ATTAINING REALISTIC SIMULATIONS OF MOBILE AD HOC NETWORKS

THESIS

Derek J. Huber, Captain, USAF

AFIT/GCO/ENG/10-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCO/ENG/10-11

ATTAINING REALISTIC SIMULATIONS OF MOBILE AD HOC NETWORKS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Derek J. Huber, BS

Captain, USAF

June 2010

AFIT/GCO/ENG/10-11

ATTAINING REALISTIC SIMULATIONS OF MOBILE AD HOC NETWORKS

Derek J. Huber, BS
Captain, USAF

Approved:

//Signed//	27 Apr 2010
Maj Mark D. Silvius, PhD (Chairman)	Date
//Signed//	27 Apr 2010
Dr. Kenneth M. Hopkinson (Member)	Date
//Signed//	27 Apr 2010
Maj Ryan W. Thomas, PhD (Member)	Date

AFIT/GCO/ENG/10-11

Abstract

Mobile Ad-hoc Networks (MANET) are comprised of wireless systems that communicate without the assistance of centrally managed routers or base stations. MANET research and development has increased due to computing technologies offering smaller, faster, smarter, and more power efficient platforms to operate on. Largely the testing and evaluation of new and existing MANET protocols has resided in simulation environments. This is due in part to the complexities and expenses incurred when conducting real world tests. Many researchers have come to recognize that these current simulations tend to assume away critical components of the MANET domain. These assumptions are made either to simplify the physical layer of the simulation so that the protocol can be tested or out of necessity because the current simulation platforms are not capable of providing a more realistic physical layer simulation environment. This thesis is focused on addressing these assumptions that affect the physical layer of the MANET protocol by gathering data in the real world and then modifying the simulation environment to model as closely as possible to the gathered results. This modified environment is then compared to the basic MANET simulation environment by analyzing packet delivery and propagation effects of both models.

Acknowledgements

I would like to express my sincere appreciation to all of my faculty advisors, Lt Col Stuart Kurkowski and Maj Todd Andel, for their guidance and support throughout the course of this thesis effort. Also I would like to extend my gratitude to Maj Mark Silvius for filling the role as my advisor during the last few months of my thesis work when Lt Col Kurkowski had departed AFIT.

I of course would never have completed this work without the support of my amazing wife and two loving daughters. Many people have sacrificed to assist me with my thesis and I’ll always be grateful for their time and efforts.

Derek J. Huber

Table of Contents

List of Figures

List of Tables

ATTAINING REALISTIC SIMULATIONS OF MOBILE AD HOC NETWORKS

1. Introduction

Wireless (a.k.a. mobile) communications have become ingrained in day to day operations of both personal and professional arenas within the modern world. Computational mobility is becoming one of the key factors in design and purchase of Information Systems. Wireless communications in operation today follow closely to Moore's Law by getting faster, smaller, and cheaper roughly every two years. One example of mobility as a driving force in today's computing environment is the fact that laptop sales have overcome desktop PC sales [2]. This is attributed primarily to users and businesses wanting the flexibility of a portable computing system. There exist many additional examples of wireless communication devices gaining popularity among users such as cell phones, in particular "smart phones" that provide data communication in addition to voice.

These devices that communicate wirelessly require a network topology to support the varied communications that propagate between devices. This topology can adhere to one of three different main approaches, either an infrastructure based approach, an ad-hoc approach, or a hybrid of the two. Currently most commercial wireless networks, as well as wired networks, follow an infrastructure approach, which means there is a central node(s) or base station(s) that performs the routing of packets internally and externally to the network. In a wireless network this is called an Access Point (AP) or wireless router, this device makes the routing decisions for the nodes connected to it. Figure 1-1 illustrates a basic infrastructure based wireless network. The nodes are configured to send their traffic to the central management node (a.k.a. AP) where it will then route the traffic accordingly.

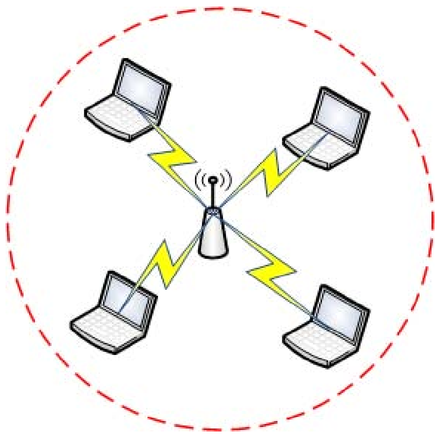


Figure 1-1. Example of an infrastructure based network

Ad-hoc networks differ significantly from infrastructure based networks as stated previously since there is no central routing node in the topology. Figure 1-2 illustrates a basic ad-hoc network where each node can pass traffic with its peers as long as they are within each other's coverage areas (red dashed circles). In this environment every node will require a protocol that can effectively discover routes and then transmit packets without any overarching administration.

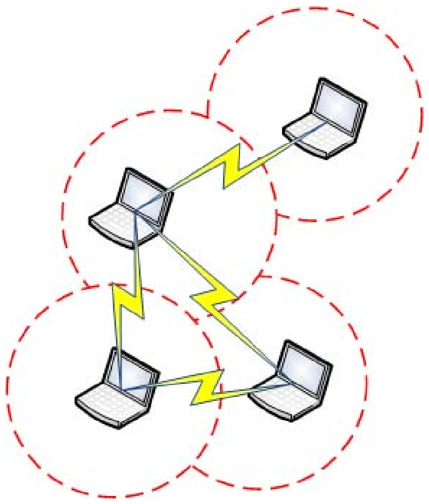


Figure 1-2. Example of an ad-hoc based network

Only a few commercial applications exist today that follow a Mobile Ad-hoc NETworking (MANET) approach. One such example is from the Swedish company

TerraNet that is using Peer-to-Peer technology to allow users to communicate directly between two or more cellular handsets without the need for a base station [5]. Mobile ad-hoc networking, even with limited acceptance in commercial applications to date, has been utilized quite heavily in the research community and is presently being pursued as a viable option for future wireless communication applications. MANET technology is also being researched for possible application into military wireless communications. MANETs are studied because of the many potential benefits that can be extracted from communication systems when there is greater mobility and smaller infrastructure footprints. This will drive a higher demand for MANETs in the computing world of the near future.

1.1. Problem Statement

There are multiple scenarios where MANETs can prove to be a vital communications resource. One possible scenario is after a disaster, natural or man-made, the need for communications between emergency responders, government(s) officials, military, and victims is imperative. In large scale disasters the communications grid can be damaged or even taken completely offline, forcing a desperate need for communications and the ability to communicate without a communication infrastructure.

MANETs offer the ability to deploy wireless nodes that can efficiently and effectively route communication traffic. These nodes can be placed in responders' vehicles and even carried by individuals. They can be deployed on Unmanned Aerial Vehicles (UAV) and flown over specific areas where communication is vital to the rescue efforts. Then as rescue teams and units move throughout an area trying to find survivors they will be able to provide Situational Awareness (SA) to their counterparts through these wireless nodes.

In this scenario and multiple others that MANETs can be utilized, they all rely on a MANET protocol's reported capabilities. A protocol's capabilities and specifications are typically acquired through simulation experiments that help to determine key aspects of the protocol such as the effective bandwidth, overhead packet rate, latency, range, etc. The protocol's capabilities are then used when a MANET designer attempts to deploy the wireless nodes. It is important that the designer realizes the real world capabilities because the deployment of these nodes will affect the traffic patterns and overall communication capability of the MANET. In the previously mentioned scenario if the designer assumes the simulated capabilities of the MANET are correct and then deploys too few or too many nodes it could jeopardize the overall rescue efforts by limiting communication between responders. This thesis is focused on addressing poor assumptions that affect the physical layer of the MANET protocol by gathering data in the real world and then modifying the simulation environment to model as closely as possible to the gathered results.

1.2. Scope

The MANET community has developed numerous protocols and their performance and reliability are constantly being tested and evaluated. These protocol evaluations need to be as realistic as possible in order to provide useful feedback to the developers. This research is focused on the development of a simulation environment that is able to accurately model real world characteristics of a MANET protocol. This work sets the stage for future development in this area. Researchers can use this thesis to assist their development of robust simulations that conform to a realistic implementation of their specified protocol.

There are many different approaches and techniques that can be explored in order to investigate the problem with current MANET simulations. This work focuses on several assumed axioms that most MANET simulations use to simplify the physical layer of the simulation environment, as identified by Kotz et al [26]. For this thesis, the simulation environment was restricted to the Network Simulator version 2 (NS-2) simulation engines to evaluate these assumptions. This work does not intend to address all possible scenarios that can arise in a MANET implementation and is instead intended to be the groundwork for a more robust simulation environment at the physical layer level.

1.3. Organization

This chapter provided a general overview of the research conducted in this thesis and the main problem in the MANET field that it attempts to address. Chapter Two describes the general workings of MANETs and discusses two different types of protocols. It also discusses in detail the problems facing MANET simulations particularly modeling the physical layer. It concludes by reviewing the *Click* modular software architecture that is used as the test bed's backbone in this thesis. Chapter Two is intended for those with a general knowledge of MANETs and is written to provide background on the current MANET research projects and the problems that this thesis is aimed at solving. Chapter Three covers the experimental methodology and the overall system that was developed to create realistic physical layer MANET simulations. Chapter Four documents the performance of the realistic simulation environment to the basic simulation environment. Chapter Five summarizes the work and results that were accomplished in this thesis and concludes by offering recommendations for further research and development.

2. Background

This chapter is devoted to the background of Mobile Ad-hoc NETworking (MANET) design, protocols, and simulation techniques. In addition to the background on MANETs and network simulation, the *Click* Modular Software Architecture will be introduced [24]. The capabilities of *Click* are discussed along with previous work utilizing *Click* for MANET protocol development and testing.

This thesis focuses on the physical layer representation within MANET simulations and the assumptions that are made in these simulation environments. Since MANET technology can be done on multiple platforms and through various techniques and tactics a general MANET overview is required. The mechanisms used in creating MANETs are covered in the first section of this chapter along with the general workings of two MANET protocols, Dynamic Source Routing (DSR) and Destination Sequence Distance-Vector (DSDV). These two protocols are used in this research so a basic understanding is needed. Next the ability to simulate MANETs is investigated since this is where the initial problems are introduced that this work addresses. Within this section, the numerous studies on MANET simulations that have been done in regards to their strengths and weaknesses are reviewed. Finally, the *Click* Modular Software Architecture has been developed as the backbone to the physical test bed. The *Click* architecture is discussed and how its granular design allows for MANET protocol implementation. In addition to the workings of *Click*, previous research is analyzed that has been conducted with the *Click* framework to develop MANET protocols.

These various topics are discussed in this chapter because an understanding of the workings of a MANET is required to realize the contributions of this thesis work. When viewing the Open System Interconnection (OSI) Reference Model, Figure 2-1, the

MANET protocols discussed in the next few sections operate within layers 2-4, but as with all protocols they depend heavily on a physical layer that is functioning correctly. This thesis work is focused on the physical layer and the simulation environment's ability to model this layer as close to the real world as possible.

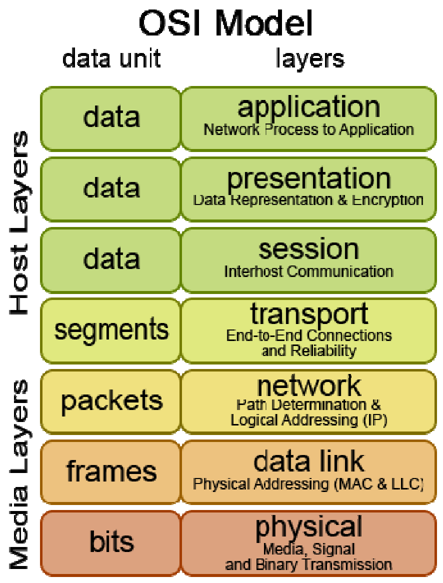

OSI Model
data unit
layers
Host Layers
Media Layers
data
application
Network Process to Application
data
presentation
Data Representation & Encryption
data
session
Interhost Communication
segments
transport
End-to-End Connections and Reliability
packets
network
Path Determination & Logical Addressing (IP)
frames
data link
Physical Addressing (MAC & LLC)
bits
physical
Media, Signal and Binary Transmission

Figure 2-1. OSI Reference Model (URL: http://upload.wikimedia.org/wikipedia/commons/2/2b/Osi-model.png)

2.1. Mobile Ad-hoc NETworks (MANET)

Ad-hoc networking concepts have been present throughout history in most communications. The ability to transmit messages by passing them through multiple nodes is not unique to the computing domain of today's modern world. Governments and militaries have experienced the benefits of relaying messages along multiple paths and messengers from source to destination. Naturally when a new technology is introduced these same benefits are expanded upon and yield even greater benefits. Ad-hoc networking/routing began its initial development within the computing domain, in particular the wireless spectrum, when ALOHAnet in 1970 was developed to allow the University of Hawaii to link up with the many islands it had facilities on. "Even though ALOHAnet was originally implemented for a fixed single-hop wireless network, the basic idea was compelling and applicable to any environment where access to a common resource had to be negotiated among a set of uncoordinated nodes" [30]. This opened the door for future research into Ad-hoc technologies and concepts. Ad-hoc networks are today classified as self-organizing and adaptive networks, which means they can create and dissolve networks without the need for system administration or more importantly fixed infrastructure [39].

The Internet Engineering Task Force (IETF) created a working group, Mobile ad-hoc networks, in order to standardize the many protocols and specifications that were emerging in this domain. Their purpose is stated as, "…to standardize IP routing protocol functionality suitable for wireless routing application within both static and dynamic topologies with increased dynamics due to node motion or other factors" [29]. They have identified that there are two different categories that most MANET routing protocols can be classified under: Reactive MANET Protocol (RMP) and Proactive MANET Protocol (PMP). The working group currently has multiple drafts and Request For Comments (RFC) out in the community for a variety of protocols and specifications that fall under these two tracks. The working group is also currently pursuing two specific protocols, Dynamic MANET On-demand (DYMO) and Optimized Link State Routing Protocol version 2 (OLSRv2), an RMP and PMP respectively [29]. It is yet to be seen if these two protocols will become the official IETF standards and consequently gain acceptance and commercial production within the industry.

Other MANET publications and groups have divided the protocols into two groups as well, commonly referred to as On-Demand Routing Protocols or Table-Driven Routing Protocols, respectively matching to RMP and PMP [30]. There are some protocols that fall under a hybrid of the two classifications, but most tend to follow one or the other. Since the work in this thesis uses just one protocol from each track the small differences a hybrid protocol offers is not discussed.

2.1.1 Reactive Routing

Reactive MANET protocols adhere to the principle that due to the mobility of the wireless nodes, all possible routes within the network do not need to be discovered only when traffic needs to be sent between nodes. The goal of this is to decrease bandwidth usage by the nodes when sending administrative packets. There is no need to send periodic routing updates throughout the network as seen with Proactive Routing [30]. There are several protocols within the MANET community that adhere to a Reactive approach such as: Dynamic Source Routing (DSR) [20], Ad-hoc On-Demand Distance Vector (AODV) [37], Temporally Ordered Routing Algorithm (TORA) [33], and DYMO Routing [12]. The DSR protocol is discussed in the following section.

2.1.1.1. Dynamic Source Routing (DSR) Protocol

DSR was developed to be a mobile ad-hoc protocol with the initial design and concepts published in 1994. It has gone through numerous revisions and was most recently submitted in 2007 to the IETF as The Dynamic Source Routing (DSR) Protocol for Mobile Ad-hoc Networks for IPv4 (RFC 4728). The protocol is designed to be "…completely self-organizing and self-configuring," which is the underlying principle of a MANET protocol [20]. The designers wanted to keep the protocol efficient and reduce the amount of overhead packets needed to pass traffic between nodes. The RFC claims that DSR can be used in networks comprised of 200 nodes and work reliably even with high rates of mobility within that network. There are two main mechanisms within DSR that allow the protocol to operate within the ad-hoc domain, Route Discovery and Route

Maintenance. In fact most MANET protocols, reactive or proactive, utilize these two mechanisms in order to learn about the network. Route Discovery allows a node to send traffic to another node when the source node does not have a route stored in its route cache. Route Maintenance is utilized in DSR MANETs to ensure existing routes within a node's route cache are still viable.

2.1.2 Proactive Routing

Proactive Routing differs from Reactive in that every node within the network will first discover every possible route in the network before passing traffic. These routes are stored within a routing table on each node. They usually resemble a wired network routing protocol because they attempt to route packets along the optimal path [36]. In order to remain self-configuring and self-organizing, most PMPs utilize two mechanisms, Route Discovery and Route Maintenance, in order to remain self-configuring and self-organizing. The PMP used in this thesis work is the Destination Sequence Distance-Vector Protocol (DSDV) [37], which does not officially state that it utilizes Route Discovery and Route Maintenance, but it does however implement these two mechanisms.

2.1.2.1. Destination Sequence Distance-Vector (DSDV) Protocol

Destination Sequence Distance-Vector was developed in 1994 for use in MANETs by Charles Perkins and Pravin Bhagwat. This was one of the first MANET protocols developed and the authors pulled most of the techniques and mechanisms from the wired protocols in use at that time. The main algorithm it is based on is the Distributed Bellman-Ford (DBF), "…where each node maintains a table that contains the shortest distance and the first node on the shortest path to every other node in the network" [30]. Unlike DSR there is no overarching attempt to limit the amount of overhead packets being sent across the network. The Route Discovery process in DSDV occurs for each node once it comes online. The newly “up” node will advertise its current routing table by broadcasting it to all of its neighbors within its wireless coverage area. Just as in DSR the Route Maintenance process occurs when a node/link has gone down. The big difference is that every node in the MANET is notified of the broken link and their routing tables are updated and broadcasted throughout the network.

2.1.3 Challenges

There are multiple challenges that every MANET protocol faces that infrastructure based topologies don’t face. Some are protocol specific, but most are due to the basic characteristics of a MANET. This section only summarizes these challenges and does not attempt to derive any of the underlying problems. They are reviewed to show that other factors are affecting MANETs in addition to the problems this thesis addresses. The first and foremost challenge is due to the operation of a network in the radio spectrum. Radio communications are susceptible to a lot more environmental influences than wired communications. Numerous studies have shown that wind, temperature, humidity, etc. all impact radio transmissions in either positive or negative trends. This thesis is utilizing hardware that operates in the 2.4GHz band, which in the US is an open frequency that anyone can utilize without a license. With the increase of wireless nodes utilizing the open band, it becomes saturated with traffic that distorts or blocks MANET transmissions [39].

Another challenge facing MANET communications is energy efficiency. Since MANETs are inherently formed by mobile nodes that are usually operating on battery power alone, the active role each node plays in routing traffic now has a significant impact on the uptime of each node. Transmitting and receiving packets wirelessly requires a great deal of power and can quickly reduce the amount of time a node can stay online while on battery power. This is a problem in general to all wireless topologies, but MANET nodes consume even more power since they are utilized by their peers to route traffic throughout the network [39].

Lastly every MANET faces higher security risks either through malicious or poorly configured nodes. The fact that MANET traffic is dependent on other peer nodes for routing to the end destination introduces the potential for man in the middle attacks. This means a malicious node could easily alter packets without the sender or destination node realizing it. MANETs also face risks in poorly configured nodes because nodes that do not accurately display correct routing information can increase congestion and result in undeliverable packets if the poorly configured node advertises a route that doesn't exist.

2.2. Network Simulation Environments

Network topologies have been simulated since their inception in the computing domain. Simulation is a useful process to undertake when a new protocol is being developed, but it is also beneficial to simulate existing protocols when implemented in new and original topologies. Simulation in a computer domain is primarily done in environments that use discrete-event methods to simulate behavior of a network. This means the observer is only concerned with the system being simulated at discrete points in time [18]. Most simulation engines were first designed to simulate wired networking topologies, but now the major simulators offer the capability to simulate wireless networks.

The most notable simulation engines that are used widely in research and commercial endeavors are: Network Simulator version 2 (NS-2) [19], Optimized

Network Engineering Tools (OPNET) [33], and Global Mobile Information Systems Simulation Library (GloMoSim) [30]. NS and GloMoSim are both provided as open use software to researchers, whereas OPNET is a commercial simulation engine. While all of these engines can simulate MANETs, the one used in this thesis work is NS-2 because of the ability to model the physical layer within NS-2.

2.2.1 Network Simulator version 2 (NS-2)

NS-2 is a discrete simulator that was developed to assist researchers doing work with computer networks. It started as a spinoff to the REAL Network Simulator in 1989 and has undergone three major versions [19]. Each of the version changes do not allow for simple backwards compatibility of older code and scenario files so a lot of users tend to utilize older versions instead of updating NS-2 to the latest version. The code utilized in this thesis work requires the use of NS-2 up to patch level 34. The following paragraphs discuss features of NS-2 that may or may not be found in the other versions.

NS-2 is founded on two programming languages Tcl and C++. C++ programming knowledge is not needed unless new functionality is to be added. NS-2 also comes prepackaged with a variety of existing classes and configurations that can be used to construct various scenarios. Tcl programming knowledge, however, is required in order to generate even basic scenario files. Scenario files are Tcl scripts that simulate a simple network topology. These scripts can then be run within the NS-2 simulation engine, which has integrated the Network Animator (NAM) for graphical feedback. There are numerous example configurations in NS-2 including MANET protocol scenario files containing DSDV implementations [19].

Figure 2-2 shows a screen capture of an example Tcl script running within NS-2. This example came from Marc Greis's online tutorial and is composed of four nodes with node 2 acting as a router for the other three nodes [19]. It will only forward data from node 0 and node 1 onto node 3. It also illustrates a basic queue within NS-2 by showing the red and blue packets queuing up on node 2.

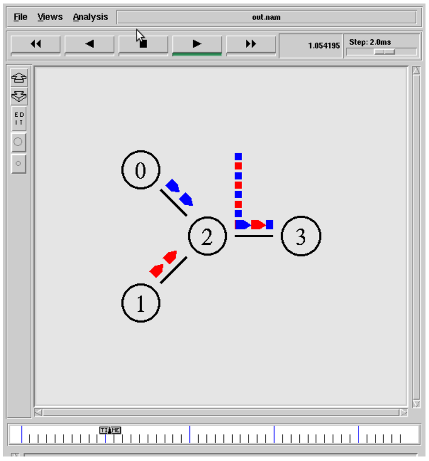


Figure 2-2. NS-2 Simulation Environment NAM Graphical Display

NS-2, like the other network simulators, models the wireless radio propagation of each node using the Friss-space (a.k.a. Free space) attenuation ($\frac{1}{r^2}$) at near distances and approximates to Two Ray Ground ($\frac{1}{r^4}$) at far distances. The approximation assumes specular reflection off a flat ground plane [15]; meaning NS-2 assumes there are no signal obstructions. These different propagation models are used by NS-2 to determine if the packet is successfully received by the destination node. In addition to the propagation models, NS-2 utilizes an omni-directional antenna that has unity gain in all directions [15]. This means the propagation coverage area is represented as a perfect circle surrounding the wireless node at a radius specified by transmission power, transmit gain, receive gain, receive threshold, wavelength, frequency, and other factors. NS-2 does offer another propagation model, Shadowing, which was added to its base installation to provide a more realistic radio model. The shadowing model is used to apply fading effects to the propagation models since the received power at a certain distance in the physical world is a random variable due to multipath propagation effects. The shadowing model still assumes a perfect circle as the other two models do, but it extends it by stating that "…nodes can only probabilistically communicate when near the edge of the communication range" [15]. The shadowing model does offer a more realistic approach to MANET simulation since the radio signal will drop off depending on the user supplied value, but it still only models an antenna coverage pattern of a circle.

2.2.2 MANET Simulations Current State

The Mobile Ad-hoc research community has been simulating different MANET protocols for quite some time, mainly because it requires an extensive amount of resources to build actual MANETs for experimentation. A problem with these simulations is that they make numerous assumptions to simplify the simulation environment. Not only are numerous assumptions made, but the simulations that are run are not rigorous in their methodology, or in their analysis. "An opinion is spreading that one cannot rely on the majority of the published results on performance evaluation studies of telecommunication networks based on stochastic simulation, since they lack credibility" [33].

These assumptions and lack of rigor are the main reason the results of the simulations are not trustworthy. Assuming critical characteristics of a protocol are constant or not a factor in the outcome renders the results invalid. Without accurate simulations, protocol development and testing has limited effectiveness since only a subset of the protocol's attributes is evaluated. The MANET community needs reliable simulations that can model real world attributes. New protocols are being developed and advertised as secure and reliable but the simulation models are unable to provide an accurate depiction of how the new protocol will actually perform, "…according to the feeling of the MANET community, there is an important lack of real experiments that prove the feasibility of wireless protocols" [11].

Cavilla et al. have identified that most MANET simulations are not modeled for indoor environments thus ignoring the effects that an indoor environment may introduce to a MANET. Most notably they identify that indoor environments are smaller than traditional outdoor MANET test environments. These smaller environments usually contain multiple obstacles to node mobility and radio propagation [10] .

There are also six axioms that are accepted and utilized in today's MANET simulations [26]. Though widely held, they are the source of certain problems in MANET simulations. Each of these axioms is implemented in the basic propagation models within NS-2 as well as the other network simulators. These axioms are also grounded very heavily in the physical layer since they are all concerned with the MANET's ability to transmit packets effectively between nodes. In the physical world these axioms are handled by the radio and hardware of each node, but in simulation, NS-2 has to handle these physical layer axioms.

Axiom 0 is that *the world is flat*, meaning that all of the nodes are on the same horizontal plane. This introduces simulation errors because link quality and signal strength varies depending on the elevation of each node. If the elevation of the node is different, then the signal will be affected accordingly. This axiom is also important due to line-of-sight (LOS) propagation that affects the wireless signal's transmission. LOS is affected by the topography of the land in outdoor environments. It is also affected by obstructions such as trees, buildings, or other objects in between the wireless nodes in both outdoor and indoor environments. While most MANETs can deliver traffic with a few obstructions in a small proximity, the simulations that are done routinely assume that only open air exists between nodes. This assumed axiom is accepted by most of the MANET community because the belief of assuming the world is flat will not affect MANETs since they usually have a small coverage area. Kotz et al. state that there are two things a MANET researcher must do in order to account for this axiom within their MANET simulations: "a) use a detailed and realistic terrain model, accounting for the effects of terrain, or b) clearly condition their conclusions as being valid only on flat, obstacle-free terrain" [26]. NS-2 simulations implement this axiom because NS-2 currently only offers 2D terrain support meaning that all nodes are on the same horizontal plane. There is however some research projects that have extended NS-2's ability to include the Z axis by implementing a Fresnel zone base propagation model [15].

Axiom 1 is that *a radio's transmission area is circular*, which never occurs in the physical world since each antenna has a different coverage area based on environmental factors and antenna construction. Also each MANET protocol will affect the coverage area of a node differently due to the number of nodes, traffic overhead, and other factors that the protocol introduces to the transmission area. Axiom 2 is that *all radios have equal range*, which ties in very closely with axiom 1. In fact, since the coverage area is not circular each radio has a different range based on environmental factors and the angle between the sender and receiver antennas. Just as in axiom 1, the MANET protocol in use will affect the range of the radios even if they are relatively the same [26]. NS-2 adheres to these two axioms because all of the propagation models (Free space, Two Ray Ground, Shadowing) simulate the radio's coverage/transmission area as an ideal circle [16]. There is no disparity between the ranges of the radios in the models other than applying the fading effects of the shadowing model. Even these fading effects will be applied consistently to all nodes in the simulation so that there is no distinct difference between them.

Axiom 3 is *if I can hear you, you can hear me (symmetry).* This is a poor assumption since the entire premise of a MANET protocol is that the nodes are mobile. Just because node A can send to node B at one instant in time does not mean that node B can send to node A after the message is received since the nodes may have been traveling in opposite directions and may now be out of the coverage area [26].

Axiom 4 is *if I can hear you at all, I can hear you perfectly*, which is an optimistic assumption since most transmissions that occur between two nodes on the fringes of their coverage areas will experience occasional dropped packets or bit errors. This means they can still communicate sporadically, but not without error [26].

Finally, axiom 5 states that s*ignal strength is a simple function of distance*. It is true that signal fading is well understood, but the equation usually employed does not necessarily represent environmental factors that increase or decrease the radio's range such as: obstruction, reflection, refraction, and scattering [26]. This axiom ties back into axiom 0, in that the environment will play a key role in affecting the wireless transmission. This assumption also ties back into axiom 1 in that the coverage area is equal at all angles around the node and the strength of the signal will decrease only because of distance.

Simulations are useful but in their current state they have limited benefits for testing and evaluating the various MANET protocols. The community has realized that key environmental and technical aspects have yet to be accurately modeled in any of the major simulation engines such as OPNET Modeler, NS-2, or GloMoSim.

2.3. *Click* Modular Software Architecture

In addition to the creation of a simulation environment for MANET protocols, the *Click* software architecture has been utilized within each of the nodes in the physical test bed. The *Click* software architecture was developed in the Computer Science department at the Massachusetts Institute of Technology (MIT). It was developed to be a “flexible, modular software architecture for creating routers” [23]. Since each node in a MANET needs to essentially perform the same actions of a router, *Click* should be able to mimic a MANET protocol. The *Click* architecture allows for flexible and detailed node development and packet handling because each *Click* router/node is built with fine-grained components called elements.

When developing a node, different elements are selected and then connected in a directed graph. In addition, new elements can be created by the developer and integrated into the already existing elements. The creators envisioned that each element would represent a simple routing task, such as decrementing an IP packet’s time-to-live field [24]. Figure 2-3 shows a very basic *Click* configuration, depicted as a directed graph with the three elements that comprise it. In general all it does is read packets from the network (FromDevice(eth0)), counts them (Counter), and then throws them away (Discard) [24].

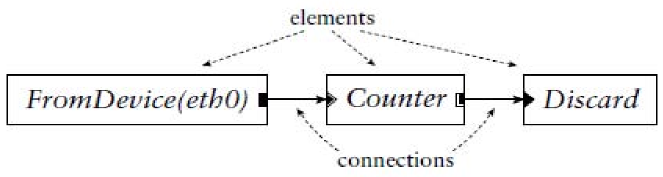


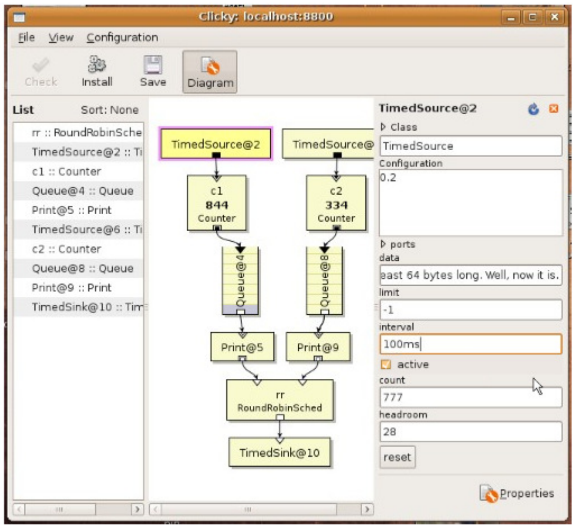
Figure 2-3. A simple Click router configuration: reads packets from the network device, counts the packets, and then discards them

Click was developed to operate as an extension to the Linux kernel, with each of the elements being a C++ object. Each connection between the elements represents paths for packets to travel, and *Click* supports two different kinds of connections, push and pull. Most software routers implement push connections since the packets start at the source element and are then passed to each corresponding element along the directed graph. Pull connections are the opposite of push since the destination element starts the packet transfer by asking the source element to send it a packet. The developers made it illegal to have a connection between a push and pull port, each connection must be made between two ports of the same type. *Click* was also developed with agnostic ports as well, which will act as a push or pull port depending on which type of port they are connected to, which thus simplifies some of the node development. The only way to allow packet flow between opposing ports is to utilize a queue element. The queue element has a push input port and a pull output port thus allowing for packets to be stored from the push connection and then later retrieved from the pull connection [24].

There are multiple elements within the *Click* distribution, all performing various functions usually as granular as possible so that the design remains modular. One other benefit is that *Click* elements can be written and integrated into configurations with existing elements thus making the code reusable.

2.3.1 Clicky simulation environment

As with most software architectures that are used to develop routers and other network objects it is useful to simulate multiple occurrences. The developers of *Click* created a simulation environment, *Clicky*, that assists in *Click* development. It was developed to graphically display the directed graphs that result from the *Click* configuration code. Figure 2-4 shows the Graphical User Interface (GUI) of *Clicky*. It can display the configuration code as a directed graph, as seen in Figure 2-4, or as the actual *Click* code. The *Click* configuration code can also be run within *Clicky* to view packet traversal through the code. *Clicky* can also read and write to handlers in live configurations. This thesis work utilized *Clicky* to generate directed graphs and assist in configuration code development but not as a simulation environment.



TimedSource@2
Class: TimedSource
Configuration: 0.2
ports
data: east 64 bytes long. Well, now it is.
limit: -1
interval: 100ms
active
count: 777
headroom: 28

Figure 2-4. Clicky's simulation environment GUI

2.3.2 MANET Protocols implemented within Click

There have been multiple research projects within the *Click* community to develop *Click* configurations for MANET protocols. The most prominent project is the "MIT Roofnet 802.11b/g mesh network" also known as the "Grid ad-hoc networking project" [36]. Other work in this area includes multiple Master's Theses (reviewed in the next sections), which develop the *Click* configuration code and then correlate it against an alternate implementation of the MANET protocol [8][6][14][22][40]. Since this thesis work utilizes DSR and DSDV, the work on those two protocols are the most meaningful, but other prior work in this area has provided benefits to the work done in this thesis.

2.3.2.1. Reactive Protocols

There are a few research projects that have been conducted to create *Click* configurations for RMPs. Bart Braem completed his Master's Thesis on implementing an Ad-hoc On-Demand Distance Vector (AODV) in *Click* [8]. His work includes the development of new *Click* elements and the integration of those elements with other existing *Click* elements into an AODV *Click* configuration. The configuration code was adapted to work within NS-2 by following Neufeld et al. [32] work. He then evaluated his implementation by comparing his AODV implementation to an open implementation of AODV, AODV-UU, which was developed at Uppsula University. His results demonstrated that his implementation of AODV within *Click* performed better when the simulations were small in scale. Once they were scaled up, AODV-UU performed better since it is an NS standard protocol, and *NSClick* added overhead to each node being processed. His thesis did contain all of his *Click* developed configuration code as well [8].

Doshi et al. created a *Click* configuration file for DSR in order to create a minimum energy aware MANET protocol that tried to limit the amount of power required to route traffic. His paper has a detailed directed graph of his implementation along with the five main packet headers that were used in his DSR implementation. His conclusion, based off of testing a *Click* DSR implementation against a minimum energy *Click* DSR implementation, was that the routing protocol's energy consumption can be reduced by a factor of three times at low speeds and seven to eleven times at high speeds. He did not include any code or example configurations within his submitted paper for future researchers to utilize [14].

Lastly another major project within *Click* to create RMPs is the Grid Ad-hoc Networking Project [36]. This project was founded by the Parallel & Distributed Operating System Group out at MIT. It also contains the Roofnet project at MIT, which is a rooftop network of around 50 nodes throughout the students' apartments in Cambridge. Figure 2-5 shows a map of the rooftop network that comprises the Roofnet. This project has also spawned other projects like the NetEquality 70-node deployment in Portland, OR [36]. They primarily utilize the in-house developed SRCR MANET protocol, but they did develop a Perl script for a DSR *Click* configuration. This script requires some node specific input from the user and is somewhat deprecated as it was written in 2003 [36].

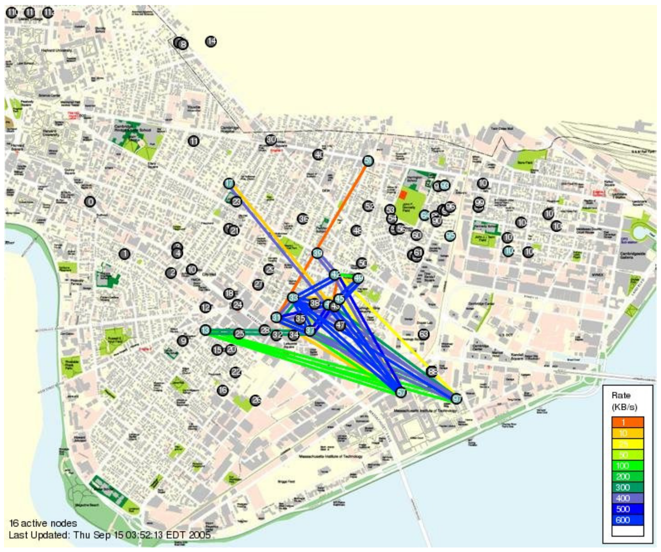


Figure 2-5. Grid/Roofnets Link Map of MIT with nodes and link data rate

2.3.2.2. Proactive Protocols

There are also some research projects that have been accomplished to create *Click* configurations for PMPs. One project was completed by Joachim Klein as a Master's Thesis to create a *Click* implementation of Optimized Link State Routing (OLSR) [22]. Klein created new elements within *Click* in order to route packets according to the OLSR specification. He validated his implementation by evaluating the computational performance of each node's ability to handle routing between the other nodes. This was performed within a test bed that was developed at his university. He identified that his implementation scaled poorly when the node count was increased, but it did still adhere to the OLSR specifications [22].

The Grid/Roofnet project also created *Click* configurations for DSDV in addition to the DSR configuration previously mentioned. They utilized it the same way as the DSR configurations and used it throughout their Roofnet nodes that were deployed across MIT's community. In addition they also created another Perl script that would generate a DSDV *Click* configuration file for the operator based on supplied node information [36].

2.4. Summary

This chapter's primary focus was on MANET protocols, and their implementations that are key components to this thesis work. The first section dealt exclusively with general information of MANET topologies and protocols. The two protocols discussed in depth due to their application in the following chapters of this work were DSR and DSDV, as an example of a reactive and proactive MANET protocol respectively. These protocols are used throughout this work to analyze physical MANETs and simulations of MANETs.

The second section reviewed network simulations and some of the current problems associated with MANET simulations. Lastly, the *Click* modular software architecture was reviewed in order to gain an understanding of how MANET protocols were developed within it. This section was also used to cover the previous work done by the MANET community within *Click*. All of the previous work discussed had some bearing in designing and developing the test bed, simulation environment, and experiments in Chapter 4 of this work.

3. Methodology

This thesis work was done to provide more reliable and realistic MANET simulation environments at the physical layer by integrating key characteristics of the MANET domain. The previous chapter discussed the MANET protocols that are used and previous MANET work that has been conducted within NS-2 and the *Click* software architecture. The work presented in this chapter is based on the analysis of Kotz et al. [26] that current MANET simulation research is overly simplified.

3.1. Problem Definition

3.1.1 Goals and Hypothesis

As stated the goal of this work is to develop a simulation environment that can accurately model a MANET protocol. The resulting simulation environment will be modeled after a real world MANET test bed to ensure it accurately follows the characteristics of the physical layer that previous simulation attempts have omitted or simplified. The MANET protocol being evaluated can be customized in both the test bed and the simulation due to the granularity of the *Click* software architecture and the NS-2 simulation engine. Since almost every characteristic of the protocol and wireless medium can be customized, a high level of granularity in the resulting simulation environment is permitted.

The test bed data is expected to offer a realistic baseline to develop the simulation environment. The parameters were changed within NS-2 and *Click* showing that MANET protocol simulations can align closely with realistic results by altering the physical layer in the simulation environment. This will then allow more realistic models to be generated within the simulation environment.

While the new simulation environment is expected to overcome the shortcomings of previous simulations, this development may add unexpected consequences to the simulation engine. These may include longer simulation processing times, or more computer system resources may be required to run the simulations. These consequences will not alter the resulting data, but may lengthen the processing of it.

3.1.2 Approach

The simulation environment developed for this thesis work models the test bed experiments without the assumed axioms, specifically 1 and 2, that were outlined in Chapter Two with the end result showing that the test bed's physical layer can be modeled within NS-2. This is accomplished by changing various parameters within NS-2 in order to alter the wireless propagation coverage, signal strength, and packet reception of the nodes. This thereby ensures that the developed system models as closely as possible to a real world implementation of a MANET protocol's physical layer.

3.1.3 Design

The overall design of the system developed for this research required both physical test bed development and simulation development. The physical test bed is required for this work to establish preliminary results, detailing specific examples of the MANET protocols and the hardware that they depend on. The nodes were built utilizing the *Click* software architecture discussed in Chapter Two of this report. The test bed was then used to generate data for preliminary scenarios to observe the real world performance of the MANET protocols and the hardware (HW) employed in these tests.

The system design is similar to the methodology employed by Kotz et al. [26], in that physical nodes were utilized to establish a MANET's physical layer capabilities. Kotz then used this information to compare and develop a simulation environment within the Simulator for Wireless Ad-hoc Networks (SWAN). The system design for this work used *Click* MANET nodes in the physical world and then NS-2 MANET nodes in the simulation environment.

Both the NS-2 and *Click* implementation of the MANET protocols adhere to the current RFCs and guidelines on how packets are handled between nodes. In the realm of this research the two are treated as equals and either one is capable of providing the routing configuration for the nodes. Figure 3-1 shows a high level diagram of how the system was designed in order to evaluate the effects of the axioms on a MANET's physical layer and not the overall routing protocol. The top layers of both environments are equivalent since they both adhere to the current RFC standards. The bottom layer in blue illustrates the area of study in regards to this work and the specified goal, which is to achieve equivalency between the two environments at this level.

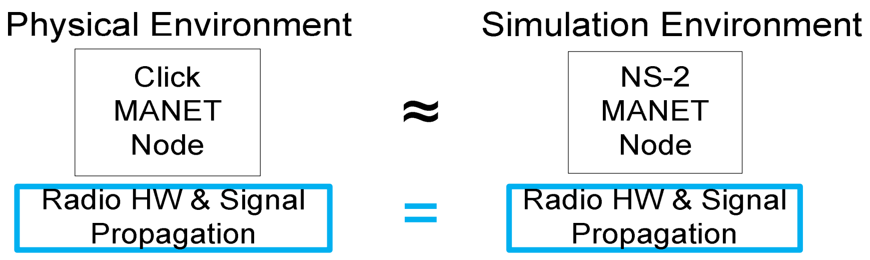


Figure 3-1. Design of Simulation Environment equated to Physical Environment

3.1.4 Services

The main service this simulation environment provides is the routing of traffic between nodes using a MANET protocol. The system is given packets originating from nodes that are part of the system. The secondary service, which will be evaluated throughout this research effort, that the system provides is the ability to route these packets by adhering to a more realistic environment. This is done by routing the traffic to the destination, while avoiding the assumption(s) made by the axioms discussed previously. The system will demonstrate the ability to model the real world by delivering the traffic in correlation to the physical test bed.

3.1.5 Boundaries

As previously stated the system was developed to use NS-2's simulation engine to route MANET traffic and to also route traffic through the use of the *Click* nodes in the physical world. The simulation environment is the System Under Test (SUT). The nodes are the components of the system and their ability to send traffic while adhering to the MANET protocols is the Component Under Test (CUT). Once the system demonstrates it can effectively model a MANET protocol's physical layer, other MANET protocol simulations can be developed in the same fashion.

The system requires inputs, i.e. Workload Parameters, to test its ability to send traffic between the nodes. The System Parameters are the characteristics of the system that if changed affect the metrics and output of the system [4]. The metrics observed within the system are used to compare it to the MANET test bed and also to a basic NS-2 MANET implementation that employs the assumed axioms. Finally, the output of the system reports on the traffic delivery between the nodes in relation to the system parameter being evaluated. The diagram below, Figure 3-2, illustrates the NS-2 MANET Simulation Environment. The details of each area within this diagram are discussed within the next few sections of this chapter.

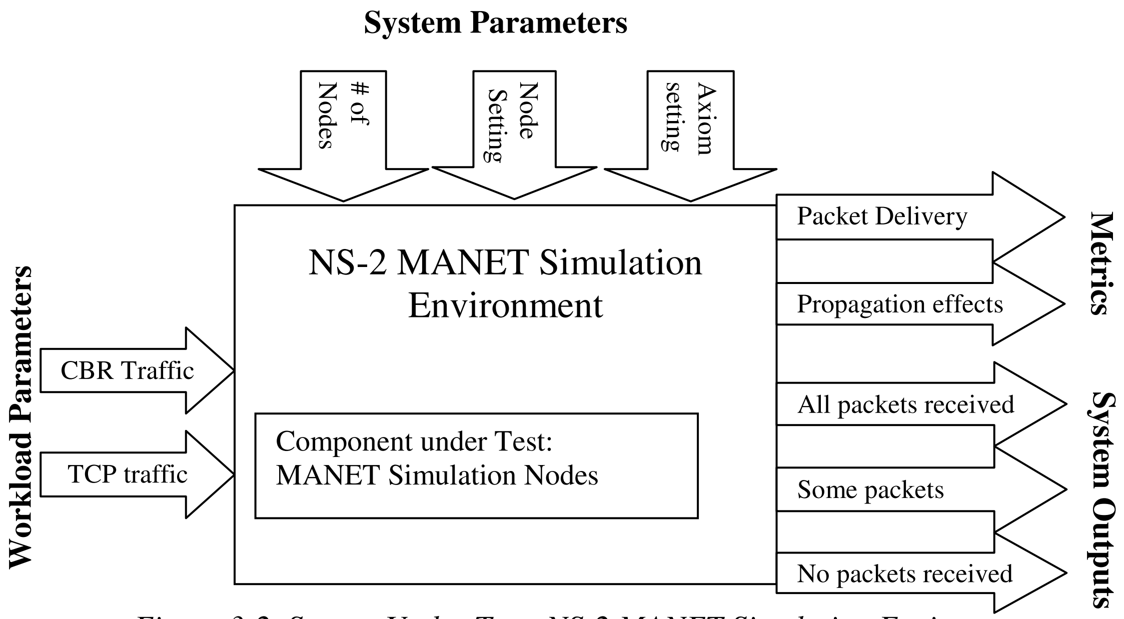


Figure 3-2. System Under Test: NS-2 MANET Simulation Environment

3.1.6 Workload Parameters

The system has two different workloads submitted to it; Transmission Control Protocol (TCP) traffic and Constant Bit Rate (CBR) Traffic. They are both used in many wireless experiments within NS-2 [19]. Each one offers benefits to these tests and both types of workloads are examined due to the different types of traffic that may be present on a MANET. The traffic will never leave the system and depending on the different axiom being evaluated will have few hops between nodes. This means both a packet's source and destination node will be a component of the system. The traffic will be delivered based off the settings entered into the NS-2 scenario files in order to accurately test the delivery between each node. The viable routes between nodes are known prior to submitting the traffic to the simulation environment so that the system tests the delivery of packets and not the ability to discover routes. The ability of the MANET protocol to discover routes does affect the packet delivery and is not measured since this work is interested in the system's ability to model the physical layer.

CBR traffic is used as a workload parameter to reduce the complexity of the system and to limit any variation that may affect the system output inadvertently. Since CBR allows the operator to specify how many packets should be sent per a specified time period, each node has an equal opportunity of receiving the exact same amount of packets. This is very useful when determining the effects that a different propagation model has on each node and their location in the simulation. This makes the analysis of the packet delivery metric straightforward, as discussed later in this chapter.

TCP traffic was selected for the tests as a workload parameter because it offers a more realistic traffic pattern since most network traffic is either UDP or TCP and arrives to nodes at random rates. TCP traffic within NS-2 will also require the destination node to acknowledge receipt of a packet since TCP is a connection-oriented protocol. This workload parameter allows the system to show its ability to handle a larger volume of traffic at varying rates than with the CBR traffic.

3.1.7 System Parameters

The parameters that can affect system performance are defined as the number of nodes in the system, the node settings, and also the axiom(s) that are being evaluated. These parameters can all be changed within the NS-2 scenario file and do affect the output of the system.

The number of nodes in the system affects the ability to route traffic from source to destination. One example is a sparsely populated simulation where the node's coverage areas do not overlap, which means the transmission of traffic between nodes is limited. On the other hand if there are a large number of nodes in the simulation, congestion may exist on the network since the wireless medium will be shared between them. The tests that will be conducted in this report will only require a few nodes in order to affirm or disprove the axioms discussed in the previous chapter. This parameter has a minimal amount of impact on the system considering the tests that are conducted for this thesis work.

The node settings are configured through NS-2 to mimic the hardware utilized in the physical test bed, Proxim Orinoco 802.11 a/b/g cards. These settings are always applied equally to every node in the system to ensure there are no discrepancies between node settings. They include wireless hardware characteristics such as frequency, data rate, antenna model, queue length and type, channel type, Media Access Control (MAC) type, and others that can be specified for each scenario that is run in the simulation.

Lastly the axioms that most MANET simulations are comprised of will be used to influence the system. These axioms will affect how nodes route traffic when they are being evaluated. One example would be if the propagation of the nodes wireless signal was changed from a perfect circle to another shape that resembles the real world.

The system is affected differently when these parameters are altered or modified for different experiments. The system may be sensitive to slight changes to these settings, which in effect alters the system outputs. These parameters must be applied systematically to ensure accurate results are obtained.

3.1.8 Performance Metrics

The performance metrics cover two areas that are used to model the simulation environment to the test bed. These areas are also used to compare the realistic simulation results to the simulation models that utilize the assumed axioms. The two areas are: packet delivery and propagation effects. These metrics can be measured effectively through NS-2's trace files and NAM files. Since each test assumes static positioning of the nodes, the “best” route a packet should take can be configured prior to the injection of the workload and thus its performance can be accurately measured.

Packet delivery is measured by the ratio of packets that are sent from the originating node to those that are received by the destination node. CBR traffic allows this metric to be standardized at each node with a known maximum amount of packets received. TCP traffic is not standardized between each node but instead will continually transmit packets as long as there is a viable link to the node it is currently configured to send to.

Propagation effects are observed visually through the use of the Institute of Telematics techniques that alter propagation models within NS-2 and generate the resulting model in LaTeX [20]. These models can be adjusted according to the physical test bed results so that the radio propagation for each node in NS-2 mimics the physical nodes actual propagation pattern.

3.1.9 Factors

Factors are the parameters of the system that will be changed deliberately within the experiments in order to garner differing results. The parameters of the system that were selected to be factors are shown in Table 3-1:

Table 3-1. Factors that influence the system and its outputs

FACTOR	LEVELS
Traffic	TCP and CBR
Propagation type	Two Ray Ground and Shadowing
Propagation Pattern	Basic (small and large) and APProxy

These factors were chosen because of their ability to illustrate the simulation environments success at modeling the MANET test bed’s physical layer. They allow for distinct differences to be seen between the realistic simulation and the simulations that adhere to the assumed axioms. These factors will be changed individually in each experiment to reduce the variability of the tests.

CBR and TCP traffic were selected as factors because they offer two different packet types that will be sent within the simulation environment. TCP traffic will be sent at a varying rate and will utilize as much of the link as possible. Whereas CBR traffic is sent at a set rate that is specified in the Tcl script to be at a maximum of 200 packets a second. If the models are similar then the two different types of traffic will produce results that map closely to each other. These traffic types are used in other NS-2 MANET simulations and provide basic traffic analysis [19].

The propagation type was selected to be a factor because in NS-2 this configuration is what governs how the wireless signal will radiate out from each node. The propagation type is configured to be the same for each node and is not changed during run time of the simulation. The different types of propagation models that are used in the experiments are NS-2's default circular propagations, Two Ray Ground and Shadowing, and the real world observed propagation model implemented through the Asymmetric Propagation Proxy (APProxy). The first two are included modules within NS-2's all-in-one installation files while APProxy requires additional modules to be loaded in order to implement it.

Within each of these propagation models different variables can be set in order to affect their coverage range. The Two Ray Ground model assumes an ideal propagation condition between the nodes. It is primarily useful in short distances between nodes, but it does represent the problems experienced with axiom 1 and 2. The NS-2 manual states, "Two Ray Ground model basically represents the communication range as a circle around the transmitter. If a receiver is within the circle, it receives all packets. Otherwise, it loses all packets [15].” The Shadowing model on the other hand still represents the propagation pattern as a circle around the node, but it tries to account for fading effects by allowing the signal to gradually lower the received power in relation to the distance between the nodes. On the other hand, the APProxy model can apply its own attributes to the nodes in combination with the Two Ray Ground and Shadowing models. “NS-2 only supports symmetric propagation models, which means the link quality between two nodes is always equal [20].” This is why the Institute of Telematics developed the APProxy model. Figure 3-3 shows the algorithm that is used to create an APProxy model. The first step is for the proxy to calculate the angle between the transmitter and receiver. This angle is then linearly mapped to a gain value, which is then used to modify the transmission power of the node. Once this has been accomplished the other propagation models, i.e. Two Ray Ground and Shadowing, will apply their attributes to the nodes.

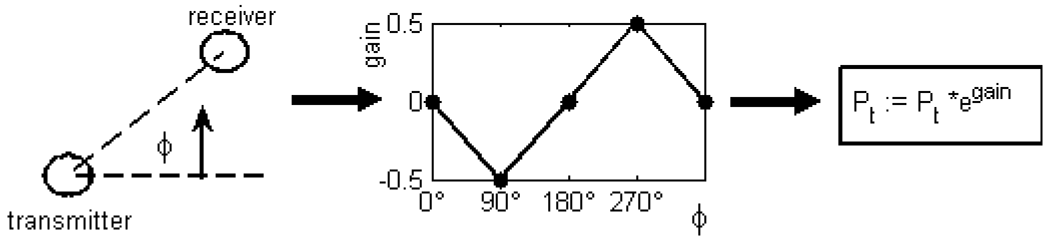


Figure 3-3. APProxy algorithm: 1) angle calculation 2) angle mapped to gain value 3) transmission power is modified by gain value [20]

3.2. Summary

Adhering to the guidelines outlined in this chapter ensures that the experiments are conducted without bias to the possible outcomes. The overarching goal is to develop a MANET simulation environment that is able to model key characteristics of a physical MANET test bed. The simulation environment is expected to show, through analysis of the data, that the axioms in current MANET simulations need to be addressed accordingly. Otherwise the simulation will over simplify the MANET protocol and leave out key aspects of its functionality.

The resulting metrics from the tests will be used as the comparison data against the data retrieved from the physical test bed. This comparison will show if the developed system is an effective model or if it still lacks the precision to model the MANET protocol. It will also be compared against two basic NS-2 implementations that adhere to the assumed axioms. Chapter Four will introduce the experimental design and techniques used to gather the data. It will conclude with analysis of the results and provide some conclusions that can be drawn from them.

4. Results and Analysis

This chapter is devoted to the results of the physical test bed experiments and the simulation experiments. The results are discussed and analyzed to determine if an NS-2 simulation environment can mimic the physical layer of a real world MANET implementation. Since the overall hypothesis of this work is to show the ability of simulations to model the real world environment, the procedures taken to accomplish this are discussed in detail. Lastly the performance of these simulations that mimic the test bed results will be compared to the simulations that adhere to the axioms.

4.1. Evaluation Technique

The evaluation technique used is a combination of the measurement and simulation techniques. The measurement technique gathers results from the physical MANET test bed and the simulation environments, while the simulation technique is used for the MANET NS-2 simulation environment. Both the physical test bed and simulation environment utilized MANET protocols in order to transmit traffic between the nodes.

The environments that MANET protocols operate in are very diverse, which is why *Click* and NS-2 were used to provide the needed flexibility to alter nodes and test the characteristics of the protocol. It is expected that the physical test bed will show the assumed axioms are in fact important factors to account for in a MANET simulation.

4.2. Experimental Design

The simulation environment is designed to conduct experiments based off the factors specified in Chapter Three. This means it requires 12 experiments: 2 (Traffic type) * 2 (propagation model) * 3 (propagation patterns) = 2 * 2 * 3 = 12 experiments.

These experiments allow for evaluation of different scenarios associated with the three factors.

Figure 3-1 showed the experimental plan for this thesis work, which was to use the *Click* configurations in the physical experiments and then use the NS-2 configurations for the simulation experiments. The system in its current state is able to conduct experiments in the physical and simulation domains. The NS-2 implementation does generate the needed results to evaluate the assumed axioms. Figure 4-1 shows the overall system design as seen in the previous chapter. Once again the focus of this thesis is the ability to model the physical node's radio HW and signal propagation as noted in the blue boxes.

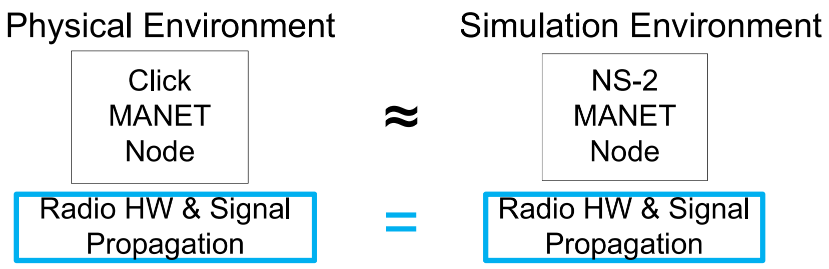


Figure 4-1. Simulation environment design

4.3. Experiment Procedures

This section covers the procedures that were completed in order to build both the physical test bed and the simulation environment. Not every step or technique is covered due to the numerous coding and configuration changes that were experimented with before arriving at the current solution. This section summarizes the problems that were faced in developing these two platforms.

4.3.1 Physical

The first step that needed to be accomplished to evaluate the axioms and determine how to effectively model the real world in simulation was to build the physical test bed. Three Dell Latitude laptops were configured to be the wireless nodes used in the test bed. Two of the laptops had wireless Network Interface Cards (NIC) built into the case. While these cards were sufficient to conduct the experiments, it was decided to reduce the variability and use the exact same model card in each node. The card selected was a Proxim 8480-WD Orinoco 802.11a/b/g ComboCard. The benefit to using these cards was that their drivers have been implemented in most Linux operating systems since they are popular cards with developers and researchers. Another benefit of these cards is that their NS-2 configuration parameters have already been identified [43].

4.3.1.1. Click Installation

The nodes then required the *Click* software architecture to be loaded on them so that the MANET protocol could be implemented. Since *Click* was developed to operate on a *nix platform the nodes were loaded with Ubuntu 9.10 as their operating system. Ubuntu was chosen because it allows for plug and play hardware setups such as the wireless cards that were being used in these nodes. In fact it only required a software update in order to get the cards functioning properly. The update installed the MadWiFi driver since it supported the cards chipset. *Click* 1.7.0rc1 was then installed on the nodes with the options `–disable-linuxmodule` and `–enable-all-elements`. These options installed *Click* to operate only in userlevel and with the ability to implement any element that was packaged with the installation files. Userlevel was selected because kernel level *Click* nodes require a patched generic kernel to be used instead of a basic installation of Ubuntu.

After reviewing the files and past research efforts with *Click* and MANET implementations, it was discovered that the Grid/Roofnet Project from MIT had created two Perl scripts that would create the *Click* configuration code for DSR or DSDV depending on the information supplied when the script was ran. These files are within the

Click program files at `~Clickinstalldir/conf` and are called make-dsr-config.pl and make-dsdv-config.pl. These Perl scripts were run with the following command on each of the nodes (node 1 is used as the example) [36]:

```
./make-dsr-config.pl -i wlan1 -a 10.0.0.1 -u > dsr1.click
```

This command specified the interface to pull the MAC address from, the address for the node, and to run it at userlevel instead of at kernel level. The next step was to setup the wireless interface to operate in ad-hoc mode with the correct settings. Each node had the following code entered in to the command line as root [36].

```
ifconfig wlan0 down   (Built-in wireless card)
ifconfig wlan1 down   (Must be down to configure)
iwconfig wlan1 mode ad-hoc essid click channel 1
iwconfig wlan1 txpower 1
ifconfig wlan1 up
ifconfig wlan1 10.0.0.1
```

The *Click* configuration code was then ran on each of the nodes to allow the nodes to connect to each other. This was done by executing the following command: `click ~/clickinstalldir/conf/dsr1.click`. The code was developed in 2003 so when it was ran there were some errors and warnings due to deprecation of the code since this work was using a newer version of *Click*. These issues were fixed with the code and the elements that were not utilized or needed for this work were removed. After the code was altered and updated, it was run under the same command and the nodes were able to establish connectivity to each other. This was accomplished by sending simple Internet Control Message Protocol (ICMP or "ping") packets between the nodes. The directed graphs in Appendix B: and Appendix C: detail the basic elements that make up the *Click* nodes for both MANET protocols. These directed graphs detail the flow of packets once they reach each node and are passed from the wireless interface to the *Click* code for processing and then passed back to the interface for delivery per the processing conducted by *Click*.

4.3.1.2. Physical Test Bed Data Gathering Experiments

The nodes were fully operational and could correctly send and receive packets between the three of them. The next step was to take them out into the field to gather physical data points that would later be used to construct the simulation environment in this thesis. These tests used *Click* nodes as described above and were conducted with both the DSR and DSDV protocol with the results from both protocols being similar.

4.3.1.2.1 Open Field Test Bed Experiment

The first test environment was an open field that consisted of no obstructions such as buildings or trees. A satellite image of the test area is shown in Figure 4-2. The red circle in the image and the computer node in the middle of it were inserted to detail where the test took place within the image. The test was setup with a center node (Node one) that was stationary, with the other node (Node two) carried away from the center node until the signal was lost between them. Node two was then carried back towards node one until the connection was restored. Node two was then carried way from the center node a second time to establish an accurate measurement. Once node two lost the connection a flag was placed in the ground marking the spot. This was repeated for eighteen more spots surrounding node one. Figure 4-2 details the distance and shape of the coverage area of the wireless signal based on this very specific time and environment test. The figure shows the center node with the wireless card on the right hand side if a user is looking down on the node. The goal of this test was to determine the propagation range of the wireless nodes and the overall shape that they project. Each flag was measured from the central node, node one, along with the measurements between the 19 spots located around the propagation area. This allowed for the test results to be effectively graphed within Microsoft Visio, showing the propagation coverage to be more of an egg shape rather than a perfect circle as indicated by axiom one and two. This reflects the physical reality of the antenna gain pattern of the NIC's transceiver.

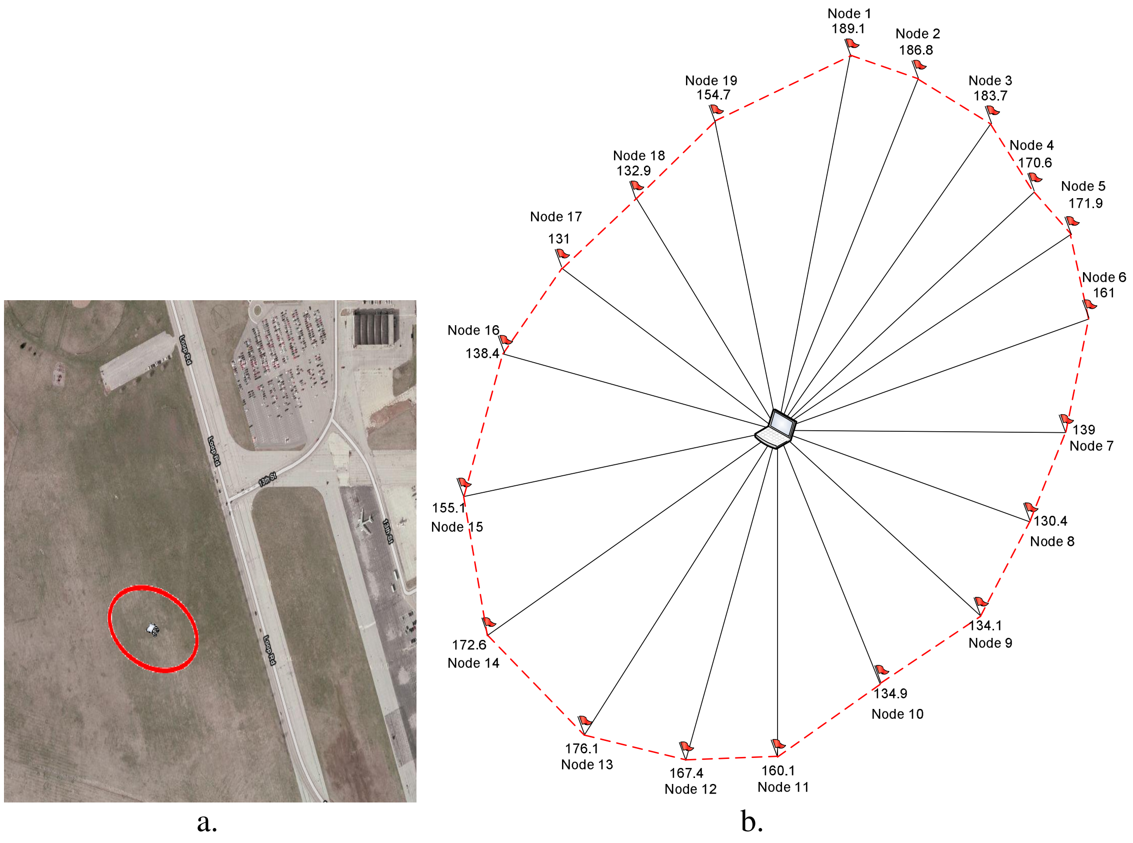
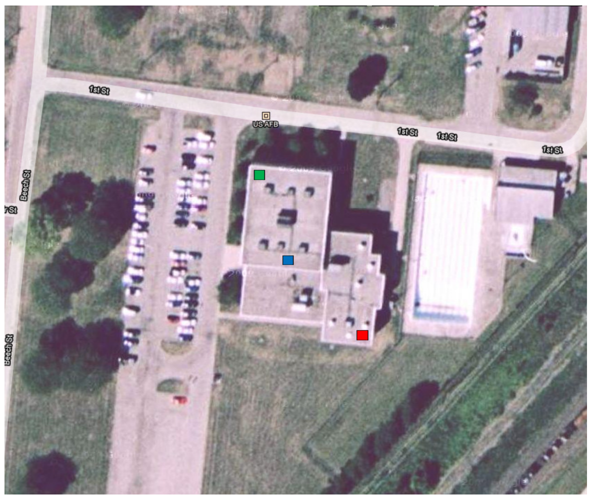


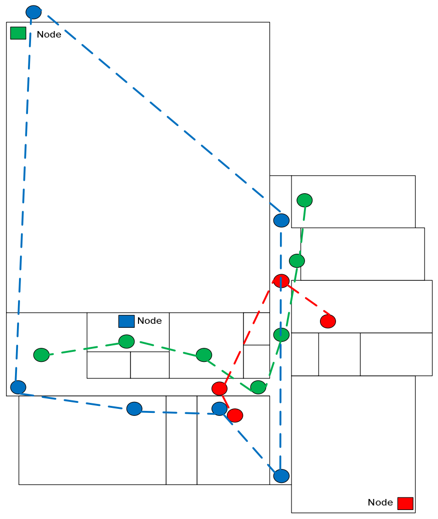
Figure 4-2. a. Satellite image of outdoor experiment location (©2010 DigitalGlobe) and b. Open Field Experiment (measurements in feet)

4.3.1.2.2 Indoor Obstruction Test Bed Experiment

The next test that was conducted with the physical test bed was an indoor experiment to determine the propagation effects experienced when obstructions exist between the two nodes. The location selected was a vacant gymnasium, Jarvis Gym, on Wright Patterson Air Force Base, Ohio. This building was ideal because there were no preexisting wireless radios in the building, and it was comprised of cinder block walls and steel doors. Its layout is also desirable because it contained a large open space along with long hallways connecting multiple enclosed rooms. This layout allows the wireless signal to be altered considerably by the environment it is operating in. The tests were conducted with only two nodes with one node remaining stationary while the other node moved throughout the building attempting to maintain the connection to the stationary node. The experiment was conducted with the stationary/central node in three different locations throughout the building. Figure 4-3 shows the different locations on a satellite image of Jarvis Gym as colored boxes (Green, Blue, and Red).

The experiments were conducted following the same steps as with the Open Field Experiment described previously. The central node remained stationary and the other node moved around the central node to discover the range of the signal. As stated previously the stationary node was placed in three different locations within the building so that the experiment could be run against the varying locations and obstructions that the building offered. Figure 4-3 shows the stationary node's placement as colored boxes within the floor plan of Jarvis Gym.





a. b.

Figure 4-3. a. Satellite image of Indoor experiment location (Jarvis Gym) (© GeoEye) and b. Indoor Obstruction Experiment results

The colored circles indicate the mobile nodes position when they were last able to communicate and pass traffic to the stationary node. Inherently, the colors correspond to each other where the red circles are the mobile node’s position when communicating with the stationary node located at the red square. The signal did permeate outside of the building but the experiment did not test for signal propagation outside of the building. An early summary of the reception ability of the mobile node indicates that obstructions greatly affect the ability of the signal to remain circular (axiom 1) or the ability to assume the propagation is a simple distance to signal degradation ratio (axiom 5).

4.3.2 Simulation

4.3.2.1. Propagation Simulations

The first step in developing the simulation environment to model the physical test bed results required the ability to alter the signal propagation within NS-2. NS-2 allows for three different propagation models that can be configured per a developer’s specifications. These include Free Space, Two Ray Ground, and Shadowing. The first two models are deterministic models, meaning if a receiving node is within the propagation area of the sending node then every packet will be delivered [20]. Shadowing allows for more realistic signal propagation since the probability of nodes being able to transmit packets decreases proportionally as the nodes are distanced from each other [15].

However, these three models still do not allow for any other propagation pattern other than circular. The need for a developer specified propagation pattern lead to the discovery and implementation of a third party NS-2 module developed by the Institute of Telematics, Hamburg, Germany. This module was discussed previously in Chapter Three as the Asymmetric Propagation Proxy (APProxy). This module was added to the NS-2 simulation environment by adding new agent code, NBhAgent.cc/.h, and a new packet, Mypacket.cc/.h, in addition to the APProxy code, AsymmetricPropagationProxy.cc/.h [20]. NBhAgent.cc, NBhAgent.h, MyPacket.cc, and MyPacket.h code files were all placed in the *NS-2 installation location*/ns2.34/common/ folder. The file packet.h had to be edited so that the new packet format, Mypacket.cc, would be recognized by NS-2. The code: `static packet_t PT_MYHEADER = 63;` and `name_[PT_MYHEADER] = "myheader";` were added to their respective sections in the file. The NS-2 makefile.in was then changed within the OBJ_C section to include the following three lines of code:

```
common/NBhAgent.o \
common/Mypacket.o \
mobile/AsymmetricPropagationProxy.o \
```

NS-2 was then reconfigured and make install was run to include the new objects within NS-2. Finally LaTeX had to be installed on the system so that visual representations of the propagations could be created. Once it was installed two Tcl scripts, initialize.tcl and writeoutput.tcl, were added to *NS-2 installation location*/ns2.34 folder. After these steps were complete NS-2 could now generate propagation models as defined within the Tcl script ran in conjunction with initialize.tcl. The commands used to generate these models are: `ns simulate.tcl shadowing.tcl` and `pdflatex document.tex.` The resulting diagram of these Tcl scripts is shown in Figure 4-4.

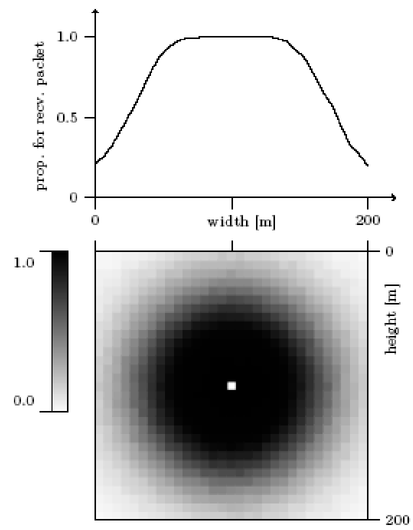


Figure 4-4. Diagram of shadowing.tcl script – top graph is the probability of receiving a packet and lower graph is a 2D area plot of the propagation pattern [20]

4.3.2.2. Traffic Simulations

Now that the signal propagation could be altered to model the physical test bed's propagation pattern, NS-2 was configured to send traffic utilizing the circular propagation models, Two Ray Ground and Shadowing, and the APProxy propagation model. The goal of these simulations was to compare the commonly used NS-2 propagation models, Two Ray Ground and Shadowing, to the modified APProxy propagation model. Appendix D: has all of the code used to create the traffic simulations described in this section. The two main Tcl scripts used for the traffic simulations were TCPegg.tcl and CBRegg.tcl. These were both created by using the University of California's simple-wireless.tcl script as a shell that was modified to suit the needs of this research.

The first modification of the script was to place the nodes within the topography of the simulation. There were two options that could have been used to simulate the Open Field Test Bed (OFTB). They both would have node 0 be stationary in the center of the topography. The difference between the possible paths was to either have one node moving around node 0 to each spot that was measured in the OFTB or to have multiple stationary nodes at each spot measured in the OFTB. The first scenario represented the experiment as it was conducted, but did not present an easily identifiable test area in NAM because as the node traveled around the central node the previous location will no longer be evident. It was determined that the second approach would allow for easier result collection. All nodes remain stationary throughout the simulation and the central node sends traffic to each node separately for a set period of time, one second. Figure 4-5 shows TCPegg.nam, which illustrates the simulation topography with the twenty nodes placed within it. This setup provided another benefit because bugs or errors in the script that affect traffic flow could be identified easily by viewing the NAM trace and seeing which node was experiencing the problems.

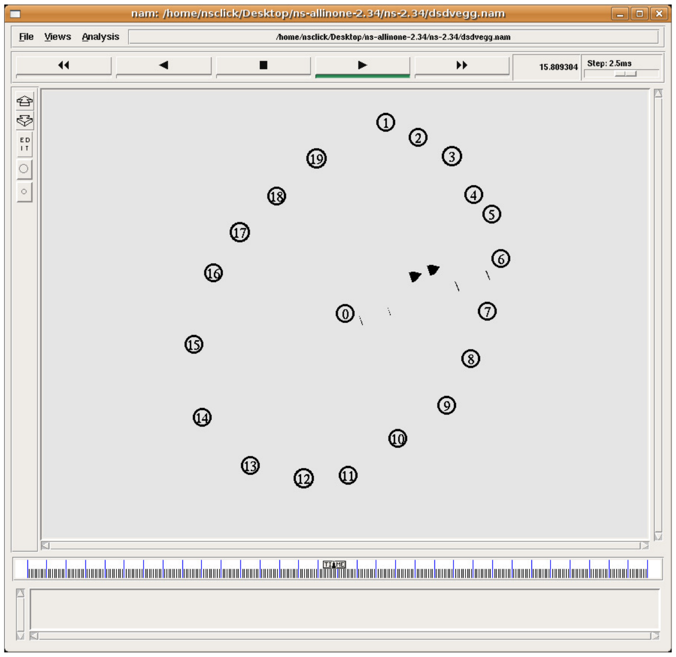

nam: /home/nsclick/Desktop/ns-allinone-2.34/ns-2.34/dsdvegg.nam
File
Views
Analysis
/home/nsclick/Desktop/ns-allinone-2.34/ns-2.34/dsdvegg.nam
15.809304
Step: 2.5ms
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19

Figure 4-5. TCPegg.nam – TCP traffic is being sent to all nodes one at time

The central node, node 0, was then configured to send traffic, either TCP or CBR depending on the test being conducted, to the nodes surrounding it. Node 0 sent traffic to each node in a clockwise rotation for exactly one second of simulation time. The CBR traffic simulations were set to 200 packets a second, while the TCP traffic simulation instead was set to send traffic continuously and with no set packet rate.

Finally the last main step to conduct the simulations was to integrate the APProxy propagation model into the traffic TCL scripts. This was done by copying the code from the APProxy.tcl script that was used to generate the LaTeX diagrams as described in the previous section of this chapter. The APProxy propagation model was applied to the nodes so that their propagation signal pattern modeled closely to the OFTB.

4.4. Analysis

This section is dedicated to the analysis of the various experiments that were conducted in order to identify if the realistic APProxy simulations do offer benefits over the Basic simulations. It also details how closely aligned the realistic NS-2 simulations are to the physical test bed. It is broken into two areas the comparison of the propagation simulation efforts and the traffic simulations.

4.4.1 Propagation Simulations

The ability to create a visual diagram of the signal propagation for the different models used allows for comparison between them. Each diagram was generated using the parameters listed in Table 4-1. The parameters do remain constant throughout the different models.

Table 4-1. Propagation parameters

NS-2 Parameters	Basic (small)	Basic (large)	APProxy
Pt: (transmit power)	0.19	0.3018	0.2018
Gt: (transmit antenna gain)	1	1	1
Gr: (receive antenna gain)	1	1	1
Freq: (channel)	2.412e9	2.412e9	2.412e9
L: (system loss factor)	1	1	1
CPThresh: (collision threshold)	10	10	10
CSThresh: (carrier sense power)	5.011872e-9	5.011872e-9	5.011872e-9
RXThresh: (receive threshold)	5.82587e-09	5.82587e-09	5.82587e-09

These parameters yielded the diagram as seen in Figure 4-6 of the propagation pattern for the Basic (small) pattern utilizing the Two Ray Ground model. The top part of the diagram shows the probability that a packet will be received by the end node. Since the Two Ray Ground model is a deterministic model every packet is shown to be received with a probability of 1 or 100% of the time as long as the receiver falls within the coverage area. The bottom portion of the diagram shows the propagation pattern as a symmetric circle around the sending node, extending out approximately 50 meters. The bottom diagram displays the probability of packet reception by varying the level of black shading of the propagation pattern, but since the probability is always 1 for Two Ray Ground; the propagation pattern is completely black.

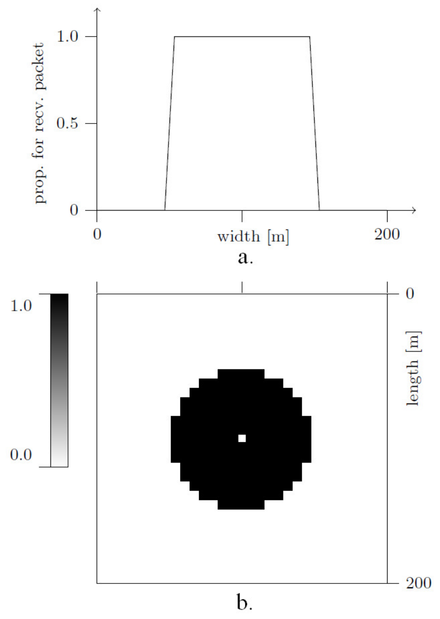


Figure 4-6. Two Ray ground circular shaped propagation – a. graph of packet reception probability b. shaded graph denoting propagation coverage area per node

The next diagram, Figure 4-7, was created by utilizing the Shadowing propagation model within the NS-2 simulation environment that had been created. This diagram is similar to the one generated by the Two Ray Ground model in that it represents a circular coverage area and covers roughly 50 meters. The difference is that the Shadowing model now accounts for packet loss due to increased distance from the sending node. The top graph in the diagram no longer has the dramatic drop-off for packet reception. Also the bottom graph has a varying level of black shading to denote the probability of a packet's reception within the coverage area.

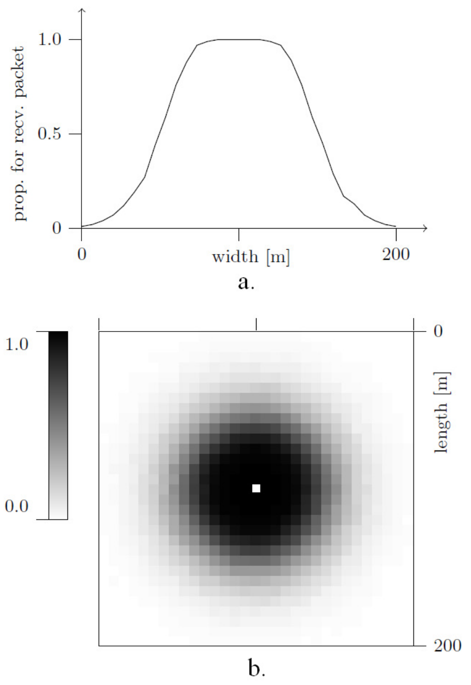


Figure 4-7. Shadowing circular shaped propagation – a. graph of packet reception probability b. shaded graph denoting propagation coverage area per node

The last two diagrams are based off the APProxy propagation module that was configured to represent the OFTB. The second, Figure 4-9, utilized the Shadowing propagation parameters so that it would be more realistic in that the packet reception probability does decrease due to distance. While the other pattern, Figure 4-8, utilized the Two Ray Ground propagation parameters so that it could be compared to the circular Two Ray Ground propagation pattern produced previously. The difference between the previous models is that the transmission power was changed according to the angle between the sender and receiver to mimic the OFTB propagation coverage area. Figure

4-8 and Figure 4-9 show the resulting diagrams where the coverage is approximately 50 meters and the pattern resembles an egg as was observed in the OFTB.

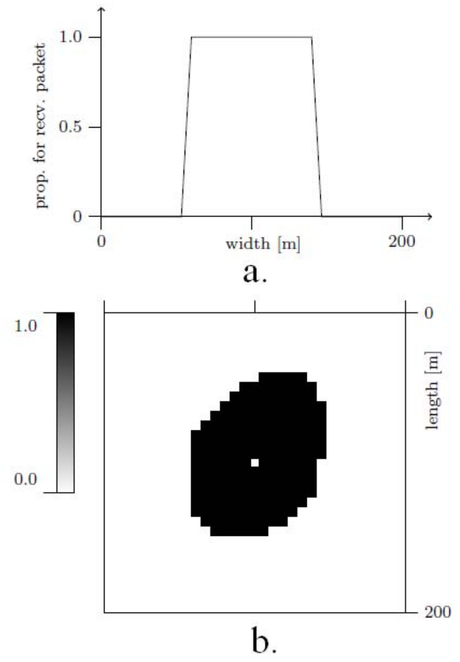


Figure 4-8. APProxy propagation based on two ray ground – a. graph of packet reception probability b. shaded graph denoting propagation coverage area per node

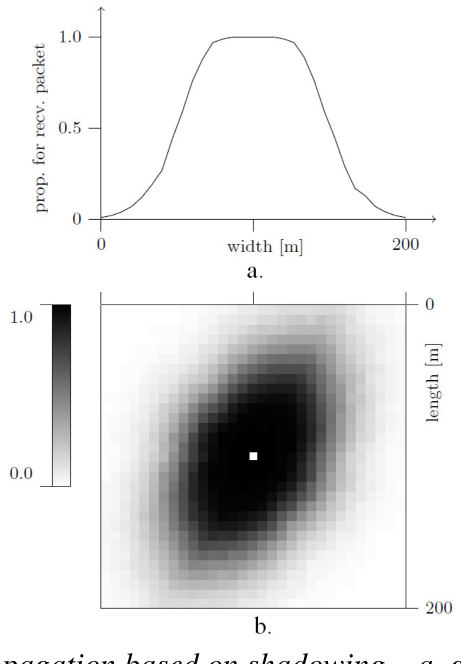


Figure 4-9. APProxy propagation based on shadowing – a. graph of packet reception probability b. shaded graph denoting propagation coverage area per node

Overlaying the circular Two Ray Ground graph on top of the APProxy Two Ray Ground graph provides for a visual analysis of the different coverage areas. This visual analysis, Figure 4-10, shows that the basic NS-2 propagation models should not be able to deliver packets to all the nodes that could receive packets within the OFTB of this thesis work. The red shaded area denotes the NS-2 Basic (small) Two Ray Ground propagation coverage and the blue shaded areas denote the APProxy propagation coverage. If the circular graph had its transmit power increased to ensure that all the nodes in the simulated OFTB would receive packets then the coverage area of the NS-2 Basic propagation would extend further out than seen in the physical world as seen in Figure 4-11.

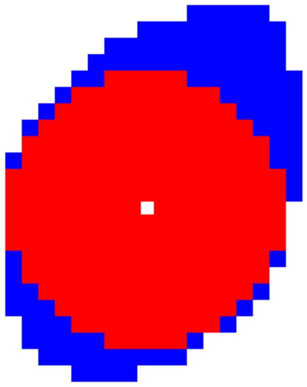


Figure 4-10. Two ray ground propagation model with the NS-2 Basic circular pattern (red) overlaid onto APProxy egg pattern (blue)

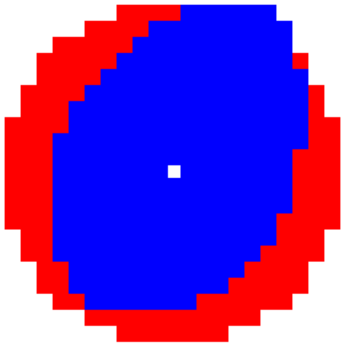


Figure 4-11. Two Ray Ground propagation model with APProxy egg pattern (blue) overlaid onto the NS-2 Basic circular pattern (red)

4.4.2 Traffic Simulations

This section of analysis consists of the comparison of packet delivery between the basic NS-2 propagation model and the APProxy propagation model that has been modified to adhere to the OFTB of this research work. The comparisons were made against the trace files that were generated when each propagation model was ran in NS-2 with CBR and TCP traffic being sent between the nodes. The basic NS-2 propagation model was set to be the Two Ray Ground model and the Shadowing model. These simulations resemble the propagation as seen in Figure 4-10 from the previous section, where the circular NS-2 default propagation resides within the modified egg shaped APProxy propagation pattern.

4.4.2.1. TCP Traffic Analysis

TCP traffic simulations were produced and compared using both propagation patterns, circular and egg-shaped. The egg-shaped model obviously had the modified APProxy model running on top of either the Two Ray Ground or Shadowing model in order to alter the pattern while the circular pattern was the default Basic NS-2 pattern. The first comparison utilized the Two Ray Ground model as the base propagation model for both the Basic (circular) and APProxy (egg-shaped) patterns. The second comparison utilized the Shadowing model as the base propagation model for both patterns. As stated previously in the procedures section, the center node, node 0, sent TCP packets to each node individually at one second increments. The packets that were received by the destination node were the ones of interest. The NS-2 trace file was filtered to include only the resulting ACK packet from the destination node. The lines in the trace file that were of interest to this work are shown in Figure 4-12. The first line shows the sending packet from node 0 (3rd column) as a TCP packet. The next line shows node 1 (3rd column) receiving the TCP packet. These lines are produced by the agent within the NS-2 Tcl script so that these comparisons can be made.

```
s 10.858952766 _0_ AGT  --- 442 tcp 1040 [0 0 0 0] ------- [0:0 1:0 32 0] [215 0] 0 0
r 10.861060985 _1_ AGT  --- 421 tcp 1060 [13a 1 0 800] ------- [0:0 1:0 32 1] [207 0] 1 0
```

Figure 4-12. Lines of interest from an NS-2 Trace file

4.4.2.1.1 TCP Basic (small) and APProxy simulation experiments

The resulting table, Table 4-2, was created and populated with the totals for each node in regards to transmitted packets from the central node to each of the external nodes individually. There were nodes that experienced a distinct difference when receiving packets through the circular pattern as compared to the egg pattern. Table 4-2 shows that for both the Two Ray Ground and Shadowing models that the circular model cannot reach all of the nodes that the egg shaped model can. This result was expected since the circular model's propagation does not cover the outlying nodes, approximately 180 feet or more further out. An unexpected finding was that nodes 12-14 did not experience packet loss in regards to the Two Ray Ground model even though the overlaid graph, Figure 4-10, of the models indicates that these nodes may not have good reception.

The next comparison utilized TCP packet delivery in the same manner by counting the total number of packets sent from the source and the total number of packets received by the destination. The only difference is that the egg-shaped and circular propagation patterns were modeled using the Shadowing NS-2 propagation as the base model instead of Two Ray Ground. The last two columns were created in Table 4-2 with this data and once again the nodes in the circular pattern experienced lower packet delivery rates than the egg pattern.

The Shadowing model did affect every node's ability to receive every packet for both propagation patterns. The Shadowing model negatively affected the circular pattern more than the egg shaped since the circular packet delivery rate was lower than the egg shaped pattern.

Table 4-2. TCP Packet Delivery Comparison between APProxy and Basic (small) propagation patterns using either Two Ray

TCP Packet Delivery Results														
Node Info		Basic (small) pattern (Circular)						APProxy pattern (Egg)						
ID	Distance (ft)	Two Ray			Shadowing			Two Ray			Shadowing			
		Tx	Rx	PD%	Tx	Rx	PD%	Tx	Rx	PD%	Tx	Rx	PD%	
1	189.1	1	0	**0**	1	0	**0**	250	250	**100**	1	0	**0**	
2	186.8	1	0	**0**	5	3	**60**	251	251	**100**	9	8	**89**	
3	183.7	1	0	**0**	2	0	**0**	253	253	**100**	12	10	**83**	
4	170.6	244	244	**100**	1	0	**0**	249	249	**100**	2	0	**0**	
5	171.9	252	252	**100**	1	0	**0**	254	254	**100**	2	0	**0**	
6	161	255	255	**100**	1	0	**0**	249	249	**100**	1	0	**0**	
7	139	258	258	**100**	99	99	**100**	249	249	**100**	119	118	**99**	
8	130.4	257	257	**100**	116	116	**100**	256	256	**100**	141	131	**93**	
9	134.1	251	251	**100**	117	117	**100**	256	256	**100**	132	120	**91**	
10	134.9	257	257	**100**	120	120	**100**	254	254	**100**	141	140	**99**	
11	160.1	251	251	**100**	47	36	**77**	255	255	**100**	71	70	**99**	
12	167.4	251	251	**100**	29	16	**55**	260	260	**100**	1	0	**0**	
13	176.1	249	249	**100**	2	0	**0**	250	250	**100**	1	0	**0**	
14	172.6	247	247	**100**	20	18	**90**	253	253	**100**	48	48	**100**	
15	155.1	243	243	**100**	2	0	**0**	248	248	**100**	90	76	**84**	
16	138.4	249	249	**100**	115	115	**100**	256	256	**100**	117	116	**99**	
17	131	254	254	**100**	93	92	**99**	249	249	**100**	160	160	**100**	
18	132.9	250	250	**100**	115	115	**100**	249	249	**100**	136	133	**98**	
19	154.7	249	249	**100**	42	35	**83**	252	252	**100**	55	50	**91**	

In order to understand and visually see the comparison between the different simulation runs, line graphs were produced using this data. The graphs, Figure 4-13 and Figure 4-14, show the difference between each sending pair's packet delivery metric when using the Shadowing model. The darker line with the diamond markers indicate the total number of TCP packets sent by node 0, while the lighter line with the square markers indicate the total number of TCP packets received by the destination node. For TCP traffic within NS-2 it appears that the node will only send TCP traffic if it knows it can deliver to the intended recipient. Within the trace files there are various NS-2 administrative and discovery packets being sent between the nodes that do clutter up the results. Comparing both graphs shows that the Egg shaped pattern provides a higher level of packet delivery with the Shadowing model applied even though it was not successful in transmitting at all to 6 of the 19 nodes. The Circular shaped pattern on the other hand was unable to transmit packets to 7 of the 19 nodes and had lower packet delivery metrics to all nodes when compared to the egg shaped pattern. The NS-2 node placement is shown to the right of the graphs for reference and the nodes that have received a 0% packet delivery metric have been grayed out for a quick visual comparison. These graphs also provided relevant results when observing how NS-2 handles TCP traffic with wireless nodes. The results show that when a destination node is outside of the sending node's coverage area then the sending node will only attempt a few packet transmissions or even none at all. This can be attributed to either a nonexistent route between the sender and destination nodes or because NS-2's TCP implementation is unable to create a reliable connection between the sending and receiving nodes [16].

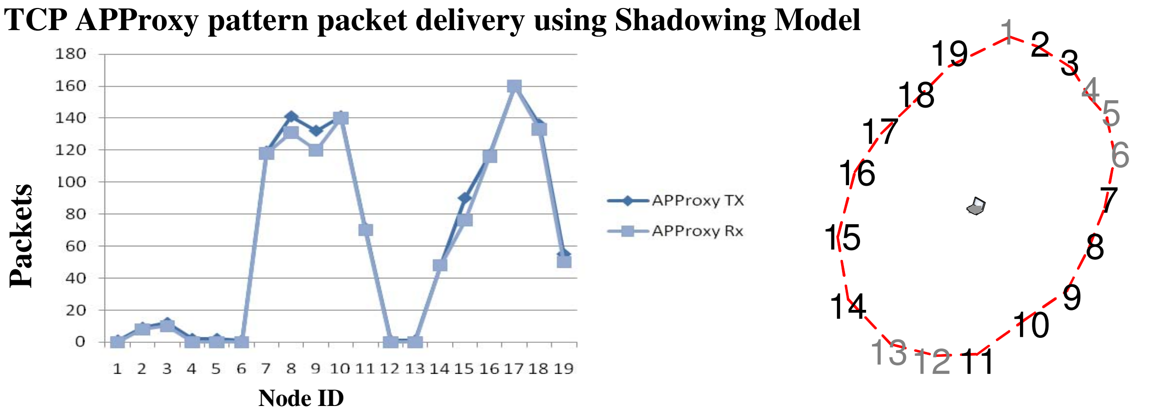


Figure 4-13. Line graph of packet delivery using the Shadowing APProxy pattern

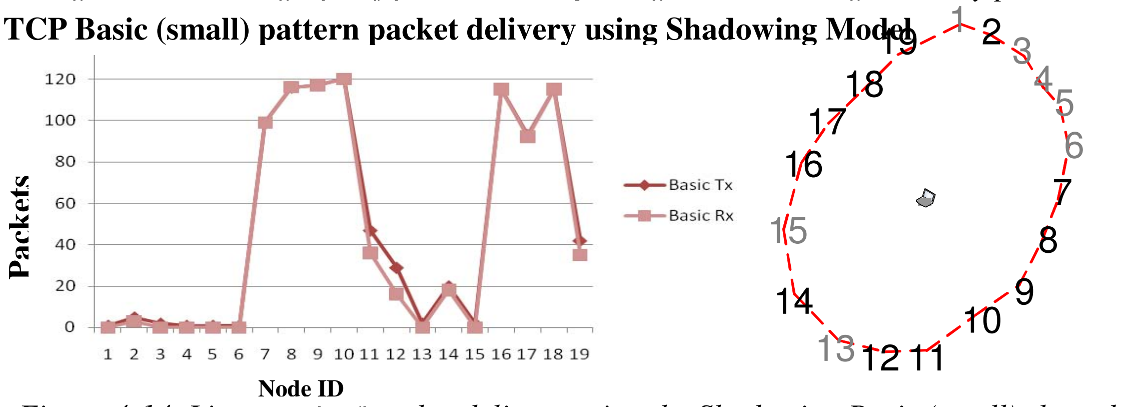


Figure 4-14. Line graph of packet delivery using the Shadowing Basic (small) shaped propagation pattern

The next two graphs produced from the data in Table 4-2 are Figure 4-15 and Figure 4-16. The first graph, Figure 4-15, shows the comparison between the APProxy pattern and NS-2's Basic (small) pattern when they both are utilizing the Two Ray Ground model. The red line corresponds to the Basic pattern and the blue line corresponds to the APProxy pattern. The important result highlighted in this graph is the inability of the Basic (small) pattern to send traffic successfully to nodes 1-3. The Basic pattern was however capable of successfully transmitting packets to all other nodes at the same rate as the APProxy pattern.

TCP packet reception for APProxy and Basic (small) using Two Ray Ground model

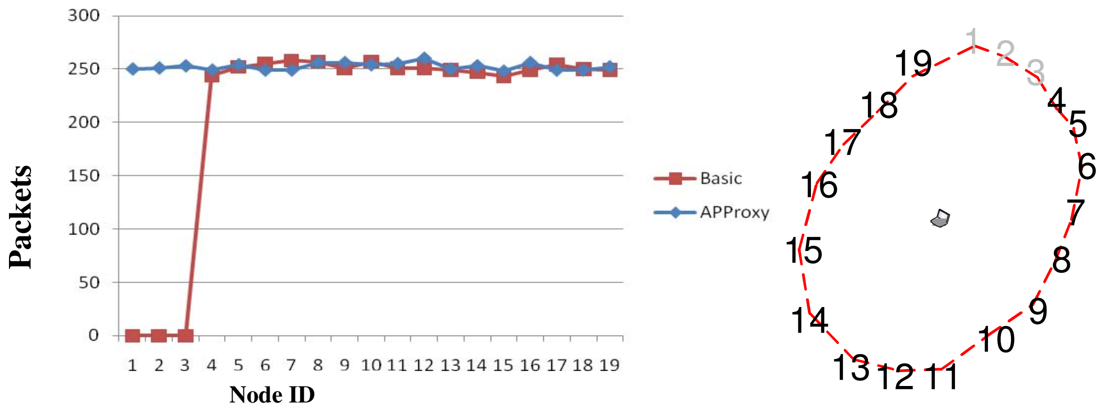


Figure 4-15. Line graph of packet reception between Basic and APProxy patterns with Two Ray Ground model

The next graph, Figure 4-16, also compares the Basic (small) pattern to the APProxy pattern just as Figure 4-15 does except that this graph shows the differences when the Shadowing model is used instead of the Two Ray Ground model. The first result that is shown in this graph is that the Shadowing model decreases the total packets received per node to 160 or less. This is due to the fading effects that are utilized in NS-2's Shadowing model. The next result detailed in this graph is that the APProxy pattern has a greater than or equal packet reception count when compared to the Basic pattern at every node except node 12. When the results are averaged between all the nodes the

Basic pattern averages 46.42 TCP packets per second and the APProxy pattern averages 62.11 TCP packets per second. This equates to a difference in favor of APProxy of 15.68 TCP packets per second or a 33.79% higher packet reception rate than the Basic pattern.

TCP packet reception for APProxy and Basic (small) using Shadowing model

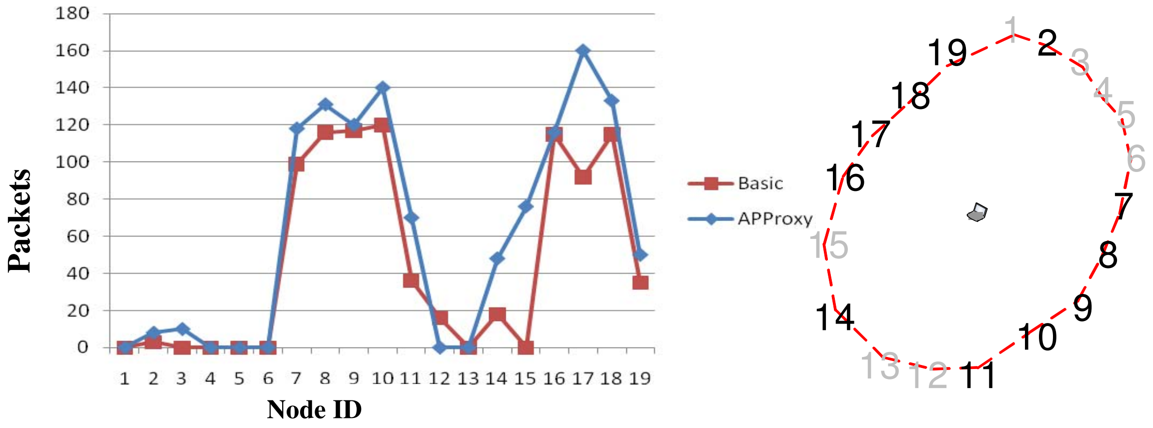


Figure 4-16. Line graph of packet reception between Basic and APProxy patterns with Shadowing model

The last graph, Figure 4-17 was of packet reception results for all four different simulation runs. This graph is a culmination of the graphs depicted in Figure 4-15 and Figure 4-16. The results for the Two Ray Ground model in NS-2 prove that MANET research does utilize the assumed axioms discussed in Chapter Two of this thesis. In particular axioms 1 and 2, deal with the physical layer's coverage area being represented as a perfect circle with every node having the same range. This graph shows that the APProxy pattern running on top of the Shadowing model may be more realistic when using the values used in this research work. This graph also has the NS-2 node placement to the right of it, and any node that experienced a 0% packet delivery metric were grayed out regardless of model or pattern.

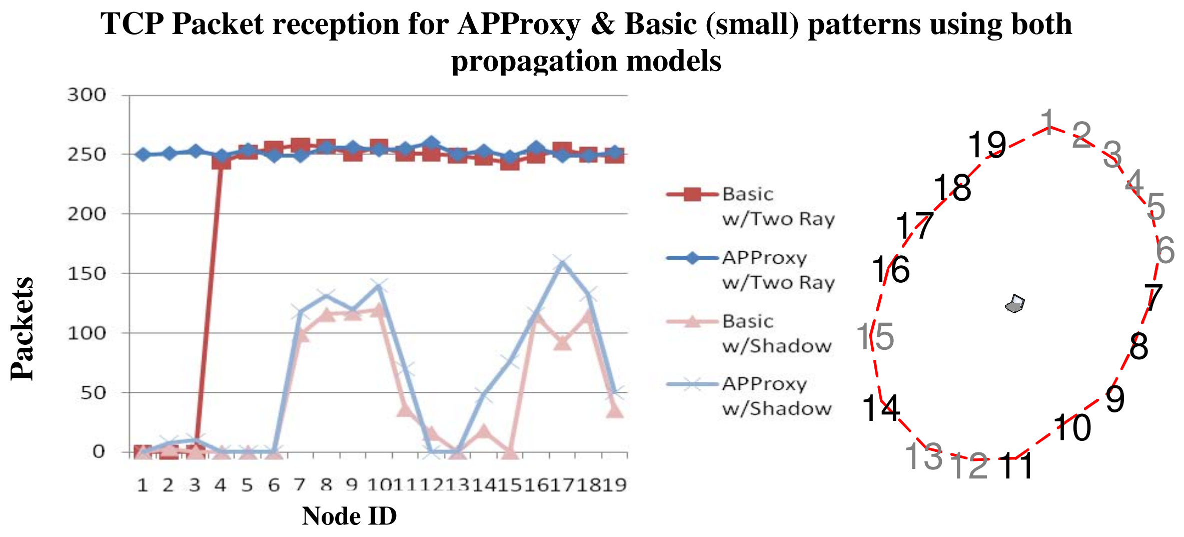


Figure 4-17. Line graph of packet receptions between all four TCP simulation runs

This graph can be broken down into several areas in order to understand the results that it presents. The lighter blue and red lines indicate the shadowing model's packet reception of the receiving node, whereas the darker blue and red lines indicate the Two Ray model's packet reception. Blue lines correspond to the modified APProxy pattern (Egg shaped), which correlates to the propagation pattern observed in the OFTB. The Red lines correspond to the basic NS-2 propagation pattern (Circular), which is what axiom 1 states is a common mistake in current MANET simulation environments since the pattern is a perfect circle around the node.

The graph shows that when the Shadowing model is used with TCP traffic then there is a decrease in packet reception at the destination node as compared to the Two Ray propagation models. The Two Ray line graphs indicate a perfect signal to the destination node so every packet that was sent by the source node is received by the destination node as long as the node is within the source's coverage area. Using the Two Ray Ground model, only the APProxy pattern was able to transmit packets to every node in the environment. The basic pattern was unable to transmit any packets to nodes 1-3, which was expected based on the simulation coverage graph as seen in Figure 4-18. In addition to nodes 1-3 the basic pattern was also expected to have poor transmission rates to nodes 12 and 13, but these nodes had a 100% packet delivery metric with the Two Ray model. The ability of the basic pattern to deliver packets to those nodes was unexpected and shows that the basic pattern's results are not exactly in line with the graph in Figure 4-18. This unexpected result leads to the assumption that the propagation simulation graphs generated in section 4.4.1 are not perfect representations of the exact coverage area of each pattern.

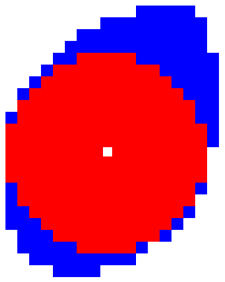


Figure 4-18. Basic pattern overlaid onto APProxy pattern (same as Figure 4-10)

The shadowing line graphs for both patterns also produced relevant results because both patterns were unsuccessful at delivering a single packet to nodes 1, 4, 5, 6, and 13. The expected results of both patterns when the shadowing model was used was that the packet received count at each destination node would be lower than the Two Ray model's results, but not a complete inability to receive at those nodes. In addition to both patterns inability to successfully send packets to those nodes, the packet delivery metric is very similar for each destination node between the patterns. In fact the basic pattern has a better packet delivery percentage on some nodes. This is because the basic pattern was able to receive a packet better at the destination nodes, but this metric is slightly misleading when used with TCP traffic. The reason it is misleading is while the basic pattern did have a better ratio of received packets to sent packets (packet delivery metric) the basic pattern also consistently sent fewer TCP packets to the destination nodes. This is seen when comparing the two patterns line graphs in Figure 4-17. While the APProxy pattern does perform better overall than the basic NS-2 pattern, the difference between the two patterns is small for some nodes and even performs better for one, node 12, in the Basic pattern results.

4.4.2.1.2 TCP Basic (large) and APProxy simulation experiments

The next TCP traffic simulation experiment that was conducted in order to illustrate the differences between the Basic pattern and the APProxy pattern was to create the Basic pattern shown in Figure 4-11. This pattern is referred to as the Basic (large) pattern throughout this section and in Table 4-3. The reason this simulation was conducted and these results were generated was to show that current MANET simulations would inflate the results if the coverage pattern was set to a higher value to include every node in the scenario modeled after the OFTB. The APProxy pattern data that was used to fill in this table is the same data in Table 4-2.

Table 4-3. TCP Packet Delivery Comparison between APProxy and Basic (large) propagation patterns using either Two Ray Ground or Shadowing models

TCP Packet Delivery Results													
Node Info		**Basic (large) pattern (Circular)**						**APProxy pattern (Egg)**					
ID	**Distance (ft)**	**Two Ray**			**Shadowing**			**Two Ray**			**Shadowing**		
		Tx	**Rx**	**PD%**	**Tx**	**Rx**	**PD%**	**Tx**	**Rx**	**PD%**	**Tx**	**Rx**	**PD%**
1	189.1	250	250	**100**	1	0	**0**	250	250	**100**	1	0	**0**
2	186.8	251	251	**100**	57	56	**98**	251	251	**100**	9	8	**89**
3	183.7	254	254	**100**	1	0	**0**	253	253	**100**	12	10	**83**
4	170.6	245	245	**100**	106	95	**90**	249	249	**100**	2	0	**0**
5	171.9	246	246	**100**	1	0	**0**	254	254	**100**	2	0	**0**
6	161	252	252	**100**	105	105	**100**	249	249	**100**	1	0	**0**
7	139	249	249	**100**	198	198	**100**	249	249	**100**	119	118	**99**
8	130.4	248	248	**100**	231	231	**100**	256	256	**100**	141	131	**93**
9	134.1	251	251	**100**	212	212	**100**	256	256	**100**	132	120	**91**
10	134.9	258	258	**100**	228	228	**100**	254	254	**100**	141	140	**99**
11	160.1	252	252	**100**	172	168	**98**	255	255	**100**	71	70	**99**
12	167.4	259	259	**100**	117	117	**100**	260	260	**100**	1	0	**0**
13	176.1	253	253	**100**	1	0	**0**	250	250	**100**	1	0	**0**
14	172.6	252	252	**100**	132	132	**100**	253	253	**100**	48	48	**100**
15	155.1	245	245	**100**	174	174	**100**	248	248	**100**	90	76	**84**
16	138.4	248	248	**100**	212	212	**100**	256	256	**100**	117	116	**99**
17	131	255	255	**100**	218	218	**100**	249	249	**100**	160	160	**100**
18	132.9	253	253	**100**	219	219	**100**	249	249	**100**	136	133	**98**
19	154.7	251	251	**100**	154	154	**100**	252	252	**100**	55	50	**91**

The first graph produced to compare the Basic (large) pattern to the APProxy pattern is Figure 4-19. In comparing this data the first result that differs from the other comparison between APProxy and the Basic (small) pattern is that the Basic (large) pattern is able to successfully transmit to every node when using the Two Ray Ground model. The Basic (small) pattern was unable to transmit successfully to nodes 1-3. The similarity to the Basic (small) pattern is that if the receiving node is in the coverage area of the sender then those nodes will receive every packet sent within the one second time slice. This means there is once again little difference between the APProxy pattern and the Basic pattern when the Two Ray Ground model is used as long as the pattern covers every node.

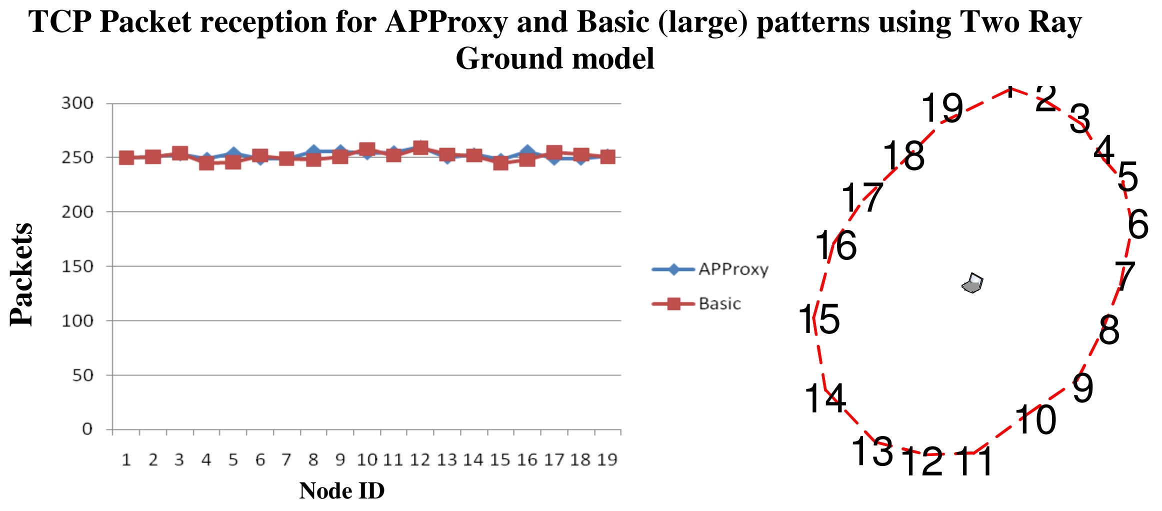


Figure 4-19. Line graph of packet reception between Basic and APProxy patterns with Two Ray Ground model

The next graph produced using the results in Table 4-3 is Figure 4-20. This graph illustrates the inflation in packets received when the Basic pattern utilizes a higher transmit power in order to represent the coverage area seen in Figure 4-11. The Basic (large) pattern has a higher or equal packet reception count when compared to the APProxy pattern at every node except node 3. This is a complete reversal from the results seen when comparing the Basic (small) pattern to the APProxy pattern in that the APProxy pattern had higher packet reception counts at every node except one. When the results are averaged between all the nodes the Basic (large) pattern averages 132.58 TCP packets per second and the APProxy pattern averages 62.11 TCP packets per second. This equates to a difference in favor of the Basic (large) pattern of 70.37 TCP packets per second or a 113.47% higher packet reception rate than the Basic pattern.

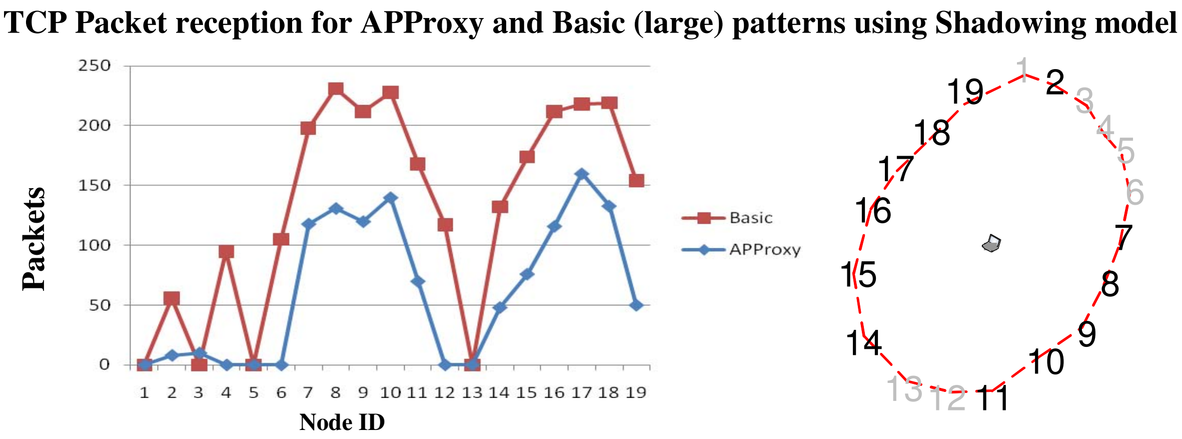


Figure 4-20. Line graph of packet reception between Basic and APProxy patterns with Shadowing model

4.4.2.2. CBR Traffic Analysis

The CBR traffic results were gathered in the same manner as the TCP traffic results. The trace files were filtered for the same two types of packets indicating the packet delivery metric between the central node and each of the outlying nodes. The only difference is that the packet type was now CBR instead of TCP.

4.4.2.2.1 CBR Basic (small) and APProxy simulation experiments

Table 4-4 was produced using the summation of these packets under their respective columns. The difference in values indicates a divergence between the two different traffic types used in this research, TCP and CBR. Since the CBR traffic in this research uses the User Datagram Protocol (UDP), there were no reply acknowledgement packets from the receiver. In addition, NS-2 treated this traffic differently, because it sent the packets regardless of the receiving node's ability to hear them. This is shown in the discrepancy between the Tx and Rx of packets using the Shadowing model. The last interesting observation is that the traffic was configured to send 200 packets per second, but NS-2 was only able to send approximately 180 (181 from node 7 on). This may be due to the administrative packets that were sent within each NS-2 trace file.

Table 4-4. CBR Packet Delivery Comparison between APProxy and Basic (small) propagation patterns using either Two Ray Ground or Shadowing models

CBR Packet Delivery Results													
Node Info		Basic (small) pattern (Circular)						APProxy pattern (Egg)					
ID	Distance (ft)	Two Ray			Shadowing			Two Ray			Shadowing		
		Tx	Rx	PD%	Tx	Rx	PD%	Tx	Rx	PD%	Tx	Rx	PD%
1	189.1	180	0	**0**	180	0	**0**	180	180	**100**	180	0	**0**
2	186.8	180	0	**0**	180	13	**7**	180	180	**100**	180	17	**9**
3	183.7	180	0	**0**	180	0	**0**	180	180	**100**	180	0	**0**
4	170.6	180	180	**100**	180	0	**0**	180	180	**100**	180	0	**0**
5	171.9	180	180	**100**	180	0	**0**	180	180	**100**	180	0	**0**
6	161	180	180	**100**	180	0	**0**	180	180	**100**	180	0	**0**
7	139	181	181	**100**	181	0	**0**	181	181	**100**	181	0	**0**
8	130.4	181	181	**100**	181	176	**97**	181	181	**100**	181	181	**100**
9	134.1	181	181	**100**	181	181	**100**	181	181	**100**	181	172	**95**
10	134.9	181	181	**100**	181	177	**98**	181	181	**100**	181	165	**91**
11	160.1	181	181	**100**	181	48	**27**	181	181	**100**	181	23	**13**
12	167.4	181	181	**100**	181	5	**3**	181	181	**100**	181	0	**0**
13	176.1	181	181	**100**	181	6	**3**	181	181	**100**	181	16	**9**
14	172.6	181	181	**100**	181	0	**0**	181	181	**100**	181	0	**0**
15	155.1	181	181	**100**	181	94	**52**	181	181	**100**	181	108	**60**
16	138.4	181	181	**100**	181	86	**48**	181	181	**100**	181	138	**76**
17	131	181	181	**100**	181	177	**98**	181	181	**100**	181	181	**100**
18	132.9	181	181	**100**	181	179	**99**	181	181	**100**	181	181	**100**
19	154.7	181	181	**100**	181	90	**50**	181	181	**100**	181	77	**43**

Line graphs were produced for the CBR traffic in the same way as for the TCP traffic. Once again, this work compared the two patterns to their packet delivery results under the Shadowing model, Figure 4-21 and Figure 4-22. The egg shaped pattern again had a better packet delivery metric, but not for every node and not as great of a margin. Nodes 9, 10, 11, 12, and 19 in the circular pattern simulation all had a better packet delivery metric, with most of the other nodes being within a few packets of the egg shaped pattern. This is interesting in that the different types of traffic, TCP or CBR, affect the packet delivery metric differently. The NS-2 node placement is shown to the right of the graphs and the nodes with a 0% packet delivery metric have been grayed out. Unlike TCP traffic in NS-2, CBR (UDP) traffic will allow the sending node to send packets to the destination node, even if the node is not in the sender's coverage area. This is important to note since the packet metric for CBR traffic will now be influenced by a constant send rate of approximately 180 packets per node. Another interesting observation in the two graphs below when comparing the CBR traffic to the TCP traffic results is that even with the Shadowing model in use some destination nodes were able to receive all the packets that were sent in the one second time period. This was not true for the TCP traffic results, since not a single node was able to receive more than half of the packets sent under the Shadowing model as compared to the Two Ray model.

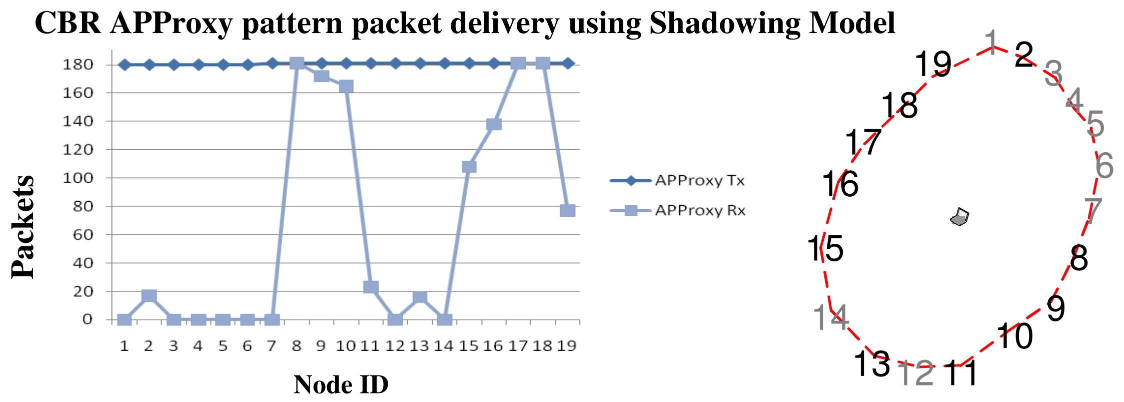


Figure 4-21. Line Graph of packet delivery using the Shadowing Egg shaped propagation pattern

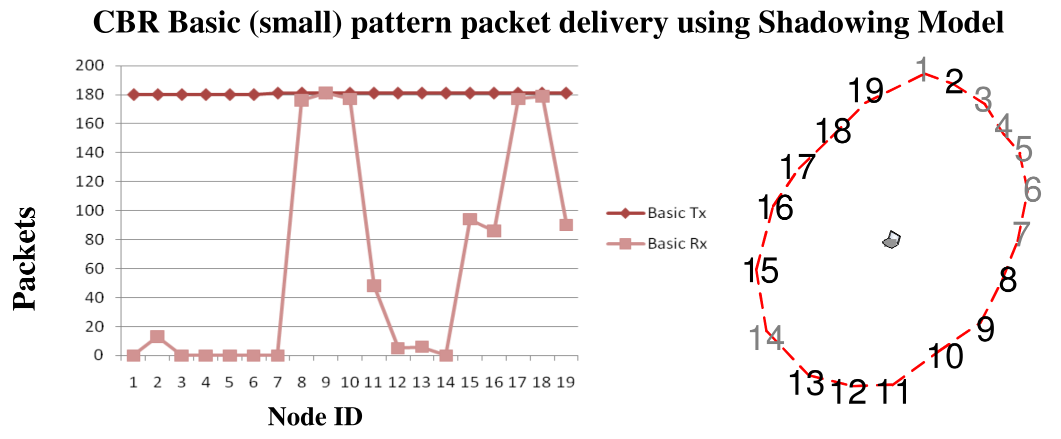


Figure 4-22. Line Graph of packet delivery using the Shadowing Circular shaped propagation pattern

The next two graphs produced from the data in Table 4-4 are Figure 4-23 and Figure 4-24. These graphs produce similar results for the CBR traffic as they did for the TCP traffic (Figure 4-15 and Figure 4-16) with slight differences. The difference between TCP and CBR packet reception in the first graph is that the CBR traffic will only send a maximum of 200 packets per second and due to NS-2's processing overhead, only 180 packets are sent each second. The graph shows the same results as TCP when comparing the two patterns, Basic to APProxy, because the APProxy pattern is able to successfully transmit packets to all nodes whereas the Basic pattern cannot send to nodes 1-3. This result is the most important to this research since it again shows the inability of the Basic pattern to transmit packets to all the nodes.

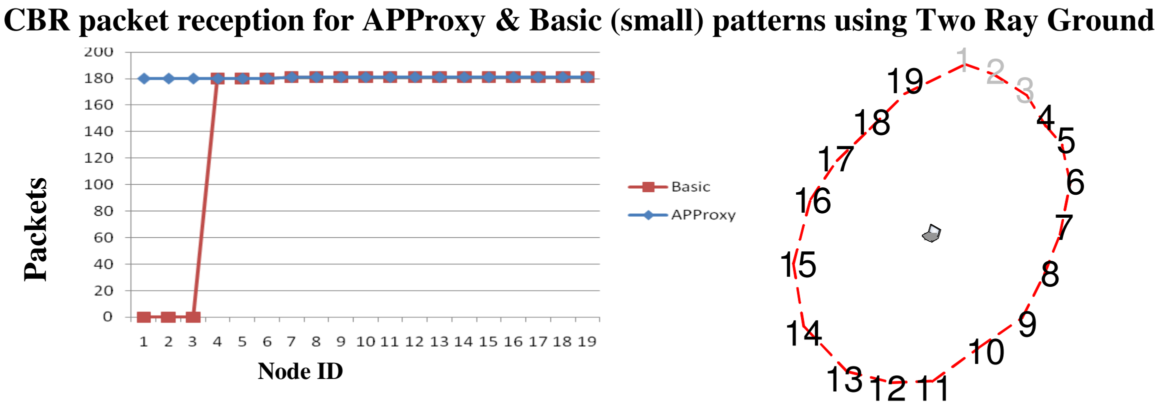


Figure 4-23. Line graph of packet reception between Basic and APProxy patterns with Two Ray Ground model

The next graph, Figure 4-24, again compares the Basic (small) pattern to the APProxy pattern while using the Shadowing model as the basis for both patterns in the same way Figure 4-16 does. The difference between the TCP and CBR traffic results in this graph is that the two patterns are more closely aligned. The Basic pattern when sending CBR traffic is able to successfully transmit more packets to 5 of the 19 nodes than the APProxy pattern. When the results are averaged between all the nodes the Basic pattern averages 64.84 CBR packets per second and the APProxy pattern averages 66.26

CBR packets per second. This is only a difference in favor of APProxy of 1.42 CBR packets per second on average or a 2.19% higher packet reception rate than the Basic pattern.

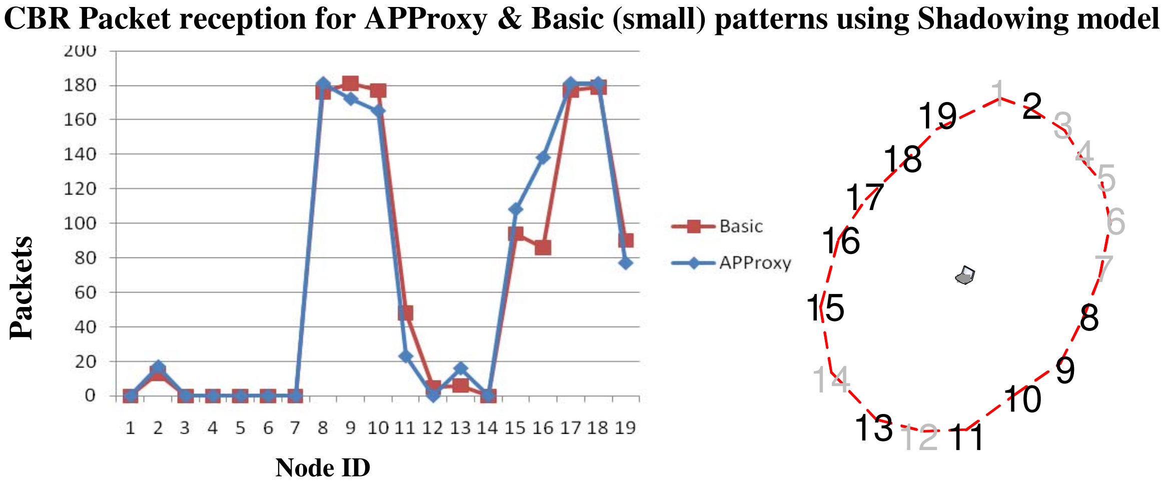
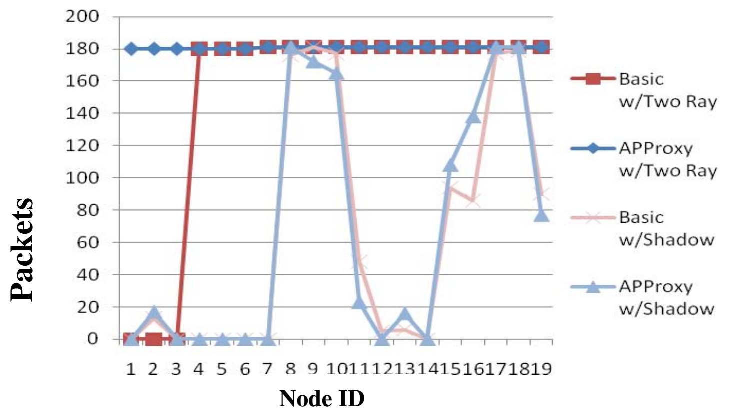


Figure 4-24. Line graph of packet reception between Basic and APProxy patterns with Shadowing model

Finally the last line graph created from the results in Table 4-4, Figure 4-25, was another graph indicating the packet reception between all four different simulations passing CBR traffic. This graph again shows that the egg shaped pattern performs better than the circular pattern, but there is a significant difference between it and the TCP traffic graph. The shadowing model does not directly affect every node's packet delivery metric as it did with TCP traffic. In addition as stated earlier the circular pattern aligns more closely with the egg shaped pattern with the Shadowing model when using CBR traffic. Also the improvement of the APProxy pattern over the Basic pattern is less pronounced when using the CBR protocol over the TCP protocol. There is however very few network deployments that will only use the CBR (UDP) protocol. Most deployments will be a hybrid of the two protocols with the majority being TCP traffic. This graph also has the NS-2 node placement to the right of it, and any node that experienced a 0% packet delivery metric was grayed out regardless of model or pattern.

CBR packet reception for APProxy & Basic (small) patterns using both models



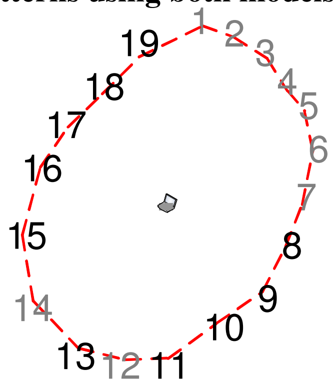


Figure 4-25. Line graph of Packet reception between all four CBR simulation runs

This graph can be broken down into several areas in order to understand the results that it presents. The darker blue and red lines indicate the shadowing model's packet reception of the receiving node, whereas the lighter blue and red lines indicate the Two Ray model's packet reception. Blue lines correspond to the modified APProxy pattern (Egg shaped), which correlates to the propagation pattern observed in the OFTB. The Red lines correspond to the basic NS-2 propagation pattern (Circular), which is what axiom 1 states is a common mistake in current MANET simulation environments since the pattern is a perfect circle around the node. The line graphs are similar in shape to the same graph produced with the results from the TCP traffic simulations. One difference between Figure 4-17 and Figure 4-25 is the number of packets sent to each node. The interesting difference between the two figures was mentioned previously in this section, which is the fact that some destination nodes in both patterns, Basic and APProxy, were able to receive the same amount of packets in either model, Two Ray or Shadowing.

Other than those items the two figures do produce relatively the same results in that the Basic pattern when using the Two Ray Ground model is unable to transmit a single packet successfully to nodes 1-3. The figures are also similar because the APProxy pattern when using the Two Ray Ground model is capable of receiving 100% of the packets sent to every single destination node at each time slice. The other similarity is that the Shadowing model degrades the packet delivery metric significantly for most of the nodes. Once again the basic NS-2 pattern is does perform worse overall, but this time there are four destination nodes, 9, 10, 11, and 19, that receive more packets with the basic pattern than the APProxy pattern.

4.4.2.2.2 CBR Basic (large) and APProxy simulation experiments

The traffic simulation experiment, to illustrate the difference between the patterns seen in Figure 4-11, was conducted with CBR traffic. This thesis work refers to the NS-2 Basic pattern as the Basic (large) pattern throughout this section and in Table 4-5. This simulation was conducted to show how inflated the results would be if the transmit power was increased on the Basic NS-2 pattern to ensure complete coverage of all nodes. The APProxy data used for the comparisons in this section is the same as in Table 4-4.

Table 4-5. CBR Packet Delivery Comparison between APProxy and Basic (large) propagation patterns using either Two Ray Ground or Shadowing models

CBR Packet Delivery Results													
Node Info		Basic (large) pattern (Circular)						APProxy pattern (Egg)					
ID	Distance (ft)	Two Ray			Shadowing			Two Ray			Shadowing		
		Tx	Rx	PD%	Tx	Rx	PD%	Tx	Rx	PD%	Tx	Rx	PD%
1	189.1	180	180	**100**	180	0	**0**	180	180	**100**	180	0	**0**
2	186.8	180	180	**100**	180	114	**63**	180	180	**100**	180	17	**9**
3	183.7	180	180	**100**	180	0	**0**	180	180	**100**	180	0	**0**
4	170.6	180	180	**100**	180	67	**37**	180	180	**100**	180	0	**0**
5	171.9	180	180	**100**	180	13	**7**	180	180	**100**	180	0	**0**
6	161	180	180	**100**	180	180	**100**	180	180	**100**	180	0	**0**
7	139	181	181	**100**	181	181	**100**	181	181	**100**	181	0	**0**
8	130.4	181	181	**100**	181	181	**100**	181	181	**100**	181	181	**100**
9	134.1	181	181	**100**	181	181	**100**	181	181	**100**	181	172	**95**
10	134.9	181	181	**100**	181	181	**100**	181	181	**100**	181	165	**91**
11	160.1	181	181	**100**	181	181	**100**	181	181	**100**	181	23	**13**
12	167.4	181	181	**100**	181	181	**100**	181	181	**100**	181	0	**0**
13	176.1	181	181	**100**	181	166	**92**	181	181	**100**	181	16	**9**
14	172.6	181	181	**100**	181	161	**89**	181	181	**100**	181	0	**0**
15	155.1	181	181	**100**	181	180	**99**	181	181	**100**	181	108	**60**
16	138.4	181	181	**100**	181	181	**100**	181	181	**100**	181	138	**76**
17	131	181	181	**100**	181	181	**100**	181	181	**100**	181	181	**100**
18	132.9	181	181	**100**	181	180	**99**	181	181	**100**	181	181	**100**
19	154.7	181	181	**100**	181	180	**99**	181	181	**100**	181	77	**43**

The first graph produced compares the Basic (large) pattern to the APProxy pattern in Figure 4-26. In comparing this data, the first difference from the results in Table 4-4 is that the Basic (large) pattern is able to successfully transmit to every node when using the Two Ray Ground model. The Basic (small) pattern was unable to transmit successfully to nodes 1-3. This is a similar result as seen with the TCP traffic simulations, meaning there is little difference between the APProxy pattern and the Basic (large) pattern when the Two Ray Ground model is used.

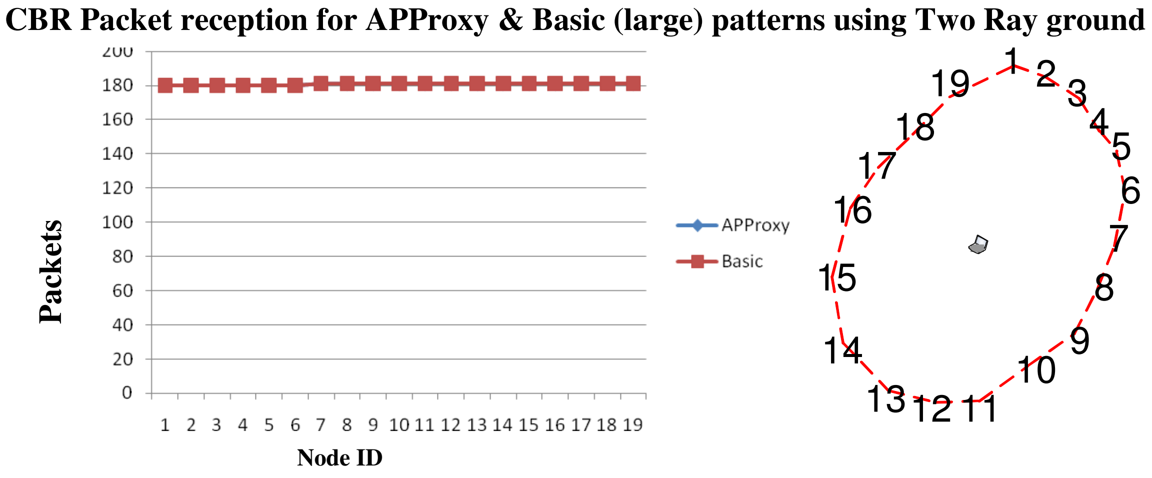


Figure 4-26. Line graph of packet reception between Basic and APProxy patterns with Two Ray Ground model

The next graph, Figure 4-27, was produced in order to illustrate the inflated results that are generated when using a higher transmit power for the Basic pattern. Just as with the TCP results the Basic (large) pattern out performs the APProxy pattern. Not a single node performs better with the APProxy pattern in the comparison. In fact, the Basic (large) pattern using the Shadowing model performs at the same level as the Two Ray Ground for 12 of the 19 nodes. When the results are averaged between all the nodes, the Basic (large) pattern averages 141.53 CBR packets per second and the APProxy pattern averages 66.26 CBR packets per second. This equates to a difference in favor of the Basic (large) pattern of 75.26 CBR packets per second or a 113.58% higher packet reception rate than the Basic pattern.

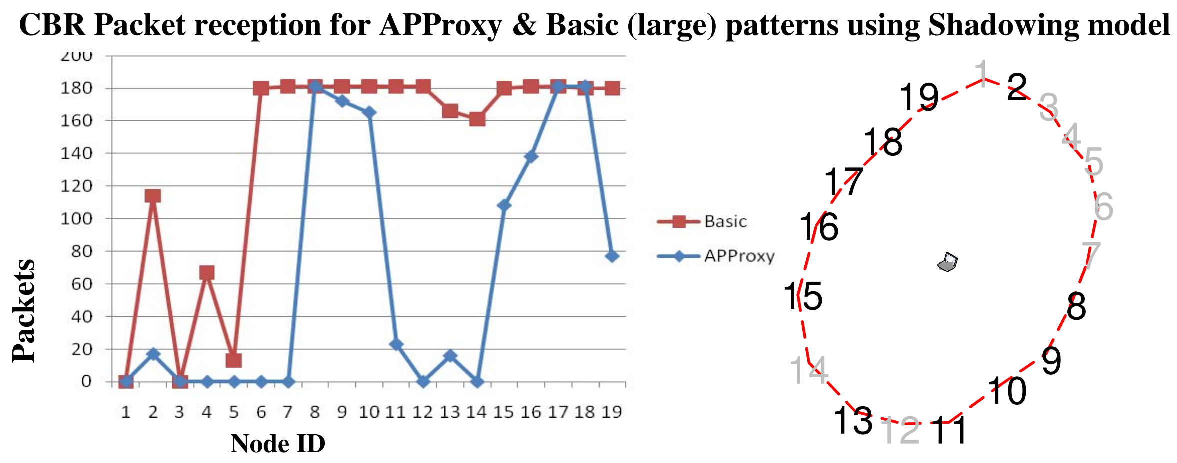


Figure 4-27. Line graph of packet reception between Basic and APProxy patterns using Shadowing model

4.4.2.3. Additional Traffic Analysis

While reviewing the graphs of TCP and CBR traffic for both patterns another assumed axiom was observed. The Two Ray Ground graphs, Figure 4-15 and Figure 4-23, show the problem of axiom 4, "If I can hear you, I can hear you perfectly". NS-2 uses this assumed axiom for the Two Ray Ground model as stated in Chapter Two in that any node in the coverage area of the sending node will receive all packets sent to it. The Two Ray Ground graphs are in contrast with the Shadowing model graphs, Figure 4-16 and Figure 4-24, which do remove assumed axiom 4 from the simulation environment. These graphs clearly showcase the shortcomings when using NS-2's Two Ray Ground propagation model since every packet was delivered to the destination node unless it was outside of the coverage area. This obviously does not hold true in the real world since there is packet loss due to obstructions or the distance between the sender and receiver.

4.4.3 Impact and Contributions

The impact and contributions of this research to the community demonstrates that simulation environments can realistically model the physical layer of a MANET protocol implementation. The procedures that are outlined in this work detail the steps needed for future developers to create their own realistic physical layer NS-2 models. This work illustrates the specific results of a physical test bed experiment that was later modeled in an NS-2 simulation environment. This demonstrated the differences at the physical layer between two basic NS-2 simulation environments and the environment modeled after the physical test bed (APProxy). Two basic environments were built to showcase two common coverage area scenarios that a MANET developer/researcher utilizes in current simulations. The first basic coverage area completely resided within the actual physical coverage area results (Figure 4-10) and the second covered the entire physical coverage area (Figure 4-11).

The results, for the basic NS-2 simulations and the modified APProxy simulations, demonstrated that the realistic propagation pattern (APProxy) avoids the axioms discussed in Chapter Two. The results support the findings [26] that MANET simulations are utilizing axioms that assume away physical layer attributes. The reason is that NS-2 does have the built-in capability to modify the Basic circular propagation model's coverage pattern. When a researcher or developer utilizes NS-2 to test or evaluate a MANET protocol, their options are limited to the basic circular pattern running on top of either the Two Ray Ground model or the Shadowing model. They can either enlarge the circular pattern to completely cover what the real world pattern would cover, or they can set the circular pattern to have a coverage area that resides within the non-circular physical world's coverage area. Both scenarios utilizing the basic pattern, as shown in the results of this chapter, will not provide an accurate realistic model since nodes will either have a lower or higher packet reception rate when compared to the realistic APProxy pattern.

If the basic patterns are built on top of the Two Ray Ground model then they adhere to axiom 4, “If I can hear you, I can hear you perfectly”, since the Two Ray Ground model assumes a perfect signal to any node in the sender’s coverage area. NS-2 does allow for fading effects when the Shadowing model is utilized with the basic pattern, but the Shadowing model still adheres to axiom 1 and 2 by applying a circular pattern equally to all nodes in the simulation. The results that were discussed in the previous sections of this chapter do indicate that changing the signal pattern to a realistic pattern does provide for a more accurate coverage area. The work outlined in this thesis should be implemented in order to have a more robust simulation environment that depicts a realistic physical layer.

4.5. Summary

The basic testing conducted in this chapter indicates that the assumptions (axioms) made by MANET researchers in simulations do overlook key characteristics of the MANET environment. These physical layer assumptions can be simulated in NS-2 and can offer a better understanding of how the protocol in question will operate when deployed in the physical domain. The procedures detailed in this chapter demonstrate the knowledge gained from researching the information contained in Chapter Two. The test bed was developed using code and instructions from the Grid/Roofnet project and the simulation environment was developed by adhering to some of the techniques outlined by Marc Greis, the Institute of Telematics, and other previous NS-2 research.

5. Conclusions and Recommendations

The conclusions are based off the knowledge gained from previous work that was discussed in chapter two and how the results, observed within chapter four, were able to address the overall problem discussed in chapters one and three. Future research ideas are also discussed in this chapter for those that may be interested in extending this work.

5.1. Conclusions of Research

This thesis work has been conducted to develop a MANET simulation environment that adheres more closely to the real world. The problem and a generic scenario were introduced in Chapter One detailing the overall need for a more realistic simulation environment. Chapter Two covered the MANET protocols that this thesis utilized and their general configurations. In addition, Chapter Two covered network simulators and the current state of MANET simulations. It also covered the *Click* software architecture since *Click* was used to gather the physical test bed data that was later used to develop the simulation environment. Chapter Three explained the simulation environment that was developed and the various workloads and parameters that would be a part of it. Chapter Four detailed the experiments that were conducted and provided analysis on the resulting data. This data was analyzed for trends and the conclusions were offered based off these identified trends.

This thesis demonstrates the capability to simulate MANET protocols in close alignment with their real world deployments. In particular the propagation of the wireless signal can be altered to mimic specific radios and environments that do affect the propagation pattern in the real world. The ability to take the same setup, e.g. propagation coverage area, signal strength, etc, in the physical world and then apply them to the simulation is very important when creating MANET models. The benefits that researchers will see by implementing more realistic techniques in generating these models are a better understanding of MANET routing, coverage areas, efficiency, and overall performance of the protocol.

The testing done in Chapter Four demonstrates that current MANET simulations that assume the axioms defined by Kotz et al. do overlook key aspects of the MANET protocols. Most notably the propagation observed in the physical world when applied to the simulation allows for an accurate deployment of MANET nodes. The ability to generate propagation models that mimic a specific real world wireless radio signal will assist researchers and developers in deploying realistic MANETs that can effectively and efficiently route traffic.

5.2. Future Research

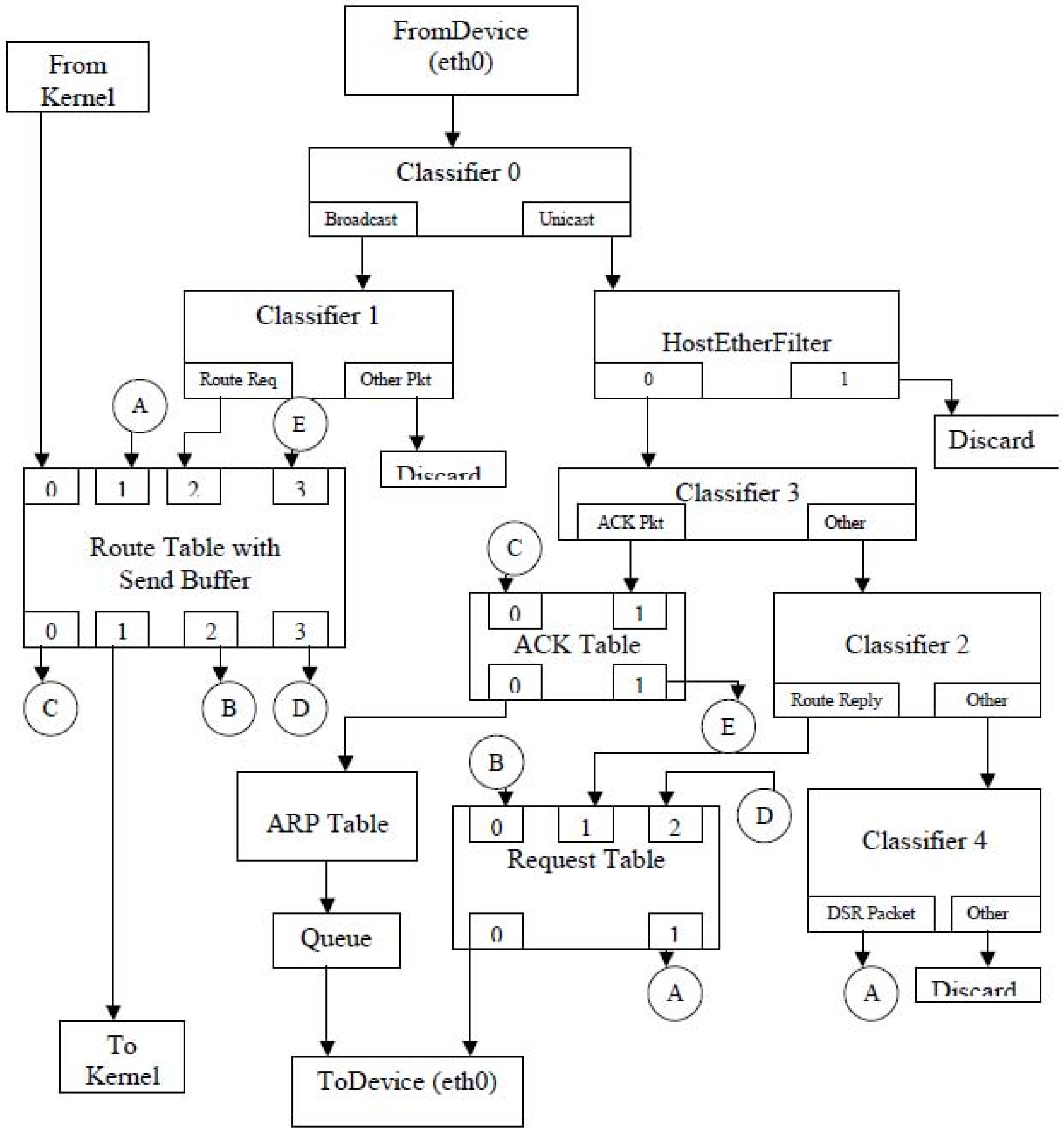
This work has laid the foundation for future efforts in attaining realistic MANET simulations. While this work focused on NS-2 and *Click* technologies, the techniques and methodology discussed can be utilized for other simulation and MANET technology research. Specifically regarding the research conducted in this thesis, the next step would be to extend this work by testing other existing MANET protocols or developing new protocols within *Click.* These can then be tested in the physical test bed and NS-2 simulation environment. There exists a need for other assumed axiom evaluations such as a seventh axiom, which would be the “Trustworthiness of nodes” axiom. Since most MANET simulations assume that a node will route traffic to its neighbors with no malicious intent, this assumed axiom can be evaluated to determine if a malicious node negatively affects a MANET deployment and the levels at which a malicious node might be tolerated.

Appendix A: Acronyms List

AFIT – Air Force Institute of Technology
AODV – Ad-hoc On-demand Distance Vector
AP – Access Point
APProxy – Asymmetric Propagation Proxy
CBR – Constant Bit Rate
CUT – Component Under Test
DBF – Distributed Bellman Ford
DSDV – Destination-Sequenced Distance Vector
DSR – Dynamic Source Routing
DYMO – Dynamic MANET On-demand
FIFO – First In First Out
GloMoSim – Global Mobile Information Systems Simulation Library
HW – Hardware
ICMP – Internet Control Message Protocol
IETF – Internet Engineering Task Force
IP – Internet Protocol
MAC – Media Access Control
MANET – Mobile Ad-hoc NETwork
MIT – Massachusetts Institute of Technology
NAM – Network Animator
NIC – Network Interface Card
NS – Network Simulator
OFTB – Open Field Test Bed
OLSR – Optimized Link State Routing
OPNET – Optimized Network Engineering Tools
OSI – Open System Interconnection
PD% – Packet Delivery Percent
PMP – Proactive MANET Protocol
RFC – Request For Comments
RMP – Reactive MANET Protocol
RREQ – Route Request
Rx – Receive
SA – Situational Awareness
SUT – System Under Test
SWAN – Simulator for Wireless Ad-hoc Networks
TCP – Transmission Control Protocol
TORA – Temporally Ordered Routing Algorithm
Tx – Transmit
UDP – User Datagram Protocol

Appendix B: DSDV *Click* Configuration Code

Directed Graph



DSR Click Directed Graph [14]

Code for the Physical Nodes

All code comments are in blue font while *Click* code remains in standard black font.

```
// This file generated with the following command:
// make-dsdv-config.pl -i wlan1 -a 10.0.0.1 -u

// this configuration performs routing lookup *after* the interface
// queue, and only works with one interface.

AddressInfo(me 10.0.0.1 00:20:a6:61:58:a5);
```

```
elementclass TTLChecker {
  // expects grid packets with MAC headers --- place on output path to
  // decrement the IP TTL for next hop and provide traceroute support.
  // push -> push
  // output [0] passes through the Grid MAC packets
  // output [1] produces ICMP error packets to be passed back to IP
  // routing layer

  input -> cl :: Classifier(19/03, -);
  cl [1] -> output; // don't try to dec ttl for non-IP packets...

  cl [0]
    -> MarkIPHeader(82)
    -> cl2 :: IPClassifier(src host != me, -);

  cl2 [0]-> dec :: DecIPTTL; // only decrement ttl for packets we don't
originate
  cl2 [1] -> output;

  dec [0] -> output;
  dec [1] -> ICMPError(me, 11, 0) -> [1] output;
};

li :: GridLocationInfo2(0, 0, LOC_GOOD false);

elementclass FixupGridHeaders {
  $li | // LocationInfo element
  input
    -> FixSrcLoc($li)
    -> SetGridChecksum
    -> output;
};

elementclass ToGridDev {
  // push, no output
  $dev |
  input -> cl :: Classifier(12/7ffe, // LinkStat 1
                            12/7ffd, // LinkStat 2
                      19/02, 19/03);
  prio :: PrioSched;
  cl [0] -> probe_counter :: Counter -> probe_q :: Queue(5) -> [0]
prio;
  cl [1] -> probe_counter;
  cl [2] -> route_counter :: Counter -> route_q :: Queue(5) ->
FixupGridHeaders(li) -> [1] prio;
  cl [3] ->  data_counter :: Counter ->  data_q :: Queue(5)
    -> data_counter_out :: Counter
    -> tr :: TimeRange
    -> lr :: LookupLocalGridRoute2(me:eth, me:ip, nb)
    -> FixupGridHeaders(li)
    -> data_counter_out2 :: Counter
    -> tr2 :: TimeRange
    -> [2] prio;
  prio
    -> dev_counter :: Counter
    -> t :: PullTee
    -> ToDevice($dev);
  t [1] -> SetTimestamp -> Discard;
};

elementclass FromGridDev {
  // push, no input
  // `Grid' packets on first output
  // `LinkStat' packets on second output
  $dev, $mac |
  FromDevice($dev, PROMISC false)
    -> t :: Tee
    -> HostEtherFilter($mac, DROP_OWN true)
    -> cl :: Classifier(12/7fff, 12/7ffe, 12/7ffd, -);
  cl [0]  // `Grid' packets
    -> ck :: CheckGridHeader
    -> [0] output;
  cl [1]  // `LinkStat 1' packets
    -> [1] output;
  cl [2]  // `LinkStat 2' packets
    -> [1] output;
  cl [3] // everything else
    -> [2] output;
  t [1] -> Discard;
  ck [1] -> Print('Bad Grid header received', TIMESTAMP true, MAXLENGTH
166) -> Discard;
};

elementclass GridLoad {
  // push, no input

  // DATASIZE should be the size of the desired UDP packet (including
  // ethernet, Grid, and IP headers), plus 2 for alignment.  It must
  // be at least 120.  Most of this is stripped off to be re-used
  // later, avoiding expensive pushes in the UDP/IP and Grid
  // encapsulation.
  src :: InfiniteSource(ACTIVE false, DATASIZE 120)
    -> Strip(112) // 14 + 60 + 8 + 20 + 8 + 2 = 112
                  // (eth + grid + grid_encap + ip + udp + 2 for
alignment)
    -> seq :: IncrementSeqNo(FIRST 0, OFFSET 0)
    -> SetIPAddress(me)
    -> StoreIPAddress(4)
    -> udp :: UDPIPEncap(me, 1111, 0.0.0.0, 8021)
    -> count :: Counter
    -> tr :: TimeRange
    -> output;
  sc :: Script;
}

ls2 :: Idle;
ls :: LinkStat(ETH me:eth, SIZE 148 );
metric :: HopcountMetric();

nb :: DSDVRouteTable(60000, 15000, 7500, 1000,
                 me:eth, me:ip,
                 MAX_HOPS 100,
                 METRIC metric,
                 VERBOSE false
```

```
                  );

grid_demux :: Classifier(19/03,     // encapsulated (data) packets
                   19/02);   // route advertisement packets

arp_demux :: Classifier(12/0806 20/0001, // arp queries
                  12/0800);        // IP packets

// handles IP packets with no extra encapsulation
ip_demux :: IPClassifier(dst host me,     // ip for us
                   dst net me/24); // ip for Grid network

// handles IP packets with Grid data encapsulation
grid_data_demux :: IPClassifier(dst host me,     // ip for us
                        dst net me/24); // ip for Grid network

// dev0
dev0 :: ToGridDev(wlan1);
from_dev0 :: FromGridDev(wlan1, me:eth)
from_dev0 [0] -> Paint(0) -> grid_demux
from_dev0 [1] -> Paint(0) -> probe_cl :: Classifier(12/7ffe, 12/7ffd);

probe_cl [0] -> ls ->  probe_switch :: Switch(-1) -> dev0;
probe_cl [1] -> ls2 -> probe_switch;

// support for traceroute
dec_ip_ttl :: TTLChecker -> dev0;
dec_ip_ttl [1] -> ip_demux;

grid_demux [0] -> CheckIPHeader( , 82) -> grid_data_demux;
grid_demux [1] -> nb -> dev0;

ip_input :: CheckIPHeader -> GetIPAddress(16) -> ip_demux;

to_host_encap :: KernelTun(me/24, HEADROOM 68, MTU 1432) -> ip_input;

from_dev0 [2] -> Discard;

ip_demux [0] -> to_host_encap;  // loopback packet sent by us, required
on BSD userlevel
ip_demux [1] -> GridEncap(me:eth, me:ip) -> dec_ip_ttl;   // forward
packet sent by us

grid_data_demux [0] -> Strip(82) -> to_host_encap;  // receive packet
from net for us
grid_data_demux [1] -> dec_ip_ttl;                                //
forward packet from net for someone else

// UDP packet generator
load :: GridLoad -> ip_input;
```

Appendix C: DSR *Click* Configuration Code

Clicky generated Directed Graph

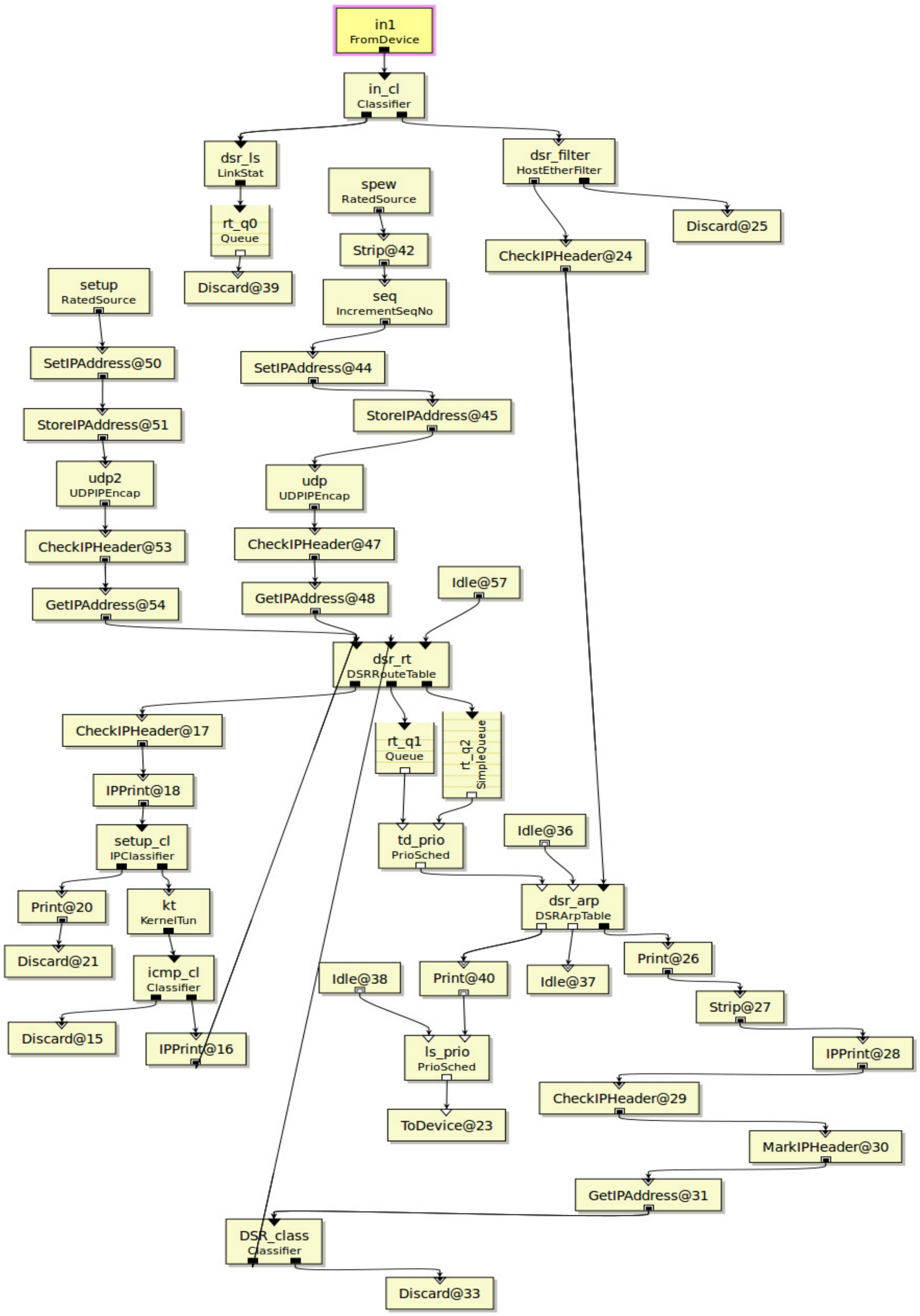


Figure C-1. DSR directed graph – generated through Clicky

Code for the Physical Nodes

All code comments are in blue font while *Click* code remains in standard black font.

```
AddressInfo(me0 10.0.0.1);
AddressInfo(my_ether 00:20:a6:61:58:a5);

rt_q2 :: SimpleQueue(10); // just ahead of todevice

dsr_ls :: LinkStat(ETH my_ether, SIZE 148) -> rt_q0 :: Queue(5);

dsr_lt :: LinkTable(IP me0);
dsr_rt :: DSRRouteTable(me0, dsr_lt, OUTQUEUE rt_q2, USE_BLACKLIST 1);

dsr_arp :: DSRArpTable(me0, my_ether);

in1 :: FromDevice(wlan1, PROMISC 0);

dsr_filter :: HostEtherFilter(my_ether,1);

in_cl :: Classifier(12/7FFF, -);
in_cl[0] -> dsr_ls;

// [1]dsr_arp takes incoming packets, and passes them through
// unchanged to output 1, adding entries to an ARP table
in1 -> in_cl;
in_cl[1] -> dsr_filter; // non-probes

// packets destined for this host
kt :: KernelTun(me0/24);

kt -> icmp_cl :: Classifier(20/0302, -);

icmp_cl[0] -> Discard; // icmp 'protocol not supported'
icmp_cl[1] -> IPPrint(0rt, CONTENTS true, NBYTES 128) ->
[0]dsr_rt[0] -> CheckIPHeader ->
                IPPrint(rt0, CONTENTS true, NBYTES 128) ->
                setup_cl :: IPClassifier(udp port 8022, -);

setup_cl[0] -> Print(setup) -> Discard;
setup_cl[1] -> kt;

ls_prio :: PrioSched -> ToDevice(wlan1);

dsr_filter[0] -> CheckIPHeader(14) -> [2]dsr_arp;

// drop packets with my ethernet source address
dsr_filter[1] -> // Print(Mine) ->
                 Discard;

dsr_arp[2] -> Print(_in, NBYTES 192) ->
              Strip(14) ->
              IPPrint ->
              CheckIPHeader() ->
              MarkIPHeader() ->
              GetIPAddress(16) ->
              DSR_class :: Classifier(09/C8, -);  // DSR packets

DSR_class[0] -> // Print(DSR) ->
                [1]dsr_rt;
```

```
DSR_class[1] -> // Print(Other) ->
                Discard;

// packets to send out on the wireless; dsr_arp puts on the ethernet
// header based on its ARP table
td_prio :: PrioSched;
dsr_rt[1] -> rt_q1 :: Queue(20) -> [0]td_prio;
dsr_rt[2] -> rt_q2 -> [1]td_prio;
td_prio -> [0]dsr_arp;

Idle -> [1] dsr_arp [1] -> Idle;

Idle -> [0]ls_prio;
rt_q0 -> Discard;

dsr_arp[0] -> Print(out, NBYTES 192) ->
              [1]ls_prio;

// packet spewer for throughput tests.
spew :: RatedSource(ACTIVE false, RATE 700, DATA
'00000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000')
        -> Strip(42) // 14 + 20 + 8 = 42
                     // (eth + ip + udp)
        -> seq :: IncrementSeqNo(FIRST 0, OFFSET 0)
        -> SetIPAddress(me0)
        -> StoreIPAddress(4)
        -> udp :: UDPIPEncap(me0, 1111, 0.0.0.0, 8021)
        -> CheckIPHeader
        -> GetIPAddress(16)
        -> [0]dsr_rt;

// setup the DSR source route
setup :: RatedSource(ACTIVE false, RATE 1, DATA 'xxx')
        -> SetIPAddress(me0)
        -> StoreIPAddress(4)
        -> udp2 :: UDPIPEncap(me0, 1111, 0.0.0.0, 8022)
        -> CheckIPHeader
        -> GetIPAddress(16)
        -> [0]dsr_rt;

Idle -> [2]dsr_rt;
```

Appendix D: NS-2 Simulation scenarios (TCL scripts)

These scripts were created by combining previous written code with the in-house developed code. This code has not been extensively tested to be “bug-free” and may cause issues with other systems. It is provided “AS IS” and no guarantees are expressed or implied with it. DSDVegg.tcl is the only scenario file listed in this appendix that details the code needed to run the simulations that generated the traffic results. All other simulations that generated traffic results were very similar to this script and only deviated per the settings discussed in Chapter Four. DSDVapproxy.tcl, Initialize.tcl, WriteOuptut.tcl, and Simulate.tcl are the only scenario files listed in this appendix that detail the code used to generate the propagation results. All other simulations that generated propagation results were very similar to these scripts and only deviated per the settings discussed in Chapter Four. Comments are in blue font for all scripts.

TCPegg.tcl

```
# Copyright (c) 1997 Regents of the University of California.
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions
# are met:
# 1. Redistributions of source code must retain the above copyright
#    notice, this list of conditions and the following disclaimer.
# 2. Redistributions in binary form must reproduce the above copyright
#    notice, this list of conditions and the following disclaimer in
the
#    documentation and/or other materials provided with the
distribution.
# 3. All advertising materials mentioning features or use of this
software
#    must display the following acknowledgement:
#      This product includes software developed by the Computer Systems
#      Engineering Group at Lawrence Berkeley Laboratory.
# 4. Neither the name of the University nor of the Laboratory may be
used
#    to endorse or promote products derived from this software without
#    specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS''
AND
# ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
# ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE
LIABLE
# FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
# DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS
# OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY
# OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
OF
# SUCH DAMAGE.

#
======================================================================
# Define options
#
======================================================================
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(netif)          Phy/WirelessPhy            ;# network interface
type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue
type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         500                        ;# max packet in ifq
set val(nn)             20                         ;# number of
mobilenodes
set val(rp)             DSDV                       ;# routing protocol
#
======================================================================
# Main Program
#
======================================================================

#
# Initialize Global Variables
#
set ns_          [new Simulator]
set stoptime     32

# set up topography object
set xsize 120
set ysize 120
set topo      [new Topography] $topo load_flatgrid $xsize $ysize

#
# Create and activate trace files.
#
set tracefd     [open "dsdvegg.tr" w]
set namtrace      [open "dsdvegg.nam" w]
$ns_ trace-all $tracefd
```

```
$ns_ namtrace-all-wireless $namtrace $xsize $ysize

#
# Create God
#
set god_ [create-god $val(nn)]

#
#set up signal propagation
#
$val(mac) set dataRate_ 11Mb
$val(mac) set basicRate_ 1Mb

Propagation/APProxy set minRandomGain_ 0;
Propagation/APProxy set maxRandomGain_ 0;
Propagation/APProxy set maxGain_ 10.0;
Propagation/APProxy set angleNumber_ 6;

$val(netif) set Pt_ 0.2818
$val(netif) set CPThresh_ 10.0
$val(netif) set CSThresh_ 5.011872e-12
$val(netif) set RXThresh_ 5.82587e-09
$val(ant) set Gt_ 1.0
$val(ant) set Gr_ 1.0
$val(netif) set freq_ 2.412e9
$val(netif) set L_ 1.0
Propagation/Shadowing set pathlossExp_ 2.0
Propagation/Shadowing set std_db_ 2.8
Propagation/Shadowing set dist0_ 1.0

set opt(propInstance) [new Propagation/APProxy]
$opt(propInstance) propagation [new Propagation/Shadowing]

$opt(propInstance) profile [expr [expr 31*31-1] /2] 0 1 0 0 .5 0

#
#  Create the specified number of mobilenodes [$val(nn)] and "attach"
them
#  to the channel.

# configure node

        $ns_ node-config -adhocRouting $val(rp) \
                    -llType $val(ll) \
                    -macType $val(mac) \
                    -ifqType $val(ifq) \
                    -ifqLen $val(ifqlen) \
                    -antType $val(ant) \
                    -propInstance $opt(propInstance) \
                    -phyType $val(netif) \
                    -channelType $val(chan) \
                    -topoInstance $topo \
                    -agentTrace ON \
                    -routerTrace ON \
                    -macTrace OFF \
                    -movementTrace OFF
```

```
      for {set i 0} {$i < $val(nn) } {incr i} {
            set node_($i) [$ns_ node]
            $ns_ initial_node_pos $node_($i) 5  ; # size the nodes for
nam
            $node_($i) random-motion 0          ; # disable random
motion
      }

#
# Provide (X,Y,Z=0) coordinates for nodes
#
$node_(0) set X_ 60.0
$node_(0) set Y_ 60.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 72
$node_(1) set Y_ 117
$node_(1) set Z_ 0.0

#189.1
$ns_ at 0.1 "$node_(0) setdest 60 60 0"
$ns_ at 0.1 "$node_(1) setdest 72 117 0"

#186.8
$ns_ at 0.1 "$node_(2) setdest 82 113 500"
#183.7
$ns_ at 0.1 "$node_(3) setdest 92 107 500"
#170.6
$ns_ at 0.1 "$node_(4) setdest 99 96 500"
#171.9
$ns_ at 0.1 "$node_(5) setdest 104 90 500"
#161
$ns_ at 0.1 "$node_(6) setdest 107 77 500"
#139
$ns_ at 0.1 "$node_(7) setdest 103 61 500"
#130.4
$ns_ at 0.1 "$node_(8) setdest 98 47 500"
#134.1
$ns_ at 0.1 "$node_(9) setdest 91 33 500"
#134.9
$ns_ at 0.1 "$node_(10) setdest 76 23 500"
#160.1
$ns_ at 0.1 "$node_(11) setdest 61 12 500"
#167.4
$ns_ at 0.1 "$node_(12) setdest 48 11 500"
#176.1
$ns_ at 0.1 "$node_(13) setdest 32 15 500"
#172.6
$ns_ at 0.1 "$node_(14) setdest 18 30 500"
#155.1
$ns_ at 0.1 "$node_(15) setdest 15 51 500"
#138.4
$ns_ at 0.1 "$node_(16) setdest 21 73 500"
#131
$ns_ at 0.1 "$node_(17) setdest 29 85 500"
#132.9
$ns_ at 0.1 "$node_(18) setdest 40 96 500"
```

```
#154.7
$ns_ at 0.1 "$node_(19) setdest 52 107 500"

set startxmittime 10

set tcp [new Agent/TCP]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp
$ns_ attach-agent $node_(1) $sink
$ns_ connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns_ at [expr $startxmittime + 0.1] "$ftp start"
$ns_ at [expr $startxmittime + 1.0] "$ftp stop"

set tcp2 [new Agent/TCP]
$tcp2 set class_ 2
set sink2 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp2
$ns_ attach-agent $node_(2) $sink2
$ns_ connect $tcp2 $sink2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ns_ at [expr $startxmittime + 1.1] "$ftp2 start"
$ns_ at [expr $startxmittime + 2.0] "$ftp2 stop"

set tcp3 [new Agent/TCP]
$tcp3 set class_ 2
set sink3 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp3
$ns_ attach-agent $node_(3) $sink3
$ns_ connect $tcp3 $sink3
set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3
$ns_ at [expr $startxmittime + 2.1] "$ftp3 start"
$ns_ at [expr $startxmittime + 3.0] "$ftp3 stop"

set tcp4 [new Agent/TCP]
$tcp4 set class_ 2
set sink4 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp4
$ns_ attach-agent $node_(4) $sink4
$ns_ connect $tcp4 $sink4
set ftp4 [new Application/FTP]
$ftp4 attach-agent $tcp4
$ns_ at [expr $startxmittime + 3.1] "$ftp4 start"
$ns_ at [expr $startxmittime + 4.0] "$ftp4 stop"

set tcp5 [new Agent/TCP]
$tcp5 set class_ 2
set sink5 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp5
$ns_ attach-agent $node_(5) $sink5
$ns_ connect $tcp5 $sink5
set ftp5 [new Application/FTP]
$ftp5 attach-agent $tcp5
```

```
$ns_ at [expr $startxmittime + 4.1] "$ftp5 start"
$ns_ at [expr $startxmittime + 5.0] "$ftp5 stop"

set tcp6 [new Agent/TCP]
$tcp6 set class_ 2
set sink6 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp6
$ns_ attach-agent $node_(6) $sink6
$ns_ connect $tcp6 $sink6
set ftp6 [new Application/FTP]
$ftp6 attach-agent $tcp6
$ns_ at [expr $startxmittime + 5.1] "$ftp6 start"
$ns_ at [expr $startxmittime + 6.0] "$ftp6 stop"

set tcp7 [new Agent/TCP]
$tcp7 set class_ 2
set sink7 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp7
$ns_ attach-agent $node_(7) $sink7
$ns_ connect $tcp7 $sink7
set ftp7 [new Application/FTP]
$ftp7 attach-agent $tcp7
$ns_ at [expr $startxmittime + 6.1] "$ftp7 start"
$ns_ at [expr $startxmittime + 7.0] "$ftp7 stop"

set tcp8 [new Agent/TCP]
$tcp8 set class_ 2
set sink8 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp8
$ns_ attach-agent $node_(8) $sink8
$ns_ connect $tcp8 $sink8
set ftp8 [new Application/FTP]
$ftp8 attach-agent $tcp8
$ns_ at [expr $startxmittime + 7.1] "$ftp8 start"
$ns_ at [expr $startxmittime + 8.0] "$ftp8 stop"

set tcp9 [new Agent/TCP]
$tcp9 set class_ 2
set sink9 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp9
$ns_ attach-agent $node_(9) $sink9
$ns_ connect $tcp9 $sink9
set ftp9 [new Application/FTP]
$ftp9 attach-agent $tcp9
$ns_ at [expr $startxmittime + 8.1] "$ftp9 start"
$ns_ at [expr $startxmittime + 9.0] "$ftp9 stop"

set tcp10 [new Agent/TCP]
$tcp10 set class_ 2
set sink10 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp10
$ns_ attach-agent $node_(10) $sink10
$ns_ connect $tcp10 $sink10
set ftp10 [new Application/FTP]
$ftp10 attach-agent $tcp10
$ns_ at [expr $startxmittime + 9.1] "$ftp10 start"
$ns_ at [expr $startxmittime + 10.0] "$ftp10 stop"
```

```
set tcp11 [new Agent/TCP]
$tcp11 set class_ 2
set sink11 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp11
$ns_ attach-agent $node_(11) $sink11
$ns_ connect $tcp11 $sink11
set ftp11 [new Application/FTP]
$ftp11 attach-agent $tcp11
$ns_ at [expr $startxmittime + 10.1] "$ftp11 start"
$ns_ at [expr $startxmittime + 11.0] "$ftp11 stop"

set tcp12 [new Agent/TCP]
$tcp12 set class_ 2
set sink12 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp12
$ns_ attach-agent $node_(12) $sink12
$ns_ connect $tcp12 $sink12
set ftp12 [new Application/FTP]
$ftp12 attach-agent $tcp12
$ns_ at [expr $startxmittime + 11.1] "$ftp12 start"
$ns_ at [expr $startxmittime + 12.0] "$ftp12 stop"

set tcp13 [new Agent/TCP]
$tcp13 set class_ 2
set sink13 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp13
$ns_ attach-agent $node_(13) $sink13
$ns_ connect $tcp13 $sink13
set ftp13 [new Application/FTP]
$ftp13 attach-agent $tcp13
$ns_ at [expr $startxmittime + 12.1] "$ftp13 start"
$ns_ at [expr $startxmittime + 13.0] "$ftp13 stop"

set tcp14 [new Agent/TCP]
$tcp14 set class_ 2
set sink14 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp14
$ns_ attach-agent $node_(14) $sink14
$ns_ connect $tcp14 $sink14
set ftp14 [new Application/FTP]
$ftp14 attach-agent $tcp14
$ns_ at [expr $startxmittime + 13.1] "$ftp14 start"
$ns_ at [expr $startxmittime + 14.0] "$ftp14 stop"

set tcp15 [new Agent/TCP]
$tcp15 set class_ 2
set sink15 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp15
$ns_ attach-agent $node_(15) $sink15
$ns_ connect $tcp15 $sink15
set ftp15 [new Application/FTP]
$ftp15 attach-agent $tcp15
$ns_ at [expr $startxmittime + 14.1] "$ftp15 start"
$ns_ at [expr $startxmittime + 15.0] "$ftp15 stop"

set tcp16 [new Agent/TCP]
```

```
$tcp16 set class_ 2
set sink16 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp16
$ns_ attach-agent $node_(16) $sink16
$ns_ connect $tcp16 $sink16
set ftp16 [new Application/FTP]
$ftp16 attach-agent $tcp16
$ns_ at [expr $startxmittime + 15.1] "$ftp16 start"
$ns_ at [expr $startxmittime + 16.0] "$ftp16 stop"

set tcp17 [new Agent/TCP]
$tcp17 set class_ 2
set sink17 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp17
$ns_ attach-agent $node_(17) $sink17
$ns_ connect $tcp17 $sink17
set ftp17 [new Application/FTP]
$ftp17 attach-agent $tcp17
$ns_ at [expr $startxmittime + 16.1] "$ftp17 start"
$ns_ at [expr $startxmittime + 17.0] "$ftp17 stop"

set tcp18 [new Agent/TCP]
$tcp18 set class_ 2
set sink18 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp18
$ns_ attach-agent $node_(18) $sink18
$ns_ connect $tcp18 $sink18
set ftp18 [new Application/FTP]
$ftp18 attach-agent $tcp18
$ns_ at [expr $startxmittime + 17.1] "$ftp18 start"
$ns_ at [expr $startxmittime + 18.0] "$ftp18 stop"

set tcp19 [new Agent/TCP]
$tcp19 set class_ 2
set sink19 [new Agent/TCPSink]
$ns_ attach-agent $node_(0) $tcp19
$ns_ attach-agent $node_(19) $sink19
$ns_ connect $tcp19 $sink19
set ftp19 [new Application/FTP]
$ftp19 attach-agent $tcp19
$ns_ at [expr $startxmittime + 18.1] "$ftp19 start"
$ns_ at [expr $startxmittime + 19.0] "$ftp19 stop"

#
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $stoptime.0 "$node_($i) reset";
}

proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
    global ns_ namtrace
    $ns_ flush-trace
    close $namtrace
}

$ns_ at $stoptime "stop"
$ns_ at $stoptime.01 "puts \"NS EXITING...\" ; $ns_ halt"

puts "Starting Simulation..."
$ns_ run
```

CBRegg.tcl

Due to reasons unknown, the CBR traffic could only operate on 7 nodes maximum per TCL script. Even though it is represented as one script in this appendix, in order for it to run in the NS-2 environment developed for this thesis it had to be divided into three different scripts.

```
# Copyright (c) 1997 Regents of the University of California.
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions
# are met:
# 1. Redistributions of source code must retain the above copyright
#    notice, this list of conditions and the following disclaimer.
# 2. Redistributions in binary form must reproduce the above copyright
#    notice, this list of conditions and the following disclaimer in
the
#    documentation and/or other materials provided with the
distribution.
# 3. All advertising materials mentioning features or use of this
software
#    must display the following acknowledgement:
#      This product includes software developed by the Computer Systems
#      Engineering Group at Lawrence Berkeley Laboratory.
# 4. Neither the name of the University nor of the Laboratory may be
used
#    to endorse or promote products derived from this software without
#    specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS''
AND
# ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
# IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE
# ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE
LIABLE
# FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL
# DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS
# OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
# HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
STRICT
```

```
# LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY
WAY
# OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY
OF
# SUCH DAMAGE.

#
======================================================================
# Define options
#
======================================================================
set val(chan)           Channel/WirelessChannel    ;# channel type
set val(prop)           Propagation/TwoRayGround   ;# radio-propagation
model
set val(netif)          Phy/WirelessPhy            ;# network interface
type
set val(mac)            Mac/802_11                 ;# MAC type
set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue
type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)         50                         ;# max packet in ifq
set val(nn)             20                         ;# number of
mobilenodes
set val(rp)             DSDV                       ;# routing protocol
#
======================================================================
# Main Program
#
======================================================================

#
# Initialize Global Variables
#
set ns_           [new Simulator]
set stoptime      32

# set up topography object
set xsize 120
set ysize 120
set topo       [new Topography] $topo load_flatgrid $xsize $ysize

#
# Create and activate trace files.
#
set tracefd     [open "dsdvegg3.tr" w]
set namtrace      [open "dsdvegg3.nam" w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $xsize $ysize

#
# Create God
#
set god_ [create-god $val(nn)]

$val(mac) set dataRate_ 11Mb
$val(mac) set basicRate_ 1Mb
```

```
Propagation/APProxy set minRandomGain_ 0;
Propagation/APProxy set maxRandomGain_ 0;
Propagation/APProxy set maxGain_ 10.0;
Propagation/APProxy set angleNumber_ 6;

$val(netif) set Pt_ .2018
$val(netif) set CPThresh_ 10.0
$val(netif) set CSThresh_ 5.011872e-12
$val(netif) set RXThresh_ 5.82587e-09
$val(ant) set Gt_ 1.0
$val(ant) set Gr_ 1.0
$val(netif) set freq_ 2.412e9
$val(netif) set L_ 1.0

set opt(propInstance) [new Propagation/APProxy]
$opt(propInstance) propagation [new Propagation/TwoRayGround]

$opt(propInstance) profile [expr [expr 31*31-1] /2] 0 1 0 0 .5 0

#
#  Create the specified number of mobilenodes [$val(nn)] and "attach"
them
#  to the channel.

# configure node

        $ns_ node-config -adhocRouting $val(rp) \
                   -llType $val(ll) \
                   -macType $val(mac) \
                   -ifqType $val(ifq) \
                   -ifqLen $val(ifqlen) \
                   -antType $val(ant) \
                   -propInstance $opt(propInstance) \
                   -phyType $val(netif) \
                   -channelType $val(chan) \
                   -topoInstance $topo \
                   -agentTrace ON \
                   -routerTrace ON \
                   -macTrace OFF \
                   -movementTrace OFF

      for {set i 0} {$i < $val(nn) } {incr i} {
            set node_($i) [$ns_ node]
            $ns_ initial_node_pos $node_($i) 5  ; # size the nodes for
nam
            $node_($i) random-motion 0          ; # disable random
motion
      }

#
# Provide initial (X,Y, for now Z=0) coordinates for mobilenodes
#
$node_(0) set X_ 60.0
$node_(0) set Y_ 60.0
$node_(0) set Z_ 0.0
```

```
$node_(1) set X_ 72
$node_(1) set Y_ 117
$node_(1) set Z_ 0.0

#189.1
$ns_ at 0.1 "$node_(0) setdest 60 60 0"
$ns_ at 0.1 "$node_(1) setdest 72 117 0"

#186.8
$ns_ at 0.1 "$node_(2) setdest 82 113 500"
#183.7
$ns_ at 0.1 "$node_(3) setdest 92 107 500"
#170.6
$ns_ at 0.1 "$node_(4) setdest 99 96 500"
#171.9
$ns_ at 0.1 "$node_(5) setdest 104 90 500"
#161
$ns_ at 0.1 "$node_(6) setdest 107 77 500"
#139
$ns_ at 0.1 "$node_(7) setdest 103 61 500"
#130.4
$ns_ at 0.1 "$node_(8) setdest 98 47 500"
#134.1
$ns_ at 0.1 "$node_(9) setdest 91 33 500"
#134.9
$ns_ at 0.1 "$node_(10) setdest 76 23 500"
#160.1
$ns_ at 0.1 "$node_(11) setdest 61 12 500"
#167.4
$ns_ at 0.1 "$node_(12) setdest 48 11 500"
#176.1
$ns_ at 0.1 "$node_(13) setdest 32 15 500"
#172.6
$ns_ at 0.1 "$node_(14) setdest 18 30 500"
#155.1
$ns_ at 0.1 "$node_(15) setdest 15 51 500"
#138.4
$ns_ at 0.1 "$node_(16) setdest 21 73 500"
#131
$ns_ at 0.1 "$node_(17) setdest 29 85 500"
#132.9
$ns_ at 0.1 "$node_(18) setdest 40 96 500"
#154.7
$ns_ at 0.1 "$node_(19) setdest 52 107 500"


set startxmittime 10

#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
```

```
set null0 [new Agent/Null]
$ns_ attach-agent $node_(1) $null0
$ns_ connect $udp0 $null0
$ns_ at [expr $startxmittime + 0.1] "$cbr0 start"
$ns_ at [expr $startxmittime + 1.0] "$cbr0 stop"

#Create a UDP agent and attach it to node n0
set udp1 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp1

# Create a CBR traffic source and attach it to udp0
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

set null1 [new Agent/Null]
$ns_ attach-agent $node_(2) $null1
$ns_ connect $udp1 $null1
$ns_ at [expr $startxmittime + 1.1] "$cbr1 start"
$ns_ at [expr $startxmittime + 2.0] "$cbr1 stop"

#Create a UDP agent and attach it to node n0
set udp2 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp2

# Create a CBR traffic source and attach it to udp0
set cbr2 [new Application/Traffic/CBR]
$cbr2 set packetSize_ 500
$cbr2 set interval_ 0.005
$cbr2 attach-agent $udp2

set null2 [new Agent/Null]
$ns_ attach-agent $node_(3) $null2
$ns_ connect $udp2 $null2
$ns_ at [expr $startxmittime + 2.1] "$cbr2 start"
$ns_ at [expr $startxmittime + 3.0] "$cbr2 stop"

#Create a UDP agent and attach it to node n0
set udp3 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp3

# Create a CBR traffic source and attach it to udp0
set cbr3 [new Application/Traffic/CBR]
$cbr3 set packetSize_ 500
$cbr3 set interval_ 0.005
$cbr3 attach-agent $udp3

set null3 [new Agent/Null]
$ns_ attach-agent $node_(4) $null3
$ns_ connect $udp3 $null3
$ns_ at [expr $startxmittime + 3.1] "$cbr3 start"
$ns_ at [expr $startxmittime + 4.0] "$cbr3 stop"

#Create a UDP agent and attach it to node n0
set udp4 [new Agent/UDP]
```

```
$ns_ attach-agent $node_(0) $udp4

# Create a CBR traffic source and attach it to udp0
set cbr4 [new Application/Traffic/CBR]
$cbr4 set packetSize_ 500
$cbr4 set interval_ 0.005
$cbr4 attach-agent $udp4

set null4 [new Agent/Null]
$ns_ attach-agent $node_(5) $null4
$ns_ connect $udp4 $null4
$ns_ at [expr $startxmittime + 4.1] "$cbr4 start"
$ns_ at [expr $startxmittime + 5.0] "$cbr4 stop"

#Create a UDP agent and attach it to node n0
set udp5 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp5

# Create a CBR traffic source and attach it to udp0
set cbr5 [new Application/Traffic/CBR]
$cbr5 set packetSize_ 500
$cbr5 set interval_ 0.005
$cbr5 attach-agent $udp5

set null5 [new Agent/Null]
$ns_ attach-agent $node_(6) $null5
$ns_ connect $udp5 $null5
$ns_ at [expr $startxmittime + 5.1] "$cbr5 start"
$ns_ at [expr $startxmittime + 6.0] "$cbr5 stop"

#Create a UDP agent and attach it to node n0
set udp6 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp6

# Create a CBR traffic source and attach it to udp0
set cbr6 [new Application/Traffic/CBR]
$cbr6 set packetSize_ 500
$cbr6 set interval_ 0.005
$cbr6 attach-agent $udp6

set null6 [new Agent/Null]
$ns_ attach-agent $node_(7) $null6
$ns_ connect $udp6 $null6
$ns_ at [expr $startxmittime + 6.1] "$cbr6 start"
$ns_ at [expr $startxmittime + 7.0] "$cbr6 stop"

#Create a UDP agent and attach it to node n0
set udp7 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp7

# Create a CBR traffic source and attach it to udp0
set cbr7 [new Application/Traffic/CBR]
$cbr7 set packetSize_ 500
$cbr7 set interval_ 0.005
$cbr7 attach-agent $udp7

set null7 [new Agent/Null]
```

```
$ns_ attach-agent $node_(8) $null7
$ns_ connect $udp7 $null7
$ns_ at [expr $startxmittime + 7.1] "$cbr7 start"
$ns_ at [expr $startxmittime + 8.0] "$cbr7 stop"

#Create a UDP agent and attach it to node n0
set udp8 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp8

# Create a CBR traffic source and attach it to udp0
set cbr8 [new Application/Traffic/CBR]
$cbr8 set packetSize_ 500
$cbr8 set interval_ 0.005
$cbr8 attach-agent $udp8

set null8 [new Agent/Null]
$ns_ attach-agent $node_(9) $null8
$ns_ connect $udp8 $null8
$ns_ at [expr $startxmittime + 8.1] "$cbr8 start"
$ns_ at [expr $startxmittime + 9.0] "$cbr8 stop"

#Create a UDP agent and attach it to node n0
set udp9 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp9

# Create a CBR traffic source and attach it to udp0
set cbr9 [new Application/Traffic/CBR]
$cbr9 set packetSize_ 500
$cbr9 set interval_ 0.005
$cbr9 attach-agent $udp9

set null9 [new Agent/Null]
$ns_ attach-agent $node_(10) $null9
$ns_ connect $udp9 $null9
$ns_ at [expr $startxmittime + 9.1] "$cbr9 start"
$ns_ at [expr $startxmittime + 10.0] "$cbr9 stop"

#Create a UDP agent and attach it to node n0
set udp10 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp10

# Create a CBR traffic source and attach it to udp0
set cbr10 [new Application/Traffic/CBR]
$cbr10 set packetSize_ 500
$cbr10 set interval_ 0.005
$cbr10 attach-agent $udp10

set null10 [new Agent/Null]
$ns_ attach-agent $node_(11) $null10
$ns_ connect $udp10 $null10
$ns_ at [expr $startxmittime + 10.1] "$cbr10 start"
$ns_ at [expr $startxmittime + 11.0] "$cbr10 stop"

#Create a UDP agent and attach it to node n0
set udp11 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp11
```

```
# Create a CBR traffic source and attach it to udp0
set cbr11 [new Application/Traffic/CBR]
$cbr11 set packetSize_ 500
$cbr11 set interval_ 0.005
$cbr11 attach-agent $udp11

set null11 [new Agent/Null]
$ns_ attach-agent $node_(12) $null11
$ns_ connect $udp11 $null11
$ns_ at [expr $startxmittime + 11.1] "$cbr11 start"
$ns_ at [expr $startxmittime + 12.0] "$cbr11 stop"

#Create a UDP agent and attach it to node n0
set udp12 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp12

# Create a CBR traffic source and attach it to udp0
set cbr12 [new Application/Traffic/CBR]
$cbr12 set packetSize_ 500
$cbr12 set interval_ 0.005
$cbr12 attach-agent $udp12

set null12 [new Agent/Null]
$ns_ attach-agent $node_(13) $null12
$ns_ connect $udp12 $null12
$ns_ at [expr $startxmittime + 12.1] "$cbr12 start"
$ns_ at [expr $startxmittime + 13.0] "$cbr12 stop"

#Create a UDP agent and attach it to node n0
set udp13 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp13

# Create a CBR traffic source and attach it to udp0
set cbr13 [new Application/Traffic/CBR]
$cbr13 set packetSize_ 500
$cbr13 set interval_ 0.005
$cbr13 attach-agent $udp13

set null13 [new Agent/Null]
$ns_ attach-agent $node_(14) $null13
$ns_ connect $udp13 $null13
$ns_ at [expr $startxmittime + 13.1] "$cbr13 start"
$ns_ at [expr $startxmittime + 14.0] "$cbr13 stop"

#Create a UDP agent and attach it to node n0
set udp14 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp14

# Create a CBR traffic source and attach it to udp0
set cbr14 [new Application/Traffic/CBR]
$cbr14 set packetSize_ 500
$cbr14 set interval_ 0.005
$cbr14 attach-agent $udp14

set null14 [new Agent/Null]
$ns_ attach-agent $node_(15) $null14
$ns_ connect $udp14 $null14
```

```
$ns_ at [expr $startxmittime + 14.1] "$cbr14 start"
$ns_ at [expr $startxmittime + 15.0] "$cbr14 stop"

#Create a UDP agent and attach it to node n0
set udp15 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp15

# Create a CBR traffic source and attach it to udp0
set cbr15 [new Application/Traffic/CBR]
$cbr15 set packetSize_ 500
$cbr15 set interval_ 0.005
$cbr15 attach-agent $udp15

set null15 [new Agent/Null]
$ns_ attach-agent $node_(16) $null15
$ns_ connect $udp15 $null15
$ns_ at [expr $startxmittime + 15.1] "$cbr15 start"
$ns_ at [expr $startxmittime + 16.0] "$cbr15 stop"

#Create a UDP agent and attach it to node n0
set udp16 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp16

# Create a CBR traffic source and attach it to udp0
set cbr16 [new Application/Traffic/CBR]
$cbr16 set packetSize_ 500
$cbr16 set interval_ 0.005
$cbr16 attach-agent $udp16

set null16 [new Agent/Null]
$ns_ attach-agent $node_(17) $null16
$ns_ connect $udp16 $null16
$ns_ at [expr $startxmittime + 16.1] "$cbr16 start"
$ns_ at [expr $startxmittime + 17.0] "$cbr16 stop"

#Create a UDP agent and attach it to node n0
set udp17 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp17

# Create a CBR traffic source and attach it to udp0
set cbr17 [new Application/Traffic/CBR]
$cbr17 set packetSize_ 500
$cbr17 set interval_ 0.005
$cbr17 attach-agent $udp17

set null17 [new Agent/Null]
$ns_ attach-agent $node_(18) $null17
$ns_ connect $udp17 $null17
$ns_ at [expr $startxmittime + 17.1] "$cbr17 start"
$ns_ at [expr $startxmittime + 18.0] "$cbr17 stop"

#Create a UDP agent and attach it to node n0
set udp18 [new Agent/UDP]
$ns_ attach-agent $node_(0) $udp18

# Create a CBR traffic source and attach it to udp0
set cbr18 [new Application/Traffic/CBR]
```

```
$cbr18 set packetSize_ 500
$cbr18 set interval_ 0.005
$cbr18 attach-agent $udp18

set null18 [new Agent/Null]
$ns_ attach-agent $node_(19) $null18
$ns_ connect $udp18 $null18
$ns_ at [expr $startxmittime + 18.1] "$cbr18 start"
$ns_ at [expr $startxmittime + 19.0] "$cbr18 stop"

#
# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at $stoptime.0 "$node_($i) reset";
}

proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
    global ns_ namtrace
    $ns_ flush-trace
    close $namtrace
}

$ns_ at $stoptime "stop"
$ns_ at $stoptime.01 "puts \"NS EXITING...\" ; $ns_ halt"

puts "Starting Simulation..."
$ns_ run
```

APProxy.tcl

```
# customize simulation
set gridX   31; # number of nodes in a horizontal line
set gridY   31; # number of nodes in a vertical line
set numberOfPackets     2000; # packets used for sending
set opt(width)          200;  # topography width
set opt(height)         200;  # topography height

Propagation/APProxy set minRandomGain_ 0;
Propagation/APProxy set maxRandomGain_ 0;
Propagation/APProxy set maxGain_ 10.0;
Propagation/APProxy set angleNumber_ 6;

# parameters for shadowing
Phy/WirelessPhy set Pt_ 0.15
Phy/WirelessPhy set CPThresh_ 10.0
Phy/WirelessPhy set CSThresh_ 5.011872e-12
Antenna/OmniAntenna set Gt_ 1.0
Antenna/OmniAntenna set Gr_ 1.0
Phy/WirelessPhy set freq_ 2.412e9
Phy/WirelessPhy set L_ 1.0
Propagation/Shadowing set pathlossExp_ 2.0
Propagation/Shadowing set std_db_ 2.8
Propagation/Shadowing set dist0_ 1.0
```

```
Phy/WirelessPhy set RXThresh_ 3.3e-8

# set approxy propagation model
set opt(propInstance) [new Propagation/APProxy]
$opt(propInstance) propagation [new Propagation/Shadowing]

$opt(propInstance) profile [expr [expr $gridX*$gridY-1] /2] 0 1 0 0 .5
0
```

Initialize.tcl

```
set ns_     [new Simulator]
set chan_   [new $opt(chan)]
set topo_   [new Topography]

#set basic packet tracing
set tracefd     [open "trace.tr" w]
set namtrace      [open "dsdvapproxy.nam" w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(width) $opt(height)

#create god object
set god_    [create-god $opt(numOfNodes)]

# create topology object
$topo_ load_flatgrid $opt(width) $opt(height)

# general node configuration
if { [info exists opt(propInstance)] } {
      $ns_ node-config -propInstance $opt(propInstance)
} else {
      $ns_ node-config -propType $opt(prop)
}

$ns_ node-config \
      -adhocRouting $opt(routing) \
      -llType $opt(linkLayer) \
      -macType $opt(macLayer) \
      -ifqType $opt(ifq) \
      -ifqLen $opt(lenIfq) \
      -antType $opt(ant) \
      -phyType $opt(netif) \
      -topoInstance $topo_ \
      -channel $chan_ \
      -agentTrace ON \
      -routerTrace ON \
      -macTrace OFF \
      -movementTrace OFF

for {set i 0} {$i < $opt(numOfNodes)} {incr i} {
      # init mobile nodes
      set node_($i) [$ns_ node]

      # disable random motion
      $node_($i) random-motion 0

      # register node.
```

```
        $god_ new_node $node_($i)

        # initial node position
        $ns_ initial_node_pos $node_($i) 10

        # init agent
        set agent_($i) [new $opt(agent)]

        # attach agent to node.
        $node_($i) attach $agent_($i)
}

set dWidth [expr 1.0*$opt(width)/[expr $gridX-1]];
set dHeight [expr 1.0*$opt(height)/[expr $gridY-1]];

for {set i 0} {$i < $gridX } {incr i} {
        for {set j 0} {$j < $gridY } {incr j} {
                set nodeId [expr $i*$gridY + $j];
                $node_($nodeId) set X_ [expr $i*$dWidth];
                $node_($nodeId) set Y_ [expr $j*$dHeight];
                $node_($nodeId) set Z_ 0.000000000000;
        }
}

for {set j 0} {$j < $numberOfPackets } {incr j} {
     $ns_ at [expr 20 + $j*1] "$agent_($centerNode) send"
}
$ns_ at [expr 30 + $numberOfPackets]  "$ns_ halt"

proc stop {} {
    global ns_ tracefd
    $ns_ flush-trace
    close $tracefd
    global ns_ namtrace
    $ns_ flush-trace
    close $namtrace
}

puts "Starting Simulation..."
# start simulation
$ns_ run;
```

WriteOutput.tcl

```
set graph1   [open "graph1.tex" w]
set graph2   [open "graph2.tex" w]
set topology [open "topologySize.tex" w]

set xWidth 5;     # width of output
set yWidth [expr 1.0*$xWidth*$opt(height)/$opt(width)];
set dWidth [expr 1.0*$xWidth/$gridX];
set dHeight [expr 1.0*$yWidth/$gridY];

puts $topology "\\newcommand{\\topHeight}{-$yWidth}"
puts $topology "\\newcommand{\\topHeightLabel}{$opt(height)}"
puts $topology "\\newcommand{\\topWidthLabel}{$opt(width)}"
```

```
for {set i 0} {$i < $gridX } {incr i} {
            set nodeId [expr $i*$gridY + [expr $gridY-1]/2];
            if {$nodeId == $centerNode} { continue; }

            set packetReceived [$agent_($nodeId) set packetCount_];
            set packetRate [ expr [expr
$packetReceived*100]/$numberOfPackets ];
            puts $graph1 "[expr $i*$xWidth/[expr $gridX-1.0]] [expr
3.0*$packetRate/100]"
}

for {set i 0} {$i < $gridX } {incr i} {
      for {set j 0} {$j < $gridY } {incr j} {
            set nodeId [expr $i*$gridY +$j];
            if {$nodeId == $centerNode} { continue; }
            set packetReceived [$agent_($nodeId) set packetCount_];
            set packetRate [ expr [expr
$packetReceived*100]/$numberOfPackets ];

            set posX [expr $i*$dWidth-0.01]
            set posY [expr $j*$dHeight-$yWidth-1.01]
            set posX2 [expr $i*$dWidth+$dWidth+0.01]
            set posY2 [expr $j*$dHeight+$dHeight-$yWidth-0.99]

            puts $graph2 "\\fill \[black!$packetRate\] ($posX ,$posY)
rectangle ($posX2,$posY2);"
      }
}

close $graph1;
close $graph2;
close $topology;
```

Simulate.tcl

```
set opt(chan)           Channel/WirelessChannel; # channel layer
set opt(netif)          Phy/WirelessPhy;         # physical layer
set opt(ant)            Antenna/OmniAntenna;     # antenna model
set opt(macLayer)       Mac;                     # mac layer
set opt(ifq)            Queue/DropTail;          # message queue, ifq
set opt(lenIfq)         50;                      # max packet in ifq
set opt(linkLayer)      LL;                      # link layer
set opt(routing)        DumbAgent;               # routing protocol
set opt(agent)          Agent/NbhAgent;          # simulation agent

Agent/NbhAgent set packetCount_ 0

# topology
set gridX   31; # number of nodes in a horizontal line
set gridY   31; # number of nodes in a vertical line
set numberOfPackets     2000; # packets used for sending
set opt(width)          200;  # topography width
set opt(height)         200;  # topography height

set argArray [split $argv];
```

```
if { $argc != 1 } {
      puts "use: ns simulate.tcl <propagation file>";
} else {
      source  [lindex $argArray 0];

      set opt(numOfNodes)      [expr $gridX*$gridY];   # number of
mobilenodes
      set centerNode           [expr [expr $opt(numOfNodes)-1] /2]

      source  "initialize.tcl";
      source  "writeOutput.tcl";
}
```

Bibliography

[1] Altman, E. and Jimenez, T. "NS Simulator for beginners," *Lecture Notes 2003-2004*. University de Los Andes, Merida, Venezuela and Sophia-Antipolis, France. (Dec. 2003). URL: http://www-sop.inria.fr/members/Eitan.Altman/COURS-NS/n3.pdf.

[2] Andel, T. R. and Yasinac, A. "On the Credibility of Manet Simulations." *Computer*, vol.39, no.7, pp.48-54. (July 2006). URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1657907&isnumber=34707

[3] Bains, L. "Laptop Sales Overtake Traditional Desktop PC Sales Over Holiday," *www.Switched.com.* (Jan. 2009). URL: http://www.switched.com/2009/01/09/laptops-overtake-traditional-desktop-pcs/.

[4] Baldwin, R. CSCE 554: Performance Analysis and Design - Course slides. Graduate School of Engineering and Management. Air Force Institute of Technology, Wright-Patterson AFB OH. (Jun. 2009).

[5] BBCNews. "Mobile System Promises Free Calls," *BBC MMIX*. (Sep. 2007). URL:http://news.bbc.co.uk/2/hi/technology/6987784.stm.

[6] Bhandare, S., Doshi, S., Brown, T. X., and Sanghai, S. "Comparison of two wireless ad-hoc routing protocols on a hardware test-bed," *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE* , vol.2, no., pp.1168-1173 vol.2, 20-20 March 2003. URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1200536&isnumber=27029.

[7] Bhargaven, K., Gunter, C. A., Kim, M., Lee, I., Sokolsky, O., and Viswanathan, M. "Verisim: Formal Analysis of Network Simulations," *IEEE Transactions on Software Engineering* Vol. 28, No. 2 (Feb. 2002), 129-145. URL:http://seclab.uiuc.edu/cgunter/dist/BhargavanGKLOSV02.pdf.

[8] Braem, B. *Implementatie En Evaluatie Van Ad-hoc On-Demand Distance Vector Routing*. MS Thesis. Universiteit Antwerpen, Belgium. (2004-2005). URL:https://euterpe.cmi.ua.ac.be/~bbraem/thesis/thesis.pdf.

[9] Brown, T. X., Doshi, S., Jadhav, S., Henkel, D., and Thekkekunnel, R. "A Full Scale Wireless Ad-hoc Network Test Bed," *Proc. of International Symposium on Advanced Radio Technologies*, Boulder, CO. (Mar. 2005). URL:http://ecee.colorado.edu/~timxb/timxb/pubs/05isart.pdf.

[10] Cavilla, A. L., Baron, G., Hart, T. E., Litty, L., and de Lara, E. "Simplified Simulation Models for Indoor MANET Evaluation are not Robust," *Proc. SECON*, University of Toronto. (2004). URL: http://www.cs.toronto.edu/~delara/papers/secon2004/secon.pdf.

[11] Cavin, D., Sasson, Y., and Schiper, A. "On the Accuracy of MANET Simulators," *POCM '02*. (Oct. 2002). URL: http://portal.acm.org/citation.cfm?id=584490.584499.

[12] Chakeres, I. and Perkins, C. "Dynamic MANET On-demand (DYMO) Routing," *IETF Internet Draft*. (Feb. 2008). URL: http://ianchak.com/dymo/draft-ietf-manet-dymo-12.txt.

[13] "*Click* Programming Manual," URL: http://www.read.cs.ucla.edu/click/progman.html.

[14] Doshi, S., Bhandare, S., and Brown, T. X. "An on-demand minimum energy routing protocol for a wireless ad-hoc network," *SIGMOBILE Mob. Comput. Commun. Rev*. 6, 3 (Jun. 2002), 50-66. URL: http://doi.acm.org/10.1145/581291.581300.

[15] Dzambaski, A., Trajanov, D., Filiposka, S., and Grnarov, A. "Ad hoc networks simulations with real 3D terrains," *15th Telecommunications forum TELFOR 2007*, Serbia. (Nov. 2007). URL: http://2007.telfor.rs/files/radovi/02_10.pdf.

[16] Fall, K. and Varadhan, K. "The *ns* Manual (formerly *ns* Notes and Documentation)." *The VINT Project,* UC Berkeley, LBL, USC/ISI, and Xeror PARC. URL:http://www.isi.edu/nsnam/ns/tutorial/.

[17] Frodigh, M., Johansson, P., and Larsson, P. "Wireless *ad-hoc* networking – The art of networking without a network," *Ericsson Review*, No. 4. (2000). URL:http://www.ericsson.com/ericsson/corpinfo/publications/review/2000_04/files/2000046.pdf.

[18] Fujimoto, R. M., Permulla, K. S., and Riley, G. F. *Network Simulation.* Berkeley: Morgan & Claypool Publishers, 2007.

[19] Greis, M. "Tutorial for the Network Simulator "ns"," Information Sciences Institute. URL: http://www.isi.edu/nsnam/ns/tutorial/index.html.

[20] Institute of Telematics. "Visualisation of NS-2 propagation models." Hamburg University of Technology, Germany. URL:https://wiki.ti5.tu-harburg.de/wsn/ns2/extension_propagation.

[21] Johnson, D., Hu, Y., and Maltz, D. "The Dynamic Source Routing Protocol (DSR) for Mobile Ad-hoc Networks for IPv4." *RFC4728, Internet Engineering Task Force.* (Feb. 2007). URL: http://tools.ietf.org/html/rfc4728.

[22] Klein, J. *Implementation of an ad-hoc routing module for an experimental network.* MS Thesis. Universität Stuttgart, Stuttgart Germany, and the Universidat Politècnica de Catalunya, Barcelona Spain, (Apr. 2005). URL:http://www.read.cs.ucla.edu/click/publications.

[23] Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M. F. "The *Click* Modular Router," *ACM Trans. Comput. Syst.* 18, 3 (Aug. 2000), 263-297. URL:http://doi.acm.org/10.1145/354871.354874.

[24] Kohler, E. *The Click Modular Router*. Ph.D. Thesis. Massachusetts Institute of Technology, MA. (Feb. 2001). URL: http://pdos.csail.mit.edu/papers/click:kohler-phd/thesis.pdf.

[25] Kotz, D., Newport, C., Gray, R.S., Liu, J., Yuan, Y., and Elliott, C. "Experimental Evaluation of Wireless Simulation Assumptions," *Dartmouth Computer Science Technical Report TR2004-507*. (Jun. 2004). URL:http://www.cs.dartmouth.edu/reports/TR2004-507.pdf.

[26] Kotz, D., Newport, C., Gray, R.S., Liu, J., Yuan, Y., and Elliott, C. "Experimental Evaluation of Wireless Simulation Assumptions," *MSWiM '04*. (Oct. 2004). URL:http://cmc.cs.dartmouth.edu/cmc/papers/kotz:axioms-tr2.pdf.

[27] Kurkowski, S., Camp, T., and Colagrosso, M. "Manet Simulation Studies: The Incredibles," SIGMobile Mobile Computing Comm. Rev., vol. 9, no. 4, pp. 50-61. (2005). URL: http://portal.acm.org/citation.cfm?doid=1096166.1096174.

[28] Larsson, T. and Hedman, N. *Routing Protocols in Wireless Ad-hoc Networks--A Simulation Study*. MS Thesis. Lulea University of Technology, Stockholm, Sweden. URL: http://www.ietf.org/old/2009/proceedings/99mar/slides/manet-thesis-99mar.pdf.

[29] Macker, J. and Chakeres, I. "Mobile Ad-hoc Networks (manet)." IETF MANET Working Group. URL: http://www.ietf.org/dyn/wg/charter/manet-charter.html.

[30] Mobile Systems Laboratory. *Global Mobile Information Systems Simulated Library (GloMoSim).* University of California, Los Angeles, CA. URL:http://pcl.cs.ucla.edu/projects/glomosim/.

[31] Murthy, C. S. R. *Ad-hoc Wireless Networks Architectures and Protocols*. New Jersey: Prentice Hall, 2004.

[32] Neufeld, M., Jain, A., and Grunwald, D. "Network protocol development with *NSClick*," *Wireless Networking*. 10, 5 (Sep. 2004), 569-581. URL:http://portal.acm.org/citation.cfm?id=1027343.

[33] OPNET Technologies, Inc. *Optimized Network Engineering Tools (OPNET).* Bethesda, MD. URL: http://www.opnet.com/.

[34] Park, V. and Corson, S. "Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification," *IETF Internet-Draft*. (Jul. 2001). URL:http://tools.ietf.org/html/draft-ietf-manet-tora-spec-04.

[35] Pawlikowski, K., Jeong, H. D., and Lee, J.S. “On the credibility of simulation studies of telecommunication networks,” *IEEE Communications*. 40(1): 132-139. (Jan. 2002). URL:http://www.cosc.canterbury.ac.nz/krys.pawlikowski/publications/credibility.0101.pdf.

[36] PDOS. “The Grid Ad-hoc Networking Project.” Parallel & Distributed Operating System Group, Massachusetts Institute of Technology, MA. URL: http://pdos.csail.mit.edu/grid/.

[37] Perkins, C. E., Belding-Royer, E. M., and Das, S. R. “Ad-hoc On-demand Distance Vector (AODV) Routing,” *IETF Internet Draft.* (Feb. 2003). URL:http://tools.ietf.org/html/draft-ietf-manet-aodv-13.

[38] Perkins, C. E. and Bhagwat, P. “Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers,” *SIGCOMM 94*. (Aug 1994). URL:http://www.cs.virginia.edu/~cl7v/cs851-papers/dsdv-sigcomm94.pdf.

[39] Toh, C. –K. *Ad-hoc Mobile Wireless Networks: Protocols and Systems.* New Jersey: Prentice Hall, 2001.

[40] Tornquist, A. *Modular and Adaptive Ad-hoc Routing in Click*. MS Thesis. University of Colorado, Colorado. (2001). URL:http://systems.cs.colorado.edu/Networking/ModularAdhoc/adhoc.pdf.

[41] Wang, Y. “A Tutorial of 802.11 Implementation in ns-2.” *MobiTec Lab, CUHK.* URL:http://www.cs.utexas.edu/~erozner/tr_ns802_11.pdf.

[42] Wireless Communications Technologies Group. “NIST DSR Model Readme File,” National Institute of Standards and Technology. (Dec. 2000). URL:http://www.antd.nist.gov/wctg/DSRreadme.pdf.

[43] Xiuchao, W. “Simulate 802.11b Channel within NS2.” National University of Singapore, Singapore. (2004). URL:http://www.comp.nus.edu.sg/~wuxiucha/research/reactive/publication/Simulate80211ChannelWithNS2.pdf

REPORT DOCUMENTATION PAGE						*Form Approved* *OMB No. 074-0188*
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to an penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**						
1. REPORT DATE *(DD-MM-YYYY)* 13-06-2010			**2. REPORT TYPE** Master's Thesis			**3. DATES COVERED** *(From – To)* Sep 2008 – Jun 2010
4. TITLE AND SUBTITLE Attaining Realistic Simulations of Mobile Ad-hoc NETworks						**5a. CONTRACT NUMBER**
						5b. GRANT NUMBER
						5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Huber, Derek, J., Captain, USAF						**5d. PROJECT NUMBER**
						5e. TASK NUMBER
						5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765						**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GCO/ENG/10-11
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) INTENTIONALLY LEFT BLANK						**10. SPONSOR/MONITOR'S ACRONYM(S)**
						11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.						
13. SUPPLEMENTARY NOTES						
14. ABSTRACT Mobile Ad-hoc Networks (MANET) are comprised of wireless systems that communicate without the assistance of centrally managed routers or base stations. MANET research and development has increased due to computing technologies offering smaller, faster, smarter, and more power efficient platforms to operate on. Largely the testing and evaluation of new and existing MANET protocols has resided in simulation environments. This is due in part to the complexities and expenses incurred when conducting real world tests. Many researchers have come to recognize that these current simulations tend to assume away critical components of the MANET domain. These assumptions are made either to simplify the physical layer of the simulation so that the protocol can be tested or out of necessity because the current simulation platforms are not capable of providing a more realistic physical layer simulation environment. This thesis is focused on addressing these assumptions that affect the physical layer of the MANET protocol by gathering data in the real world and then modifying the simulation environment to model as closely as possible to the gathered results. This modified environment is then compared to the basic MANET simulation environment by analyzing packet delivery and propagation effects of both models.						
15. SUBJECT TERMS Wireless Communications, Simulations, Computer Networks, Mobile Ad-hoc Networks, *Click* Modular Router, NS-2						
16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Mark D. Silvius, Maj, USAF (ENG)	
REPORT U	**ABSTRACT** U	**c. THIS PAGE** U	UU	122	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636, ext; e-mail: Mark.Silvius@afit.edu	

Standard Form 298 (Rev: 8-98)
Prescribed by ANSI Std. Z39-18