism, with comparable repeatability and tight control. The costs per tested node are higher than simulation. Also, emulation is not as scalable as simulation.

There are several emulators for wired and wireless networks, but we focus on emulators for wireless ad hoc networks. The purpose of these emulators varies: some are built to allow the testing of protocols on real hardware, while others are used to replicate real-world network environments. In the latter case, emulation is used to form a dynamic topology among the nodes. This allows easy in-lab testing without physically moving the nodes around for a full-scale experiment.

Emulators can be categorized according to their degree of emulation, control, support for mobility, and how they model wireless links. Furthermore the emulation is characterized by the degree of virtualization employed by the emulator. Emulation through virtualization is realized by setting up parallel network stacks on a single Operating System (OS), e.g., as employed in NEMAN [11]. In contrast, real world testbeds have a one-to-one node mapping, where each node is a single emulation machine. With virtualization we can host multiple virtual nodes on test runs with multiple nodes in a controllable environment.

2.3.1 Categorization of Emulators

Our categorization of MANET emulators includes four key aspects influencing the functional properties of MANETs: which layers are emulated, the control mechanism, mobility modeling and wireless medium modeling. The chosen implementation of the above properties determines the limitations and capability of the emulator in terms of fidelity, scalability, and performance.

2.3.1.1 Degree of Emulation

Physical Layer Emulator

In a physical layer emulator, all network layers except the physical layer are implemented in a real system. Here only the physical layer is emulated. Physical layer emulators modify the radio signal emitted by the wireless interfaces of the nodes to replicate the effects that the radio waves would experience in a real-world setup. In the EWANT [12] emulator the emitted signal is attenuated to scale down the physical area required for experiments. The output of an antenna after attenuation is multiplexed by using a 1:4 RF multiplexer. Each node thus has four antennas attached to it. By placing these antennas at suitable positions and choosing which of these four antennas is used for transmission, mobility is emulated on a physically static network. The ORBIT [13] lab emulator is based on a similar technique as EWANT: it scales the radio range by transmitting at low power levels and emulates movement by changing connectivity of a large 2-dimensional array of radio nodes. Thus in both EWANT and ORBIT by switching between different antennas, the physical characteristics of the medium change and physical layer is emulated.

MAC Layer Emulator

In a MAC layer emulator, all network layers except the MAC and physical layer are implemented in a real system. MAC layer emulators simply determine the nodes that should receive a given frame. Filters are used to either allow or drop a certain frame. If a node is emulated to be within radio range of another node, a filter tool allows the exchange of frames between them, and if the nodes are out of each other's range, the respective frames are dropped. Filters are placed in either a central machine or they are dispersed into the whole network where every node contributes to controlling the network topology. In a centralized system all nodes send their frames to a central machine, which then determines the nodes that should receive the frames. Decentralized emulators distribute this decision making into the network and are based on available network filter tools such as IPTABLES or specifically designed filter tools such as APE mackill [14].

2.3.1.2 Control

Centralized control

Centralized control is achieved via a central server emulating the node mobility or the state of the wireless medium. Every node in the emulated network sends traffic to this central core server, where the decision is made whether to forward, drop or alter the packet based on the network topology and the conditions of the wireless medium at the moment. For example, the central server of JEmu [15] creates connectivity on a frame level based on the current topology and on current conditions of the wireless medium. NEMAN [11] emulator also has a centralized control mechanism. This technique is simple to implement but has the drawbacks of creating a bottleneck and putting too much processing load on the central server. MobiNet [16], on the other hand, introduces multiple core control nodes which help in sharing the processing load and in mitigating of the bottleneck created by a central control entity.

Distributed control

Distributed control does not involve one central node in the decision making for processing a packet. Every node in the network is able to decide whether it will forward, drop or alter the packet. Every node in the network, instead of forwarding the traffic to a central node, processes the traffic independently, forwarding or dropping it. This decision is made either based on the current topology of the network or based on a pre-calculated scenario file stored on the node. MobiEmu [17] nodes control their packets using filtering tools based on timing information provided by a central server. The central server synchronizes the nodes in the network by broadcasting timing information, which is used by other nodes to implement the filters for that particular time from a pre-stored scenario file. EMWIN [18] and EMPOWER [19] also determine basic on/off connectivity from a predefined event or scenario list.

2.3.1.3 Mobility

Mobility of the network nodes to emulate real network behavior is achieved using three different techniques: real mobility, channel emulation, and logical connectivity.

Real Mobility

Real mobility is introduced by altering the physical position of the nodes by carrying them or using robots or by switching antennas between fixed propagation locations. DAWN [20] is an example of such an emulator, where movement patterns are terrain and user specific. Real mobility is achieved either in a predefined or in a random fashion. To be able to replicate the mobility of nodes for a certain test, an additional tracking system is required. For example, nodes may be equipped with a GPS receiver to achieve accurate positioning. Even using a tracking system it is still difficult to replicate a certain mobility pattern because of changing weather conditions or the presence of irregular obstacles.

Mobility by Channel Emulation

Channel emulation is based on stationary nodes, whose radio signals are altered to emulate the properties of a time varying radio channel. In EWANT [12] the nodes are stationary and mobility is achieved by switching radios according to a scenario file. Each node in the emulator has four external antennas attached. These are placed such that when switching takes place between these antennas connectivity with other nodes changes. Thus this on/off connectivity with other nodes emulates mobility. ORBIT [13] applies a similar antenna switching approach, and is based on a 400-radio node grid.

Mobility by Logical Connectivity

Mobility modeling thus far discussed is based on physical layer manipulation. However, radio frequency solutions are costly due to the additional equipment needed, such as attenuators and of the additional antennas as used in EWANT and ORBIT. Another approach to overcome these limitations is to apply logical connectivity on a

packet or frame level. While the physical nodes are stationary, as in channel emulation, a logical connectivity matrix is calculated from the scenario description or the virtual distances between nodes. Filtering is then used to block or accept packets from certain nodes. The testbeds JEmu [15], MobiNet [16], EMWIN [18], EMPOWER [19], NEMAN [11], MobiEmu [17] and MNE [21] support logical connectivity.

2.3.1.4 Wireless Medium Modeling

Emulation also requires modeling the wireless medium to include the effects of physical interaction between nodes. Emulators can be classified based upon how the radio channel, co-channel interference, and medium access mechanism are replicated.

Modeling the Radio Channel

Physical channel undergoes variations which are time based, dependent upon node location/mobility and the surrounding wireless environment. Radio link emulation models these varying characteristics of the channel, such as shadowing, free space attenuation, slow and fast fading, and scattering. As some emulators are scaled down version of the actual network, by employing attenuators, they already capture these radio channel effects, as in EWANT [12]. Other emulators such as MNE [21] and MobiEmu [17] are built on a real IEEE 802.11 network so they naturally incorporate the channel properties. Some also apply traffic shaping, as is used by Net [22], which adds radio channel effects on packets flowing through its emulated nodes.

Modeling Co-Channel Interference

Co-channel interference is the phenomenon where two nodes using the same frequency band experience interference. Emulators that use the physical layer without any changes incorporate co-channel interference naturally. MobiNet [16] emulates co-channel interference using the logical connectivity approach. In that case, the centralized emulation controller drops a frame if the virtual signal strength of two colliding frames is below a certain threshold.

Modeling Medium Access

Emulators sometimes employ a wired network to emulate wireless behavior, hence careful planning is required to model medium access into the emulator. In addition to access delays and dynamic bandwidth limitations, link layer functionalities such as backoff timers and frame collisions have to be considered. In a centrally controlled emulator, as all the traffic is directed to a central controller, these properties are controlled by the core node based on a scenario file, as in JEmu [15]. In a distributed control scenario, traffic shapers are sometimes added on each emulated node to incorporate these effects.

2.3.2 Emulator Examples

Some network emulators are described below.

EWANT [23] is an emulator which scales the test environment by using an attenuator. This significantly reduces the space requirements for complex experiments and greatly increases the ease of experimental setup. EWANT adds a 1:4 multiplexer to the attenuators, using four antennas for each node. These antennas are placed such that when a node switches between any of these antennas, the network topology changes. Thus EWANT emulates mobility of nodes by selecting between differently placed antennas for transmission.

Another focus in building systems to be used for emulation-based experiments is to allow multiple researchers to share the resources available. This problem has been targeted in many different systems, such as ORBIT, PlanetLab, and MIT RON. Some of the projects only aim for providing a set of resources for experimenters to build their own emulation-based experiment setup. ORBIT [13], for example, emulates mobility using spatial switching, providing an excellent resource for testing in 802.11 networks. It consists of an indoor radio grid emulator and an outdoor network testbed. The indoor emulator does not require any physically moving parts as it emulates mobility by

switching over an array of spatially distributed radios, whereas the outdoor network is a complete testbed discussed later in the chapter.

Some protocols have complex implementations, which makes re-implementing them for a simulation prone to errors and inaccuracies. For this reason, some researchers have made the real protocol implementations available as open source software, and packaged them in a way that can be incorporated in simulators. In ENTRAPID [24], the network stack from FreeBSD is packaged such that it works in the user level, so that protocol developers can experiment with their own protocols incorporated into the stack. The topology and the physical layer in ENTRAPID are simulated. This controlled environment allows developers to implement and verify their work before deployment in the field.

In ModelNet [25], the environment is divided into two sets of hosts called core nodes and edge nodes. Core nodes are used to subject the traffic to the bandwidth and congestion constraints, latency, and losses of the targeted network topology. The edge nodes are the real hosts whose traffic is routed through the virtual network. While ModelNet is targeted to wired IP networks, MobiNet is an extension of the same approach but it targets MANETs. MobiNet also enables the edge node to use virtualization to multiplex multiple edge nodes on a single machine.

Another approach used is placing traffic shapers between the protocol stack and the network device. This keeps the real protocol stack and programs running while the rest of the network is simulated. For example, in NET-Shaper [22], parameters such as bandwidth and delay are controlled by a user-space program. A similar approach is taken in EMPOWER [19], which targets wired IP networks, and EMWIN [18], which is based on EMPOWER but targets MANETs. Filters based on IPTABLES are also employed to simulate the connectivity between nodes in the network. As an example, MNE [21] is a distributed system which abstracts physical layer effects and mobility changes by IPTABLES-based packet filtering controlled from a central controller.

An alternative to using traffic shaper modules in the kernel is the use of the universal TUN/TAP driver, which is designed for implementing tunneling using user

level programs. With the developments that allow multiple operating systems to run on one base operating system, other approaches have recently become possible. User Mode Linux (UML) provides a virtual Linux kernel running in user mode. UML is used for implementing virtual nodes that are then connected by a virtual network. One of the examples of this approach is the Virtual Network User Mode Linux (VNUML) [26] tool, which is a networking application using User Mode Linux that allows running a Linux kernel with its own resources like memory, disk, process space, etc. as a normal process in the physical host. This technique will be further explained and explored in the next chapter.

Another original direction is explored by Haeberlen et al. in their system called Monarch [27]. In Monarch, the idea is to use the latency observed between the host on which the virtual sender and the virtual receiver resides and a remote host in the Internet. For this purpose, for every packet the virtual sender wants to send, Monarch captures it and sends a probe packet of the same size to a remote host associated with the virtual receiver. When a reply is received, the virtual receiver is allowed to receive the packet. In the direction from virtual receiver to virtual sender, Monarch passes the packets without delay. This way, both the sender and the receiver observe the round trip time obtained from the probe packet. However, only transport layer is targeted through this approach.

2.4 Real World Testbeds

“A real world testbed is a perfectly normal instance of the system under test (SUT) in a particular experiment, used for various experimental objectives such as collecting data to be interpreted or monitoring performance.” [28]

In a real world experiment, all parts of the system are fully functional in a real-world environment. The whole network is deployed and tested under realistic experimental conditions. Thus, no potentially wrong or inaccurate assumptions are made. Real-world experiments comprise all the effects present in a network and can provide

feedback for simulation or emulation. Furthermore, a real-world experiment is the ultimate way to prove that an algorithm or protocol works as expected. As propagation and mobility parameters are often out of the experimenter's hands, this makes it difficult to repeat experiments and to fully understand and correctly interpret their results. The drawbacks of real-world experiments are the lack of repeatability and tight control as well as the limited scalability mainly caused by high costs of hardware, software and manpower.

To minimize some of the above mentioned problems, testbeds are usually a scaled down version of the actual system under test. For example, when it comes to the testing of a routing protocol we can employ ten nodes and observe the effect of the routing algorithm and measure key performance metrics. From these results we can attempt to extrapolate how this same routing algorithm will perform when deployed in a real network. The results derived from testbed-based experiments are not always indicative in all dimensions relevant to real deployment of a network. Scalability is such a dimension that can not be judged based on the experiment conducted on a set of 10 nodes in our earlier example.

2.4.1 Testbed Examples

Some examples of wireless testbeds currently being used are given here.

Maniac Challenge [29] is a NSF funded challenge in the field of ad hoc networking. The idea behind this research challenge is to understand the cooperation and interoperability of nodes in an ad hoc environment. Participants join to form an ad hoc network and organizers generate traffic destined to participant nodes. Organizer nodes or source nodes generate equal amount of traffic destined to each participant. Participants compete to maximize the number of packets received, with the least amount of energy consumed. Thus the node with the least amount of packets forwarded for other nodes and having the maximum number of packets received that were destined to it wins. An open source API is developed to enable participants to manipulate the forwarding decision

made by the routing protocol. It provides functionalities like accepting the packet, dropping the packet or changing the next hop of the packet. Thus the node has to behave in an intelligent manner in order to survive these challenging network conditions.

MiNT [30] is a testbed that utilizes an attenuator to scale down testbed size. It adds attenuators to node antennas, thus enabling it to make use of the physical radio link, but in a physically reduced environment. It creates mobility by placing these nodes on remote controlled robots which follow a pre-defined path according to the scenario file. These nodes have a central controller which is used to issue commands and collect statistics during the experiment. As attenuators scale down the space requirements of the testbed, they lessen the effort required in setting up the network.

WHYNET [31] is a testbed that is a result of a multi-institution effort, and it is used for the evaluation of next-generation wireless protocols in a realistic and scalable manner. Beyond physical experimentation and simulation, the WHYNET framework also supports emulation that uses physical and simulated elements in different combinations. This flexibility between simulation, emulation and real testbed testing enables the researchers to achieve the desired tradeoff between the realism, degree of control, repeatability and cost per node.

The Network Research Testbed, NRT [32], at UCLA consists of about 20 laptops and 60 PDAs with 802.11 cards. Several ad hoc network protocols have been implemented in NRT, including unicast routing protocols such as DSDV and ODMRP, multicast routing protocols such as ODMRP multicast and DVMRP, IP protocols such as Mobile IP, network management protocols such as SNMP, and transport protocols such as TCP Reno and TCP Westwood.

The outdoor field network testbed of ORBIT [13] consists of 50 nodes located at Rutgers University. Each node is equipped with two mini 802.11 a/b/g cards, allowing evaluation of protocols in real life settings. After sanity checking on ORBIT's indoor emulator this testbed is utilized for real life scenario testing.

CMU Monarch [33] is a testbed designed and implemented to evaluate the performance of an ad hoc network in the field. The network consisted of moving car-mounted nodes and some stationary nodes.

Ritter et al. [34] have approached the problem of experimentation in Mobile Ad-hoc Networks by building a real world testbed that allows perform measurements and covers from the radio layer to the application layer. The hardware used is unconventional, consisting of an embedded device having a core controller (Motorola 68HC912) and I/O peripherals as well as different wireless and wired network connectors. The device is equipped with Bluetooth and a 433 MHz RF module.

Dawn [20], another approach to real life testbed experimentation, consists of embedded devices used to replicate ad-hoc networks and do experimentation. Two basic components of the testbed are DAWN switches and endpoints. The switches are wireless routers with the DAWN software and the endpoints are notebooks attached to the routers using Ethernet. The software is developed on a FreeBSD system. The topology is controlled manually by people carrying the nodes and moving around to form a network.

In MIT Roofnet [35], an experimental multi-hop 802.11b mesh network, real changing conditions for the wireless channel between the nodes distributed on the roofs of some buildings near the MIT campus are used for testing routing in wireless networks. Roofnet consists of about 50 nodes.

In Ad hoc Protocol Evaluation testbed, APE [14], a software package is installed on each node, with the goal of simplifying testing in ad hoc networks. It contains an execution environment, with several protocol implementation and tools for data analysis. Thus a testbed can be created without significant effort.

This chapter outlined some of the major testing approaches for mobile ad-hoc networks. Although more costly and less scalable than simulation, emulation has advantages over other techniques when it comes to providing experimental control in a flexible environment that is easily reproducible. Emulation being available at an early developmental stage allows thorough testing before deployment in real scenarios.

The next chapter discusses emulation through virtualization and the challenges this technique poses. Virtualization is categorized and an approach towards an emulator using virtualization is described. Later some challenges like scalability, mobility and promiscuous listening are discussed.

3 Emulation through Virtualization

As described in the previous chapter, there are a number of emulators designed to replicate wireless ad hoc networks. We approach emulation by using virtualization to create a setup where general analysis and testing of protocols is possible. This emulator particularly focuses on evaluating the performance and behavior of different cooperation strategies employed in a MANET. This chapter gives details about the implementation and working of our emulator.

Virtualization is the creation of a virtual rather than a physical entity, which can be some application, storage device, network resources or even a whole operating system. For example, an operating system can be virtualized through the use of software that enables hardware to run multiple instances of the same operating system. Virtualization has targeted three areas, described next.

Storage virtualization enables consolidation of storage devices on multiple network devices to appear as single storage device. *Server virtualization* hides the nature of the physical resources, such as processors and operating systems, from the program. *Network virtualization* involves the consolidation or manipulation of available network resources, like splitting the available bandwidth into channels that can be assigned to independent servers.

3.1 Virtual Emulation

Emulators are used as a development and testing platform, usually employed in research, pre-production or for educational purposes. By introducing virtualization in the emulation process, resources used are a fraction of those required when working without virtualization. This environment is easier to manipulate, thus achieving greater flexibility in testing or prototyping network protocols and algorithms. Also, emulators based on virtualization require less manpower to conduct experiments, thus enabling experiments

to be reproducible. Such an environment for emulation is created by joining multiple virtual machines through a virtual network. Implementations of these virtualized networks include VNUML [26], NetKit [36], Xen [37], VMWare [38], and MLN [39]. These are capable of specifying and deploying virtualization scenarios involving complex virtual network topologies.

The goal of our virtual emulator is to be scalable, with minimum amount of overhead. It should allow real life implementations to be ported into the emulator without significant changes. It should be capable of testing different protocols employed in MANETs. The emulator should also contain features that specifically allow the comparison of performance and behavior of different cooperation strategies in a selfish MANET. We adopt VNUML for our emulation, due to its relatively short learning curve, ease of use and deployment, scalability and fidelity to real world scenarios.

3.1.1 Virtual Network User Mode Linux (VNUML)

VNUML is an open source application tool designed to help build a virtual network running on Linux machines. VNUML is based on User Mode Linux (UML), a secure way of running multiple instances of virtual Linux on a single physical Linux machine. VNUML is an application that encapsulates the intricate details of creation and connection of these virtual instances using a virtual network.

3.1.1.1 Networking under User Mode Linux

Virtual instances of User Mode Linux are applications running on the Linux kernel as user processes. As these processes run on top of the kernel, they do not interact directly with the hardware. So, virtual instances of Linux do not have access to the network interfaces of the physical machine. Network interfaces in these virtual instances of Linux are software-based and an arbitrary number of interfaces can be brought up in each of these virtual instances. These interfaces are combined together by using a UML switch, which behaves similarly to a layer 2 switch. Thus, virtual instances of Linux are

able to communicate with one another by sending traffic through this switch as shown in figure 3.1.

To be able to communicate with the host or the network outside of the physical machine, these virtual instances use TUN/TAP network drivers. TUN and TAP are virtual drivers, where TAP simulates an Ethernet device, and TUN simulates the network layer. For connecting the UML instances, a host's TAP device connects to the UML switch. This way the host can access all UML instances that are connected to the UML switch. Another approach is to directly connect the virtual interface of UML to the TUN/TAP device on the host, bypassing the UML switch. Also, by using IP forwarding or Network Address Translation (NAT) these UML instances are able to connect to the outside world through the physical host's interface.

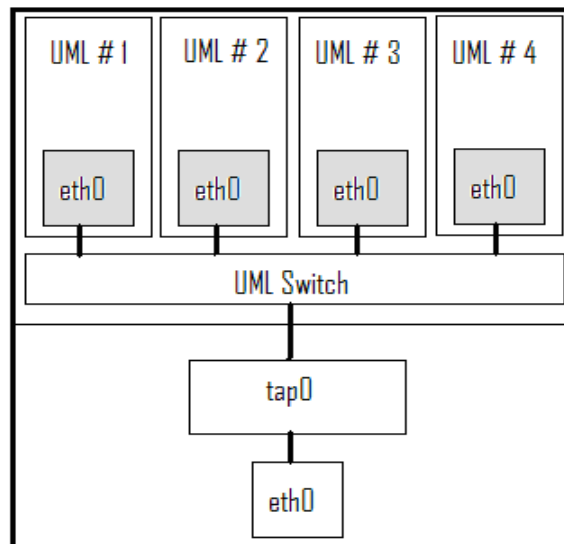


Figure 3.1: Networking with UML

3.1.1.2 Parts of VNUML

VNUML consists of two main parts: the eXtended Markup Language (XML) based VNUML language, and the parser, which invokes UML to build and manage the network described in XML. An example of a network description using XML is shown in figure 3.2.

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/share/xml/vnuml/vnuml.dtd">
<vnuml>
  <global>
    <version>1.7</version>
    <simulation_name>final</simulation_name>
    <ssh_key>/root/.ssh/identity.pub</ssh_key>
    <automac offset="0"/>
    <vm_mgmt type="private" network="192.168.0.0" mask="24" offset="100">
      <host_mapping/>
    </vm_mgmt>
    <vm_defaults>
      <filesystem
        type="cow">/usr/share/vnuml/filesystems/root_fs_tutorial</filesystem>
      <console id="0">xterm</console>
    </vm_defaults>
  </global>
  <net name="Net1" mode="virtual_bridge" external="ath0" />
  <vm name="umlB1">
    <mem>50M</mem>
    <if id="1" net="Net1">
      <ipv4>10.10.0.22</ipv4>
    </if>
    <forwarding type="ip"/>
  </vm>
  <vm name="umlB2">
    <if id="1" net="Net1">
      <ipv4>10.10.0.43</ipv4>
    </if>
    <forwarding type="ip"/>
  </vm>
  <host>
    <hostif net="Net1">
      <ipv4 mask="255.255.255.0">10.10.0.12</ipv4>
    </hostif>
    <physicalif name="ath0" type="ipv4" ip="10.10.0.12"/>
    <forwarding type="ip"/>
  </host>
</vnuml>

```

Figure 3.2: VNUML Language

On this XML description, calling the VNUML parser starts the virtual network scenario. After the experiment, the VNUML parser is called again to dismantle the scenario and release the resources.

Each of the UML instances is built on a Linux root file system located on the host machine. We created a customized root file system for our emulator by making several

additions to the standard root file system. The challenges faced and the additions made for our emulator are discussed in the next section.

3.2 Challenges of Virtual Emulation of Ad Hoc Networks

To assess the behavior of various cooperation strategies in a selfish ad hoc network, our emulator must enable nodes in the network to listen promiscuously to traffic sent by their neighbors. Promiscuous listening enables nodes to maintain reputation information for other nodes to which these nodes forwards traffic. This reputation information in turn enables the nodes in the network to punish or respond to other nodes by building strategies based on reputation data. The punishment and response behavior is built by using the API developed for the MANIAC Challenge [29]. The MANIAC Challenge API, as discussed in the previous chapter, has the ability to forward, drop or redirect packets.

In addition to these requirements, the emulator should be able to scale for large network testing. It should also be capable of including different mobility patterns in the emulation. To summarize, the challenges discussed here comprise of:

1. Mobility emulation;
2. Promiscuous listening; and
3. Scalability.

The remainder of the chapter discusses the various features and changes made in VNUML.

3.2.1 Mobility Emulation

Thus far we have explained that VNUML is used to build a network environment that is connected and has virtual instances of Linux that can be manipulated just like an ordinary physical machine. It is necessary for this VNUML-based network to be able to

use mobility models that accurately represent mobile nodes that will eventually utilize the protocol under test. Only in this type of scenario is it possible to determine whether the proposed protocol will be useful when implemented in a MANET. MobiEmu [17] is used to emulate mobility on top of a VNUML-based fixed network of virtual machines.

3.2.1.1 MobiEmu

MobiEmu is a software platform for testing and analyzing ad hoc networks. It is based on a fixed network of n machines emulating a mobile ad-hoc network of n nodes. These fixed n machines can be real or virtual and in our scenario we created them virtually using VNUML. As in an ad hoc network, the topology of the network changes frequently based upon the movement of nodes. Thus, to replicate the effect of nodes moving in and out of one another's transmission range MobiEmu dynamically installs and removes packet filtering rules. The goal of a packet filter is to create a partially connected virtual network on top of a fully connected real network so that testing and analyzing of ad hoc networks can be accomplished in laboratory settings. Several such systems have been developed and used for fixed networks; these include NIST Net [40] and dummynet [41].

MobiEmu is based on a master/slave architecture as shown in figure 3.3, where the master runs on a separate machine and communicates with the slaves using a dedicated control channel and the slave runs on each of the emulated machines. The emulation is scenario driven, and a scenario consists of information about movement and location of every node. Each of the virtual instances in our emulation runs a MobiEmu slave controller. The master node controls all the slave nodes and keeps them synchronized. It broadcasts timing information and enforces the current topology using packet filtering.

Scenario files detailing node connectivity and movement throughout the emulation period are loaded using the graphical user interface of MobiEmu. MobiEmu accepts the same scenario format as used in the ns2 network simulator. That is, any files generated by CMU's setdest tool can be used to drive the emulation.

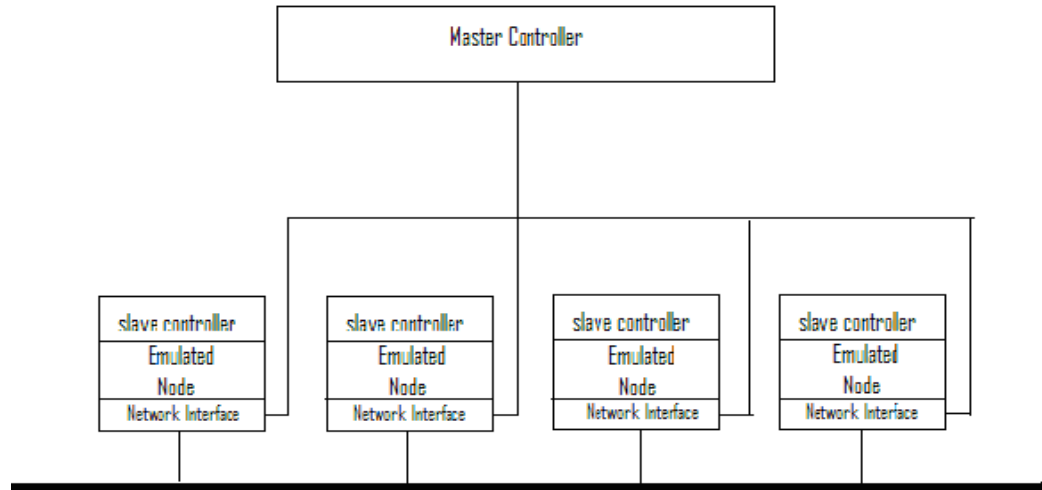


Figure 3.3: MobiEmu Architecture

Emulation is started by the master node triggering the master clock or emulation clock. After the emulation clock starts, it goes through the scenario file checking for topology changes. The master keeps broadcasting the emulation time, enabling the slaves to enforce the topology associated with the broadcast timestamp.

3.2.1.2 Mobility Models

Scenario files used in the emulation are based on mobility models. There are two types of mobility patterns used for simulation and emulation of ad hoc networks: traces and synthetic models. Traces are those mobility patterns that are observed in real life systems. Traces provide accurate information, especially when they involve a large number of participants and an appropriately long observation period. Synthetic models attempt to represent the mobility pattern of nodes by utilizing mathematical modeling.

NS-2 generated mobility scenarios files follow random waypoint mobility model. In this model, a node selects a random destination within the network, a speed and then moves to the selected destination at the selected speed. Once the node reaches the destination, the node rests for a pause time, and then repeats the process by selecting a new destination and speed and resumes movement. As we try to replicate pedestrian movement from MANIAC Challenge environment, the mobility of the nodes is randomly

selected between 0 and 5 m/s to represent walking speeds with a pause time of 10 seconds.

Pedestrian mobility is a mobility model obtained by observing the actual movement patterns of people. MANIAC Challenge traces are a collection of such movement traces from participant nodes. Compared to ns-2 generated synthetic model based on the random way point model, these traces reflect a more realistic behavior of node movement and their changing connectivity. Also, these traces capture link changes due to radio channel variations, co-channel interference, contention for medium access and route flapping effects.

Our approach involved using ns-2 generated mobility scenario files using CMU's setdest tool and extracting traces from the logs of actual experiments conducted during the MANIAC Challenge. This involved extracting the topology from every node involved in the MANIAC Challenge and then parsing the data to produce an ns2 format scenario file. Both traces and synthetic topology files generated through setdest have been used in our experiments.

3.2.2 Promiscuous Listening

Promiscuous mode is a configuration of a wireless network card that makes the card pass all traffic it receives to the central processing unit, rather than just packets addressed to it. It is a feature normally used for packet sniffing.

To replicate promiscuous listening, each virtual node in a network needs to be able to hear traffic not only addressed to it but all the traffic generated by its neighboring virtual nodes. If the emulation is not virtualized and is based on physical machines, promiscuous mode is easily achieved. However, in a virtual emulator comprising of multiple physical machines we must emulate broadcast of packets to achieve promiscuous listening. This is due to the fact that virtual machines residing on the same physical machine communicate through the internal UML switch.

For example, suppose nodes A, B and C are emulated within one physical machine and nodes D and E are emulated in a different physical machine as shown in figure 3.4. In this example topology, when node A forwards a message to node C, traffic

passes through node B. In a real wireless network, all neighboring nodes of node B would be able to hear node B forwarding this packet towards its final destination. In the virtual, emulated network, node A and node D, both being one-hop neighbors of node B, are unable to hear any packets forwarded by node B. For node A to hear these packets it needs to listen promiscuously on the internal UML switch through which the traffic is flowing, but still node D, which is also a neighbor of node B, is unable to hear this forwarding, as it resides on a different physical machine and traffic traverses from node A to node C via the internal UML switch of emulator node 1. Thus we need to emulate promiscuous listening by generating additional packets which make node A and node D both aware of the forwarding performed by node B, as explained next.

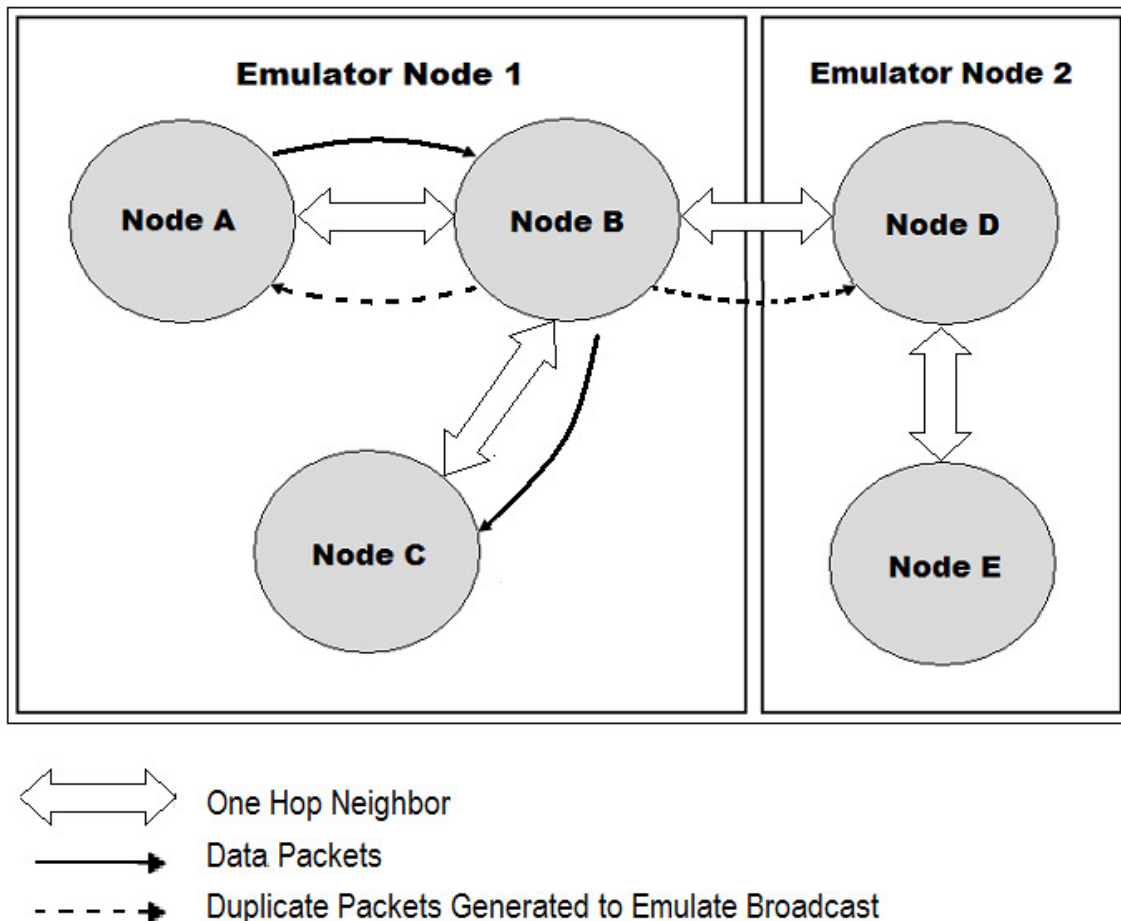


Figure 3.4: Packet Traffic from Node A to Node C

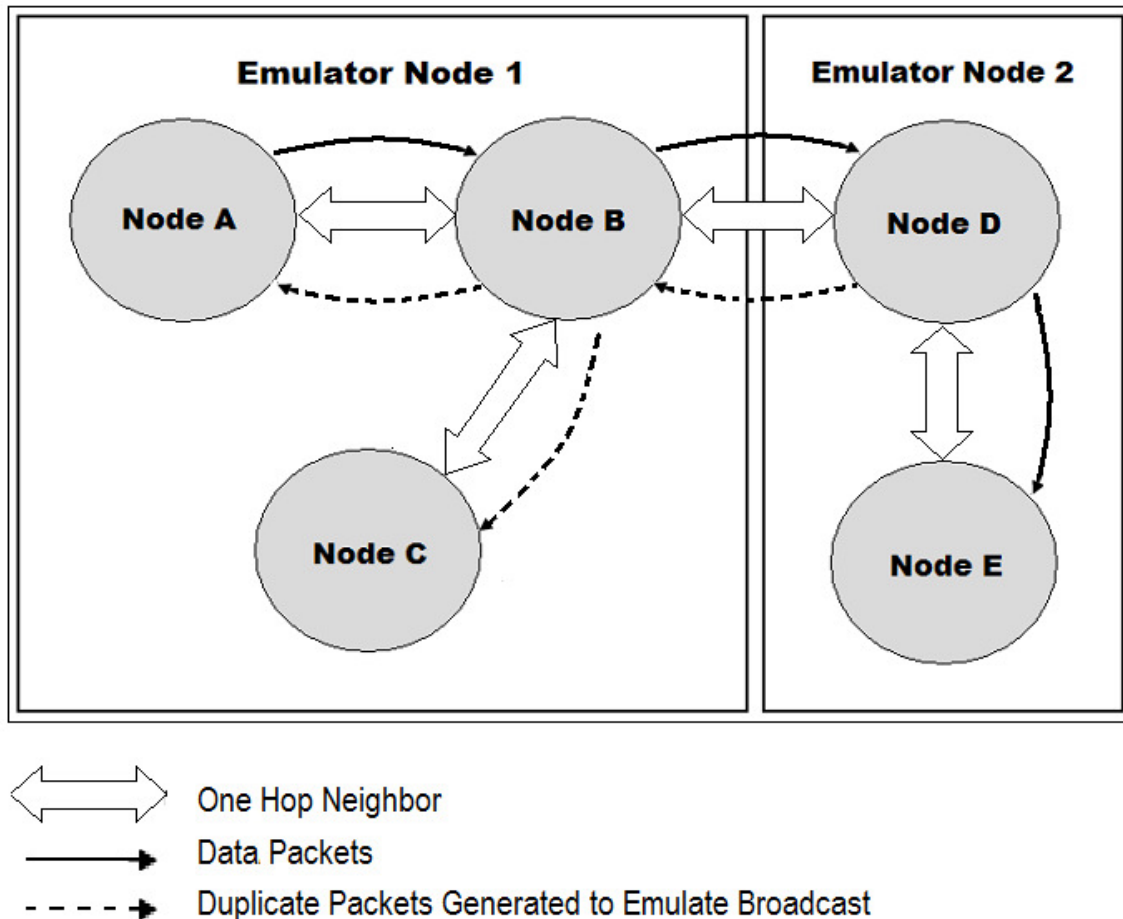


Figure 3.5: Packet Traffic from Node A to Node E

The TTL field in the IP datagram header is designed to prevent a datagram from endlessly moving in a loop. In every datagram, the TTL field is set to a positive integer. While passing through intermediary nodes the TTL field gets decremented and eventually either the packet reaches its final destination or is discarded when the TTL field reaches zero.

We use the TTL field to emulate a broadcast mechanism, where every node is able to listen to packets forwarded by its neighbors. For every packet that is forwarded by any node a replica of the same packet but with the TTL field set to 2 is send to all one hop neighbors of that node except the next hop of the packet. (The TTL field within the duplicate packet is set to 2 rather than 1 because if it is set to 1 the TTL field decrements

to 0 upon reaching the neighboring node and the packet is dropped before the neighbor can extract any useful information from it.) When a packet with TTL field set to 2 reaches a node, the TTL field is decremented to 1 and the node listening for packets with TTL field set to 1 is able to extract information regarding any neighboring node that has forwarded the packet. In this way as shown in figure 3.4 node B generates duplicate broadcast packets that are sent to nodes A and D.

Similarly, in figure 3.5 node A is generating traffic destined to node E and as the traffic messages are forwarded by the intermediate nodes, these intermediate nodes generate duplicate broadcast packets indicating to their neighbors that a message has been forwarded by them. This broadcast message is sent to the entire one-hop neighbor list except the next hop of the packet. Due to these duplicate messages node B is able to listen to forwarding performed by node D, which resides on a different physical machine. This process of generating duplicate messages makes this multiple machine based emulator work like a normal ad hoc network.

3.2.3 Scalability

Scalability is the ability of a system to either manage increasing load in an elegant manner, or to be easily enlarged. For example, it can refer to the capability of a system to increase total throughput under an increased load when additional resources are provided.

As explained previously, VNUML-based virtual machines are capable of communicating to the outside world through the physical machine. This communication with the external network makes it scalable, allowing the emulator to benefit best of both the virtual and the real world environment. An overview of the emulator is shown in figure 3.6.

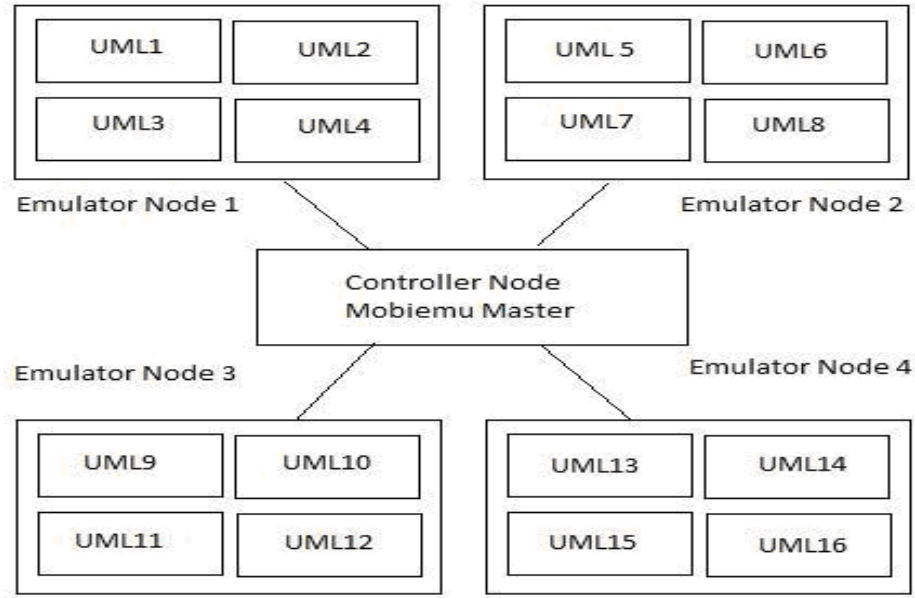


Figure 3.6: Emulator Setup

Scalability in VNUML is achieved by connecting multiple physical machines together in an ad hoc network and then running virtual machines through VNUML on every physical machine on this ad hoc network. Thus the virtual machines running on top of every physical machine in this scenario become part of the network. In this way multiple physical nodes participating in an ad hoc network can collectively build an emulator, and by increasing the number of physical nodes involved we can increase the size of the network emulated.

Using this approach also helps to model the wireless medium. If only one machine is used, all the traffic flows remain inside just one physical machine and hence traffic is not affected by channel variation and losses. Wireless connection of multiple physical machines causes the traffic to be sent out of the wireless interfaces of every physical machine. This helps incorporate the effects of radio channel impairments, co-channel interference and contention for medium access to the traffic flowing in the emulator.

Although VNUML is scalable, the physical interface may act as a bottleneck. This happens due to limitations of the physical interface in supporting multiple virtual machines.

Combining the scalability, mobility and promiscuous listening feature with virtualization, we created an emulator capable of comparing various forwarding strategies in a MANET of selfish nodes.

This chapter gives a brief overview of the features of the emulator being used. It describes how scalability, mobility, and promiscuous listening techniques are supported in our emulator. The next chapter describes the results of experimentation with some cooperation strategies in a MANET. Results are based on scenarios using both MANIAC Challenge routing table traces and ns2 mobility traces generated according to the random waypoint model.

4 Characterizing Cooperation Strategies in MANETs

In this chapter we evaluate the performance of various cooperation strategies in a selfish network environment.

Mobile nodes in an ad hoc network face constraints on battery life and bandwidth, which may create incentives for nodes within the network to adopt greedy, selfish or malicious behavior. Characterizing such misbehavior and avoiding it are critical to maximizing the performance of the network. However, due to the lack of infrastructure in an ad hoc network it can be challenging to detect such node misbehavior. This section describes some of the methods being used to detect selfish energy-conserving strategies, greedy behavior or malicious activities of nodes in an ad hoc network.

Node misbehavior involves either selfishness or maliciousness. An example of *selfish behavior* occurs when a node does not fully participate in the packet forwarding functionality in order to conserve its limited resources, such as battery life. In contrast, *malicious behavior* is intentional misbehavior, where a node's aim is to disrupt normal network operations. These nodes may act independently or collude with other nodes in a deliberate effort to break the normal functioning of the network. In this thesis, we consider selfish behavior and assess the effectiveness of a generous tit for tat strategy for such an environment.

Different techniques can be used to counter the effects of selfish nodes. Credit exchange systems and reputation-based schemes are examples of such techniques. In a *credit exchange system*, participating nodes receive a payment for each time they forward a packet for any other node; such a system requires the ability to store the credit information and some central entity to secure these transactions. Thus these have limited application in real life scenarios. An example of a credit exchange system is Sprite [42]. *Reputation-based schemes* are systems where a node is monitored either directly or

indirectly by its neighbors. Misbehavior of these nodes is punished by reciprocating (adopting similar behavior towards that node) and/or by enforcing the partial or total isolation of selfish nodes from the network. CONFIDANT [43] and CORE [44] are based on reputation schemes. We evaluate the effectiveness of various reputation-based schemes in our emulator. The next section describes the components of reputation-based schemes.

4.1 Reputation-Based Schemes

Reputation-based systems typically consist of a three-stage architecture. The main components include:

1. Detection and analysis;
2. Building of reputation; and
3. Punishment or avoidance.

4.1.1 Detection and Analysis

In a wireless network, the detection and analysis stage of a reputation mechanism is responsible for monitoring the behavior of neighboring nodes to assess their participation in the network. Nodes are monitored and rated by nodes in their vicinity or by nodes that are the sources or recipients of the traffic.

A neighboring node monitors and rates its neighbors by eavesdropping on their transmissions. Such eavesdropping is possible by promiscuously listening to all the traffic a node overhears. For example, node A forwards a packet to node B, then listens promiscuously to transmissions of node B, waiting for it to forward this packet toward its destination. If node A overhears such a transmission, it increases its reputation indicator for node B; otherwise, node A decreases its reputation indicator for node B. However, such promiscuous listening may not always be possible due to link asymmetry or collisions, thus possibly triggering a false alarm in node A's reputation system.

A sender- or recipient-based detection and analysis stage depends on feedback from endpoint nodes (either sender or destination) to build reputation for a node. For example, node A forwards a packet to node B. If feedback from the destination indicates the successful delivery of packet then node A increases the reputation indicator for node B; if, on the other hand, no such feedback is received, then node A decreases the reputation indicator for node B. Such a recipient-based detection and analysis mechanism is proposed in [45].

4.1.2 Building Reputation

Reputation schemes provide a node with some quantitative measure of the cooperativeness of its surrounding nodes. Reputation can be based either on local or global reputation tables. Local reputation tables involve each node maintaining a rating for every other node in the network. A node then looks up this rating before forwarding any messages for a neighbor. Global reputation creates a single database that is updated according to feedback from each node in the network.

Reputation mechanisms typically use a moving window to calculate each neighbor's reputation based on a certain number of most recent observations. This moving window mitigates the effects of false positives, where a neighboring node was wrongly marked as uncooperative due to link asymmetry or other communications impairments. Also, by using a moving window rather than relying on all past observations for calculating the reputation of neighboring nodes, the system forgives and forgets with the passage of time those nodes that were uncooperative and isolated from the network. As a result of this forgiving and forgetfulness the uncooperative nodes once isolated from the network are permitted to join the network again.

4.1.3 Punishment and Avoidance

Punishment and avoidance is the stage in a reputation scheme where a node reciprocates to uncooperative neighboring nodes. Punishment is implemented by

dropping packets destined to or from an uncooperative node, whereas avoidance consists of finding a route to the destination that consists solely (or, if that is not possible, primarily) of cooperative intermediary nodes, to maximize the probability that a packet reaches its destination.

Cooperation strategies can be developed based on the above-mentioned reputation schemes. The strategies evaluated in our emulation are explained in the next section.

4.2 Strategies Employed

The strategies used in experimentation in our emulator are detailed in this section.

4.2.1 Generous Tit for Tat (GTFT)

The generous tit for tat (GTFT) strategy is a reputation-based strategy that comprises the three phases discussed in the previous section. The first phase is the assessment or evaluation of the other nodes present in the vicinity of a node. Neighboring nodes in a network evaluate each other by assessing the amount of traffic forwarding done by them. The second phase involves the creation of local reputation tables from the information collected in the first phase. This table is updated for every packet that is processed by the node employing GTFT, showing the reciprocal of the ratio of the number of packets forwarded to a particular neighboring node to the number of packets that were forwarded towards the final destination by this particular neighbor. The final, reactive phase comprises forwarding packets based on reputation information available in the table for the sender and/or the destination of the packet. This strategy is called generous based on a *generosity factor* added to the reputation indicator of each node.

For example, if a node A has a generosity factor $x\%$ and while building reputation for a neighboring node B, node A calculates that node B is forwarding only $y\%$ of the packets forwarded to it, then generosity factor implies that when node A is requested to

forward traffic originating at or destined to node B, the percentage of forwarding done by node A is given by:

$$\min(x+y, 100).$$

This added factor attempts to mitigate the effects of false positives.

4.2.2 Generous Tit for Tat with Selfish Avoidance (GTFT_SA)

Generous tit for tat with selfish avoidance (GTFT_SA) is identical to GTFT in building a reputation table for the other nodes present in the network. The difference in the implementation comes when the reputation index for a node in the reputation table falls below a certain threshold. When this threshold is crossed, then instead of forwarding packets through this node, GTFT_SA tries to avoid this node and forward packets by using a different route to the destination consisting of more cooperative nodes.

4.3 Emulation Setup

We evaluate cooperation strategies through emulation. The emulation setup consists of a total of 16 emulated nodes; in some emulated scenarios, four of these nodes are selected as traffic generators; in other scenarios, all nodes participate in generating traffic. Traffic is introduced in to the network at 15 kbps by each generator. This generation rate qualifies the emulator to be considered as a low load environment for testing.

As previously described, two types of mobility are emulated: ns-2 generated random waypoint mobility and mobility traces collected during the MANIAC Challenge.

4.4 Results of Experimentation

All results are plotted against the number of nodes using the GTFT or GTFT_SA strategy. Graphs use two variations of GTFT by varying the generosity factor (20% and 10%). Results for these two variations are listed in the graphs as GTFT_0.2 and GTFT_0.1, respectively.

The remainder of the nodes in the network of 16 emulated nodes consists of nodes with fixed percentage forwarding strategies. With this strategy, a node simply forwards a fixed percentage of packets flowing through it regardless of how other nodes in the network are behaving; these percentages are set for each node in the network. This models a simplistic form of selfish behavior. Although this strategy is not based on reputation, a fixed percentage forwarding strategy is used to create a benchmark selfish environment for testing reputation-based strategies.

4.4.1 Overall Percentage of Packets Forwarded

Figures 4.1 to 4.8 show the overall average percentage of packets forwarded by all nodes in the network; the 90% confidence interval is also shown in each plot. As seen from these plots, with increasing number of nodes using the GTFT strategy there is an increase in the percentage of packets forwarded in the network. As expected, when more nodes use GTFT strategies, which utilize a generosity factor in reciprocating to other nodes' selfish behavior, then more packets are forwarded. This is true for both GTFT versions and GTFT_SA strategies, but this effect is seen to a lesser degree in the former. GTFT_SA's feature of routing packets through nodes that are more cooperative results in an increase in the number of packets that are forwarded as compared to GTFT. Even by increasing the generosity factor from 10 percent to 20 percent we see a considerable increase in the overall number of packets forwarded, as shown in the following plots. Since the reactive phase of GTFT strategies is based on the reputation index of the sender and/or destination of the packet, all the above mentioned effects are more pronounced in figures 4.3, 4.4, 4.7 and 4.8, when all nodes act as senders. An increased number of

sender nodes means that more packets are processed in the reactive phase by nodes employing GTFT strategies.

4.4.1.1 NS2 Generated Random Waypoint Mobility

The random waypoint mobility model is a synthetic mobility model where every node moves in a zigzag pattern from one point to the next. The waypoints are uniformly distributed over a fixed predefined area and nodes randomly select the velocity and pause time before the start of each movement.

Figures 4.1 to 4.4 are based on random waypoint mobility. Figure 4.1 shows an approximately linear increase in the percentage of packets forwarded as the number of nodes using GTFT-based strategies is increased and the number of nodes using fixed forwarding is decreased. Here GTFT_SA results in the maximum number of packets being forwarded, followed by GTFT_0.2 and then by GTFT_0.1.

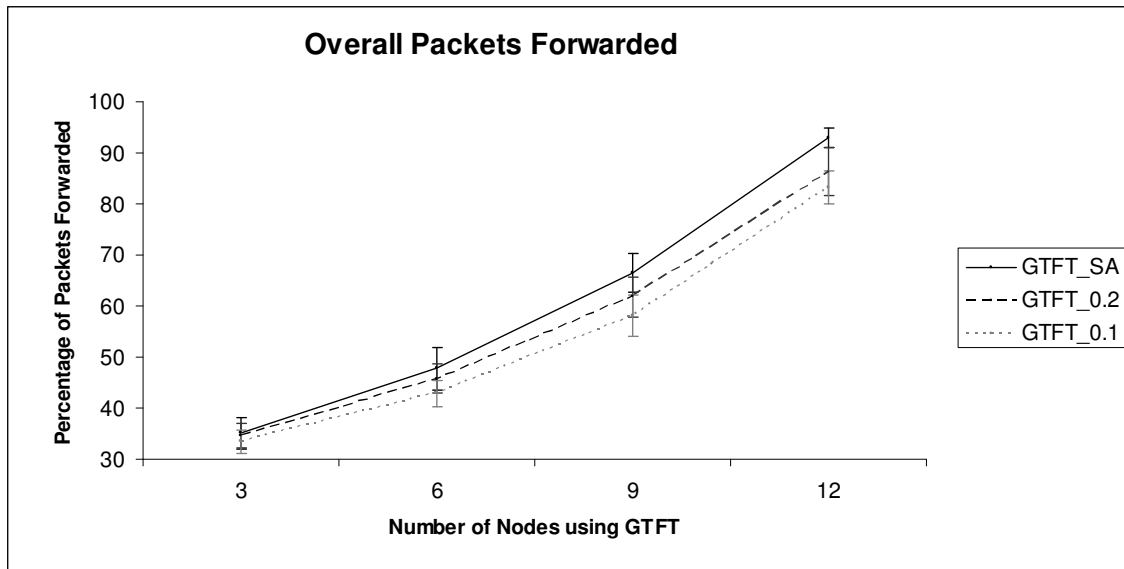


Figure 4.1: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.

Compared to figure 4.1, figure 4.2 has a higher initial point because nodes that do not adopt GTFT strategies are fixed to forward 75 instead of 25 percent of packets and the overall packet forwarding is calculated by taking into account both nodes using GTFT-based strategies and nodes that employ fixed forwarding. Regardless of the higher initial starting point, the plots reach the same range as in figure 4.1 when the number of nodes using GTFT-based strategies is maximized, i.e., when 12 nodes use GTFT-based strategies.

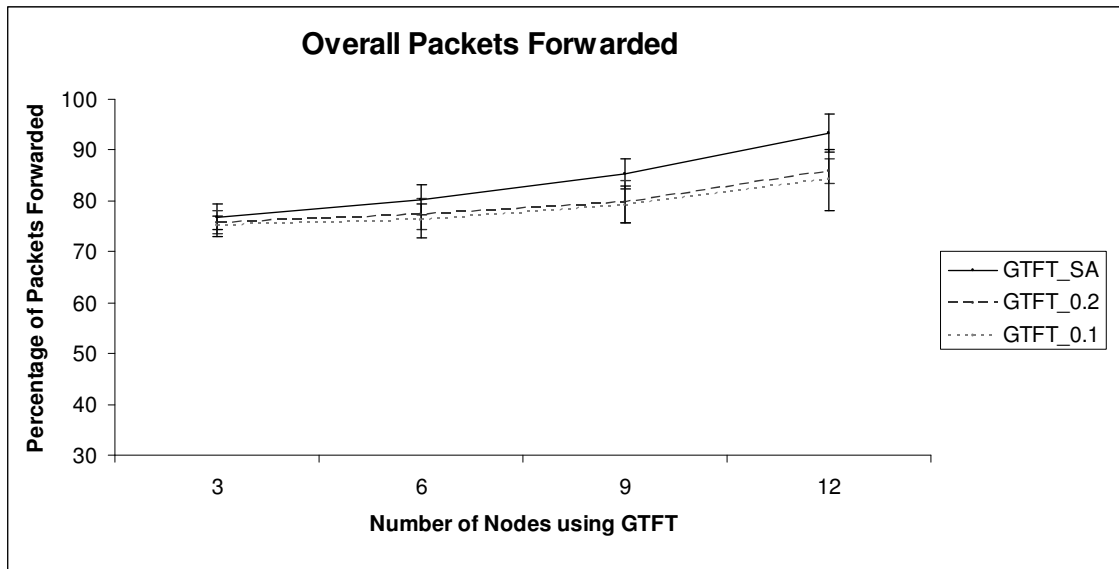


Figure 4.2: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.

In figures 4.3 and 4.4, all nodes act as traffic generators, and these plots show a slight increase in the gap between the curves for the three GTFT strategies, indicating an increase in the percentage of packets forwarded in the GTFT_SA case, compared to the other strategies. As the nodes employing GTFT reciprocate on the basis of sender and/or destination node’s reputation index and with all nodes in these graphs generating traffic more nodes have a reputation index for the reactive phase to work with. Thus a larger percentage of traffic is processed by the reactive phase of GTFT, resulting in an increase in the gap between the three curves. As expected in figure 4.3 and 4.4, compared to figure

4.1 and 4.2, there is a slight decrease in the overall percentage of packet that are forwarded. This is due to the fact that GTFT nodes have reputation index for more nodes and thus a larger percentage of packets are processed by these nodes.

Similarly to figure 4.2, figure 4.4 has a higher starting point due to the fixed forwarding percentage being set to 75 instead of 25.

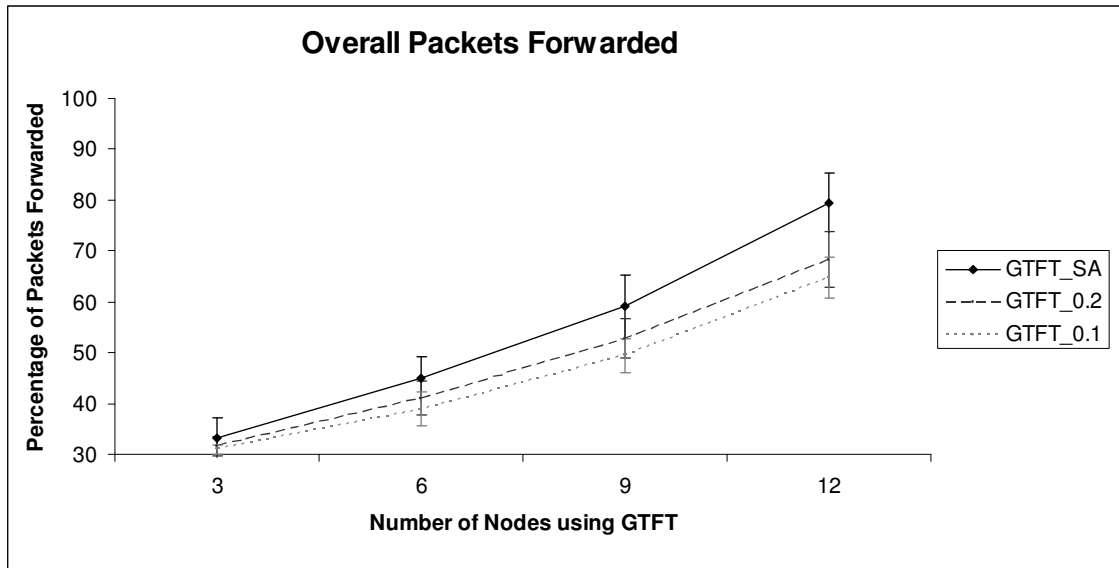


Figure 4.3: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.

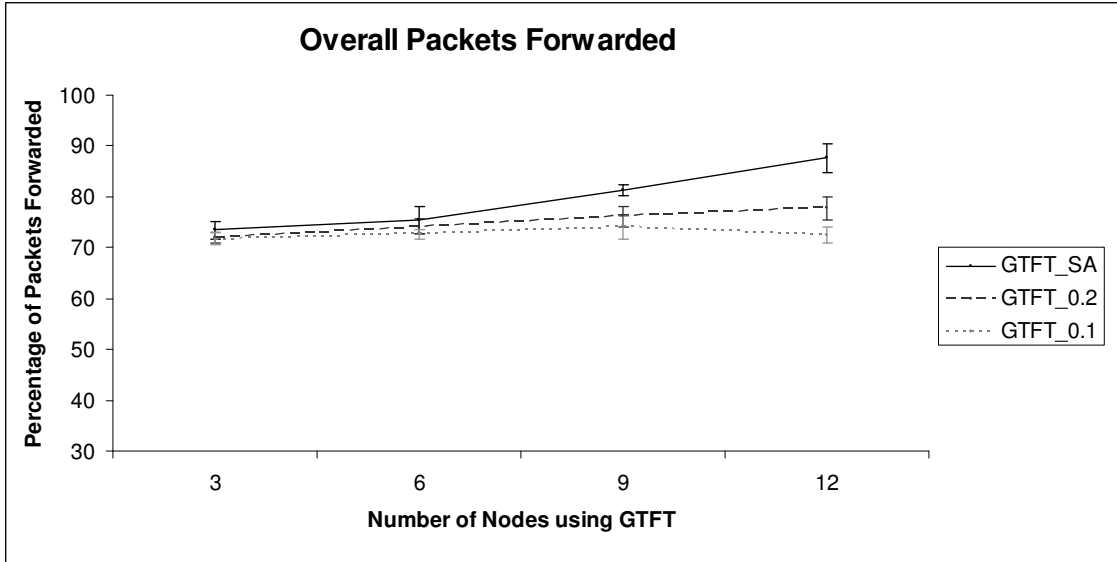


Figure 4.4: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.

4.4.1.2 MANIAC Challenge Traces

Plots in figures 4.5 to 4.8 are based on traces gathered during the MANIAC Challenge. These traces depict a more realistic behavior of node movement and their changing connectivity. Figures 4.5 and 4.6 have four nodes generating the traffic. The curves are not as smooth as those obtained using the random waypoint mobility model. This decreased smoothness is attributed to frequent route changes that cause packets received for forwarding to be dropped and affect the overall packet forwarding done by a GTFT node. Apart from this, the curves show a similar behavior than observed in the random waypoint mobility plots.

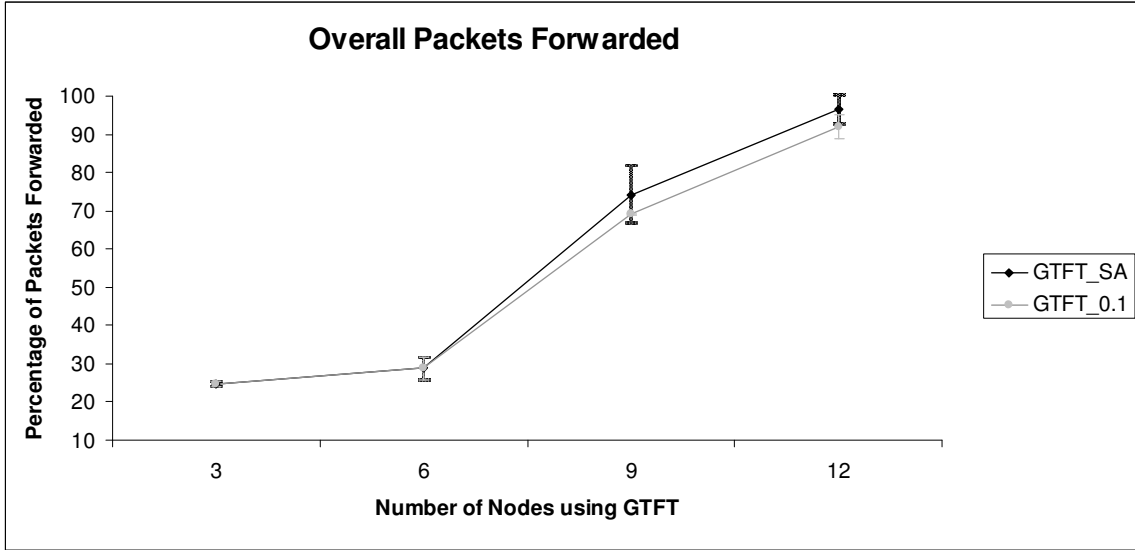


Figure 4.5: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces.

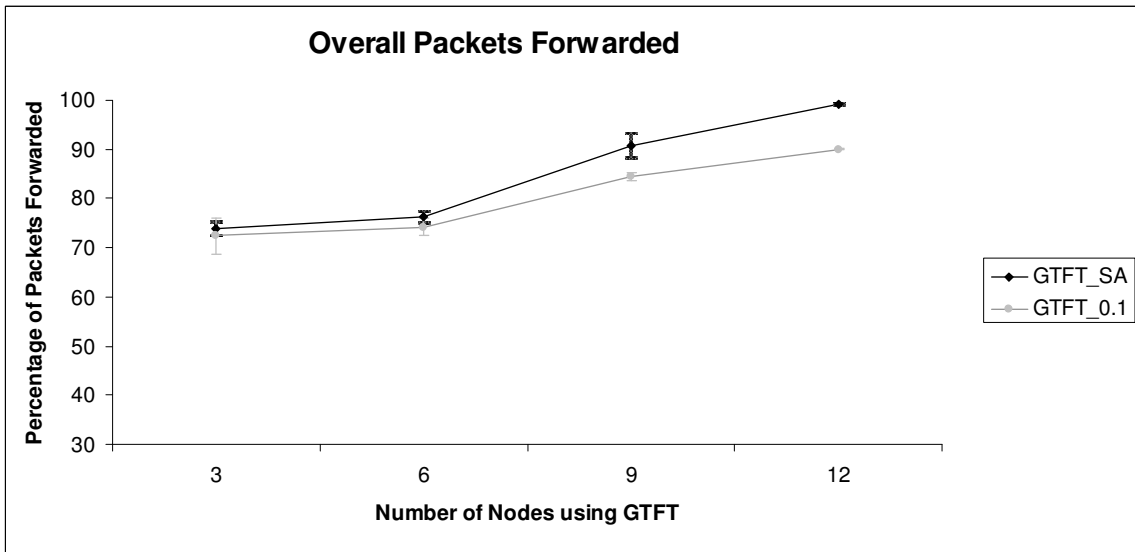


Figure 4.6: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces.

Figures 4.7 and 4.8 show plots that use MANIAC Challenge traces when all nodes are generating traffic. Compared to figures 4.3 and 4.4, these plots also show increased

non-linearity, which is also attributed to frequent route changes. However the overall curves depict similar behavior as observed in figures 4.3 and 4.4.

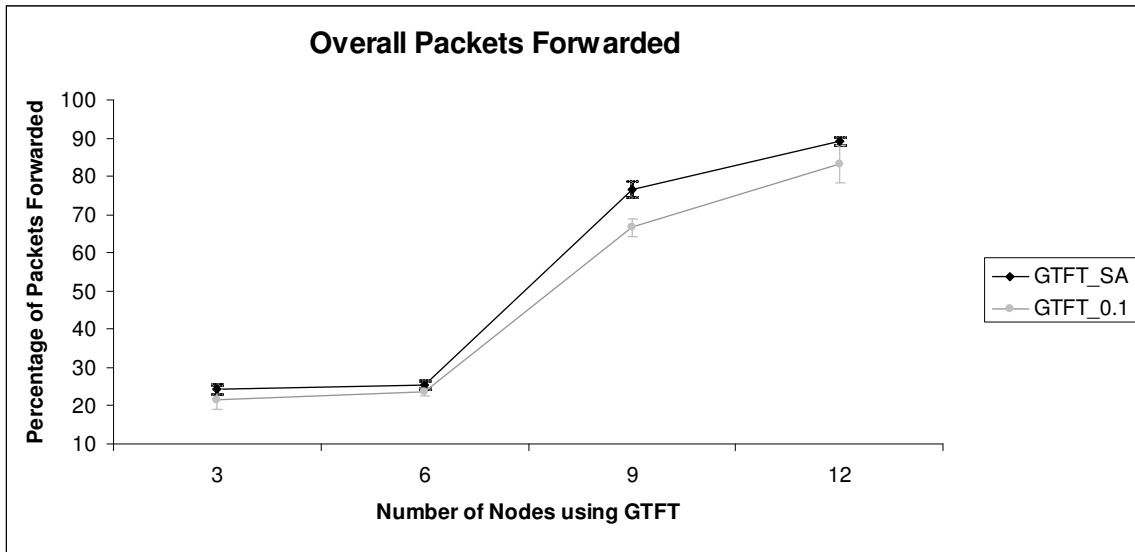


Figure 4.7: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces.

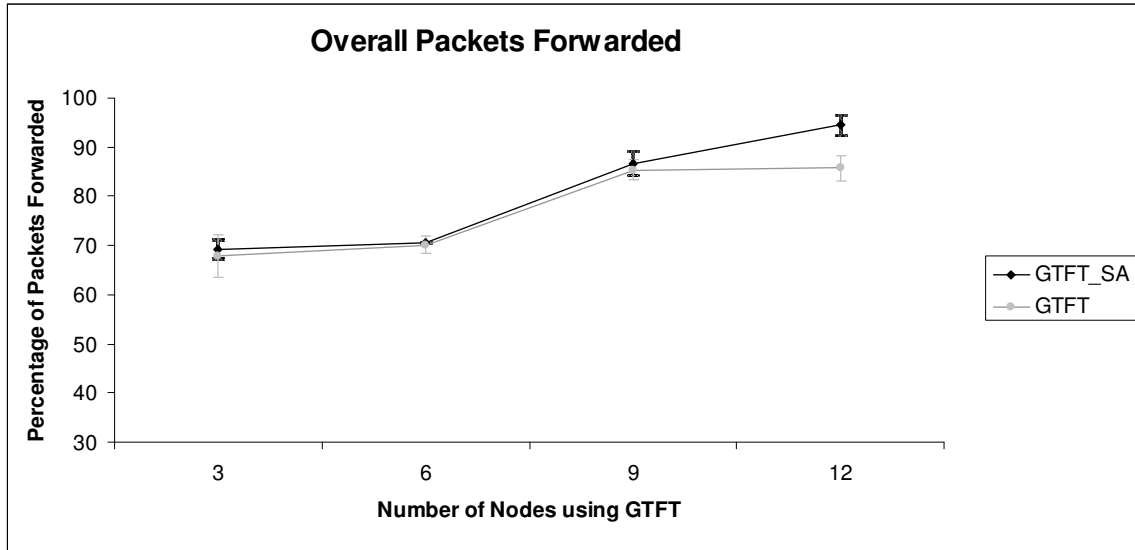


Figure 4.8: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces.

4.4.2 Packets Forwarded by Nodes Using GTFT Strategies

Figures 4.8 to 4.16 show the average percentage of packet forwarding done by nodes employing the GTFT_SA and GTFT strategies; the 90% confidence interval is also shown. The percentage of packets forwarded by a node is given by the ratio of the number of packets forwarded by a node to the number of request received by that node for forwarding packets. Packet forwarding in the case of GTFT_SA is high because, as mentioned earlier, the GTFT_SA strategy tries to find an alternate route to the final destination that avoids the uncooperative nodes in the network, which results in a higher number of packets being forwarded through cooperative GTFT_SA based nodes. Also as expected, GTFT_0.2 has a higher forwarding percentage than GTFT_0.1, due to its higher generosity factor.

4.4.2.1 NS2 Generated Random Waypoint Mobility

Figures 4.9 to 4.12 plot the percentage of packets forwarded by GTFT nodes, under the random waypoint mobility model. By increasing the number of nodes using GTFT strategies the percentage of packets forwarded by GTFT nodes increases only slightly compared to the percentage of overall packets forwarded, which showed a sharp increase as seen in figure 4.1 to 4.4. There is almost a 10% increase in the number of packets that are forwarded by GTFT_SA compared to GTFT_0.2. As explained earlier this behavior is expected due to GTFT_SA's ability to discover new routes and to continue forwarding packets using the most cooperative route to the destination.

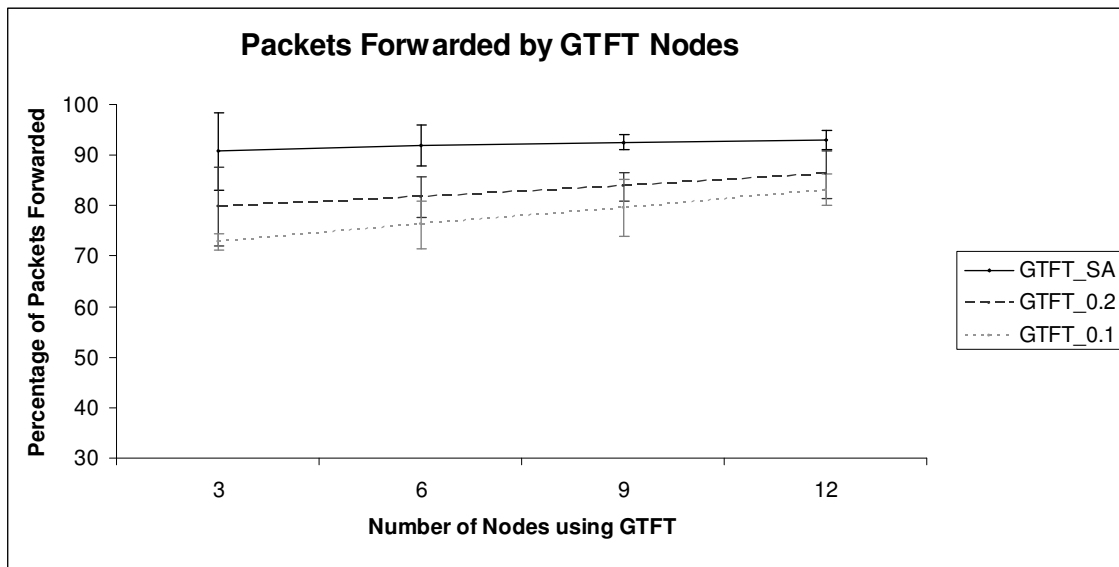


Figure 4.9: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.

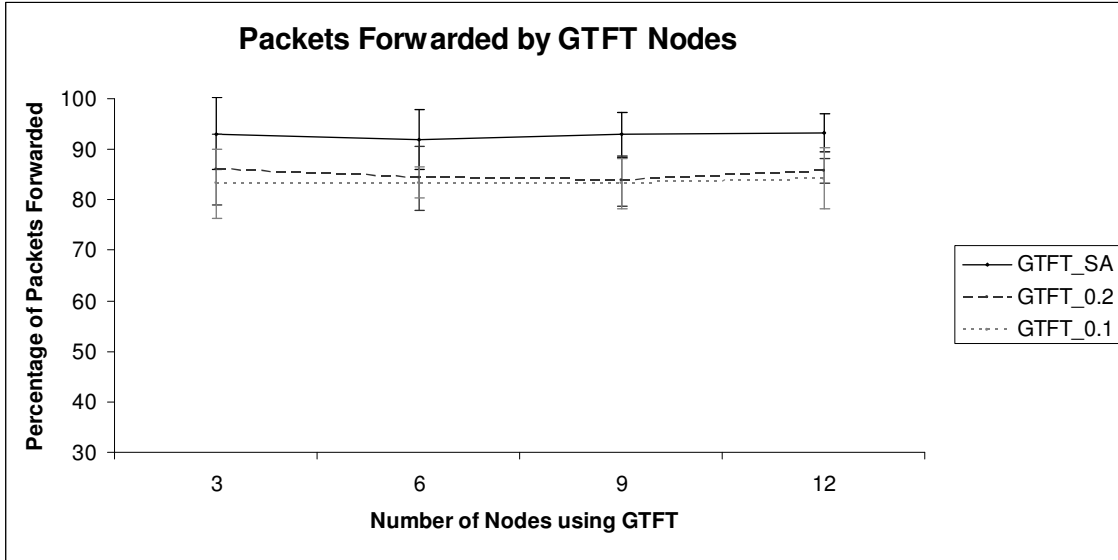


Figure 4.10: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.

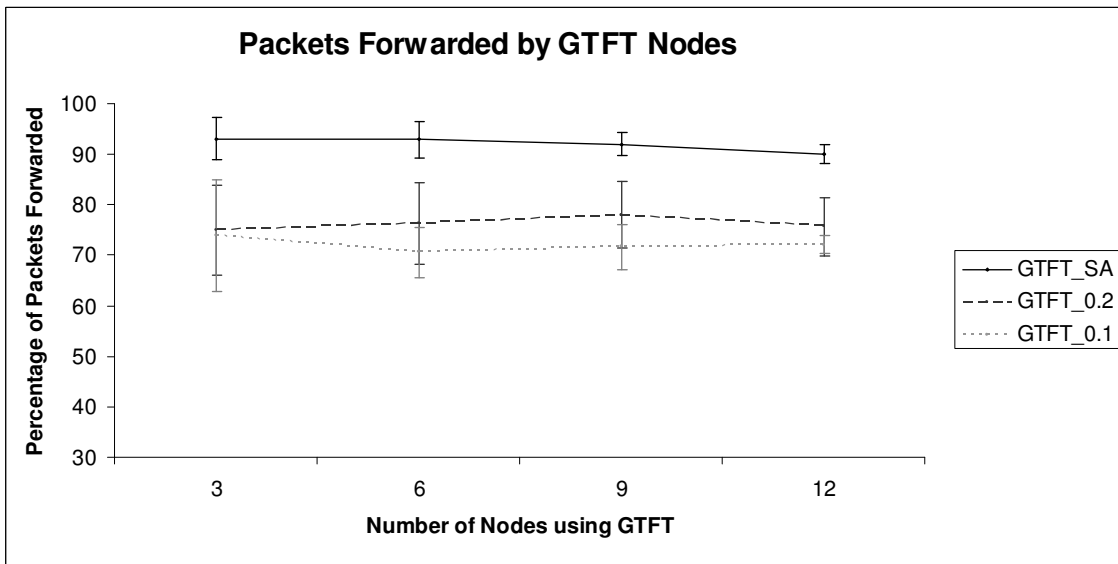


Figure 4.11: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.

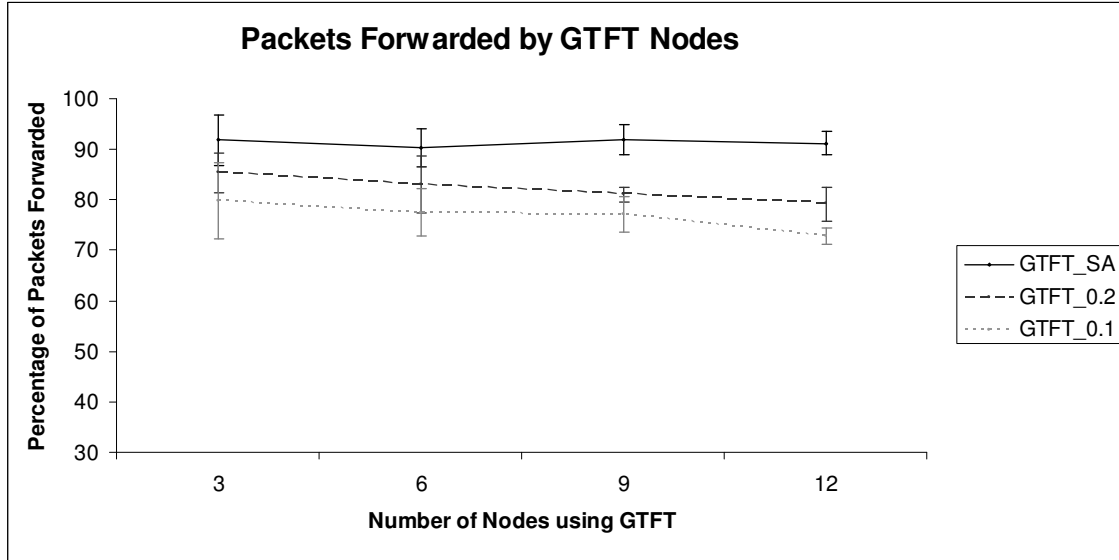


Figure 4.12: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.

4.4.2.2 MANIAC Challenge Traces

Figures 4.13 to 4.16 are generated using traces from the MANIAC Challenge. For these traces, nodes are sparsely distributed and exhibit less connectivity. This has a two-fold effect on nodes in the network: first, the nodes receive fewer requests to forward packets from other nodes; and secondly the reputation building done by GTFT-based nodes is affected as promiscuous listening is disrupted by the frequent route changes. Thus when no packets are forwarded through these nodes then according to the definition, packet forwarding results in 100 percent forwarding. Also when there is no reputation rating for a node and it is requesting a GTFT node to forward its packets, the GTFT node will forward all those packets. Both these points show why these plots have 100 percent forwarding for fewer number of nodes using GTFT. But as the number of nodes using GTFT based strategies is increased the graph takes a similar shape as the one using the ns2 generated random waypoint mobility model. All four of these graphs end up with GTFT_SA showing a higher percentage of forwarding compared to GTFT_0.1.

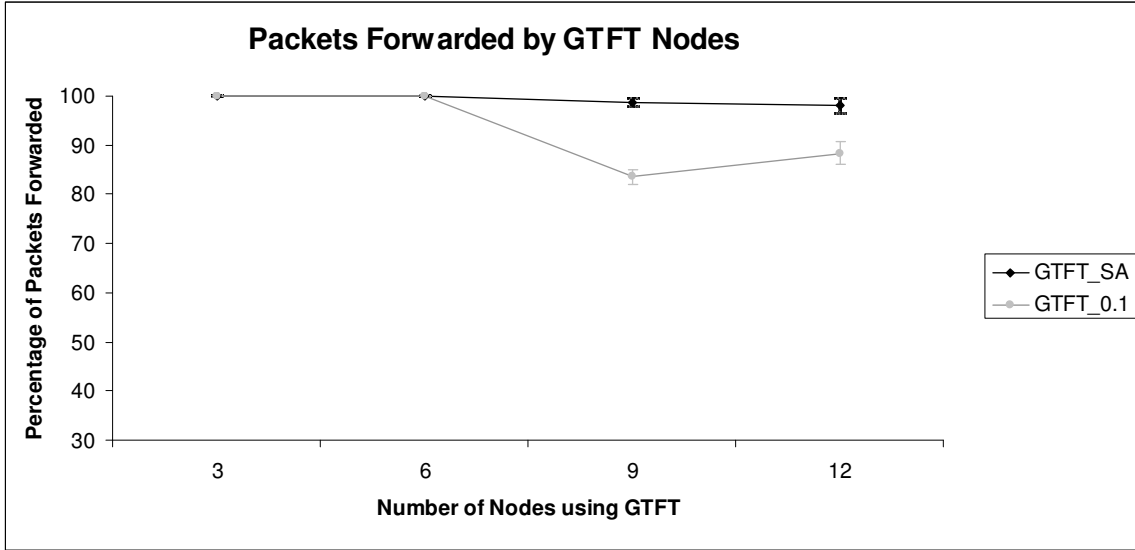


Figure 4.13: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces.

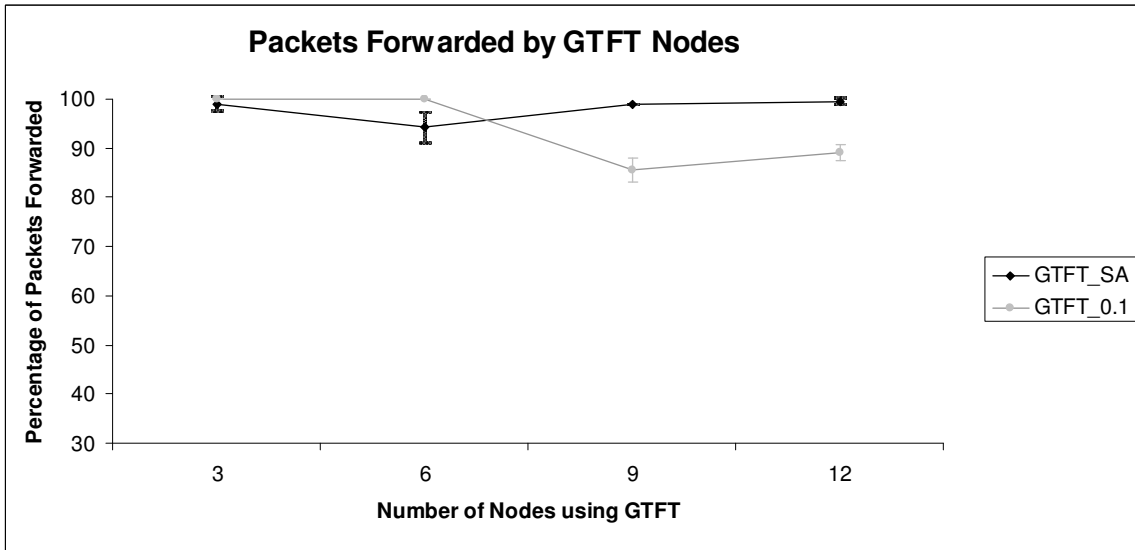


Figure 4.14: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces.

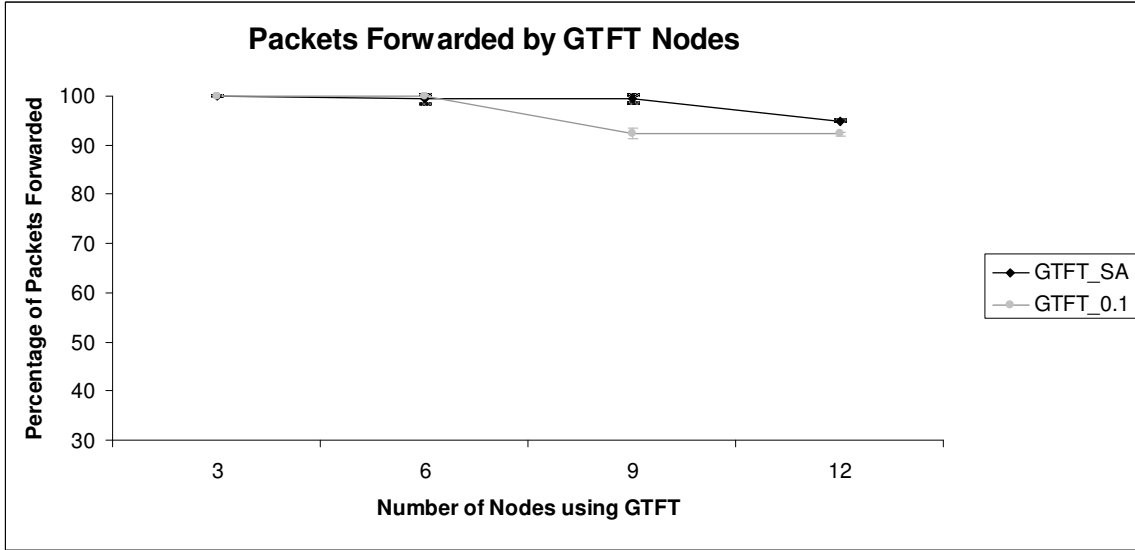


Figure 4.15: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces.

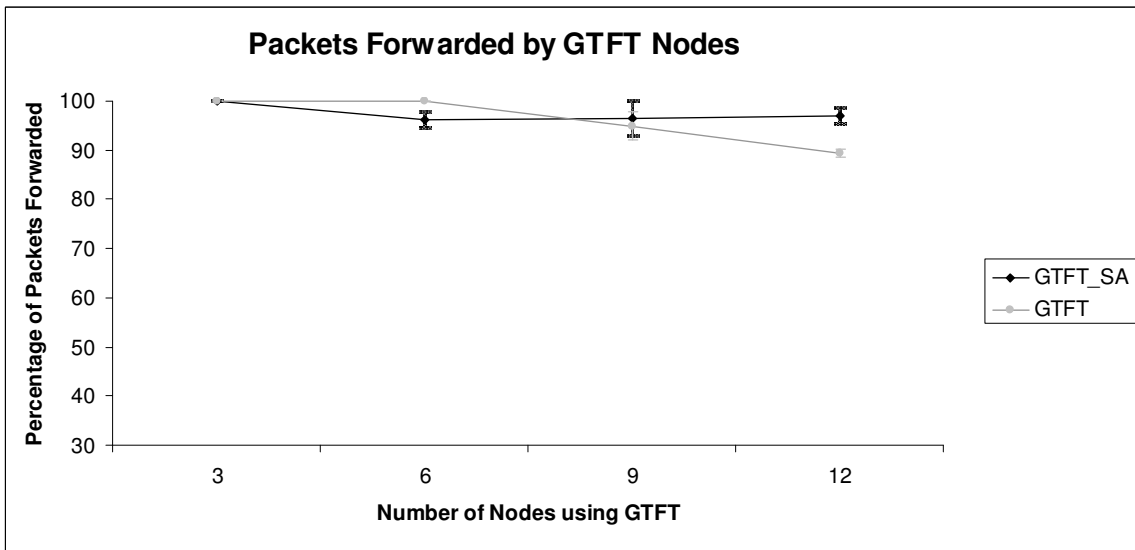


Figure 4.16: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces.

4.4.3 Percentage Packets Received

The plots in figures 4.17 to 4.32 show the average percentage of packets received by all the nodes in the network; the 90% confidence interval is also indicated. The results are broken down into the percentage of packets belonging to real time traffic that are received within a pre-set playback time and the percentage of packets received belonging to non real time traffic. Nodes acting as traffic generators send both real time and non real time UDP packets to other nodes, but only real time packets are time stamped to mimic the operation of a stored media player. The graphs for the percentage of real time packets received show the percentage of packets that were received within a set playback interval. Packets that arrive late, after the playback time has elapsed, are considered lost. All these graphs show a similar trend, where GTFT_SA shows slight improvement compared to both other variations of GTFT.

4.4.3.1 NS2 Generated Random Waypoint Mobility

Graphs in figure 4.17 to 4.24 are generated using random waypoint mobility scenarios. They show both the percentage of real time packets and non real time packets received by all the nodes in the network.

As expected, with the increase in the number of nodes using GTFT based strategies, there is an increase in the number of packets successfully reaching their destinations. This is shown in figures 4.17 and 4.18, where the total percentage of packets received starts around 50 percent but as the number of GTFT nodes is increased the network becomes more and more cooperative, resulting in an increase of real time and non real time packets received.

However, in figures 4.19 and 4.20 the increase in the number of packets received is not as sharp as in 4.17 and 4.18; due to the fact that these plots are generated in an environment with 75 percent fixed forwarding compared to 25 percent for figures 4.17

and 4.18. In figures 4.19 and 4.20 the graphs eventually reach a similar point when the number of nodes using GTFT strategies is maximized.

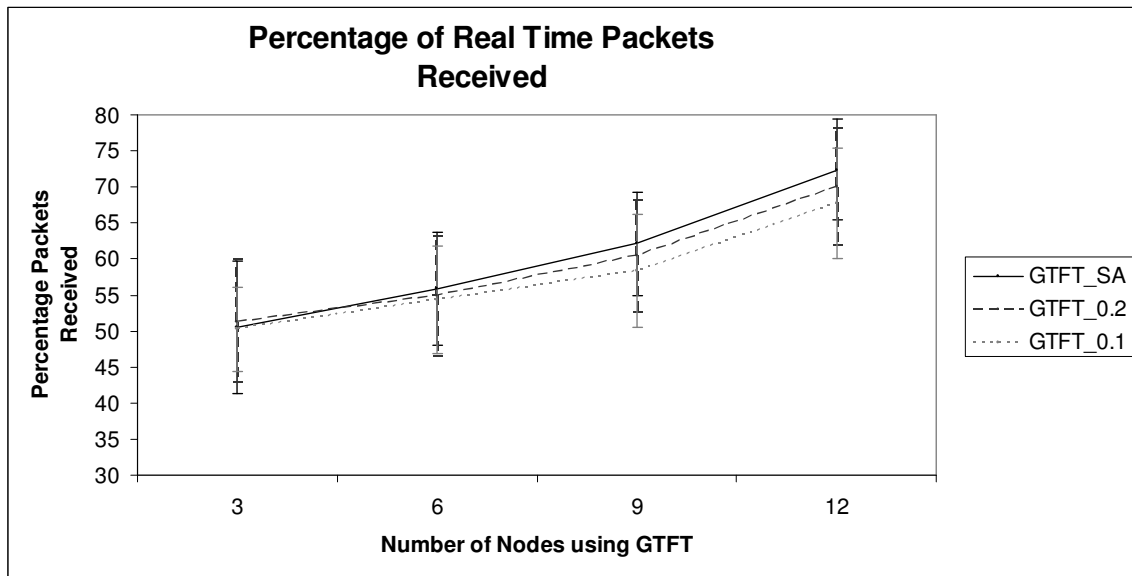


Figure 4.17: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.

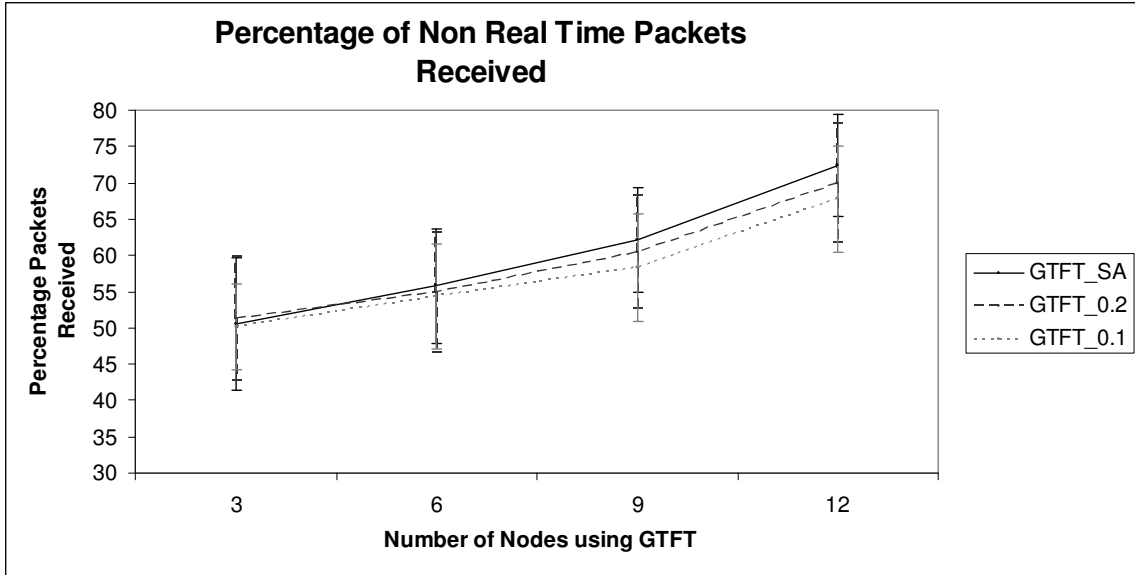


Figure 4.18: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.

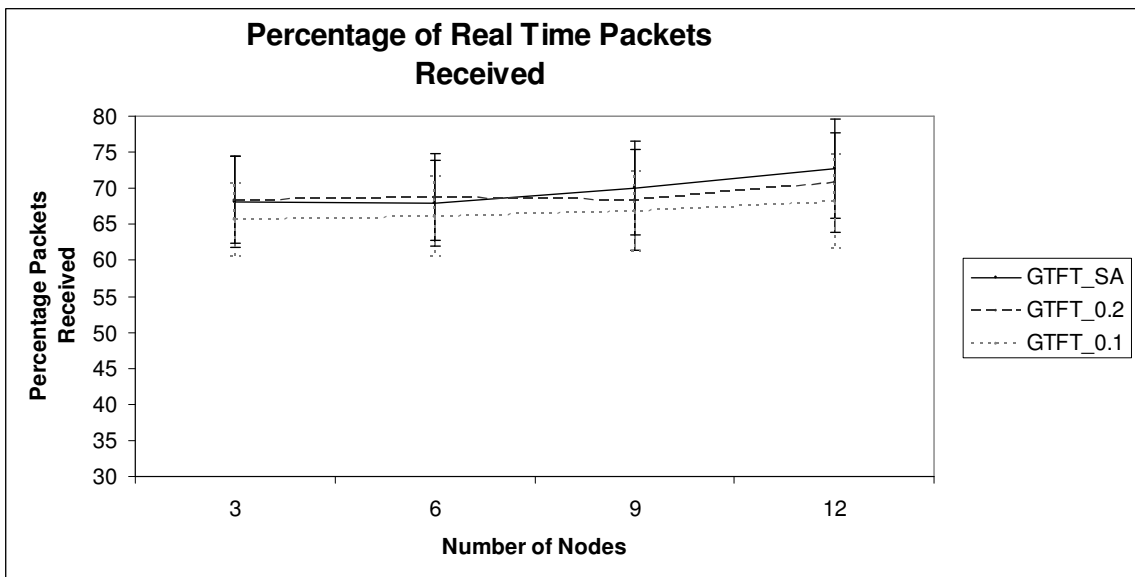


Figure 4.19: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.

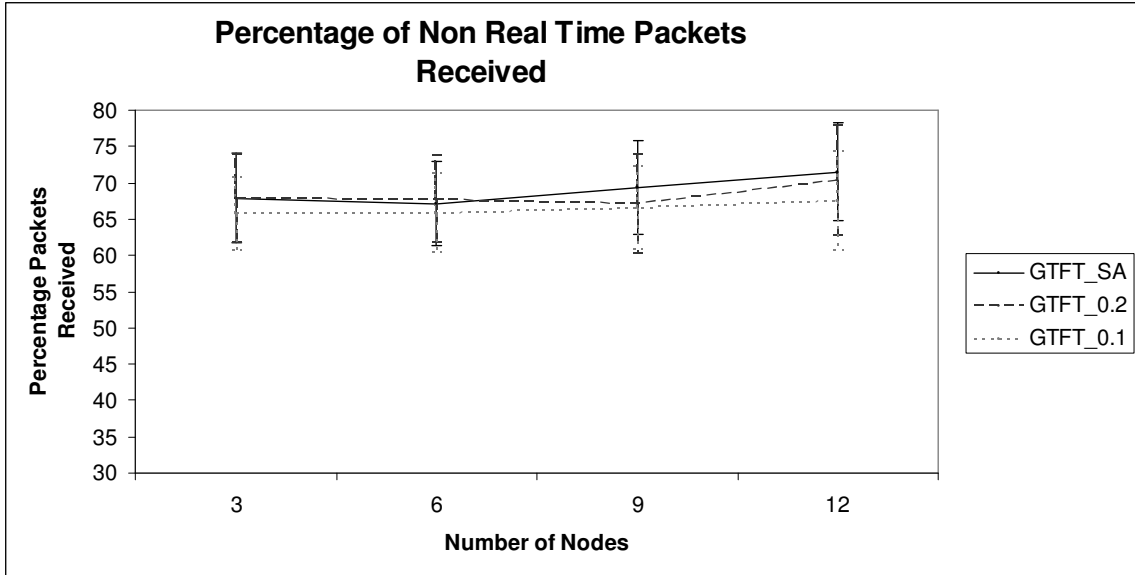


Figure 4.20: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.

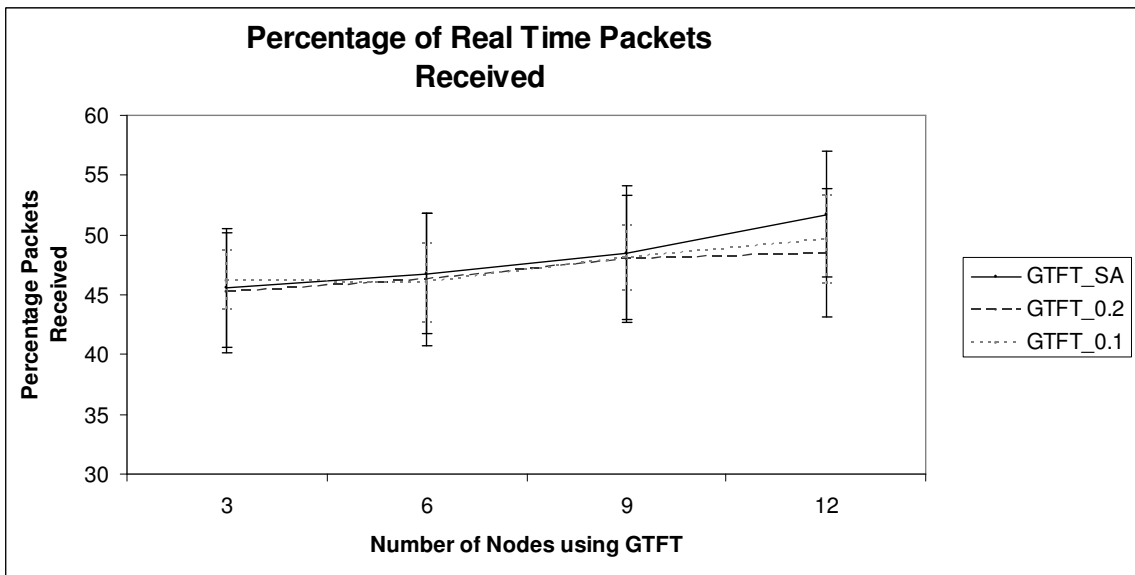


Figure 4.21: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.

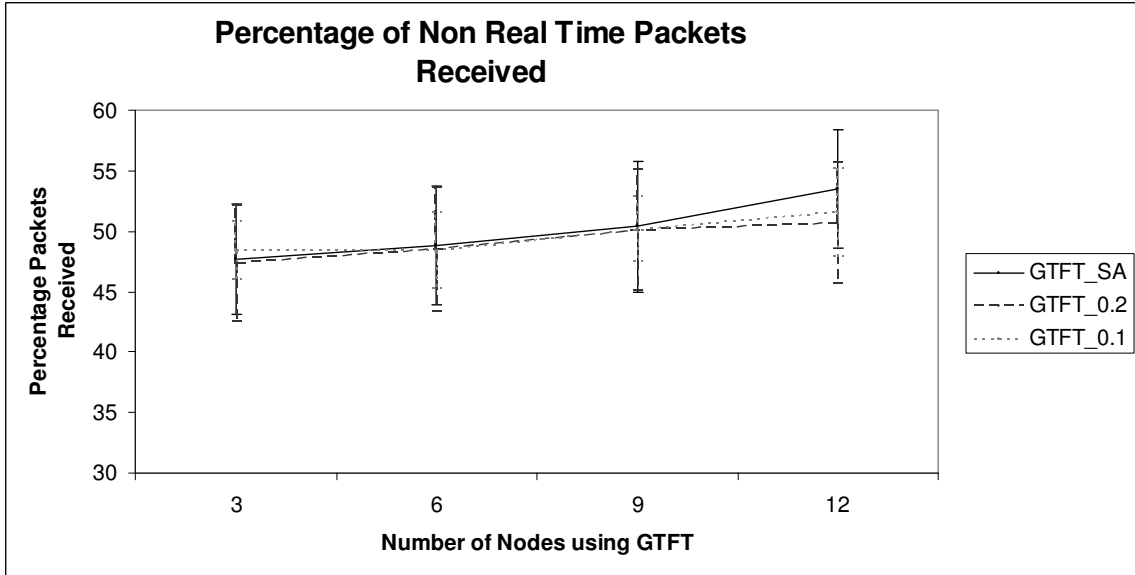


Figure 4.22: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow ns2 generated random waypoint mobility.

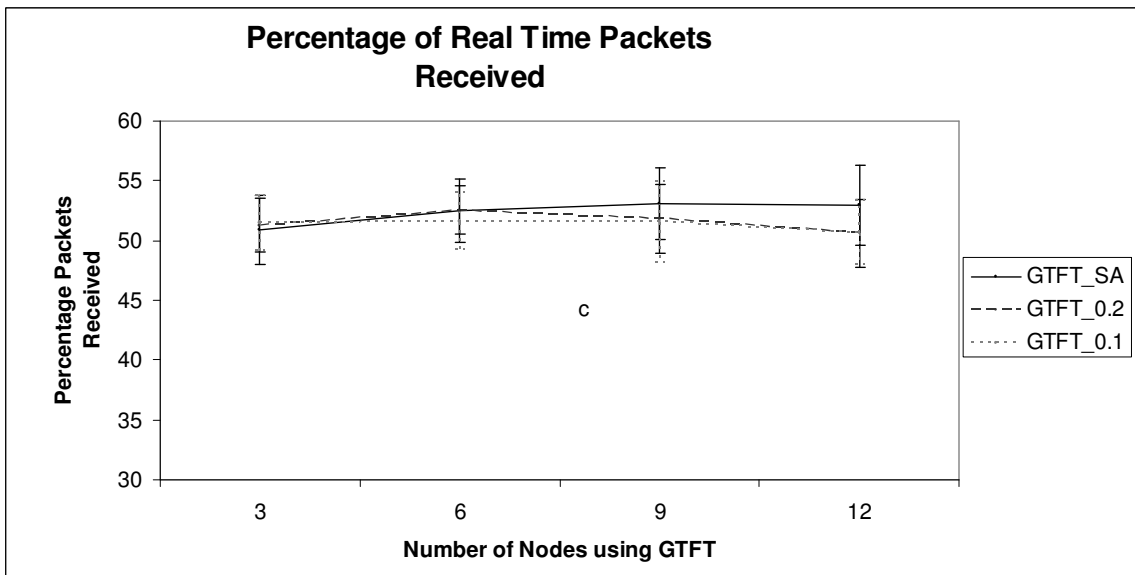


Figure 4.23: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.

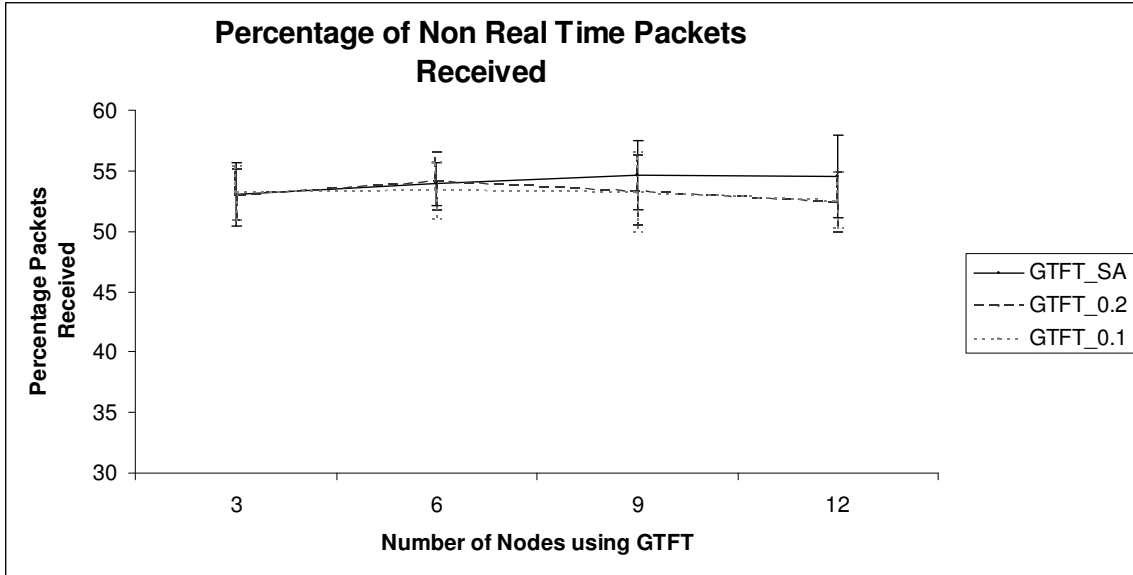


Figure 4.24: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow ns2 generated random waypoint mobility.

4.4.3.2 MANIAC Challenge Traces

Figures 4.25 to 4.32 are plotted using the MANIAC Challenge traces. These traces exhibit less connectivity among nodes, thus a considerable decrease in the percentage of packets reaching their destination is observed. This low percentage of packets reaching their destination is consistent to the results seen in the 2007 MANIAC Challenge [46] regarding the low packet delivery ratio. Comparing to their counterparts using the random waypoint model, these plots show a similar trend as the number of nodes using GTFT is increased in the network, but these plots show a lower percentage of the total packets received. This applies to both real time and non real time traffic.

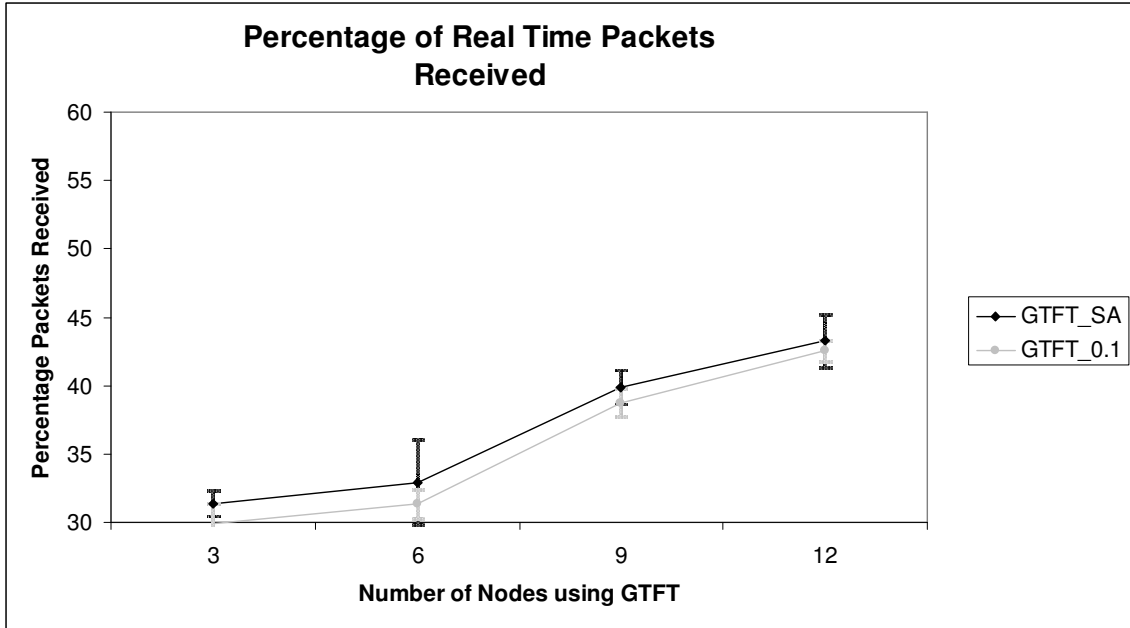


Figure 4.25: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces.

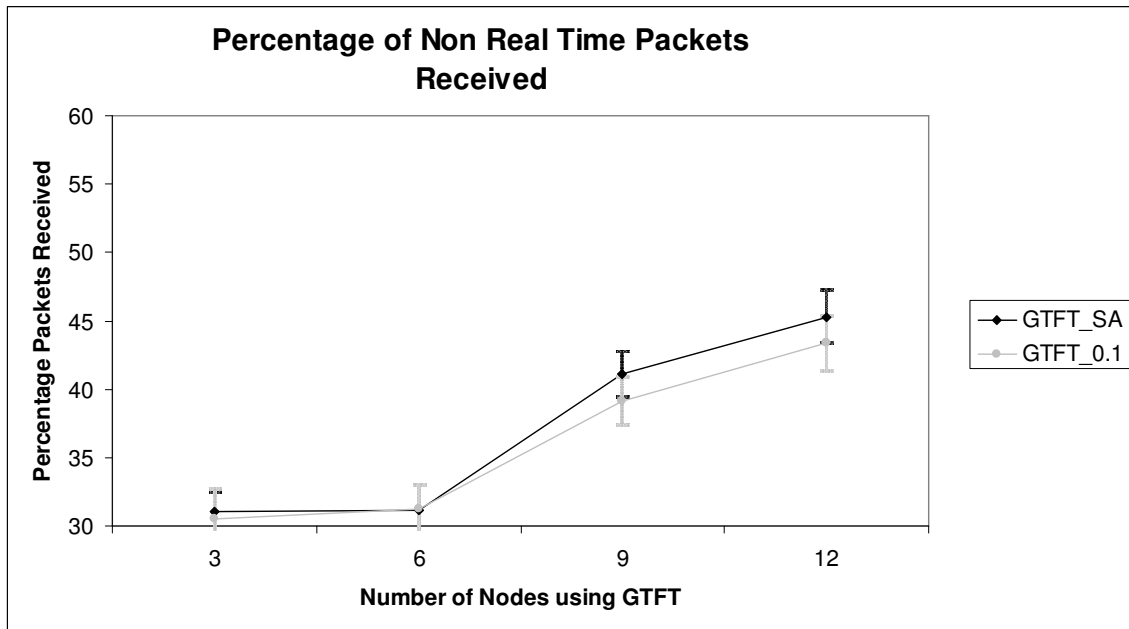


Figure 4.26: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces.



Figure 4.27: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces.

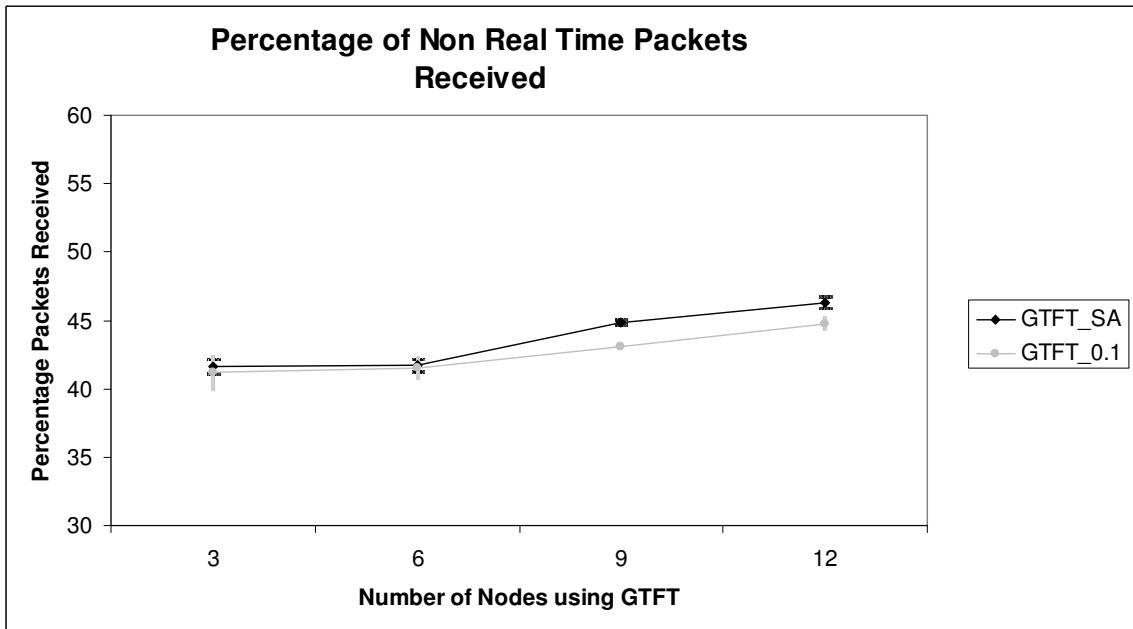


Figure 4.28: Four nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces.

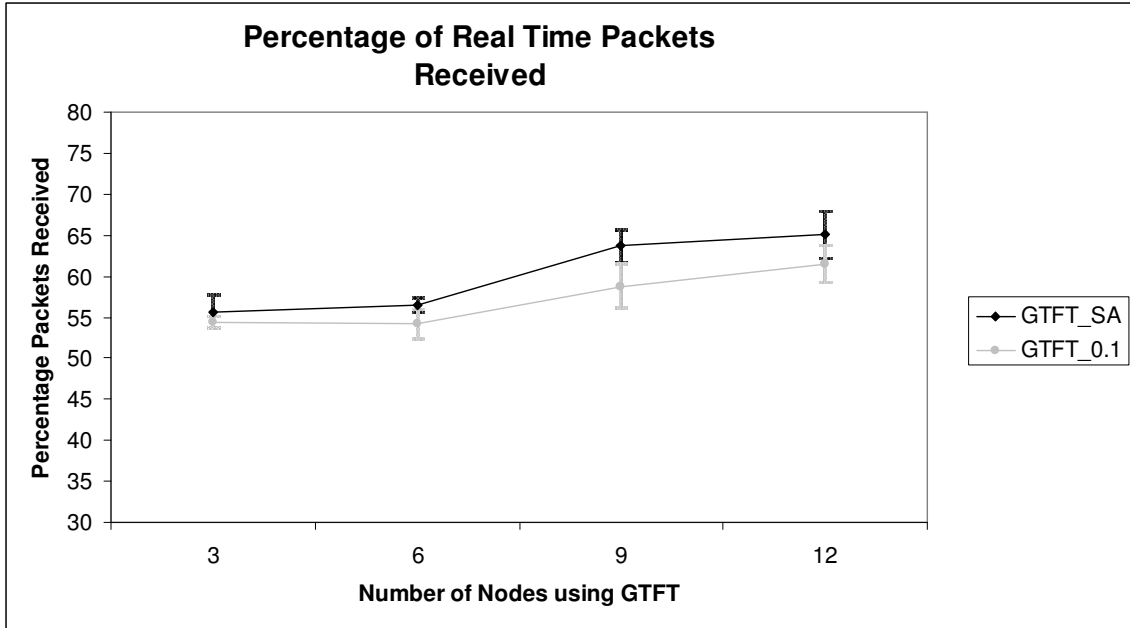


Figure 4.29: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces.

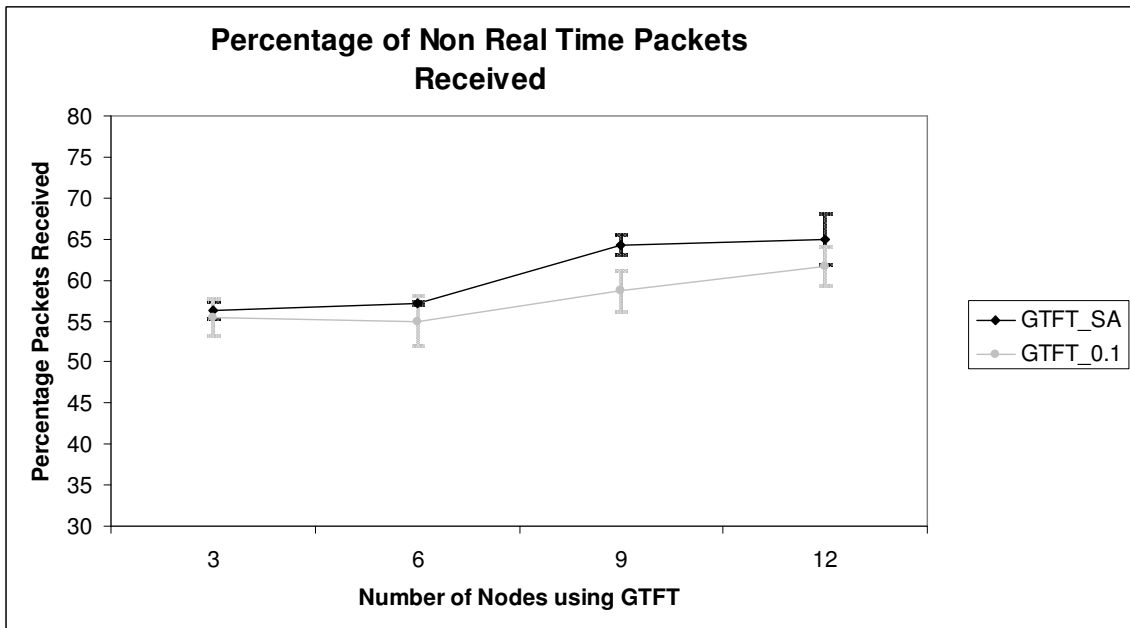


Figure 4.30: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 25% of all packets received; nodes follow MANIAC Challenge traces.

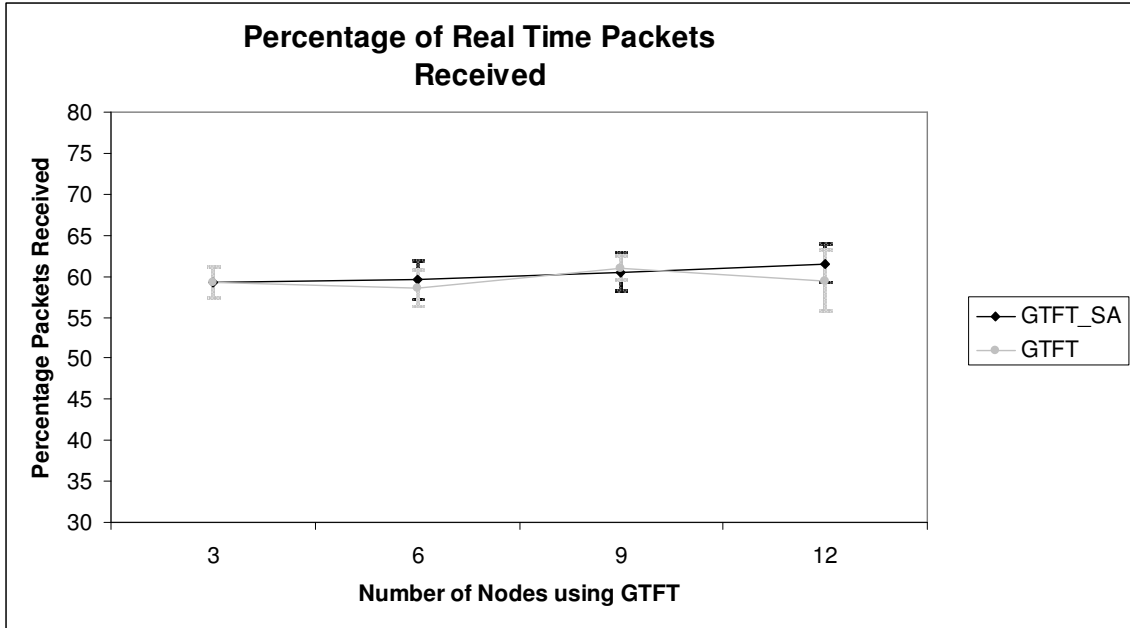


Figure 4.31: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces.

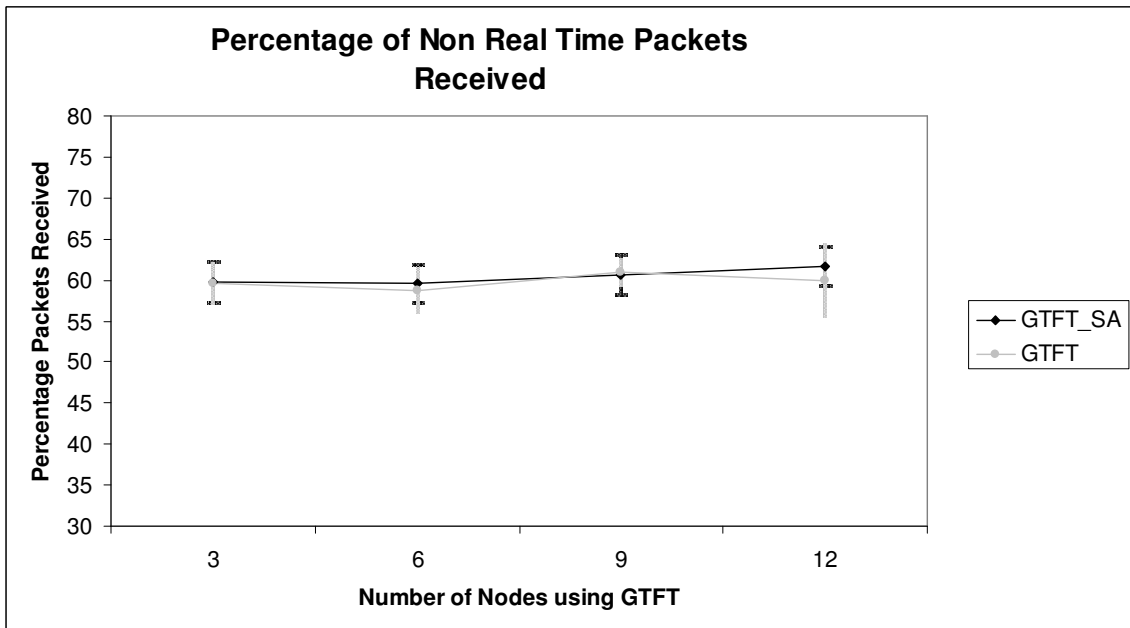


Figure 4.32: All nodes act as traffic generators; nodes that do not adopt GTFT follow a fixed percentage forwarding strategy, forwarding 75% of all packets received; nodes follow MANIAC Challenge traces.

4.4.4 Percentage Observation Error

Figure 4.33 is plotted to determine the effect of percentage observation error while calculating the reputation of a node. For example consider node A, forwarding packets to node B; suppose that either due to packet loss or link asymmetry node A is unable to listen promiscuously to node B forwarding its packets to their destinations. This inability of node A to overhear promiscuously its packets being forwarded by other nodes results in observation error. We define percentage observation error as the percentage of packets forwarded by a node that its neighbor was unable to overhear through promiscuous listening.

In our emulator implementation, the percentage observation error thus can be calculated as the ratio of the number of packets generated to emulate broadcast that cannot be correctly decoded due to asymmetric links or other communications impairments to the total amount of promiscuous packets generated by a node. Observation loss is induced in the emulator by dropping a percentage of the duplicate packets generated by nodes to emulate broadcast. Thus observation error leads a node to decrease the reputation index for other nodes in the network. The graph in figure 4.33 plots the percentage of packets received and the overall forwarding done in the network versus the percentage of observation error. Here, observation error for all the nodes in the network is increased in every run and network performance is analyzed using the total amount of forwarding and packets received in the network.

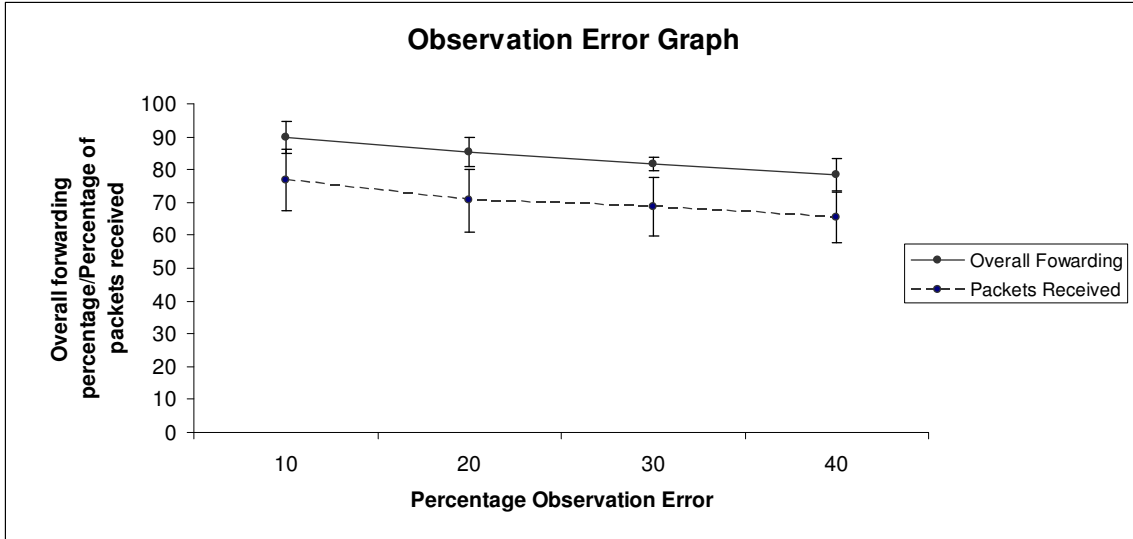


Figure 4.33: 4 nodes act as traffic generators; remaining nodes adopt GTFT_0.1 strategy; nodes follow the ns2 generated random waypoint mobility model.

As expected, the overall forwarding decreases as we induce more observation error in the network. Since not all of the traffic forwarded in the network is based on reputation tables, overall forwarding does not decrease sharply as observation error is increased. Similarly, as expected, the number of packets received decreases with an increase in observation error.

In this chapter results of experimentation are discussed and general observations regarding different trends are highlighted. The next chapter draws conclusions from these results and lays out some of the findings driven by the emulation design and experimentation. Future work direction is also discussed in the next chapter.

5 Conclusion and future work

Our main contribution involves building a test environment for analyzing cooperation strategies in MANETs. This environment is based on emulation using virtualization. Emulation requires less manpower and resources to replicate and is more flexible to changes compared to real life test beds and provides greater fidelity compared to simulation. Also emulation replicates the networking stack in every virtual node, thus it is not only limited to evaluating cooperation among nodes but can be used as a pretest tool in experiments investigating various issues in the protocol stack for a MANET.

5.1 Conclusions

Our work also involves characterizing selfish behavior and analyzing the effectiveness of cooperating strategies in a selfish mobile ad hoc network.

We conclude that by deploying the generous tit for tat (GTFT) strategy in a selfish environment a node can realize gain in performance, as indicated by the percentage of its traffic that successfully reaches the destination. GTFT nodes discourage selfish behavior in a network. As GTFT adds a generosity factor while reciprocating to other nodes in the network, the higher the generosity factor used the more cooperative the node becomes, in turn benefiting the overall performance of the network.

A further benefit in the performance is observed by using generous tit for tat with selfish node avoidance (GTFT_SA). GTFT_SA acts in a similar manner as GTFT but after the packet drop percentage crosses a certain threshold it attempts to re-route packets through more cooperative nodes.

The results obtained by deploying GTFT and GTFT_SA in a ad hoc network are consistent while using both real life (connectivity traces collected in the MANIAC Challenge) and ns-2 generated random waypoint mobility models. The network

performance is measured by accounting the percentage of packets forwarded by every node and the proportion of traffic correctly delivered to its destination.

5.2 Future Work

Future work direction involves experimentation with a larger set of strategies involving more complex reputation mechanisms. This can include using feedback from nodes outside one's one-hop neighborhood or assessing reputation according to acknowledgement of the packet being received by destination.

Assessing the effects on performance of deploying a combination of strategies and studying the interaction between these different strategies are also possible extensions to this work. Furthermore, a variety of mobility models can be included in future experiments to study the effect of different types of mobility on different cooperation strategies.

References

- 1 J. P. Buzen, "A simple model of transaction processing," *Proc. 10th Int'l Computer Measurement Group Conf. (CMG)*, 1984, pp. 835-839.
- 2 M. M. de Carvalho, "Analytical modeling of medium access control protocols in wireless networks," doctoral dissertation, Dept. of Computer Engineering, Uni. of California, Santa Cruz, 2006.
- 3 M. M. Carvalho and J. J. Garcia-Luna-Aceves, "Modeling wireless ad hoc networks with directional antennas," *Proc. 25th IEEE Conf. on Computer Comm. (INFOCOM 06)*, 2006, pp. 1-12.
- 4 M. Cagalj S. Ganeriwal, I. Aad and J. P. Hubaux, "On selfish behavior in CSMA/CA networks," *Proc. 24th IEEE Conf. Computer Comm. (INFOCOM 05)*, 2005, pp. 13–17.
- 5 L. Junghoon, "An analytical modeling approach for a large scale mobile ad hoc network," *Proc. Military Comm. Conf. (MILCOM 07)*, 2007, pp. 1-4.
- 6 Simulation, <http://www.yourdictionary.com/simulation>. Accessed on 12th Mar. 2009.
- 7 D. Cavin, Y. Sasson, and A. Schiper, "On the accuracy of manet simulators," *2nd ACM Int'l Workshop On Principles Of Mobile Computing. (POMC 02)*, 2002, pp. 38-43.
- 8 T. R. Andel and A. Yasinsac, "On the credibility of manet simulations," *IEEE Computer*, vol. 39, no. 7, 2006, pp. 48-54.
- 9 L. F. Perrone and Y. Yuan, "Modeling and simulation best practices for wireless ad hoc networks," *Proc. Simulation Conference*, 2003, pp. 685-693.
- 10 Network Emulation, http://en.wikipedia.org/wiki/Network_emulation. Accessed on 12th Mar. 2009.
- 11 M. Puzar and T. Plagemann, "Neman: A network emulator for mobile ad-hoc networks," *Proc. 8th Int'l Conf. Telecommunications (ConTEL 05)*, 2005, pp. 155- 161.
- 12 S. Sanghani, T. X. Brown, S. Bhandare and S. Doshi, "EWANT: The emulated wireless ad hoc network testbed," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC 03)*, 2003, pp. 1844–1849.
- 13 D. Raychaudhuri et al., "Overview of the ORBIT radio grid testbed for evaluation of next-generation wireless network protocols," *Proc. IEEE Wireless Comm. and Networking Conf. (WCNC 05)*, 2005, pp. 1664–1669.
- 14 The APE testbed, <http://apetestbed.sourceforge.net/>. Accessed on 12th Mar. 2009.

- 15 J. Flynn, H. Tewari, and D. O'Mahony, "JEmu: A real time emulation system for mobile ad hoc networks," *Proc. Conf. Comm. Networks and Distributed Systems Modeling and Simulation*. (CNDS 02), 2002, pp. 115–120.
- 16 P. Mahadevan, A. Rodriguez, D. Becker and A. Vahdat, "MobiNet: A scalable emulation infrastructure for ad hoc and wireless networks," *Proc. Int'l Workshop on Wireless Traffic Measurements and Modeling* (WiTMeMo 05), 2005, pp. 7–12.
- 17 Y. Zhang and W. Lei, "An integrated environment for testing mobile ad-hoc networks," *Proc. 3rd ACM Int'l Symposium on Mobile Ad hoc Networking and Computing* (MobiHoc 02), 2002, pp. 104–111.
- 18 P. Zheng and L.M. Ni, "EMWIN: Emulating a mobile wireless network using a wired network," *Proc. 5th ACM Int'l Workshop on Wireless Mobile Multimedia*. (WoWMoM 02), 2002, pp. 64–71.
- 19 P. Zheng and L.M. Ni, "Empower: A cluster architecture supporting network emulation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 7, 2004, pp. 617-629.
- 20 R. Ramanathan and R. Hain, "An ad hoc wireless testbed for scalable, adaptive QOS support," *Proc. IEEE Wireless Comm. and Networking Conf.* (WCNC 00), 2000, pp. 998-1002.
- 21 J. Macker, W. Chao, and J. Weston, "A low-cost, IP based mobile network emulator (MNE)," *Proc. IEEE Military Communications Conference*. (MILCOM 2003), 2003, pp. 481–486.
- 22 S. Maier, D. Herrscher, and K. Rothermel, "On node virtualization for scalable network emulation," *Proc. Int'l Symposium on Performance Evaluation of Computer and Telecommunication Systems*. (SPECTS 05), 2005, pp. 917–928.
- 23 E. Nordstrom, P. Gunningberg, and H. Lundgren, "A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks," *Proc. 1st IEEE Int'l Conf. Testbeds and Research Infrastructures for the Development of Networks and Communities*. (TRIDENTCOM 05), 2005, pp. 100-109.
- 24 X. W. Huang, R. Sharma and S. Keshav, "The entrapid protocol development environment," *Proc. IEEE 18th Ann. Conf. Computer and Comm. Societies*. (INFOCOM 99), 1999, pp. 1107-1115.
- 25 A. Vahdat et al., "Scalability and accuracy in a large-scale network emulator," *In Proc. 5th Symposium on Operation Systems Design and Implementation*. (OSDI 02), 2002, pp. 271-284.
- 26 VNUML wiki, http://www.dit.upm.es/vnumlwiki/index.php/Main_Page. Accessed on 12th Mar. 2009.
- 27 A. Haeberlen, D. Marcel, G. Krishna and S. Stefan, "Monarch: A tool to emulate transport protocol flows over the internet at large," *Proc. 6th ACM SIGCOMM on Internet measurement*. (IMC 06), 2006, pp. 105–118.

- 28 E. Gokturk, "A stance on emulation and testbeds, and a survey of network emulators and testbeds," *Proc. 21st European Conference on Modeling and Simulation*. (ECMS 07), 2007,
- 29 Maniac Challenge, <http://www.maniacchallenge.org/>. Accessed on 20th Aug. 2008.
- 30 P. Raniwala, A., Sharma and T. Chiueh, "MiNT: A miniaturized network testbed for mobile wireless research," *Proc. 24th IEEE Conf. on Computer Comm.* (INFOCOM 05), 2005, pp. 2731-2742.
- 31 Z. Ji et al., "WHYNET: A hybrid testbed for large-scale, heterogeneous and adaptive wireless networks" UCLA Computer Science Department Technical Report CSD-TR060002, January, 2006.
- 32 NRT, <http://www.cs.ucla.edu/ST/testbeds.html>. Accessed on 12th Mar. 2009.
- 33 D.A. Maltz, J. Broch, and D.B. Johnson. "Lessons from a full-scale multihop wireless ad hoc network testbed," *IEEE Personal Communications*, Vol. 8, no. 1, 2001, pp.8-15.
- 34 H. Ritter, M. Tian, T. Voigt, J. Schiller, "A highly flexible testbed for studies of ad-hoc network behavior," *Proc. 28th Ann. IEEE Int'l Conf. Local Computer Networks*, 2003, pp. 746.
- 35 MIT Roofnet, <http://pdos.csail.mit.edu/roofnet/doku.php?id=design>. Accessed on 12th Mar. 2009.
- 36 NetKit, <http://www.netkit.org/>. Accessed on 10th Jan. 2009.
- 37 Xen Virtual machine monitor, <http://www.cl.cam.ac.uk/research/srg/netos/xen/>. Accessed on 13th Mar. 2009.
- 38 VMware, <http://www.vmware.com>. Accessed on 13th Mar. 2009.
- 39 The MLN Project, <http://mln.sourceforge.net/>. Accessed on 13th Mar. 2009.
- 40 M. Carson and D. Santay, "Nist net: a Linux-based network emulation tool," *SIGCOMM Computer Comm. Review*. vol. 33, no. 3, 2003, pp. 111-126.
- 41 L. Rizzo, "Dummysnet: A simple approach to the evaluation of network protocols," *ACM Computer Comm. Review*, vol. 27 no. 1, 1997, pp. 31-41.
- 42 S. Zhong, J. Chen and Y.R. Yang, "Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks," *Proc. IEEE INFOCOM*. vol. 33, no. 3, 2003, pp. 1987-1997.
- 43 J. L. Boudec and S. Buchegger, "Performance analysis of the CONFIDANT protocol," *Proc. 3rd ACM Int'l Symposium Mobile Ad hoc Networking and Computing*. (MobiHoc 02), 2002, pp. 226-236.

- 44 R. Molva and P. Michiardi, "Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks," *Proc. 6th Joint Working Conf. on Comm. and Multimedia Security*. 2002, pp. 107-121.
- 45 M. T. Refaei, V. Srivastava, L. A. DaSilva and M. Eltoweissy, "A reputation-based mechanism for isolating selfish nodes in ad hoc networks," *2nd Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*. (MobiQuitous05), 2005, pp. 3-11.
- 46 V. Srivastava, A. B. Hilal, M. S. Thompson, J. N. Chattha, A. B. MacKenzie and L. A. DaSilva, "Characterizing mobile ad hoc networks - the MANIAC challenge experiment," *WINTECH 2008*, pp. 65-72.