U.S.N.A — Trident Scholar project report; no 307 (2003)

WIRELESS NETWORK DESIGN OPTIMIZED FOR MILITARY OPERATIONS IN DEGRADED LITTORAL ENVIRONMENTS USING LINK LAYER ERROR DETECTION MECHANISMS

by

Midshipman Nathan A. Fleischaker, Class of 2003
United States Naval Academy
Annapolis, Maryland

_____
(signature)

Certification of Adviser's Approval

CAPT Joseph C. McGowan, PhD, USNR
Associate Professor, Department of Electrical Engineering

_____
(signature)

_____
(date)

Acceptance for the Trident Scholar Committee

Professor Joyce E. Shade
Deputy Director of Research & Scholarship

_____
(signature)

_____
(date)

USNA-1531-2

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (FROM - TO)
05-05-2003		05-05-2003 to 05-05-2003

4. TITLE AND SUBTITLE
Wireless Newtork Design Optimized for Military Operations in Degraded Littoral Environments Using Link Layer Error Detection Mechnanisms
Unclassified

5a. CONTRACT NUMBER
5b. GRANT NUMBER
5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER
5e. TASK NUMBER
5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME AND ADDRESS
US Naval Academy
Annapolis, MD21402

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME AND ADDRESS
,

10. SPONSOR/MONITOR'S ACRONYM(S)
11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
APUBLIC RELEASE
,

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Mobile ad-hoc networking (MANET) is a wireless technology to link autonomous, mobile computers that are free to move randomly, organize themselves arbitrarily and leave or enter the network on-the-fly. Two areas of interest have been identified and studied: the effect of environmental factors on MANET performance, and methods to improve performance by faster identification of broken links. In MANET, the physical channel is affected by environmental conditions such as sea state and physical barriers that increase the frequency of link failures and degrade network performance. Link Layer Detection (LLD), a subroutine designed to identify broken node links, can potentially improve a network?s performance, but the current standard does not utilize LLD. This study examines, through computer simulations, the hypothesis that LLD will improve the simulated overall network performance. In an analysis of over five thousand simulations, LLD was found to improve performance in simulated environments approximating ideal propagation conditions. However, simulated environmental degrade decreased the performance advantages of using LLD. A break point, beyond which operation with LLD failed to yield superior performance, was identified. Design criteria for improved LLD methods, representing potential improvements in network routing with particular application to military operations, are proposed.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	Same as Report (SAR)	74	Cornell, Elizabeth ecornell@dtic.mil
Unclassified	Unclassified	Unclassified			19b. TELEPHONE NUMBER International Area Code Area Code Telephone Number DSN

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATE COVERED
	5 May 2003	

4. TITLE AND SUBTITLE
Wireless network design optimized for military operations in degraded littoral environments using link layer error detection mechanisms

5. FUNDING NUMBERS

6. AUTHOR(S)
Fleischaker, Nathan A. (Nathan Andrew), 1982-

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)		8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
US Naval Academy Annapolis, MD 21402		**Trident Scholar project report no. 307 (2003)**

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
This document has been approved for public release; its distribution is UNLIMITED.	

13. ABSTRACT: Mobile *ad-hoc* networking (MANET) is a wireless technology to link autonomous, mobile computers that are free to move randomly, organize themselves arbitrarily and leave or enter the network on-the-fly. Two areas of interest have been identified and studied: the effect of environmental factors on MANET performance, and methods to improve performance by faster identification of broken links. In MANET, the physical channel is affected by environmental conditions such as sea state and physical barriers that increase the frequency of link failures and degrade network performance. Link Layer Detection (LLD), a subroutine designed to identify broken node links, can potentially improve a network's performance, but the current standard does not utilize LLD. This study examines, through computer simulations, the hypothesis that LLD will improve the simulated overall network performance. In an analysis of over five thousand simulations, LLD was found to improve performance in simulated environments approximating ideal propagation conditions. However, simulated environmental degrade decreased the performance advantages of using LLD. A break point, beyond which operation with LLD failed to yield superior performance, was identified. Design criteria for improved LLD methods, representing potential improvements in network routing with particular application to military operations, are proposed.

14. SUBJECT TERMS:	15. NUMBER OF PAGES
Wireless Networking; Routing; Ad-hoc network; Link Layer Detection; MANET; Mobile Networking	**72**
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Abstract

Mobile *ad-hoc* networking (MANET) is a wireless technology to link autonomous, mobile computers that are free to move randomly, organize themselves arbitrarily and leave or enter the network on-the-fly. Two areas of interest have been identified and studied: the effect of environmental factors on MANET performance, and methods to improve performance by faster identification of broken links.

In MANET, the physical channel is affected by environmental conditions such as sea state and physical barriers that increase the frequency of link failures and degrade network performance. Link Layer Detection (LLD), a subroutine designed to identify broken node links, can potentially improve a network's performance, but the current standard does not utilize LLD. This study examines, through computer simulations, the hypothesis that LLD will improve the simulated overall network performance.

In an analysis of over five thousand simulations, LLD was found to improve performance in simulated environments approximating ideal propagation conditions. However, simulated environmental degrade decreased the performance advantages of using LLD. A break point, beyond which operation with LLD failed to yield superior performance, was identified. Design criteria for improved LLD methods, representing potential improvements in network routing with particular application to military operations, are proposed.

Keywords:

- Wireless Networking
- Ad-hoc network
- MANET
- Routing
- Link Layer Detection
- Mobile Networking

Acknowledgments

I am grateful for my advisor, collaborators, family and friends for their contributions and support of this work. CAPT Joseph McGowan, my advisor, spent countless hours working with me, refining and strengthening my thinking and scientific reasoning. His patience and help in making this project a reality cannot be overstated. Mr. Joe Macker, the co-chair of the IETF MANET Charter Group and the Senior Network Scientist at NRL, provided suggestions, feedback and critique throughout the progress of this project despite his busy schedule. His input gave this project direction at the crucial beginning stages. Thanks are also due to Mr. Ray Cole, section head in the IT Division at the Naval Research Laboratories, Washington DC, who provided financial aid and arranged for an internship at NRL where I learned the fundamentals of the area of research, and the software necessary to conduct this study.

The enthusiasm and encouragement of CAPT Kiepe from the Office of Naval Research helped give me a big picture vision of how this technology truly will be able to impact the Navy and Marine Corps of tomorrow. CDR Thad Welch graciously provided his suggestions and expertise in the area of wireless radio communications and propagation models. I am grateful to Professor Shade and the other members of the Trident Scholar committee whose efforts and often thankless work made this Trident program a reality.

Lastly, I am grateful and thankful for my family and friends for the support and inspiration they provided me throughout this past year. Mr. Steven Fleischaker assisted in the writing of software to automate parts of the data collection process and was always an

encouragement with his excitement at hearing about any progress that was being made as this work progressed. Finally, my inspiration and interest in research must be traced to my father. His own PhD has always been such an assumed constant in my life, and there has never been any doubt in my mind that I too would go on to do research of my own when I "grew up."

Table of Contents

1. Introduction

A. MANET

Mobile *ad-hoc* networking (MANET) is a wireless technology to link autonomous, mobile computers that are free to move randomly, organize themselves arbitrarily and leave or enter the network on-the-fly. Stationary networks, both wired and wireless, assume that fixed infrastructures (virtual or physical) are present and that routing paths rarely change. MANET must be designed with the assumption that the routing paths will frequently fail, requiring new routes to be found. Individual nodes, or computers, within a MANET are all peers and must act as hosts, routers and access points. Physical connectivity is provided by radio communications, and the network protocols must be designed to maintain virtual connectivity between nodes moving within the network [1,2].

MANET is a developing area of research that has recently attracted significant interest from both commercial and military sources. Commercial interest is fueled by the rapid increase in the number of portable laptop computers and other computing devices such as personal desktop assistants (PDAs). MANET has the potential to connect all of these "personal-size" accessories without any setup required by the users. Bluetooth networks are an example of such networking. The Bluetooth network standard is designed so that an individual's cell phone, laptop, PDA or other portable communication device can communicate through wireless "personal-area-networks" which are created and maintained on-the-fly [3].

The military has a clear interest in MANET because the course and outcome of modern warfare is shaped by communications and the speed at which information is exchanged. Modern military strategic thinking is based on "tempo of war" strategies which suggest that victory on

the battlefield is a result of collecting, processing and reacting to information faster than the enemy. MANET, by increasing the type, quantity and reliability of data that can be sent, may speed the collection and processing of data, between and within units of all sizes. MANET is particularly well suited for military operations such as platoon infantry maneuvers and amphibious operations. During amphibious operations, such as Ship to Shore Objective Maneuvers (STOM) [4] and Operational Maneuver from the Sea [5], the movement of Marines from their host Navy ships in the Amphibious Ready Group (ARG) to the area of operations is extremely critical to mission accomplishment. In such a tactical situation, battlefield dynamics can necessitate that the Marines in the assaulting force rapidly adapt to their changing environment. The rapid and reliable dissemination of information is paramount because the manner in which the Marines assault their initial objectives will often determine an operation's success.

One of the key factors in MANET network design is the capability of a network to maintain connectivity between mobile, individual nodes. The independence and mobility of nodes in a MANET means that no working connection between nodes can be assumed to be permanently effective. The MANET must be able to adapt by quickly incorporate alternative routes to maintain virtual connectivity despite a changing physical structure of the network. This is a challenge unique to a MANET as opposed to a stationary infrastructure. The military may not be able to rely on central nodes because the network must be robust enough to deal with the possibility of hard losses. Having peer-to-peer connectivity ensure that the network is not dependent on any single node.

Military applications of MANET also demand that these networks operate effectively in

degraded environments that impede the propagation of electromagnetic waves. To overcome these technical challenges, information about the status of neighboring nodes is maintained by every individual node. Several factors related to routing have a strong effect on a MANET's overall network performance: the speed of identifying failed links, and the speed at which new routes are discovered and adopted. Additionally, methods that minimize the false reports of link errors will lower routing overhead, increasing the bandwidth available for the transmission of data packets.

B. Network Design and Research

Network study and design has traditionally been based on the Open Systems Interconnection (OSI) model that divides the functions and capabilities necessary for a network to operate, placing them into seven abstracted layers which stack and interface vertically. These layers are: physical layer, data link layer, network layer, transport layer, presentation layer, session layer, and application layer [6]. Diagram 1 is a graphical

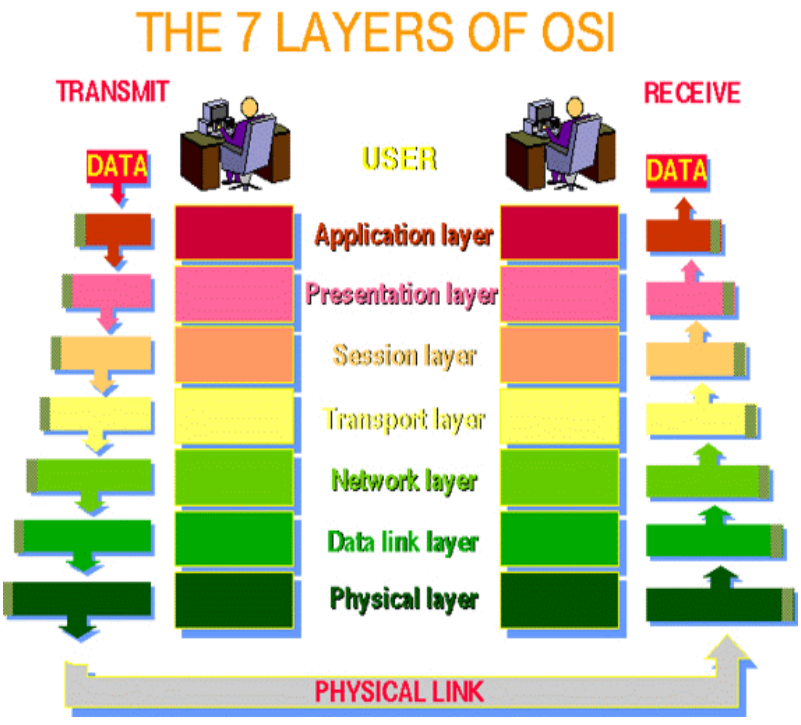

Diagram 1: Graphical representation of the OSI 7-layer model

representation of this network model. MANET and traditional wired networks contrast most at the lower three layers. Higher network layers deal more with end-to-end reliability, the generation and type of data that is sent over the network, and the presentation of this data to the final user. While higher layers can be specifically designed for MANET, abstraction between the layers will ideally make differences between MANET and wired networking invisible to these higher layers.

The physical layer of the network is the most basic level in which electromagnetic waves that carry encoded data are sent, received and processed. In the physical layer, wireless networks, such as MANET, which maintain connectivity via wireless free-space propagation in the environment, differ from wired networks in which copper wire, coaxial cable or fiber serve as the channel through which data is transmitted. The channels in a wired network tend to be insulated from the environmental factors such as weather, while the channel of a wireless network is the environment itself.

The data link layer ensures the individual bits sent by different nodes are correctly transported across the physical layer connection. This data link layer also detects errors that occur in these transmissions and moderates the access to the physical medium. These last two functions are often separated from each other, and placed in two separate sub-layers: the link layer (LL) and the Media Access Control (MAC) layer. In the OSI model, network layer functions include: breaking large chunks of data into packets of information to be sent, ensuring that packets of data are transported from the sending node to the receiving node, and routing. Routing functions, particularly important to MANET because of the high frequency of new route discovery requests relative to stationary networks, discover and store the path that is taken

between the source node and the final destination node.

Most of the technical problems associated with MANET are associated with these three lowest networking layers. Much research into MANET to date has focused on two general areas: (1) protocol design and methods for reliably carrying out and (2) confirming the multi-cast transmission of data to more than one destination. The difficult network layer issue common to both of these fields of research is the routing of packets through a network where nodes have unrestrained mobility and routes can be assumed to be only temporarily viable. The need exists for new protocols and architectures that are responsive to changes in the path between nodes and capable of prioritizing packet transmission [7]. The internet engineering task force (IETF) [8] has organized a working group to focus on MANET related issues, particularly the development of routing protocols that seek to provide solutions to the difficult problem of reliably transmitting information across a network of unknown structure.

Previous studies have developed and examined the performance of routing protocols in different scenarios, varying the network makeup by changing the number of nodes, their mobility, and the amount of data that is sent through the network [9]. In this study two areas of interest were identified and examined:

1. The potential effect of environmental factors on network performance
2. MAC layer mechanisms to more quickly identify broken links and then pass this information to routing functions in the network layer.

The physical layer of the network is the environment in which electro-magnetic waves propagate. Clearly the quality of this channel in propagating waves will affect the overall quality of the network, and in MANET this channel is affected by environmental conditions such as sea

state, ambient noise, active jamming, and physical barriers ("clutter") that cause scattering, diffraction and fading. The author of this study is unaware of any previous studies that examined the effects of a degraded environment on network performance. A degraded environment will likely cause more frequent link failures which will have a detrimental effect on network performance. It was hypothesized that improved link error detection methods in the data link layer can further improve the overall network performance.

In this study, the focus was a quantitative analysis of the simulated performance of networks in scenarios modeling movement and environments that would be encountered during a military operation, particularly the amphibious operation in which Marines assault a potentially hostile beachhead from Navy ships. Mobility scenarios modeling the movement during an amphibious operation were designed and implemented. The performance, evaluated in terms of the data throughput, of a MANET functioning to connect the assaulting vehicles was simulated while accounting, with mathematical models (described below), for the effects of environmental conditions on radio propagation. Based on these results, modifications to the MAC layer's methods for detecting link errors were proposed. The proposed methods and strategies represent a potential improvement in network routing with particular application to military operations. More robust and effective communications will directly translate into the increased effectiveness of assault forces.

C. Radio Propagation Models- Log Normal Shadowing

Several models, ranging in complexity based on the number of considered physical and environment factors, mathematically predict the propagation of electromagnetic waves in a free

space. The log-normal shadowing propagation model for attenuation in a wireless medium provides a means to simulate radio transmissions in noisy and cluttered environments [10]. In this propagation model, equations 1 and 2 below, it is assumed that the environmental interference can be represented in terms of two variables, β and σ.

$$PL(d)[dBm] = \overline{PL}(d_0) + 10\beta \log\left(\frac{d}{d_0}\right) + X_\sigma \quad (1)$$

$$P_r(d)[dBm] = P_t[dBm] - PL(d)[dBm] \quad (2)$$

Where PL(d)[dBm] is the path loss in dB received at a distance d. D_o is a reference distance and $PL(d_o)$ is the known path loss at a specific distance. The received power, P_r, is related to P_t, the transmitted power and the path loss. The path loss exponent, β, is given for an environment, and describes the rate at which the signal attenuates with distance. The stochastic noise and interference in the environment is represented by X_σ, the absolute value of a normal distribution function centered at 0 with a standard deviation of σ.

The variables, β and σ, can theoretically represent any environment. β, the exponential path loss constant, is primarily caused by multi-path fading, diffraction and other physical limitations imposed by the environment. The stochastic element caused by noise and other random interference is represented by a Gaussian distribution function that is added to the path loss. For example, ideal free-space propagation is described by β =2 and σ=0. In this situation, the above equations reduce to simplified free-space, spherical propagation of an electromagnetic wave. Table 1 shows a variety of environments along with the β and σ values associated with them [11].

In this study, the design emphasis for the methodology was fidelity in modeling the environment of an amphibious operation. In the littoral environment during a military operation, sea state and active jamming are the most likely detriments to performance. These factors can reasonably be modeled by understanding that they affect the rate of attenuation of a signal (β) and the noise that may interfere with the receipt of a signal (σ). For this study, the log normal shadowing model was assumed to model the network's physical layer propagation of radio signals.

Environment		beta
Outdoor	Free Space	2
	Shadowed Urban Area	2.75 to 5
Indoors	Line-of-sight	1.6 to 1.8
	Obstructed	4 to 6

Environment	std deviation (dB)
Outdoor	4 to 12
Office, hard partition	7
Office soft partition	9.6
Factory, line-of-sight	3 to 6
Factory, obstructed	6.8

Table 1: Log Normal Shadowing β and σ values for typical environments

2. Routing and Link Layer Detection

Routing protocols are methods, implemented in software, that are used to determine how packets of information will be transported through a network. These methods are designed to maintain the ability of any node in the network to communicate with any other node regardless of whether or not its radio transmissions can be directly received. In the current study, it was shown that the protocol designs and route discovery methods used by a network will affect that network's overall performance as compared to the same network running different routing protocols or route discovery methods. Routing protocols generate overhead which consumes part of the bandwidth that is available for data packets. Poor or sub-optimized designs can contribute to delays if the network reaction is slow to fix link failures.

MANET routing protocols may be classified as *reactive* or *proactive* based upon when they determine the route by which a data packet will reach the destination node. Proactive protocols use periodic updates to ensure nodes are aware of neighboring nodes and have up-to-date routing information. The cost of proactive routing is increased overhead caused by periodic updates. Destination Sequence Distance Vectoring (DSDV) [12] is an example of a proactive routing protocol that is based on the traditional Bellman-Ford distance vectoring approach to determining routes between nodes. Conversely, reactive protocols find routes to destination nodes only when a node attempts to send data to a node for which there is no known viable route. Two examples of reactive routing protocols are *Ad Hoc on Demand Distance Vector* (AODV) [13] and *Dynamic Source Routing* (DSR) [14]. DSR is reactive routing protocol utilizing source routing. In source routing the initiating node determines the entire path to the destination node and forwarding nodes need only maintain information about their neighboring nodes.

Past MANET studies have compared the advantages of different routing protocols in scenarios with different mobility for individual nodes. Results suggested that different protocol designs are each better suited for a specific type of MANET mobility scenario. When comparing DSR and AODV, Perkins [9] found that in high mobility scenarios, networks using AODV performed better than networks using DSR. In such high mobility scenarios, AODV was more effective at handling both the frequent link errors and the additional variability of the nodes in the network. However, in scenarios with less mobility in the nodes, DSR performed better than AODV. While both DSR and AODV are reactive routing protocols, their methods of discovering and storing routes differ. DSR uses source routing which maintains multiple routes to the same destination, while AODV maintains only one routing path at a time. In low mobility scenarios, the few link errors that exist are handled faster by DSR because multiple routes to the same destination are cached and often no new route discovery is necessary. Conversely, in high mobility scenarios, the probability that several or all of the routes cached in DSR are ineffective is high. In these high mobility scenarios, AODV's more frequent updates to the routing path yield superior performance.

The current work examines Link Layer Detection (LLD), mechanisms in the Data Link Layer designed to detect link errors and interface with the routing layer to pass on information about broken links. The use of LLD is advantageous because it frees the routing protocol from using alternative methods to replicate this function. The alternative to using MAC layer feedback in the form of LLD is the use of neighbor discovery methods by the routing protocol. Neighbor discovery mechanisms are procedures through which the routing protocol queries and receives responses from the surrounding nodes that it can communicate with directly. These methods are

discussed in the following section. Since the routing layer is further removed from the physical layer than is the data link layer, neighbor discovery methods in the routing layer must be more proactive than LLD methods and generally require more overhead which decreases the available bandwidth. However in AODV, the current default standard is to operate without LLD, using HELLO messages instead. Throughout the rest of this paper, any reference to the use of HELLO messages by a network is considered synonymous with not using LLD, and vice versa.

A. Neighbor Discovery Methods

Without LLD information from the MAC layer, AODV incorporates uses a neighbor discovery method that consists of sending "HELLO" messages on a regular basis. These HELLO messages both announce the presence of the sending node and act as requests by the sending node for information from any nodes that have recently become neighbors. In contrast, the lack of response from a previously neighboring node is an indication that the link between these two nodes has become dysfunctional. HELLO messages are sent out at a specified interval (the default interval is one second) unless a node has received broadcast packets within this time-period. The HELLO message contains the node's address and sequence number which neighboring nodes use to update their own routing information.

By using HELLO messages, each node is forced to send out at least one message during each HELLO message interval. The absence of any communications from a neighbor is interpreted as a failed connection between the two nodes causing AODV to react by removing from its routing table any routes that had included this now-broken link between nodes.

B. Link Layer Detection Methods

The IEEE 802.11 MAC layer specifications, for wireless ad-hoc networks, include a method for the identification of failed links which is based on a time-out system. Whenever the MAC layer passes information to the physical layer to be sent out to another node, the expectation is that the receipt of this data will be confirmed by the receiving node. If a data packet is sent by the MAC layer, an acknowledgment (ACK) packet is expected. When a Request-To-Send (RTS) message (a packet sent to determine if the channel to that node is clear to send information) is sent, a Clear-To-Send (CTS) message is expected in response.

When the MAC layer begins the transmission of a packet (data or RTS), a copy of the packet being sent is stored in a buffer, and a timer is started to measure the time elapsed since the packet has been sent. Individual packets are identified by a Packet ID number that is unique to each new packet sent. This packet ID number is incremented based on when the packet was sent. In addition to retrieving the appropriate value, this ID helps to place the packets in the correct order for processing. In noisy or congested environments, it is likely that packets will not arrive at their destination in the order in which they were sent.

If the MAC layer receives no response (neither ACK nor CTS packets) within the time period of the timer, a retransmission of the same packet will be attempted. The MAC layer will make several retransmission attempts to send the data before declaring that the link between these two nodes is broken and no longer effective. This number of retransmission attempts is known as the LLD 'threshold value.' The default threshold value for RTS packets is seven (7), meaning that six unacknowledged RTS packets will be sent before a link is declared bad and an error message broadcast. Similarly, the threshold value for data packets is four (4). [11]

In this study, modifications to the routing and MAC layers included turning LLD on or off, and modifying the threshold values for the number of attempted RTS packets that can be sent unacknowledged before a link is determined to be in error.

3. Materials and Methods

This study focused on network designs that were optimized for use in an amphibious operation during STOM. In such an operation, Amphibious Assault Vehicles (AAVs), Landing Craft Air Cushions (LCACs) and other types of troop and equipment carriers will move marines and their supplies from ships to a potentially hostile objective beachhead. Based on the tactical methods used by the Navy and Marine Corps for the employment of amphibious landing craft during STOM, scenarios of node mobility for testing MANET performance were designed. The littoral environment is likely to include noise and other factors that will degrade the signal strength of radio waves, potentially inhibiting communications. The scenarios used in the testing for this study simulated these degraded environments with the log normal shadowing model (discussed previously in section 2.C). This model added stochastic variation and additional attenuation to propagation of radio signals.

A. Simulation Tools

Testing of the network performance was performed with the Network Simulator, version 1b9 (NS2) [11], a detailed computer simulation of the network model. NS2 is open-source software, written in C++, that simulates every layer in a communications network. The interface with the NS2 executable is through script files written in the Tool Command Language (TCL). These scripts define such variables in a network simulation as the specific movement patterns of the nodes in the scenario, the input data rate, and the simulated propagation conditions. Of particular utility to this study is the fact that NS2 is available open source, making it both free and providing access to the source code. Modifications to the source code

were used in this study to implement and study, in NS2, new methods and non-standard specifications for routing, LLD and other network functions. The testing in this study included simulations using both existing routing protocols and MAC layer specifications, as well as simulating the results obtained from modifications to the MAC and router layers.

NS2 simulates transmission of data packets at the physical layer by modeling the attenuation and path loss of a propagating radio wave for each packet. The propagation of these signals is based on mathematical models that can be configured to take into account attenuation and stochastic variation. To the knowledge of the author, research to date has not examined the effects of different environmental conditions on any aspects of network performance. An important goal of military communications is ensuring that communications are robust regardless of external factors such as the environment. The log-normal shadowing propagation model was selected to simulate a variety of degraded environments.

NS2 reports the results of simulation tests in trace files which simply record every action that occurs in the network. Options exist to exclude or include information on the network activity at the MAC, routing and agent layers of the network. These options also significantly influence the run time for individual simulations. In this study, whenever possible, unnecessary tracing was turned off to save time and disk space. Large trace files can be analyzed by programs that parse strings, analyzing one line of the trace file at a time and determining (through IF statements) what network actions caused this statement.

B. Scenario

During amphibious operations, many different factors will influence the particular manner in which amphibious craft proceed from the ship to the beachhead objective. The number of craft used and the method by which they approach the shore will be determined by the type of mission that they are going to accomplish which can vary from a hostile amphibious assault to troop transport directed towards an already secured beachhead. However, a typical operation is usually based on the employment of a forward deployed Marine Expeditionary Unit (MEU). The MEU is built around a Marine infantry battalion, which can support up to fifteen AAV's in a single operation. Being forward deployed, these units are the most likely to be present and used during an amphibious operation. The amphibious assault craft will typically travel from the amphibious ready group, likely located beyond-the-horizon at about 30 NM away from the shore, and travel at speeds of 10 - 40 knots (approximately 5 - 20m/s). During seaborne maneuvers, ships and landing craft will "stay on station," keeping a set formation so that relative motion between the craft is minimal while they travel together towards a destination. While ideally the craft will stay on station during the entire transit, it is likely that for any variety of reasons, from mine avoidance to a changing objective, craft within the formation will have to adjust their position within the formation.

The tactical doctrine described above is used by the US Navy and Marine Corps to conduct amphibious operations, and is the basis for the scenarios we designed for use in this study. The wireless networking scenario used in this study consisted (see figure 1) of ten nodes, two of which are the source and destination for the data transmission in the network. A connection was created between these two mobile nodes, n0 and n1. Both the Transmission

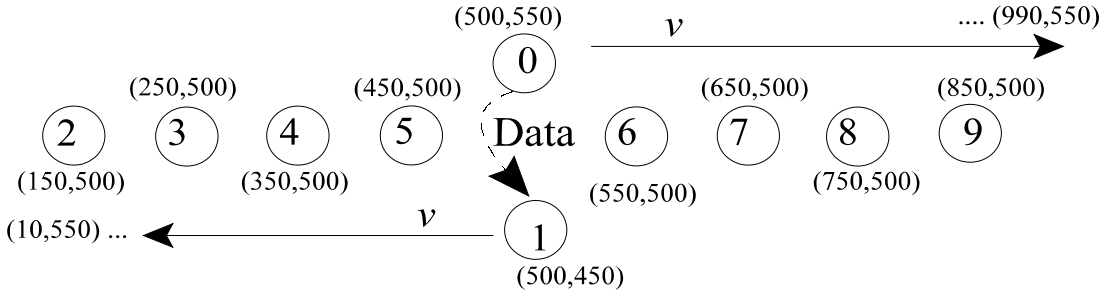

Figure 1: Ten Node Scenario (coordinates given in meters)

Control Protocol (TCP) and User Datagram Protocol (UDP) connections were tested. The traffic sources that were used simulated either File Transfer Protocol (FTP) or Constant Bit Rate (CBR) agents as appropriate to the connection. See Appendix A for the source code used to implement this scenario in an NS2 simulation.

Although both TCP and UDP connections were originally tested, the emphasis during the analysis was on UDP because of its simplicity. While TCP is the standard for transfer of data in the internet and in wired networks, it optimizes performance based on the assumption that network congestion is the primary problem in causing network delay and unacknowledged packets. TCP reduces network's input data rate to control congestion and optimize network performance. It also requires the network to become congested before it attempts to fix the problem. Using the more simplistic UDP methods removes confounding variables that are introduced by TCP's flow control methods. While optimized for wired internet connections, the flow control methods in TCP have been shown (*Gunes, et al*) to be detrimental to the performance of a MANET [15]. Gunes found that the congestion assumption used in TCP's design, that errors are caused by network congestion, was invalid for MANETs in where errors are more likely caused by failed links and absent routes. TCP's flow control methods which lower the input data rate only exacerbate network problems and often bring the MANET to a

total standstill.

In the chosen scenario, nodes n0 and n1 begin transmitting via a direct connection, but with time they move farther apart. The scenario is designed to simulate a formation of amphibious craft moving towards the objective as two additional ships move to be "on station." This scenario also ensures that the physical link originally existing between n0 and n1 will eventually fail, forcing the MAC layer to detect an error in the link and the routing protocol to discover a new route. The other nodes (n2 - n9) in the scenario do not originate data transmission, act as the destination of data or move in the scenario. However, even though these nodes are not the source of any information, because of the nature of MANET routing protocols, they are active performing functions relevant to the routing protocol: relaying broadcast ERROR messages, broadcasting HELLO messages, responding to requests for routes, and forwarding packets by acting as intermediary nodes in a multi-hop routing path.

The choice and relative simplicity in the scenario used for simulation compared to that of previous research in the field warrants a discussion. First, this network scenario was designed specifically around likely and actual scenarios that would be encountered during an amphibious operation. Additionally, one of the primary areas of interest in this study was examining the methods used by the MAC layer to detect links that break when within a network. This scenario realistically modeled the transit of amphibious craft moving into station as they transit from ship to beachhead objective, and it also ensured that link error and neighbor discover functions of the network operated with minimal influence from complex mobility in the scenario.

One goal of this study was to understand the effects of environmental factors and the differing effects on the performance of our modifications in the MAC layer. The simplified

scenario is useful for this purpose. Previous research in this field has tested network performance by varying a network topology consisting of fifty or more nodes, all moving randomly and potentially sending or receiving data simultaneously. Past research clearly shows that these factors significantly influence network performance. Thus to minimize the influence of variables whose influence on the results is not fully known, it was decided to simplify the initial scenarios used in this study as much as possible so that the effects caused by environmental factors could be isolated.

C. Variables Studied and Tests Run

In this study, two objectives were identified: 1) to examine and quantify the effects of simulated degraded environments and to determine what other factors might mitigate this negative relationship. 2) to identify modified LLD methods that will improve network performance. To achieve these goals, the tests that were run in simulation varied the following:

- β, exponential path loss: 2.0-3.0 (in this scenario, the network cannot function with values greater than 3.0)
- σ, standard deviations of the stochastic environmental element: 0-20 dB (the end value is when the network stops functioning, this point is affected by the β value)
- *Velocity of n0 and n1:* 5m/s, 10m/s and 20 m/s
- *Routing and MAC layers*: AODV is used with LLD and without LLD.
- *LLD Threshold Values*: Variations in the threshold values (default 7) of allowed unacknowledged packets (threshold values of 3, 7, 8, 11, 15 and 25 were compared).

- *Data Rate and Transmission Protocol:* CBR Data Rates varied between 100kbps, 500kbps and 1Mbps. Some TCP connected networks were also simulated

The length of the simulation is dependent on velocity of n0 and n1, the simulation lasts until n0 and n1 reach their final destination (simulation time length $\times$ v =500 m·s, 25sec for v=20m/s, 50sec for v=10m/s, 100sec for v=5m/s). In the course of this study, over five thousand different network scenarios from the parameter space of the above variables were simulated and analyzed.

The values that were chosen for velocity, β and σ correspond to reasonable values, expected during an amphibious operation. The range of speeds of 5-20m/s corresponds to speeds of approximately 10-40 knots, which are the speeds that would be expected during such an operation. For the environmental factors, Table 1 (in section 1-C) shows that the range of values chosen for β and σ represent reasonable values. The types of environments simulated range from outdoor operations in no clutter to moderately cluttered terrain, and outdoor environments in which random factors, in this case weather and sea state, are included.

Tests in this study were simulated in groups based on a common network scenario that varied only with respect to the characteristics of the simulated environment. Given the common scenario, simulations were run while varying the values for the environmental conditions. Running tests in this fashion made the effects of simulated environmental degradation on a specific scenarios readily observable, achieving the first aim of this study. The results from a group of simulations can be represented on a 2-D graph with multiple series or a 3-D contour graph. This testing procedure was then repeated around different scenarios, forming different groups of results for analysis and comparison.

D. Automated Simulation Software

During this study, every simulation in NS2 required either new source code for the scenario to be simulated or a recompilation of the NS2 simulator. New source code was required for scenarios in which only the network structure, mobility or environment changed. Of the variables used in this study to modify these factors, four were varied for the scenario source code: node velocity, β and σ values, and input data rate. For every scenario to be simulated, a new scenario file was written for NS2 to input. NS2 reads a file that specifies these (and other) criteria and runs a simulation based on this input. While the majority of the source code does not change significantly between these simulations, a different file must be written for each new simulation. For changes involving the actual implementation of the network protocols, a full recompilation of NS2 is required. In this study, recompilation of NS2 was required before conducting simulations in which the routing protocol or MAC layer were modified.

The testing of the over five-thousand separately simulated scenarios in this study was significantly eased by the development of a program to partially automate data collection and analysis. While attempting to automate the testing requiring NS2 source code modification and a recompilation of the executable, it was possible to automate the generation of individual scenarios. Further, using batch files, it was possible to automate the simulation and analysis of the scenarios. See Appendix D for selected source code from this program, *Shadows.exe*. *Shadows* produces three outputs, two batch files and a series of scenario definition files ready for simulation in NS2.

Written in visual C++, *Shadows* was designed for a Windows environment and uses a Graphical User Interface to allow the user to specify either a value or a range of values for five

different variables modified in the testing for this study. These variables are β, σ, input data rate, packet size, and node velocity. A separate scenario definition file is then created for every combination of these variables. The names of the scenario file, and the specific name of the trace-file that will result are specified by the values of the variables used in this scenario. The two batch files that are also created: first, `pre_process.bat`, runs NS2 simulations on all of the scenario files produced. Second, `post_process.bat`, runs the data analysis program on all of the trace-files that will be produced.

See Appendix E for the nomenclature of the simulation definition and trace files, and example lines from the batch files produced by *Shadows*.

E. Techniques for Simulation Analysis and Criterion for Optimization

The raw results of the network performance of scenarios simulated in NS2 are large trace files which record the activity of the simulated network. Each action taken in the network causes a new line to be written in the trace file. By changing variables defined in the scenario definition file, NS2 will write to the trace file information from any or all of the MAC, routing and/or Agent layers. Each line in the trace file includes information specifying exactly what happened during an action taken by the network by writing the following information: time elapsed since the beginning of the scenario, specific details about the type and size of the packet being sent, what happened (sent, received, forwarded, or dropped packet), what layer is currently involved in this action, the nodes between which this activity occurred, the source and destination nodes of the overall routing path, and various other flags and addresses used by the MAC and routing layers. The analysis of the trace files was complicated somewhat by the fact that NS2 has

several different formats for trace files. As NS2 has been updated and new versions have been released, the trace file format for wireless networks has changed significantly. These changes required the use of different analysis programs for each type of trace file.

In the simulations run during this study, a typical trace file, which used MAC, routing and agent traces, would be between five and eight Mb in size. The sheer quantity of data within each individual trace file is far too much for this study involving the simulation of over five thousand individual scenarios in this study's parameter space.

The criterion chosen to evaluate the network performance was the network's throughput, defined as the average data rate received by the destination node over the course of the simulation. Data was also recorded on routing overhead, good-put (the ratio of received packets to sent packets) and the ratio of received data packets to dropped data packets. Initial results led to the conclusion that good-put and the ratio of received to dropped packets were both correlated to the throughput. The routing overhead was too sensitive to the variables being studied to be a reliable evaluation criterion. Thus, the criterion for comparison and optimization in this study was the average data throughput of the network over the course of the simulation. An added advantage of using this criterion was that it reduced the need for MAC and Routing layer traces, saving disk space, speeding the simulation of individual scenarios and speeding the analysis of the resulting trace files.

To analyze the trace files created by NS2, two programs were used. One program TCPdump Rate Plot Real-time (TRPR) [16] was written at Naval Research Laboratory (NRL) and outputs the average data rate in specific time intervals. This output can be graphed to observe an "instantaneous data rate" vs time for a given simulation. These results are useful for

identifying where link errors occurred and how long the network was stopped from functioning properly before a new route was discovered. However, this analysis was, for the most part, far too detailed for the aims of this study.

Analysis of the trace files was done primarily through a program written in AWK [17], an interpreter language specifically designed for analyzing strings and text files. See Appendix B for the source code for this data analysis program. AWK programs work by reading one line of text at a time and then using string matching commands to parse through the line. The program written for trace file analysis parsed the trace files line by line, with each line corresponding to one specific event or action that occurred in the network. Based on the trace file format, the analysis program attempts to use string comparisons and IF statements to identify the type of event that occurred. However, the program is only capable of identifying specific types of events. The program then maintains a running total of the number of each event that occurred and outputs this value to the console stream at the conclusion of the program. The types of events that can be identified and are tallied are: total packets sent, data packets sent, routing packets, ACK packets sent, request packets sent, total packets dropped, routing packets dropped, data packets dropped, hello packets sent and total error messages broadcast during the simulation.

During this study, analysis of the trace files was completed as follows: the trace files were analyzed with the AWK trace file analysis program (see Appendix B) either individually from the console prompt or using the second batch file produced in the *Shadows* program. The output from this trace file analysis program was piped into a text file using the console command prompt. This text file consists of several numbers, delimited by tabs, which represent the total

number of each event that was counted in the trace file. This text file was copied into a spreadsheet program where the raw data, in the form of the number of each type of event that occurred (dropped packet, sent packet, etc), combined with information on the packet size and the time duration of the scenario, was converted into a data throughput value. Data throughput had previously been identified as the evaluation criterion to be used to quantify network performance in this study.

4. Results

This study had two specific aims: First, to examine, analyze and quantitatively describe the effects of simulated degraded environments on network performance. Second, to identify LLD methods and evaluate the relative performance of the networks in which they are implemented. Analysis of the trace files produced from the simulations studied in this study demonstrate that for most simulated scenarios, the implementation of LLD in a network meaningfully improves the data throughput of the network. Furthermore, networks using a non-standard implementation of LLD, an implementation using a threshold value of roughly double the default value, produced superior network performance in the majority of the scenarios tested. Testing also demonstrated, however, that the best network design for a given scenario is not a set standard, but a function of several factors, including environmental conditions.

Results from this study can be organized into three categories: (1) qualitative and quantitative descriptions of the relationship between simulated environmental degradation on the network and identification of factors that influence this relationship, (2) the relative network performance compared between networks implemented with and without LLD, and identification of how this relationship is affected by other factors, and (3) comparison between the relative performance of alternate LLD implementations.

A. Simulated Environmental Degradation on Network Performance

In general, simulated degrade in the environment markedly decreased the network's performance. This result is consistent with expectations, and can be clearly seen from many of the figures presented in this results section. Figures 2, 3 and 4 clearly show this trend of a

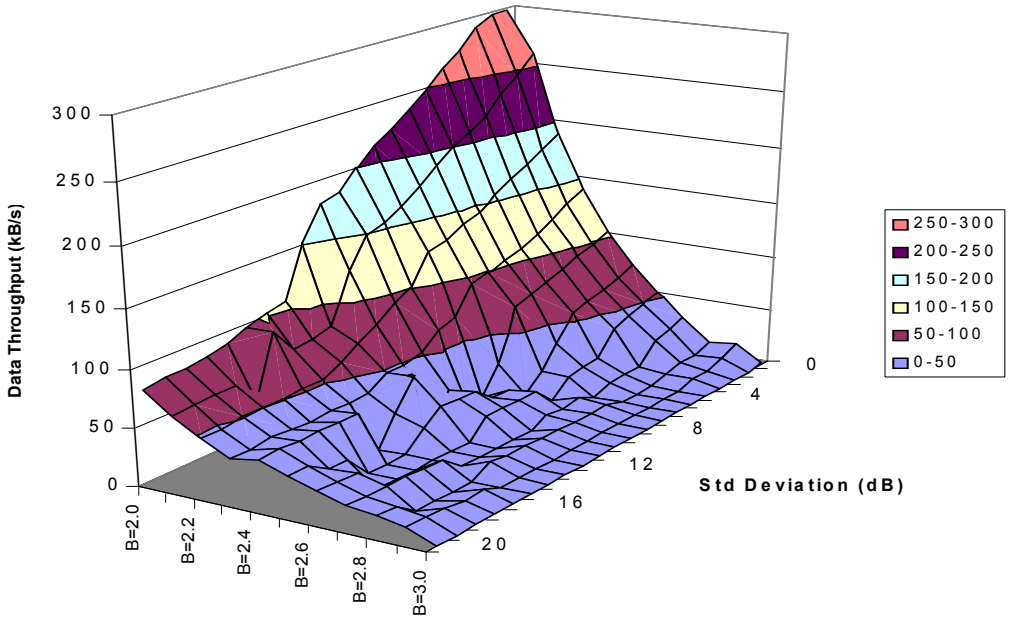

Figure 2: Input CBR=500kb, v=10, LLD OFF

From this contour, the effects of degraded environmental conditions is readily apparently. Compare the far corner, with an assumption of free-space ideal propagation to the rest.

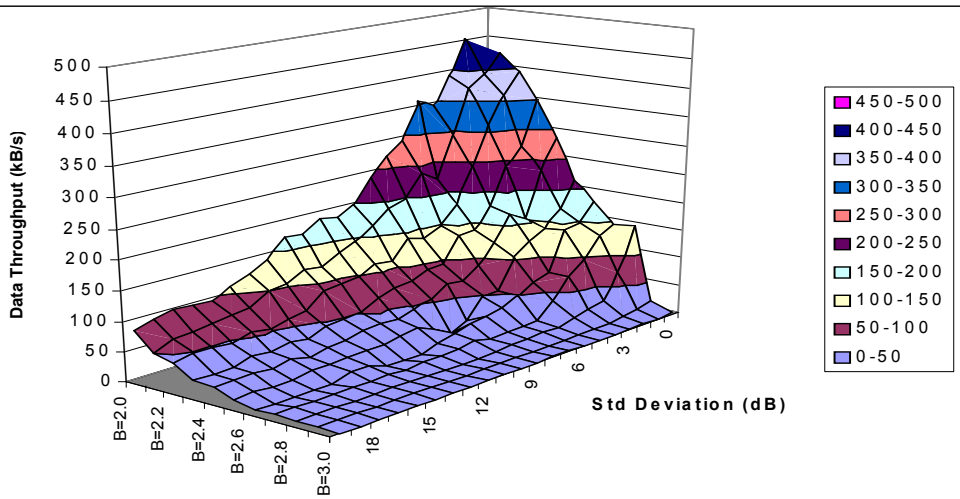

Figure 3: Input CBR=500kb, v=10, LLD ON

From this contour, clearly the general decreasing trend in performance, as factors hindering propagation in the environment increase, is not changed by LLD. However, the performance of the network is improved.

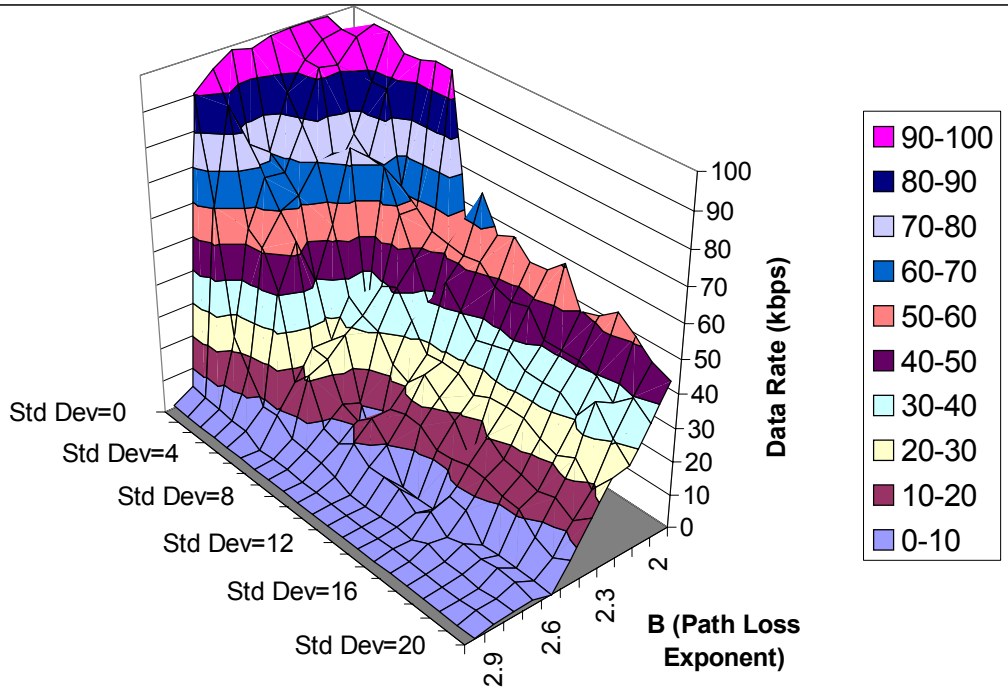

Figure 4: CBR=100kpbs, v=20, No LLD

Note that neither velocity nor data rate have discernable impacts on the trend of the network performance as environmentl degradation is increase. At higher velocities, small stochastic aberrations are apparent

substantial decreased performance with increasing simulated environmental degradation. These figures are three-dimensional contour graphs that display the performance of a network in a grouping of different scenarios. Each graph represents a single network design that is evaluated by simulation in a variety of environments modeled by varying the β and σ values over the specified parameter space. Figures 2 and 3 share the same network mobility in terms of the node velocity and input data rate, they differ only in the use of LLD by the simulated network. Figure 2 implemented standard HELLO Messaging (not using LLD), while the scenario for figure 3 implemented LLD (with the default Threshold value of 7).

In figures 2 and 3, note the marked decrease in the network performance as the environmental conditions become more degraded. An ideal environment is modeled by a β value of 2.0 and a σ of 0, representing standard $1/R^2$ attenuation with no additional noise added to the path loss. The network data throughput steadily declines with respect to an increasing β in the simulated environment. The same results held for tests in which β was held constant and σ values were decreased.

The results from the scenario represented in Figure 4 differ from the previous scenarios in both the velocity of the nodes and increased data rate. In this graph, the same trend of increasing noise decreasing network performance is clearly observed, but the specific characteristics of this trend differ because of the other variables that were varied. In this scenario, the higher velocity increased the randomness of the resulting performance. The results from a single test run of these simulations is not as smooth as the results from a lower network velocity. The results from individual runs include more random variation. However, when averaged over multiple test runs, the results follow the same trend that was observed from results with lower node velocities.

Figures 2, 3 and 4, were analyzed by examining the average rate of decrease in performance with respect to added noise in the environment in both low and high noise scenarios. This analysis reveals that networks are, on average, more sensitive to added noise in the environment in low noise scenarios (where σ is between 0 and 10 dB) relative to than in high noise scenarios (where σ is between 11 and 20 dB). This data suggests that the network is more sensitive to added stochastic noise when β values are lower, however a more inclusive examination of the data suggests that many factors other than β values influence the sensitivity

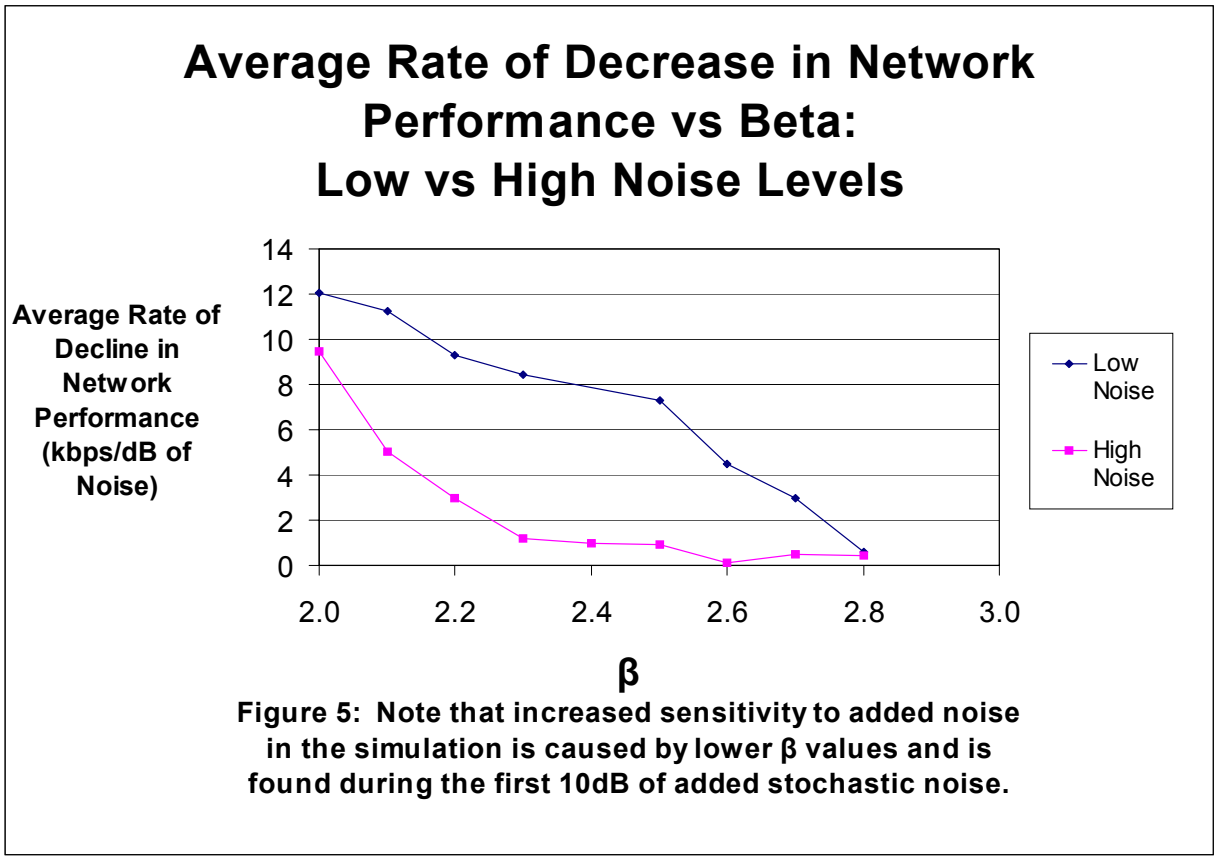

Figure 5: Note that increased sensitivity to added noise in the simulation is caused by lower β values and is found during the first 10dB of added stochastic noise.

of the network's performance to added noise. Figure 5 graphically represents these sensitivity results that are mentioned above. Note that these results are for a specific scenario in which the input data rate=500 kbps. The rate of decline in network performance is graphed versus the β values for two different series, low noise and high noise. The low noise series is the average rate of decline in network performance for the range of σ=0-10 dB. The high noise series is the average rate of the decline in network performance over the high noise range of σ=11-20 dB. Clearly, the sensitivity of network performance to added noise to the environment is related to the current environment and network factors such as LLD implementation, data input rate and node velocity.

In comparisons between TCP and UDP testing, TCP tests were more sensitive to environmental degradation than UDP tests, a result that is consistent with previous studies on TCP performance with MANET [15].

B. Performance of AODV with and without LLD

Testing that compared the results between network performance with and without LLD yielded data that supported two main results: (1) measurement of the improvement in network performance by networks using LLD relative to those using HELLO messaging and (2) the effect of LLD on a network's sensitivity to increased noise in the environment.

In both CBR and TCP testing, the use of LLD improves performance relative to networks using HELLO messaging in nearly every scenario. Several variations in the implementation of LLD were tested in this study, but for the majority of cases, the implementation of LLD using a threshold value of fifteen (15) produced the best results. However, in both very degraded environments and close to ideal environments, this implementation of LLD is not optimal. A discussion of the differences between LLD methods is located below in the next section of results.

In general, a version of LLD is superior for ideal to moderately degraded environments, while HELLO messaging improves performance in heavily degraded environments. Table 2 shows the relative comparison between networks using a TCP connection and implementing LLD or HELLO messaging. Percentage increase represents the increase in throughput, given previously in absolute terms of kbps, as a percentage increase relative to the throughput of the method with poorer performance. Clearly, in scenarios with lower values of σ, networks

implementing LLD yield superior network performance, but scenarios with higher σ values have better network performance with HELLO messaging. These general results hold regardless of the velocity at which the nodes in the network are traveling or whether the network uses TCP or UDP connections. However, in CBR testing, the ranges for which HELLO messaging yields superior performance are smaller than the ranges using TCP connections. Table 3 shows the relative improvements that were found in using types of LLD in networks over specific simulated environments. Note the smaller (or non-existent) ranges for which HELLO messaging is superior. Using the proper type of network implementation improved data throughput by between 40% -180%, from the default network routing protocol implementation, which is HELLO messaging, depending on specific environmental conditions. While in simulated environments with low dB of stochastic variation, HELLO messaging produces worse performance than networks with LLD, HELLO messaging appears to be less sensitive to the increase in noise to the environment. As the amount of noise in the path loss model increases, a cross-over point is reached after which the use of LLD no longer offers superior performance. Figure 6 clearly identifies this "break point"

v=10, Improvement found by using HELLO Messaging

Beta	Range of Xσ	Improvement (kb/s)	Improvement (%)
2.0	4.5-21	43.17	29.3
2.1	6-18	29.54	27.7
2.2	3-12	18.55	17.0
2.3	5-12	15.95	19.9
2.4	3-9	10.57	23.9

v=10, Improvement found by using HELLO Messaging

Beta	Range of Xσ	Improvement (kb/s)	Improvement (%)
2	0-4	18.18	25.43716484
2.1	0-4.5	55.70	31.47113945
2.2	0-2	11.00	4.981132075
2.3	0-4	3.63	16.24441133
2.4	0-2	10.50	30

v=20, Improvements found by using LLD

Beta	Range of Xσ	Improvement (kbps)	Improvement (%)
2.0	0-4	33.00	16.3
2.1	0-4	47.00	24.2
2.2	0-4	40.60	29.7
2.3	0-3	38.80	34.8
2.4	0-4	29.20	38.5

v=20, Improvements found by using HELLO Messaging

Beta	Range of Xσ	Improvement (kb/s)	Improvement (%)
2.0	5-19	48.58	31.8
2.1	5-16	40.68	33.8
2.2	5-14	20.02	22.0
2.3	4-10	18.85	24.9
2.4	5-9	9.41	19.2
2.5	3-7	13.93	34.2

Table 2: Comparison of Network Performance of TCP scenarios with and without LLD. Note that in the scenarios with larger β values, HELLO messaging improves the performance markedly

Comparison of average percentage improvement between LLD Thresholds

Beta	Std Deviation	Comparison of LLD Threshold=15 vs default LLD
2.00	3-10	7.28
2.10	2-10	18.54
2.20	3-10	20.41
2.30	4-10	*24.13*
2.40	1-10	21.77
2.50	1-10	33.96

Beta	Std Deviation	Comparison of Defulat LLD vs. HELLO Messages
2.00	0-10	40.45
2.10	0-10	45.99
2.20	0-7	67.91
2.30	0-8	99.67
2.40	0-8	108.99
2.50	0-6	153.69

Beta	Std Deviation	Comparison of LLD Threshold=15 vs HELLO Messages
2.00	0-10	45.51
2.10	0-10	60.13
2.20	0-10	81.68
2.30	0-9	115.94
2.40	0-8	132.80
2.50	0-9	189.75

Table 3 (Data input=100kbps, v=5m/s): The values given in the third column are presented as a percentage improvment. Note the large improvements that can be obtained by using LLD relative to HELLO messaging. However, the advantages are not over all ranges of σ values, they are restricted to some scenarios, as can be seen in the second column.

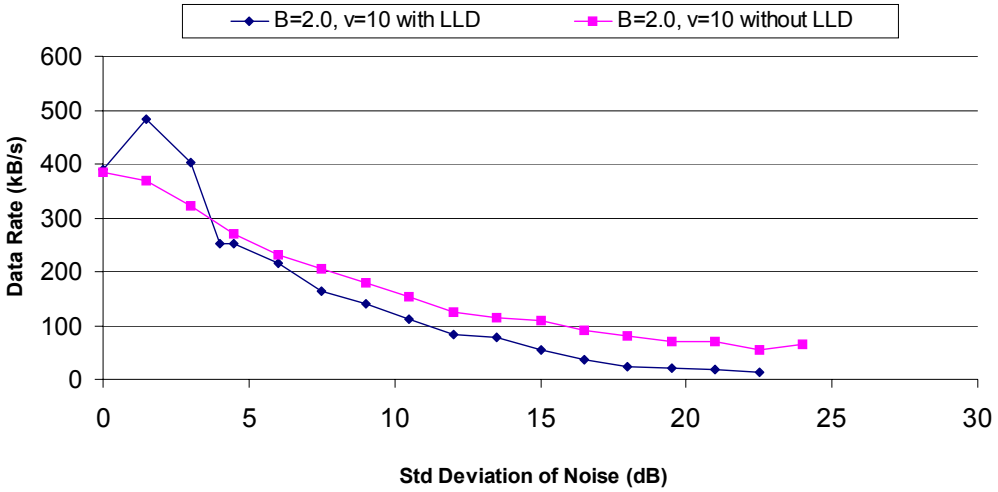

Figure 6 (using TCP): The above graph clearly demonstrates the relative sensitivity to increased σ values. The position of the "break-point" is clearly identifiable or this group of simulated environments

for a particular network scenario simulated over a grouping of environmental scenarios. Similarly, this break point was identified for every network scenario simulated in this study.

Figure 6 also clearly shows the relative sensitivity to increased environmental noise demonstrated by networks using LLD instead of HELLO messaging. While HELLO messaging yields poor network performance relative to LLD for un-degraded to moderately degraded environments, networks using HELLO messaging are less sensitive to added environmental noise. Thus as the simulated environment becomes more degraded, the relative advantage found in using LLD decreases until HELLO messaging yields superior performance. Table 4 compares the relative decrease in network performance (per added dB of random noise) between HELLO messaging and different LLD methods. HELLO messaging has a much lower sensitivity to increased noise relative to all but one of the LLD methods. The similarity between the LLD with Threshold of 25 and HELLO messaging is addressed below.

Comparison on effect of Beta and Threshold Value's effect on the Rate of Decline in Network Performance (kbps/dB noise)

Beta	NO LLD	3.00	7.00	8.00	11.00	15.00	25.00
2.00	0.49	5.37	3.41	3.41	3.01	2.89	0.06
2.10	0.43	7.79	5.64	5.82	7.33	4.79	0.13
2.20	0.38	8.09	7.84	7.84	7.33	6.46	0.63
2.30	0.49	9.04	8.61	8.75	8.50	7.58	0.60
2.40	1.76	9.00	8.46	8.92	8.89	8.10	2.28
2.50	0.33	7.40	8.44	8.57	8.36	8.12	0.52

Table 4: The data presented demonstrates the effect of β values and different LLD methods on the Network's Decline Performance. Clearly LLD increases the sensitivity to noise relative HELLO messaging.

C. LLD Threshold Improvements

The implementation of LLD in the 802.11 specification sets a threshold value for the number of unacknowledged allowed before the MAC layer will declare a link bad and broadcast an ERROR message. The default threshold value is placed at seven (7) unacknowledged

packets. Results from this study demonstrated that while the default setting produced superior results in simulated ideal environments, alternative methods of LLD produced better results for all other environmental scenarios. For the majority of simulated environments (more than 70% of the scenarios tested), performance was obtained when implementing LLD with a threshold value of fifteen (15) vice seven (7). Data throughput improved between 7% -150%, on average, depending on the specific environmental conditions in which the testing was evaluated. Table 3 lists these results, and Appendix F includes a more complete listing of all of the results.

Figures 7, 8, and 9 are 3-D contour maps that compare the network performance of different implementations of LLD across a grouping of different environmental conditions. These figures differ only in the β values that characterize the simulated network. Several results that were previously noted are observable from these graphs. First, the trend for decreased

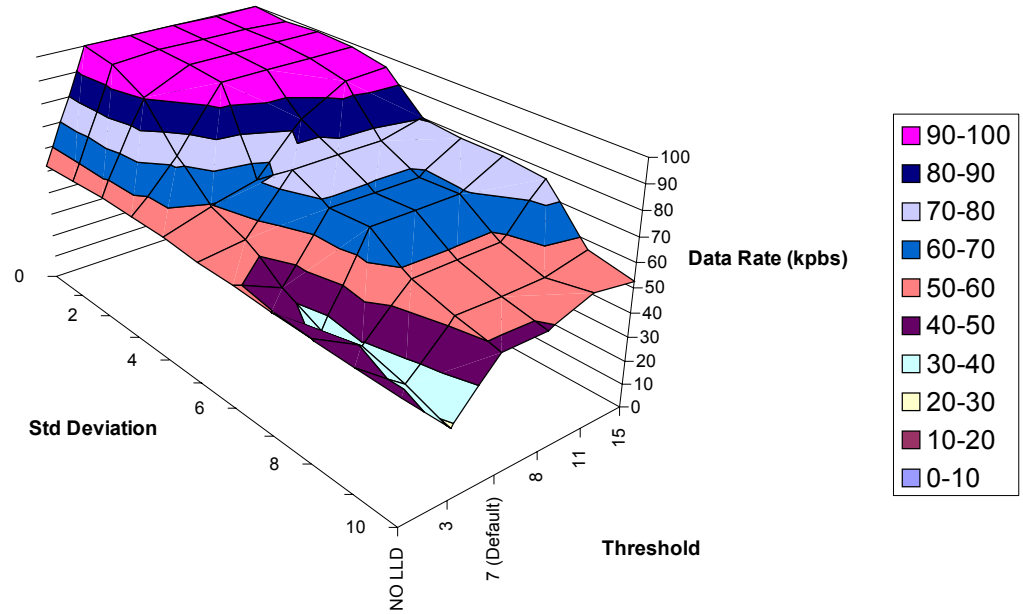

Figure 7: Data Rate vs. LLD Threshold and Std Dev

β=2.1, CBR=100kbps

Note that as these relatively low data rates, the network performance is less sensitive to simulated environmental degredation. Note the poor performance of LLD with Threshold=3.

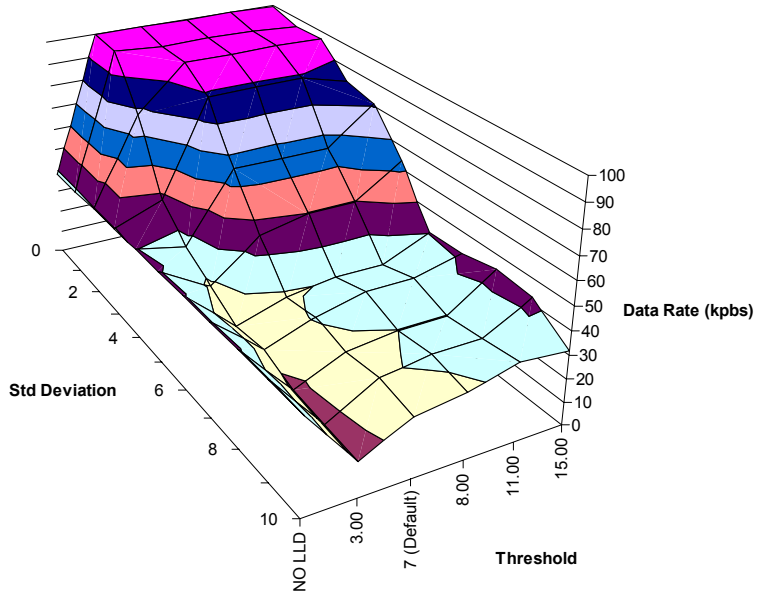

Figure 8: Data Rate vs. LLD Threshold and Std Dev, β=2.2, CBR=100kbps

Compare this figure with the two previous and future figures. As Beta values increase, the detrimental effects of adding noise becomes more pronounced.

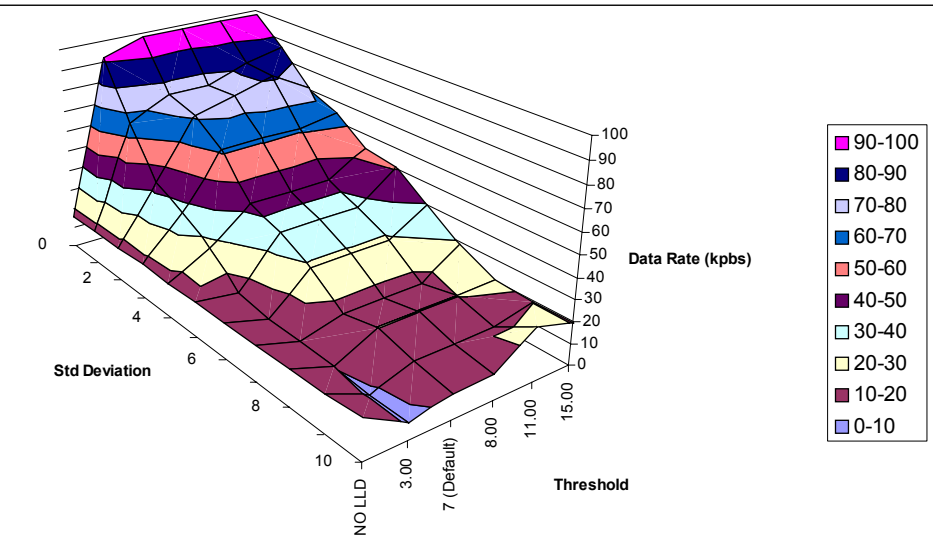

Figure 9: Data Rate vs. LLD Threshold and Std Dev, β=2.5, CBR=100kbps

Note clearly both the decrease with respect to increased noise (std Dev) in the environment. Also apparent in this figure is improved performance of some LLD method over others

performance with increasingly degraded environments is apparent. Also, the network's sensitivity to added noise in the environment clearly is influenced by several factors: the current environmental condition, the β value and current σ value, and whether or not LLD is used. Second, the contours represent the relative advantages in some implementations of LLD. In figure 9, this result is clearly visible in the downward slope in the region with σ between 4 and 7 dB. This shows the relative advantage of LLD implementation using a higher threshold value, such as 15. Again, table 3 provides a chart quantifying this performance improvement. Note that in scenarios with near ideal environments, the default setting for LLD (Threshold=7) is the best network design. In scenarios with very high environmental degradation, HELLO messaging improves performance relative to the default LLD specifications.

Table 4 compared the relative sensitivity of networks using HELLO messaging to those using LLD with varying threshold values. A similarity between the results of LLD with a threshold of 25 and networks using HELLO messaging was noted. Figure 10 is a graph comparing the performance of three network designs as the dB of stochastic variations increases. The three network implementations are no LLD, LLD with a threshold value of 7 and LLD with a threshold value of 25. First, the superiority of LLD with threshold value =7 for the range of σ=0-8 is readily apparent. Also note the relative sensitivity to increased random variation in the environment (the slope of the curves). HELLO messaging is clearly less sensitive to noise and has a slight advantage in performance in the range σ=9-10. Also note the similarity in the results between HELLO messaging and LLD with a threshold value of 25. The similarities in the network performance of networks using these two implementations held across the parameter space tested and examined in this study.

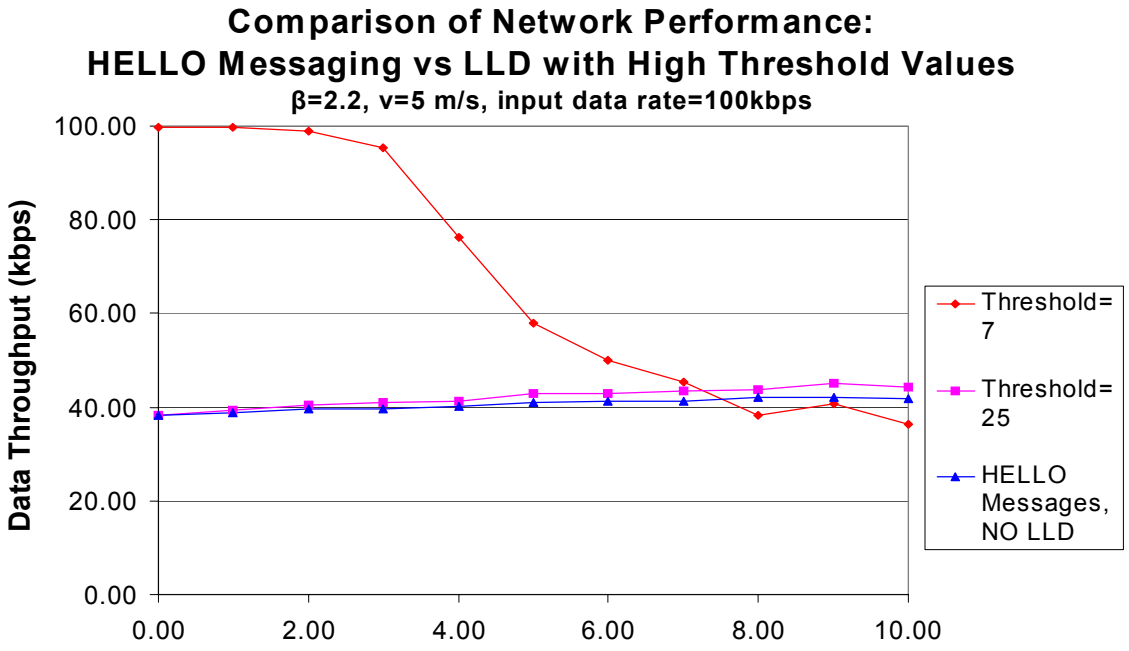

Figure 10: Note the similarity in results between LLD with Threshold=25 and HELLO Messaging. It is hypothesized that with high threshold values, RTS packets used in LLD act similar to HELLO messages

5. Discussion and Conclusions

The results from this study indicated that several changes to the existing MANET routing and MAC layer protocols may improve the performance of a MANET, especially in degraded environments representative of those encountered in the littoral during amphibious operations. Further analysis of the results was used to characterize, both quantitatively and qualitatively, the response of network performance to simulated degraded environmental conditions. Lastly, many of the factors influencing the sensitivity of a network's response to increasing noise in the environment were identified.

While the current standard implementations for the AODV routing protocol do not utilize the LLD feedback information provided by the MAC, the simulations in this study showed that using LLD consistently yielded superior network performance relative to networks operating with HELLO messaging instead of LLD. Improvements in network performance of up to 180%, on average, were observed in simulated environments. Further, testing different implementations of LLD resulted in improvements in the network throughput, relative to the default LLD implementation, of up to 30% on average. The best LLD method tested in this study used a threshold of unacknowledged packets set to slightly more than twice the default value. It was demonstrated that this method of LLD implementation offered superior performance for all scenarios except ideal and extremely degraded environments. Further, it was shown that continuing to increase the threshold value began to decrease the network performance.

The best LLD implementation for the majority (over 70%) of the scenarios tested in this study used a threshold value of fifteen (15). However in environments simulating ideal radio

propagation, using the default threshold value of seven (7) yielded superior performance. Also, in environment with very high attenuation, HELLO messaging or a LLD method with high threshold value (25) yielded the best network performance. These results suggest that the design for LLD is consistent with an assumption of ideal propagation. However, the results from this study indicate that this assumption is not valid as even slight increases in the nosie level (dB) in the environment resulted in situations where increasing the threshold value to fifteen notably improved the network performance.

The increased performance in degraded environments may be caused by the regularity of the HELLO messages that are sent out, allowing neighbors that have moved out of range of each other to identify this problem faster. Higher threshold values should make the network less sensitive to individual variations in the link between two nodes. Because LLD does not identify a reason for an unacknowledged packet, all the different problems that might cause a packet to be lost or unacknowledged are treated equally. Thus a link that is legitimately broken would be treated the same as a link that experienced a momentary burst of noise causing the packet to be unacknowledged. Many of these temporary variations or other unique situations that do not necessarily indicate that a link is broken would cause some LLD methods to "identify" false bad links. Simulated environments with more noise will have an increased number of isolated unacknowledged packets caused by random noise in the environment. This study demonstrated that a LLD method that is less sensitive to noise and other instantaneous factors may reduce the congestion in the network caused by unnecessary or premature link error transmissions. This reduced congestion can improve the overall network performance. However this study's results

also indicated that too large an increase in the threshold value will result in decreasing network performance.

In scenarios in which a very high threshold value of twenty-five (25) was simulated, the network performance relative to using a value of 15 (the overall best LLD method identified in this study) declined markedly. However, of particular interest was the observation that the performance of networks using HELLO messaging and networks using LLD with high threshold values were very similar. First, it is clear that simply increasing the threshold value will not continue to improve performance, because eventually the network will take too long to identify when link errors have occurred. However, the fact that networks using HELLO messaging (not using LLD) and networks using LLD with a high threshold behave similarly suggests that the RTS messages in LLD may begin to act similarly to HELLO messages. This is possibly explained by the fact that when the threshold value is set very high, the time-out process causes RTS packets to be sent on a regular basis, similar to HELLO messages. The results could imply that combining some of the functions of the routing and link layers in this situation might improve performance. If RTS packets act like HELLO messages in some situations in this network, then modifying the structure of the RTS packet and including more listening by neighboring nodes could further improve the network performance.

Learning that different LLD implementations will improve network performance in different simulated scenarios suggests that the development of an adaptable form of LLD may be beneficial. An adaptable LLD implementation would be software based and would change the implementation of its LLD dependent upon the environmental conditions and node mobility of the current situation. In real environments, especially those in which the military operates, the

criteria that define the propagation of radio waves in the environment will be constantly changing. These changes make the development of an adaptive protocol necessary for optimal performance in every environment. The implementation of such a LLD method would be largely in software so that such changes could happen automatically and on the fly, rather than require a hardware change-out whenever necessary.

In this project, "break" and crossover points, points at which a best type of LLD implementation was replaced by a different LLD implementation, were identified for a range of variables affecting network performance. The location and identification of these points will be important for future development and implementation of an adaptable LLD method.

As was expected, this study found that added environmental noise and interference in the propagation channel of the physical layer will significantly degrade the network's overall performance. However, the sensitivity of the network to increased environmental degradation was not linear. The results from this study were used to analyze the different factors that influenced the actual sensitivity of the network's performance to increased environmental degradation. A number of factors were identified as influencing the networks performance: the current environmental condition, the use of LLD, the type of data connection between nodes, the input data rate and the node velocity. The results suggest that a MANET has three types of sensitivity regions. Initially, if all conditions are ideal for network functionality (lower CBR input data rate, ideal environment, slower velocity) the network has a buffer zone in which it is not very sensitive to additional environmental degradation. In these scenarios, the network is already operating at maximum throughput and additional noise will not measurably affect the network's performance. The network, in such cases, is essentially insensitive to added

environmental degradation. However, this region is very small and increasing any of the factors that lessen network functionality will make the network performance sensitive to increased noise. Conversely, a network in which the factors are markedly impeding network functionality has a lower performance, but is also less sensitive to increased noise in the environment. While there does not appear to be a minimum level of network performance, the rate of decline in network performance with additional noise does decrease relative to the previous sensitivity region.

Also of interest to future studies was the observation of the similarities between the effects that node mobility and environmental conditions have on network performance. Past studies [9,15] have studied network performance in a variety of mobility scenarios, using the mobility of the nodes in the network as the stress on network performance. These studies have concluded that increased mobility decreased a network's performance and that different routing protocols were each optimized for different types of environments. Clearly there are similarities between the reaction of a MANET to node mobility and its reaction to increased environmental degradation. There is also some evidence that these two stresses are not entirely independent as it was shown that increased mobility will also increase a network's sensitivity to environmental degradation. Future studies that examine routing protocols and LLD methods for use in MANET may be designed to evaluate a network's performance while stressing the network by varying both node mobility and the environmental condition. Clearly environmental degradation has a non-linear impact on network performance, possibly affecting results from studies that assume a constant, ideal environment.

Lastly, it is useful to note some of the limitations and assumptions of this study. The objective of the testing was to record and describe the changes to the network's performance caused by simulated degraded environments. No attempts were made to improve the network performance by designing changes to the hardware required to run the network. In particular, methods to improve bandwidth were not studied because such methods would require fundamental changes to the hardware and physical layer. Such issues, while potentially valuable to network performance, were left to others for future research. Research into increasing the network bandwidth would be mutually compatible with the conclusions found in this study because of the abstraction offered by the OSI network layer model. However, the effective bandwidth available for data transmission can be improved indirectly through improvements in LLD and routing design which minimize the bandwidth required for routing functions.

6. Implications for Future Investigations

The results of this study suggest a number of area of research for future studies to develop. Both the environmental degradation and node mobility appear to influence and degrade a network's performance in similar ways. However, this study has shown that there is likely an interaction between a network's sensitivity to noise and a network's sensitivity to increased mobility. It may be very useful to analyze a network's performance with respect to both environmental degradation and node mobility to determine how these two network stresses interact and influence each other, if at all. Further simulations using more complex node mobility scenarios will validate that results can be extrapolated to all types of mobility scenarios.

A sensitivity study into all the factors that influence a network's sensitivity to increased noise in the environment would benefit this field of research. The present study demonstrated when MANET protocols and LLD methods optimized for scenarios based on too many assumptions about the factors that influence network performance can cause these designs to be sub-optimal in all but the most ideal situations. Knowing exactly what factors influence a network's performance and the sensitivity to increased noise (network performance and its derivative) will ensure that future studies and network protocol designs incorporate better assumptions about what variables and factors need to be tested and those that can be assumed to be ideal.

While more simulation studies can be run, the implementation and then testing of the proposed LLD methods in hardware would be useful. It is likely that currently LLD methods are implemented in hardware. Modifications to the MAC layer, would then require a redesign of the Network Interface Card (NIC). Since the LLD methods in this study all dealt with simply

changing threshold values, a redesign of the NIC should implement this threshold counter in software. It is faster and easier to modify software, making future testing or real networks using different LLD implementations faster. Also, an adaptable protocol would have to have the counter implemented in software so that it could be changed on the fly in the midst of an operating network.

Further work may examine the potential for improved LLD methods in the MAC layer other than using this simple method of unacknowledged packets and a threshold value. Further development of a more robust LLD method using signal strength or a signal to noise ratio method could potentially improve network performance by detecting errors in the network prior to dropping packets. Such anticipatory methods would minimize the network congestion caused by the retransmission of dropped packets and also decrease the time delay in identifying a bad link. A LLD method based on signal to noise ratio should incorporate both the noise in the environment (which is modeled by a combination of the β and σ values in the log-normal shadowing model) and trends in the signal strength. By examining the trends in the signal strength, instead of individual measurements, the LLD will make the network less sensitive to individual "bursts" of noise. However, the inclusion of the signal to noise ratio in this measurement would prevent this decreased sensitivity to noise from taking too long to identify broken links. Testing to optimize this LLD method should examine the number of past transmissions that should be recorded to identify a trend in performance and the appropriate criteria for deciding when a link is ready to break. The type of trends to use (whether they are linear, exponential or other) would also need to be optimized. Clearly, un-optimized criteria

either result in a slow response to link errors or results in too many falsely identified link errors, both of which have a detrimental effect on the network's performance.

Lastly, tests validate the log normal shadowing model, and characterize the β & σ values of real environments likely to be encountered during amphibious operations would be useful. This study assumed that the log-normal shadowing model of environmental degradation was a valid method of modeling the environment. Validating this assumption and knowing values that characterize specific littoral environments would be beneficial to future studies attempting to simulate networks operating in degraded environments.

This study defined a research area in which future work clearly has a potential to improve network routing with particular application to military amphibious operations. More robust and effective communications will directly translate into the increased effectiveness of assault forces.

Appendices

Appendix A: Source Code for Simulation Scenario

Below is one example of the source code that was used to implement the mobility and environmental aspects of a scenario being simulated in NS2. It is written in Tool Command Language (TCL) and interpreted in NS2. Because each simulated environment has a different set of criteria, each simulated scenario has different source code. The example below is from a scenario with the following specifications: node velocity of 5 m/s, data input rate of 100kbps, packet size of 1000 bytes, and simulated environmental criteria of β=2.0 and σ=0.00

The other criteria, relating to the MAC and routing layers are determined within the NS2 source code and require recompilation of NS2 between simulations.

```
# LLD-test_Shadow.txt
#unfortunately, there is no easy way to have the ns simulation detect whether LLD is on or off
...
#thus I recompile and rerun the data, being careful to properly label the resulting output files.

#starting in a diamond formation, two of the nodes move away from each other ... examine how
#long it takes the two protocols with/without LLD to notice the difference and route through the
other nodes
#data is recorded in files with the three numbers being:  path loss exponent, std deviation,
velocity


#use of Shadowing propogation model to examine the effect of environmental factors on performance
Propagation/Shadowing set pathlossExp_ 2.00         ;# path loss exponent
Propagation/Shadowing set std_db_ 0.00              ;# shadowing deviation (dB)
Propagation/Shadowing set dist0_ 1          ;# reference distance (m)
Propagation/Shadowing set seed_ 0           ;# seed for RNG
set val(velocity)      5.0                  ;# velocity (m/s) of nodes
set val(data_rate)     100kb                ;# cbr data rate
set val(packet_size)   1000                 ;# cbr packet size

#for labeling purposes ... will print specifics to the screen
puts  "AODV, Beta=2.00, Std Dev=0.00, v=5, for 100s"

# ======================================================================
# Define options (wireless variables)
# ======================================================================
set val(chan)           Channel/WirelessChannel     ;# channel type
set val(prop)           Propagation/Shadowing       ;# 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)             10                  ;# number of mobilenodes
set val(rp)             AODV                ;# routing protocol

# ======================================================================
# Main Program
```

```
# ======================================================================

#
# Initialize Global Variables
#
set ns_                [new Simulator]
$ns_ use-newtrace

#setup traces (NAM and standard trace)

set tracefd     [open AODV-B2.00-stdev0.00-v5-t100_cbr.tr w]
$ns_ trace-all $tracefd

# set up topography object
set topo       [new Topography]

$topo load_flatgrid 1000 1000

#
# Create God
#
create-god $val(nn)

#
#  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) \
                         -propType $val(prop) \
                         -phyType $val(netif) \
                         -channelType $val(chan) \
                         -topoInstance $topo \
                         -agentTrace ON \
                         -routerTrace OFF \
                         -macTrace OFF \
                         -movementTrace OFF

        for {set i 0} {$i < $val(nn) } {incr i} {
                set node_($i) [$ns_ node]
                $node_($i) random-motion 1            ;# disable random motion
        }

#
# Provide initial (X,Y, for now Z=0) co-ordinates for mobilenodes

$node_(0) set X_ 500
$node_(0) set Y_ 550
$node_(0) set Z_ 0.0

$node_(1) set X_ 500
$node_(1) set Y_ 450
$node_(1) set Z_ 0.0

$node_(2) set X_ 150
$node_(2) set Y_ 500
$node_(2) set Z_ 0

$node_(3) set X_ 250
$node_(3) set Y_ 500
$node_(3) set Z_ 0

$node_(4) set X_ 350
$node_(4) set Y_ 500
$node_(4) set Z_ 0

$node_(5) set X_ 450
```

```
$node_(5) set Y_ 500
$node_(5) set Z_ 0

$node_(6) set X_ 550
$node_(6) set Y_ 500
$node_(6) set Z_ 0

$node_(7) set X_ 650
$node_(7) set Y_ 500
$node_(7) set Z_ 0

$node_(8) set X_ 750
$node_(8) set Y_ 500
$node_(8) set Z_ 0

$node_(9) set X_ 850
$node_(9) set Y_ 500
$node_(9) set Z_ 0

$ns_ at 1.0 "$node_(0) setdest 990.0 550.0 $val(velocity)"
$ns_ at 1.0 "$node_(1) setdest 10.00 450.0 $val(velocity)"
$ns_ at 1.0 "$node_(2) setdest 150.0 500.0 $val(velocity)"
$ns_ at 1.0 "$node_(3) setdest 250.0 500.0 $val(velocity)"
$ns_ at 1.0 "$node_(4) setdest 350.0 500.0 $val(velocity)"
$ns_ at 1.0 "$node_(5) setdest 450.0 500.0 $val(velocity)"
$ns_ at 1.0 "$node_(6) setdest 550.0 500.0 $val(velocity)"
$ns_ at 1.0 "$node_(7) setdest 650.0 500.0 $val(velocity)"
$ns_ at 1.0 "$node_(8) setdest 750.0 500.0 $val(velocity)"
$ns_ at 1.0 "$node_(9) setdest 850.0 500.0 $val(velocity)"


# Setup traffic flow between nodes


##### FOR CBR TRAFFIC SOURCE #####
set source [new Agent/UDP]
$ns_ attach-agent $node_(0) $source

set sink [new Agent/UDP]
$ns_ attach-agent $node_(1) $sink
$ns_ connect $source $sink

set traffic [new Application/Traffic/CBR]
$traffic set packetSize_ $val(packet_size)
$traffic set rate_ $val(data_rate)
$traffic attach-agent $source
$ns_ at 2.0 "$traffic start"


##### FOR FTP/TCP TRAFFIC SOURCE #####
# TCP connections between node_(0) and node_(1)
#set source [new Agent/TCP]
#$ns_ attach-agent $node_(0) $source
#
#set sink [new Agent/TCPSink]
#$ns_ attach-agent $node_(1) $sink
#$ns_ connect $source $sink
#
#set ftp [new Application/FTP]
#$ftp attach-agent $source
#$ns_ at 2.0 "$ftp start"
#

# Tell nodes when the simulation ends
#
for {set i 0} {$i < $val(nn) } {incr i} {
    $ns_ at (100.1) "$node_($i) reset";
}

$ns_ at 10 "puts \"SIMULATION TIME:  10 seconds ...\""
$ns_ at 20 "puts \"SIMULATION TIME:  20 seconds ...\" "
$ns_ at 30 "puts \"SIMULATION TIME:  30 seconds ...\" "
```

```
$ns_ at 40 "puts \"SIMULATION TIME:  40 seconds ...\""
$ns_ at 60 "puts \"SIMULATION TIME:  60 seconds ...\" "
$ns_ at 70 "puts \"SIMULATION TIME:  70 seconds ...\""
$ns_ at 80 "puts \"SIMULATION TIME:  80 seconds ...\" "
$ns_ at 90 "puts \"SIMULATION TIME:  90 seconds ...\" "
$ns_ at 100 "stop"
$ns_ at 100.01 "puts \"NS EXITING...\" ; $ns_ halt"

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

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

Appendix B: Source Code for Trace File Analysis Program

Below is the source code for the data analysis awk program used to analyze the NS2 trace files created during simulations. It is written in awk, an interpreter language which is specifically designed for analyzing text files and strings. During simulations, NS2 writes a separate line to the trace file for every recorded network action. This code reads a line of text, uses if-then statements to determine what action took place and maintains a running total of the number of each action.

Data is recorded on the number of: total packets sent, data packets sent, routing packets, ack packets sent, request packets sent, total packets dropped, routing packets dropped, data packets dropped, hello packets sent and total errors found during the simulation.

```
BEGIN{  drop=0;
        all_drop=0;
        data_drop=0;
        route_drop=0;
        reply_drop=0;
        request_drop=0;
        ack_drop=0;

        data_received=0;

        all_sent=0;
        data_sent=0;
        route_sent=0;
        hello_sent=0;
        ack_sent=0;

        errors=0;
        time=0;
        print("");     }

{
#examining the sent packets
  if(/s -t/)  {
     if(/-Nl AGT/)     {
       if(/-It tcp/ || /-It cbr/) { data_sent++; }
       if(/-It ack/) { ack_sent++; }  }
     if(/-It AODV/ || /-It DSR/)      {
        route_sent++;
        if(/REPLY/) {reply_sent++;}
        if(/HELLO/) { hello_sent++; }
        if(/REQUEST/) {request_sent++;}
        if(/ERROR/)   {
#          print(drop " drops occurred before reported error at " $3 " by " $5 );
           drop=0;
           errors ++;  } } }

#examining the received packets
  if(/r -t/ && /-Nl AGT/ && !/-It ack/ ) {data_received++;}
```

```
#examining the dropped packets
  if(/d -t/)  {
#   print("packet dropped:  sent from " $5 " at " $3 " (" $31 ")");
    all_drop++;
    drop++;

    if(/-It AODV/ || /-It DSR/)              {route_drop++;}
    if(/-It tcp/ || /-It cbr/)               {data_drop++;}
    if(/-It ack/)                     {ack_drop++;}
}

}
END{   printf(data_sent+route_sent+ack_sent  "\t"  route_sent  "\t"  hello_sent  "\t"  request_sent
"\t" reply_sent "\t" errors "\t" data_received "\t" data_sent "\t\t" all_drop "\t" data_drop "\t"
route_drop "\t" ack_drop);
}
```

Appendix C: Selected Source Code from mac-802_11.cc

The code below is two selected functions from the mac-802_11.cc file used to implement Link Layer Detection in the 802.11 MAC layer protocol in NS2. These functions deal with the actions of the link layer after an RTS message or data packet is sent but not acknowledged. These functions maintain a count of the number of packets that fail to be acknowledged and call another function to pass data to the routing layer if this threshold value is exceeded. The exact threshold value is a one-line definition statement included in the mac-802_11.h file. The code below also reflects modifications made to the source code to aid print error messages to the console output to aid in debugging and further source code modification. Note that these changes can be commented out with definition statements included in the .h file.

```
/* ======================================================================
   Retransmission Routines
   ====================================================================== */
void
Mac802_11::RetransmitRTS()
{
        assert(pktTx_);
        assert(pktRTS_);
        assert(mhBackoff_.busy() == 0);

        macmib_->RTSFailureCount++;

        #ifdef FLEISCHAKER_OUTPUT
        fprintf(stderr,   "(%d)   Failed   RTS's:     %d   (%d   in   a   row)\t\t%.2f\n",   index_,
macmib_->RTSFailureCount, ssrc_,Scheduler::instance().clock());

                #ifdef FLEISCHAKER_STDOUT
                fprintf(stdout, "(Failed RTS's:   %d (%d in a row)\n", macmib_->RTSFailureCount,
ssrc_);
                #endif


        #endif  //fleischaker_output

        ssrc_ += 1;                     // STA Short Retry Count

        if(ssrc_ >= macmib_->ShortRetryLimit) {

                #ifdef FLEISCHAKER_OUTPUT
                        fprintf(stderr,  "***(%d)....RTS  threshold  exceded:%x\t\t%.2f\n",  index_,
pktRTS_, Scheduler::instance().clock());
                        #ifdef FLEISCHAKER_STDOUT
                                fprintf(stdout,  "***(%d)....RTS  threshold  exceded:%x\t\t%.2f\n",
index_, pktRTS_, Scheduler::instance().clock());
                        #endif

                #endif  //fleischaker_output
```

```
                discard(pktRTS_, DROP_MAC_RETRY_COUNT_EXCEEDED); pktRTS_ = 0;
                /* tell the callback the send operation failed
                   before discarding the packet */
                hdr_cmn *ch = HDR_CMN(pktTx_);
                if (ch->xmit_failure_) {
                         /*
                          *  Need to remove the MAC header so that
                          *  re-cycled packets don't keep getting
                          *  bigger.
                          */
                         ch->size() -= ETHER_HDR_LEN11;
                         ch->xmit_reason_ = XMIT_REASON_RTS;
                         ch->xmit_failure_(pktTx_->copy(),
                                           ch->xmit_failure_data_);
                       }



                discard(pktTx_, DROP_MAC_RETRY_COUNT_EXCEEDED); pktTx_ = 0;
                ssrc_ = 0;
                rst_cw();
        } else {

                //**********************
                #ifdef FLEISCHAKER_OUTPUT
                        fprintf(stdout, "(%d)...retransmitting RTS:%x\n",index_,pktRTS_);
                #endif  //fleischaker_verbose

                struct rts_frame *rf;
                rf = (struct rts_frame*)pktRTS_->access(hdr_mac::offset_);
                rf->rf_fc.fc_retry = 1;

                inc_cw();
                mhBackoff_.start(cw_, is_idle());
        }
}

void
Mac802_11::RetransmitDATA()
{
        struct hdr_cmn *ch;
        struct hdr_mac802_11 *mh;
        u_int32_t *rcount, *thresh;

        assert(mhBackoff_.busy() == 0);

        assert(pktTx_);
        assert(pktRTS_ == 0);

        ch = HDR_CMN(pktTx_);
        mh = HDR_MAC802_11(pktTx_);

        /*
         *  Broadcast packets don't get ACKed and therefore
         *  are never retransmitted.
         */
        if((u_int32_t)ETHER_ADDR(mh->dh_da) == MAC_BROADCAST) {
                Packet::free(pktTx_); pktTx_ = 0;

                /*
                 * Backoff at end of TX.
                 */
                rst_cw();
                mhBackoff_.start(cw_, is_idle());

                #ifdef FLEISCHAKER_VERBOSE
                        f p r i n t f ( s t d e r r ,      " L o s t      B r o a d c a s t
message\t%.2f\n",Scheduler::instance().clock());
                        #ifdef FLEISCHAKER_STDOUT
                               f p r i n t f ( s t d o u t ,      " L o s t      B r o a d c a s t
message\t%.2f\n",Scheduler::instance().clock());
                        #endif
```

```
		#endif

		return;
	}

	macmib_->ACKFailureCount++;

	#ifdef FLEISCHAKER_OUTPUT
	//NATHAN's test code- 14 FEB2003
	//apparently this code is not used often, largely because
	//the real errors are detected with the RTS/CTS messages that
	//probably don't get answered
		fprintf(stderr, "LOST ACK of DATA, number:
%d\t%.2f\n",macmib_->ACKFailureCount,Scheduler::instance().clock());
	// END OF NATHAN's TEST CODE
	#endif  //fleischaker_output


	if((u_int32_t) ch->size() <= macmib_->RTSThreshold) {
		rcount = &ssrc_;
		thresh = &macmib_->ShortRetryLimit;
	}
	else {
		rcount = &slrc_;
		thresh = &macmib_->LongRetryLimit;
	}


	(*rcount)++;


	if(*rcount > *thresh) {
		macmib_->FailedCount++;


		#ifdef FLEISCHKER_OUTPUT
			//NATHAN's test code- 14 FEB2003
			//apparently this code is not used often, largely because
			//the real errors are detected with the RTS/CTS messages that
			//probably don't get answered
			printf(stdout, "FAILED LINK number:  %d", macmib_->FailedCount);
			// END OF NATHAN's TEST CODE
		#endif  //fleischaker_output

		/* tell the callback the send operation failed
		   before discarding the packet */
		hdr_cmn *ch = HDR_CMN(pktTx_);
		if (ch->xmit_failure_) {
			 ch->size() -= ETHER_HDR_LEN11;
			 ch->xmit_reason_ = XMIT_REASON_ACK;
			 ch->xmit_failure_(pktTx_->copy(),
					   ch->xmit_failure_data_);
		 }

		discard(pktTx_, DROP_MAC_RETRY_COUNT_EXCEEDED); pktTx_ = 0;
		printf("(%d)DATA discarded: count exceeded\n",index_);
		*rcount = 0;
		rst_cw();
	}
	else {
		struct hdr_mac802_11 *dh;
		dh = HDR_MAC802_11(pktTx_);
		dh->dh_fc.fc_retry = 1;

		sendRTS(ETHER_ADDR(mh->dh_da));
		//printf("(%d)retxing data:%x..sendRTS..\n",index_,pktTx_);
		inc_cw();
		mhBackoff_.start(cw_, is_idle());
	}
}
```

Appendix D: Selected Source Code from *Shadows*

The code below is a selection from the Visual C++ source code for *Shadows.exe*, the program written to automate part of the data collection process in this study. As noted in Appendix A, each simulation test requires either a recompilation of NS2 or a new simulation source code which modifies the values appropriately. This program, *Shadows*, produces a collection the source code files modified in the appropriate places as defined by the user's input to a Graphical User Interface. It also produces two batch files, one to run the NS2 simulator with each of the generated scenarios and one to run the data analysis program on the resulting trace files. The program allows the user to specify a range of values which several different variables (β, σ, data rate, packet size, and velocity) will take. A separate scenario source file is then created for every combination of these variables, and the batch files are created accordingly.

```
// shadowsDlg.cpp : implementation file
/////////////////////////////////////////////////////////////////////////////
// CShadowsDlg dialog

CShadowsDlg::CShadowsDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CShadowsDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CShadowsDlg)
	m_check_db_vary = FALSE;
	m_check_dr_vary = FALSE;
	m_check_pe_vary = FALSE;
	m_check_ps_vary = FALSE;
	m_edit_db_base = _T("0.0");
	m_edit_db_interval = _T("1.0");
	m_edit_db_n_intervals = _T("21");
	m_edit_db_offset = _T("0.0");
	m_edit_dr_base = _T("100");
	m_edit_dr_interval = _T("0");
	m_edit_dr_n_intervals = _T("0");
	m_edit_dr_offset = _T("0");
	m_edit_pe_base = _T("2.0");
	m_edit_pe_interval = _T("0.1");
	m_edit_pe_n_intervals = _T("11");
	m_ediit_pe_offset = _T("0.0");
	m_edit_ps_base = _T("1000");
	m_edit_ps_interval = _T("0");
	m_edit_ps_n_intervals = _T("0");
	m_edit_ps_offset = _T("0");
	m_check_v_vary = FALSE;
	m_edit_v_base = _T("10");
	m_edit_v_interval = _T("10");
	m_edit_v_n_intervals = _T("2");
```

```
        m_edit_v_offset = _T("0");
        //}}AFX_DATA_INIT
        // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
        m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}


void CShadowsDlg::pe_gear (double pe, double db, long dr, long ps, long v)
{
        if (m_check_pe_vary)
        {
                pe = pe + atof(this->m_ediit_pe_offset);
                long N = atoi (m_edit_pe_n_intervals);
                for (long s = 0; s < N; ++s)
                {
                        db_gear (pe, db, dr, ps, v);
                        pe = pe + atof (m_edit_pe_interval);
                }
        }
        else
        {
                db_gear (pe, db, dr, ps, v);
        }
}

void CShadowsDlg::db_gear (double pe, double db, long dr, long ps, long v)
{
        if (m_check_db_vary)
        {
                db = db + atof(m_edit_db_offset);
                long N = atoi (m_edit_db_n_intervals);
                for (long s = 0; s < N; ++s)
                {
                        dr_gear (pe, db, dr, ps, v);
                        db = db + atof (m_edit_db_interval);
                }
        }
        else
        {
                dr_gear (pe, db, dr, ps, v);
        }
}

void CShadowsDlg::dr_gear (double pe, double db, long dr, long ps, long v)
{
        if (m_check_dr_vary)
        {
                dr = dr + atoi (m_edit_dr_offset);
                long N = atoi (m_edit_dr_n_intervals);
                for (long s = 0; s < N; ++s)
                {
                        ps_gear (pe, db, dr, ps, v);
                        dr = dr + atoi (m_edit_dr_interval);
                }
        }
        else
        {
                ps_gear (pe, db, dr, ps, v);
        }
}

void CShadowsDlg::ps_gear (double pe, double db, long dr, long ps, long v)
{
        if (m_check_ps_vary)
        {
                ps = ps + atoi (m_edit_ps_offset);
                long N = atoi (m_edit_ps_n_intervals);
                for (long s = 0; s < N; ++s)
                {
                        v_gear (pe, db, dr, ps, v);
                        ps = ps + atoi (m_edit_ps_interval);
                }
```

```
        }
        else
        {
                v_gear (pe, db, dr, ps, v);
        }
}

void CShadowsDlg::v_gear (double pe, double db, long dr, long ps, long v)
{
        if (m_check_v_vary)
        {
                v = v + atoi (m_edit_v_offset);
                long N = atoi (m_edit_v_n_intervals);
                for (long s = 0; s < N; ++s)
                {
                        jigsaw (pe, db, dr, ps, v);
                        v = v + atoi (m_edit_v_interval);
                }
        }
        else
        {
                jigsaw (pe, db, dr, ps, v);
        }
}

void CShadowsDlg::jigsaw (double pe, double db, long dr, long ps, long v)
{
        // v*t = 500
        long t = 500 / v;

        // determine fname
        // format: pe_N_db_N_dr_N_ps_N_v_N
        long peN2 = (long) (pe * 100.0);
        long dbN2 = (long) (db * 100.0);

        // batch file
        char dest[160];
        sprintf (dest, "c:\\shadows\\output\\pe_%d_db_%d_dr_%d_ps_%d_v_%d.txt",peN2,dbN2,dr,ps,v);
        // remove dest
        _unlink(dest);
        // create dest
        FILE *f2 = fopen(dest,"a+b");
        if (f2 == NULL)
                return;
        char line [160];
        jigsaw_append_file (f2, "c:\\shadows\\jigsaws\\jigsaw1.txt");

        // Propagation/Shadowing set pathlossExp_ 2.5       ;# path loss exponent \r\n
        sprintf (line, "Propagation/Shadowing set pathlossExp_ %-.2f\t\t;# path loss exponent
\r\n",pe);
        jigsaw_append_line (f2, line);

        // Propagation/Shadowing set std_db_ 4.0            ;# shadowing deviation (dB) \r\n
        sprintf (line, "Propagation/Shadowing set std_db_ %-.2f\t\t;# shadowing deviation (dB)
\r\n",db);
        jigsaw_append_line (f2, line);

        jigsaw_append_file (f2, "c:\\shadows\\jigsaws\\jigsaw2.txt");

        // set val(velocity)  20.0                  ;# velocity (m/s) of nodes \r\n
        sprintf (line, "set val(velocity)\t%d.0\t\t\t;# velocity (m/s) of nodes \r\n",v);
        jigsaw_append_line (f2, line);

        // set val(data_rate) 100kb                 ;# cbr data rate
        sprintf (line, "set val(data_rate)\t%dkb\t\t\t;# cbr data rate\r\n",dr);
        jigsaw_append_line (f2, line);

        // set val(packet_size)       1000                  ;# cbr packet size
        sprintf (line, "set val(packet_size)\t%d\t\t\t;# cbr packet size\r\n",ps);
        jigsaw_append_line (f2, line);

        jigsaw_append_file (f2, "c:\\shadows\\jigsaws\\jigsaw3.txt");
```

```
        // puts  "AODV LLD ON, Beta=2.5, Std Dev=4.0, v=20, for 25s" \r\n
        sprintf (line, "puts  \"AODV, Beta=%-.2f, Std Dev=%-.2f, v=%d, for %ds\" \r\n",pe,db,v,t);
        jigsaw_append_line (f2, line);

        jigsaw_append_file (f2, "c:\\shadows\\jigsaws\\jigsaw4.txt");

        // set tracefd     [open AODV-B2.5-stdev4.0-v20-t25_with-LLD_cbr.tr w] \r\n
        sprintf  (line,  "set  tracefd          [open  AODV-B%-.2f-stdev%-.2f-v%d-t%d_cbr.otr  w]
\r\n",pe,db,v,t);
        jigsaw_append_line (f2, line);

        jigsaw_append_file (f2, "c:\\shadows\\jigsaws\\jigsaw5.txt");

        //     $ns_ at (25.1) "$node_($i) reset"; \r\n
        sprintf (line, "    $ns_ at (%d.1) \"$node_($i) reset\"; \r\n",t);
        jigsaw_append_line (f2, line);

        jigsaw_append_file (f2, "c:\\shadows\\jigsaws\\jigsaw6.txt");

        // $ns_ at 25 "stop" \r\n
        sprintf (line, "$ns_ at %d \"stop\" \r\n",t);
        jigsaw_append_line (f2, line);

        // $ns_ at 25.01 "puts \"NS EXITING...\" ; $ns_ halt" \r\n
        sprintf (line, "$ns_ at %d.01 \"puts \\\"NS EXITING...\\\" ; $ns_ halt\" \r\n",t);
        jigsaw_append_line (f2, line);

        jigsaw_append_file (f2, "c:\\shadows\\jigsaws\\jigsaw7.txt");

        // append dest to pre-process batch
        FILE *fpre = fopen ("c:\\shadows\\output\\pre_process.bat","a+b");
        if (fpre == NULL)
                return;
        char preline[320];
        // NOTE: change formatting on preline to change output to pre-process batch file
        sprintf(preline,"ns pe_%d_db_%d_dr_%d_ps_%d_v_%d.txt",peN2,dbN2,dr,ps,v);
        strcat(preline,"\r\n");
        fwrite(preline, strlen(preline), 1, fpre);
        fclose (fpre);

        // append dest to post-process batch
        FILE *fpost = fopen ("c:\\shadows\\output\\post_process.bat","a+b");
        if (fpost == NULL)
                return;
        char postline[320];
        // NOTE: change formatting on preline to change output to pre-process batch file
        sprintf     (postline,     "awk     -f     data_sent_drop_output-excel-cbr.awk
AODV-B%-.2f-stdev%-.2f-v%d-t%d_cbr.tr >> data.txt\r\n",pe,db,v,t);
        fwrite(postline, strlen(postline), 1, fpre);
        fclose (fpost);

        fclose (f2);
}


void CShadowsDlg::jigsaw_append_file (FILE *f2, char *src)
{
        FILE *f1 = fopen (src, "rb");
        ASSERT (f1 != NULL);

        fseek(f1,0,SEEK_END);
        long position = ftell(f1);
        long len = position;

        fseek(f1,0,SEEK_SET);
        long len64 = len / 160;

        long i, s;
        char msz_final[161];

        for (i=0;i<len64;++i)
```

```
        {
                s = fread(msz_final, 160, 1, f1);
                if (s != 0)
                        fwrite(msz_final, 160, 1, f2);
        }
        len -= len64 * 160;

        if (len != 0)
        {
                s = fread(msz_final, len, 1, f1);
                if (s != 0)
                        fwrite(msz_final, len, 1, f2);
        }
        fclose (f1);
}

void CShadowsDlg::jigsaw_append_line (FILE *f2, char *line)
{
        long s = fwrite(line, strlen(line), 1, f2);
        if (s == 0)
                TRACE ("null\r\n");
        else
                TRACE (line);
}
```

Appendix E: Nomenclature of Trace and Scenario File Definition Files

The format used for naming the scenario definition file is:

pe_<*β value*>_db_<*σ value*>_dr_<*data rate in kbps*>_ps_<*packet size in bytes*>_v_<*value of node velocity*>.txt

ex: `pe_200_db_300_dr_100_ps_1000_v_5.txt`

The format used for naming the resulting trace file is:

<*routing protocol*>-B <*β value*>-stdev <*σ value*>-v <*value of node velocity*>-t<*time*>_<*cbr or tcp*>.tr

ex: `AODV-B2.00-stdev3.00-v5-t100_cbr.tr`

Note that the trace file naming system does not incorporate the packet size or data rate information. During this study, the trace files were saved initially to separate folders and then separate CDs. Each folder of CD contained only trace files from scenarios with specific criteria, and they were labeled appropriately.

The following examples are from the batch files produced by *shadows*. Also noted briefly is the syntax used for entering data into the command console:

Example line from `pre_process.bat`:

```
ns AODV-B2.00-stdev3.00-v5-t100_cbr.tr
```

ns <*filename.txt*>: calls the NS2 executable and simulates the scenario defined in *filename.txt*

Example line from `post_process.bat`:

```
awk -f data_sent_drop_output-excel-cbr.awk AODV-B2.50-stdev9.00-v5-t100_cbr.tr >> data.txt
```

awk -f <*awk file*> <*input file*> >> <*output stream*>: call the awk executable and specifies the awk program is *awk file*; *input file* is the trace file to be read and *output stream* is the file into where the output is piped.

Appendix F: Comparison of LLD Methods

Beta=2.0

							Comparison to using Threshold Value=7					Comparison to using HELLO messages						
Std Dev	NO LLD	3.00	7.00	8.00	11.00	15.00	3.00	8.00	11.00	15.00	15 %improvement	3.00	7.00	7 %improvement	8.00	11.00	15.00	15 %improvement
0.00	59.43	89.55	89.14	89.14	89.14	88.90	0.41	0.00	0.00	-0.24	-0.27	30.12	29.71	50.00	29.71	29.71	29.47	49.59
1.00	59.43	91.27	89.63	89.63	89.63	89.06	1.63	0.00	0.00	-0.57	-0.64	31.84	30.20	50.82	30.20	30.20	29.63	49.86
2.00	59.35	81.22	99.51	99.51	99.51	88.08	-18.29	0.00	0.00	-11.43	-11.48	21.88	40.16	67.68	40.16	40.16	28.73	48.42
3.00	59.10	74.20	99.43	99.43	99.43	99.92	-25.22	0.00	0.00	0.49	0.49	15.10	40.33	68.23	40.33	40.33	40.82	69.06
4.00	58.86	61.63	91.51	91.51	91.51	99.27	-29.88	0.00	0.00	7.76	8.47	2.78	32.65	55.48	32.65	32.65	40.41	68.65
5.00	58.53	62.20	77.31	77.31	77.31	99.10	-15.10	0.00	0.00	21.80	28.19	3.67	18.78	32.08	18.78	18.78	40.57	69.32
6.00	57.31	52.41	71.59	71.59	71.59	83.92	-19.18	0.00	0.00	12.33	17.22	-4.90	14.29	24.93	14.29	14.29	26.61	46.44
7.00	56.90	51.51	75.27	75.27	75.27	74.42	-23.76	0.00	0.00	-0.85	-1.13	-5.39	18.37	32.28	18.37	18.37	17.52	30.79
8.00	56.16	46.69	73.55	73.55	73.55	69.47	-26.86	0.00	0.00	-4.08	-5.55	-9.47	17.39	30.96	17.39	17.39	13.31	23.69
9.00	55.43	43.18	66.04	66.04	66.04	70.20	-22.86	0.00	0.00	4.16	6.30	-12.24	10.61	19.15	10.61	10.61	14.78	26.66
10.00	54.86	41.47	62.20	62.20	71.02	64.82	-20.73	0.00	8.82	2.61	4.20	-13.39	7.35	13.39	7.35	16.16	9.96	18.15
slope	**-0.49**	**-5.37**	**-3.41**	**-3.41**	**-3.01**	**-2.89**			**average**		***7.28***			***40.45***				***45.51***

Beta=2.1

Std Dev	NO LLD	3.00	7.00	8.00	11.00	15.00	3.00	8.00	11.00	15.00	%improvement	3.00	7.00	7 %improvement	8.00	11.00	15.00	15 %improvement
0.00	52.16	99.92	99.76	99.76	99.67	99.67	0.16	0.00	-0.08	-0.08	-0.08	47.76	47.59	91.24	47.59	47.51	47.51	91.08
1.00	53.06	97.96	99.35	99.35	99.84	97.96	-1.39	0.00	0.49	-1.39	-1.40	44.90	46.29	87.23	46.29	46.78	44.90	84.62
2.00	53.63	90.29	99.27	99.27	99.02	99.43	-8.98	0.00	-0.24	0.16	0.16	36.65	45.63	85.08	45.63	45.39	45.80	85.39
3.00	53.22	72.82	97.96	97.96	95.43	98.04	-25.14	0.00	-2.53	0.08	0.08	19.59	44.73	84.05	44.73	42.20	44.82	84.20
4.00	53.14	59.51	64.33	84.05	76.24	95.18	-4.82	19.72	11.92	30.86	47.97	6.37	11.18	21.04	30.91	23.10	42.04	79.11
5.00	51.59	54.12	76.00	76.00	58.04	80.33	-21.88	0.00	-17.96	4.33	5.69	2.53	24.41	47.31	24.41	6.45	28.73	55.70
6.00	51.59	42.86	68.33	68.33	50.04	78.44	-25.47	0.00	-18.29	10.12	14.80	-8.73	16.73	32.44	16.73	-1.55	26.85	52.04
7.00	50.45	34.86	68.98	68.98	45.47	78.44	-34.12	0.00	-23.51	9.46	13.72	-15.59	18.53	36.73	18.53	-4.98	27.99	55.49
8.00	50.04	39.10	57.55	57.55	38.12	76.72	-18.45	0.00	-19.43	19.16	33.30	-10.94	7.51	15.01	7.51	-11.92	26.68	53.31
9.00	49.47	34.45	52.49	52.49	40.82	56.33	-18.04	0.00	-11.67	3.84	7.31	-15.02	3.02	6.11	3.02	-8.65	6.86	13.86
10.00	49.14	28.65	48.98	48.98	47.43	52.41	-20.33	0.00	-1.55	3.43	7.00	-20.49	-0.16	-0.33	-0.16	-1.71	3.27	6.64
slope	**-0.43**	**-7.79**	**-5.64**	**-5.82**	**-7.33**	**-4.79**			**average**		***18.54***			***45.99***				***60.13***

Beta=2.2

Std Dev	NO LLD	3.00	7.00	8.00	11.00	15.00	3.00	8.00	11.00	15.00	%improvement	3.00	7.00	7 %improvement	8.00	11.00	15.00	15 %improvement
0.00	38.37	99.84	99.67	99.67	99.67	99.35	0.16	0.00	0.00	-0.33	-0.33	61.47	61.31	159.79	61.31	61.31	60.98	158.94
1.00	38.69	96.73	99.84	99.84	99.84	98.78	-3.10	0.00	0.00	-1.06	-1.06	58.04	61.14	158.02	61.14	61.14	60.08	155.27
2.00	39.51	78.37	99.02	99.02	99.02	98.12	-20.65	0.00	0.00	-0.90	-0.91	38.86	59.51	150.62	59.51	59.51	58.61	148.35
3.00	39.67	51.92	95.43	95.43	95.43	95.67	-43.51	0.00	0.00	0.24	0.26	12.24	55.76	140.53	55.76	55.76	56.00	141.15
4.00	40.16	47.51	76.24	76.24	76.24	83.84	-28.73	0.00	0.00	7.59	9.96	7.35	36.08	89.84	36.08	36.08	43.67	108.74
5.00	40.90	35.67	58.04	58.04	58.04	65.71	-22.37	0.00	0.00	7.67	13.22	-5.22	17.14	41.92	17.14	17.14	24.82	60.68
6.00	41.22	32.57	50.04	50.04	50.04	55.67	-17.47	0.00	0.00	5.63	11.26	-8.65	8.82	21.39	8.82	8.82	14.45	35.05
7.00	41.22	26.94	45.47	45.47	45.47	59.76	-18.53	0.00	0.00	14.29	31.42	-14.29	4.24	10.30	4.24	4.24	18.53	44.95
8.00	41.96	26.20	38.12	38.12	38.12	48.24	-11.92	0.00	0.00	10.12	26.55	-15.76	-3.84	-9.14	-3.84	-3.84	6.29	14.98
9.00	42.04	24.98	40.82	40.82	40.82	51.67	-15.84	0.00	0.00	10.86	26.60	-17.06	-1.22	-2.91	-1.22	-1.22	9.63	22.91
10.00	41.80	23.51	36.24	36.24	47.43	44.90	-12.73	0.00	11.18	8.65	23.87	-18.29	-5.55	-13.28	-5.55	5.63	3.10	7.42
slope	**0.38**	**-8.09**	**-7.84**	**-7.84**	**-7.33**	**-6.46**			**average**		***20.41***			***67.91***				***81.68***

Beta=2.3

Std Dev	NO LLD	3.00	7.00	8.00	11.00	15.00	3.00	8.00	11.00	15.00	%improvement	3.00	7.00	7 %improvement	8.00	11.00	15.00	15 %improvement
0.00	28.65	99.67	99.76	99.76	99.76	99.10	-0.08	0.00	0.00	-0.65	-0.65	71.02	71.10	248.15	71.10	71.10	70.45	245.87
1.00	28.73	98.69	97.39	97.39	97.39	95.84	1.31	0.00	0.00	-1.55	-1.59	69.96	68.65	238.92	68.65	68.65	67.10	233.52
2.00	29.31	77.63	96.73	96.73	96.73	96.65	-19.10	0.00	0.00	-0.08	-0.08	48.33	67.43	230.08	67.43	67.43	67.35	229.81
3.00	30.20	47.76	88.98	88.98	88.98	83.76	-41.22	0.00	0.00	-5.22	-5.87	17.55	58.78	194.59	58.78	58.78	53.55	177.30
4.00	30.29	36.08	65.47	65.47	65.47	80.90	-29.39	0.00	0.00	15.43	23.57	5.80	35.18	116.17	35.18	35.18	50.61	167.12
5.00	30.69	28.41	49.71	49.71	49.71	63.35	-21.31	0.00	0.00	13.63	27.42	-2.29	19.02	61.97	19.02	19.02	32.65	106.38
6.00	30.94	21.71	31.35	31.35	31.35	40.16	-9.63	0.00	0.00	8.82	28.13	-9.22	0.41	1.32	0.41	0.41	9.22	29.82
7.00	32.41	23.18	43.02	35.49	35.49	40.65	-19.84	-7.53	-7.53	-2.37	-5.50	-9.22	10.61	32.75	3.08	3.08	8.24	25.44
8.00	33.06	17.88	29.96	29.96	29.96	44.08	-12.08	0.00	0.00	14.12	47.14	-15.18	-3.10	-9.38	-3.10	-3.10	11.02	33.33
9.00	32.90	16.49	31.59	31.59	31.59	42.69	-15.10	0.00	0.00	11.10	35.14	-16.41	-1.31	-3.97	-1.31	-1.31	9.80	29.78
10.00	32.82	15.18	28.16	28.16	33.71	31.84	-12.98	0.00	5.55	3.67	13.04	-17.63	-4.65	-14.18	-4.65	0.90	-0.98	-2.99
slope	**0.49**	**-9.04**	**-8.61**	**-8.75**	**-8.50**	**-7.58**			**average**		***24.13***			***99.67***				***115.94***

Beta=2.4

Std Dev	NO LLD	3.00	7.00	8.00	11.00	15.00	3.00	8.00	11.00	15.00	%improvement	3.00	7.00	7 %improvement	8.00	11.00	15.00	15 %improvement
0.00	21.55	99.84	99.67	99.67	99.67	99.43	0.16	0.00	0.00	-0.24	-0.25	78.29	78.12	362.50	78.12	78.12	77.88	361.36
1.00	22.04	93.96	84.16	96.56	98.21	96.73	9.80	12.40	14.05	12.57	14.94	71.92	62.12	281.85	74.52	76.17	74.69	338.89
2.00	22.69	67.59	89.63	89.63	89.63	93.71	-22.04	0.00	0.00	4.08	4.55	44.90	66.94	294.96	66.94	66.94	71.02	312.95
3.00	22.61	36.98	68.33	68.33	68.33	73.55	-31.35	0.00	0.00	5.22	7.65	14.37	45.71	202.17	45.71	45.71	50.94	225.27
4.00	23.02	29.39	41.47	41.47	41.47	42.69	-12.08	0.00	0.00	1.22	2.95	6.37	18.45	80.14	18.45	18.45	19.67	85.46
5.00	24.24	20.73	42.78	42.78	42.78	42.12	-22.04	0.00	0.00	-0.65	-1.53	-3.51	18.53	76.43	18.53	18.53	17.88	73.74
6.00	24.24	14.29	25.06	25.06	25.06	42.29	-10.78	0.00	0.00	17.22	68.73	-9.96	0.82	3.37	0.82	0.82	18.04	74.41
7.00	23.84	14.20	27.43	27.43	27.43	35.59	-13.22	0.00	0.00	8.16	29.76	-9.63	3.59	15.07	3.59	3.59	11.76	49.32
8.00	37.80	14.12	25.71	25.71	25.71	38.29	-11.59	0.00	0.00	12.57	48.89	-23.67	-12.08	-31.97	-12.08	-12.08	0.49	1.30
9.00	37.47	11.67	22.04	22.04	22.04	25.63	-10.37	0.00	0.00	3.59	16.30	-25.80	-15.43	-41.18	-15.43	-15.43	-11.84	-31.59
10.00	38.04	11.92	21.14	21.14	23.10	26.53	-9.22	0.00	1.96	5.39	25.48	-26.12	-16.90	-44.42	-16.90	-14.94	-11.51	-30.26
slope	**1.76**	**-9.00**	**-8.46**	**-8.92**	**-8.89**	**-8.10**			**average**		***21.77***			***108.99***				***132.80***

Beta=2.5

Std Dev	NO LLD	3.00	7.00	8.00	11.00	15.00	3.00	8.00	11.00	15.00	%improvement	3.00	7.00	7 %improvement	8.00	11.00	15.00	15 %improvement
0.00	16.00	91.59	98.12	98.12	98.12	97.88	-6.53	0.00	0.00	-0.24	-0.25	75.59	82.12	513.27	82.12	82.12	81.88	511.73
1.00	16.73	65.96	75.33	75.33	82.38	86.03	-9.37	0.00	7.05	10.70	14.21	49.22	58.60	350.14	58.60	65.64	69.30	414.09
2.00	16.41	43.43	71.18	76.04	81.48	70.56	-27.76	4.86	10.29	-0.62	-0.87	27.02	54.78	333.83	59.64	65.07	54.16	330.05
3.00	16.98	36.41	61.55	61.55	61.55	64.33	-25.14	0.00	0.00	2.78	4.51	19.43	44.57	262.50	44.57	44.57	47.35	278.85
4.00	16.57	27.10	42.45	42.45	42.45	53.39	-15.35	0.00	0.00	10.94	25.77	10.53	25.88	156.16	25.88	25.88	36.82	222.17
5.00	17.06	14.53	36.82	36.82	36.82	50.78	-22.29	0.00	0.00	13.96	37.92	-2.53	19.76	115.79	19.76	19.76	33.71	197.61
6.00	28.98	11.18	28.90	28.90	28.90	39.59	-17.71	0.00	0.00	10.69	37.01	-17.80	-0.08	-0.28	-0.08	-0.08	10.61	36.62
7.00	17.06	11.02	16.65	16.65	16.65	28.73	-5.63	0.00	0.00	12.08	72.55	-6.04	-0.41	-2.39	-0.41	-0.41	11.67	68.42
8.00	17.55	10.20	20.08	20.08	20.08	20.57	-9.88	0.00	0.00	0.49	2.44	-7.35	2.53	14.42	2.53	2.53	3.02	17.21
9.00	17.80	8.65	15.18	15.18	15.18	19.51	-6.53	0.00	0.00	4.33	28.49	-9.14	-2.61	-14.68	-2.61	-2.61	1.71	9.63
10.00	19.27	8.00	11.92	11.92	25.39	19.43	-3.92	0.00	13.47	7.51	63.01	-11.27	-7.35	-38.14	-7.35	6.12	0.16	0.85
slope	**0.33**	**-7.40**	**-8.44**	**-8.57**	**-8.36**	**-8.12**			**average**		***33.96***			***153.69***				***189.75***

Appendix G: Glossary of Terms

β	Path loss exponent (in the Log Normal Shadowing model of radio propagation)
σ	Standard Deviation of the normally distributed random path loss factor(in the Log Normal Shadowing model of radio propagation)
AAV	Amphibious Assault Vehicle
ACK	Acknowledgment (packet)
ARG	Amphibious Ready Group
AODV	Ad-hoc On-demand Distance Vector (routing protocol)
CBR	Constant Bit Rate
CTS	Clear to Send
DSDV	Destination Sequence Distance Vectoring
DSR	Dynamic Source Routing
FTP	File Transfer Protocol
IETF	Internet Engineering Task Force
kbps	Kilobits per second
LCAC	Landing Craft Air Cushion
LL	Link Layer
LLD	Link Layer Detection
MAC	Media Access Control
MANET	Mobile Ad-Hoc networking
MEU	Marine Expeditionary Unit
NIC	Network Interface Card
NS2	Network Simulator (version 2.1b9)
OSI	Open Systems Interconnection
RTS	Request to Send
STOM	Ship to Objective Maneuvers
TCP	Transmission Control Protocol
TCL	Tool Command Language
UDP	User Datagram Protocol

References

[1] Frodigh, M. et al., "Wireless Ad Hoc Networking: The Art of Networking Without a Network," Ericsson Rev., no. 4, 2000, http://www.ericsson.com/review

[2] Frodigh, M. et al., "Future Generation Wireless Networks," IEEE Personal Communications., Vol 8, Issue 5 (October 2001): 10-17.

[3] Gehrmann, Christian. "Bluetooth Security White Paper" (April 2002). Available at http://www.bluetooth.com/upload/24Security_Paper.PDF

[4] *Ship to Objective Maneuver*. Marine Corps Combat Development Command: Quantico, Virginia., 1997. Available at: http://www.concepts.quantico.usmc.mil/stom.htm

[5] *Operational Maneuver from the Sea*. Headquarters, US Marine Corps: Washington DC, 1996. Available at: http://www.concepts.quantico.usmc.mil/omfts/docs/omftsfinal.pdf

[6] Leon-Garcia, Alberto and Widjaja, Indra. *Communication Networks: Fundamental Concepts and Key Architectures*. McGraw Hill: Boston, MA. 2000.

[7] INSIGNIA Project of Columbia University., http://comet.columbia.eu/insignia.

[8] IETF MANET Working Group. http://www.ietf.org/html.charters/manet-charter.html

[9] Perkins, C. E., Royer, E. M., and Das, S. R., "Performance Comparison of Two On- Demand Routing Protocols for Ad Hoc Networks," *IEEE Personal Communications*, (February 2001), 16-28.

[10] Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall 1996.

[11] Network Simulator Notes and Documentation, UCB/LBNL., http://www.isi.edu/nsnam/ns/

[12] Johnson, David, Malz, D., et al. "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)" http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-07.txt, IETF Internet Draft, February 2002, work in progress

[13] Perkins, C. E., Royer, E. M., and Das, S. R., "Ad Hoc on Demand Distance Vector (AODV) Routing," http://www.ietf.org/internet-drafts/draft-ietfmanet-aodv-11.txt, IETF Internet Draft, June 2002, work in progress.

[14] Johnson, D.B., Maltz, D.A., and Hu, Yih-chun. "Dynamic Soure Routing (DSR)," http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-08.txt, IETF Internet Draft, February 2003, work in progress.

[15] Gunes, M.; Vlahovic, D. "The performance of the TCP/RCWE enhancement for ad-hoc networks" *Computers and Communications, 2002. Proceedings. ISCC 2002*. (Seventh International Symposium on, Vol., Iss., 2002) Pages: 43- 48

[16] *TRPR User's guide*. Available at: http://proteantools.pf.itd.nrl.navy.mil/trpr.html

[17] *The AWK Manual*. Available at http://www.cs.uu.nl/docs/vakken/st/nawk/nawk_toc.html