

University of Windsor Scholarship at UWindsor

Electronic Theses and Dissertations

2012

Improvements in AODV towards Smart Grid Routing - Test Bed Development & Prioritized Routing for Emergency Scenarios

Shawn Andrew Ruppert

Follow this and additional works at: <http://scholar.uwindsor.ca/etd>

Recommended Citation

Ruppert, Shawn Andrew, "Improvements in AODV towards Smart Grid Routing - Test Bed Development & Prioritized Routing for Emergency Scenarios" (2012). *Electronic Theses and Dissertations*. Paper 5371.

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

Improvements in AODV towards Smart Grid Routing - Test Bed Development &
Prioritized Routing for Emergency Scenarios

by

Shawn A. Ruppert

A Thesis
Submitted to the Faculty of Graduate Studies
through Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada

2012

© 2012 Shawn A. Ruppert

Improvements in AODV towards Smart Grid Routing - Test Bed Development &
Prioritized Routing for Emergency Scenarios

by

Shawn A. Ruppert

APPROVED BY:

Dr. Subir Bandyopadhyay
School of Computer Science

Dr. Esam Abdel-Raheem
Department of Electrical and Computer Engineering

Dr. Kemal Tepe, Advisor
Department of Electrical and Computer Engineering

Dr. Sid-Ahmed, Chair of Defense
Department of Electrical and Computer Engineering

September 18, 2012

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

The Smart Grid project is a global effort towards a reliable and robust power grid. With estimated funding in the billions in just the next few years, this project will be an on-going multi-disciplinary feature of engineering. Designated as one of the subsections of the Smart Grid effort is power line monitoring or transmission line monitoring. For a reliable power grid power lines, transformers, and other field equipment must be monitored. Wireless mesh networks have already begun to deploy in different sectors and is a viable strategy for power line monitoring. They can be easily deployed with sensors and can create data paths to monitoring stations on their own. Ad-Hoc On-Demand Distance Vector Routing (AODV) is a routing protocol that has received much attention over the years and has been chosen as a basis for alteration for moving towards a more intelligent smart grid routing protocol. Wireless mesh networking protocols have generally been simulated in nearly all the research, live testing is tedious and requires hardware and test schedules. These problems are solved with the developed Java Ad-Hoc Network Test Bed which will allow even novice users to complete rudimentary network tests on live systems.

Emergencies in the field such as transmission line faults and/or transformers misbehaving need to get information to hydro companies as quick as possible. A quality of Service prioritized emergency route solves the problems of network congestion during an emergency. When a sensor detects that a power line has gone down, it will request a specialized higher speed route to the control end, therefore resulting in a lower latency and assurance of quick emergency data delivery.

DEDICATION

I would like to dedicate my Master's Thesis to a few select people:

To my Mother who supported me the entire way through and would wake up early every morning to make sure I had a good lunch prepared for school. She gave all the unconditional love a person could give and I am thankful for that.

To my Father and Stepmother who were always there with the right answer for every question and/or a helping hand with anything I have experienced along the way. My father has taught me so many things; from getting around in this world to standing up for myself.

To my Brother for being a great friend who has always had the time to sit and talk over a cold beer about all the difficulties we've faced, the music we love, and the things in life that matter.

To my great friend Mary who has listened to constant complaining from me about issues I've had throughout my graduate studies and who has opened up her heart and her home to me through the entire duration of my Master's research. I have learned many things from her; from her huge heart and shining personality.

ACKNOWLEDGEMENTS

I would first and foremost like to acknowledge Dr. Kemel Tepe, my advisor and mentor. His attitude towards academia and research has allowed myself and fellow lab members to expand our knowledge to subjects not only focused on our research areas but other new and interesting topics in the field. I would not have pursued graduate studies if I had not met Dr. Tepe and felt so welcome and at home working as part of his team. The mutual respect between himself and his students has led me to strive for quality work. Not only has he been there for guidance throughout my graduate studies but also as an understanding friend who treats his students much like a family rather than a worker that pursues his interests.

I would like to acknowledge Dr. Nabih Jaber, a man with more philosophical ideas and intelligence than I could ever hope to acquire, for his guidance and friendship. He has helped me along the way to publish my research and create quality work for Dr. Tepe. Thank you for the many long discussions and interesting ideas you shared throughout the day to break up the busy schedule.

I would also like to thank the members of the WICIP team that I had the pleasure of working with. I have received help from nearly everyone on subjects pertaining to programming, Linux kernels, wireless concepts, etc. Of all my peers, most of all I'd like to acknowledge Mr. Ahmed El Baba for his friendship and partnership. We've spent many long nights working on projects and sharing ideas together and sometimes I wonder if I'd have been able to complete the long strenuous graduate school process without him.

Thank you all.

DECLARATION OF ORIGINALITY	iii
ABSTRACT.....	iv
DEDICATION	v
ACKNOWLEDGEMENTS	vi
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF ABBREVIATIONS.....	xii
CHAPTER 1: Introduction and Background Information	1
1.1 Introduction.....	1
1.2 Smart Grid.....	3
1.3 Wireless Ad-hoc Network Routing Protocols	4
1.4 Thesis Objectives	5
1.4.1 Ad-hoc Network Test Bed.....	5
1.4.2 QoS Emergency Priority Route.....	6
1.5 Thesis Structure	6
1.6 Chapter Summery.....	6
CHAPTER 2: Literature Review	7
2.1 Introduction to AODV	7
2.1.1 AODV Functionality.....	7
2.2 Quality of Service in Ad-Hoc Networks	10
2.2.1 QoS Metrics	11
2.2.2 Non-Real-time and Real-Time QoS Requirements	11
2.3 Ad-hoc Network Performance Testing.....	12
2.4 External Testing Utilities	12
2.4.1 Iperf 2.0.4 Network Testing Tool	13
2.4.2 Script module.....	14
CHAPTER 3: Ad-Hoc Wireless Network Java Test Bed	15
3.1 Introduction	15

3.2 Ad-Hoc Network Test Bed Development.....	15
3.2.1 Test Node Preparation	18
3.2.1.1 AODV-UU install.....	19
3.2.2 Discovery of Test Nodes	19
3.2.3 Topology Setup.....	22
3.2.3.1 Topology Table.....	22
3.2.3.2 Topology Decoder	23
3.2.4 Run Iperf Test.....	26
3.2.5 Node/Listener Software	29
3.3 Chapter Summary.....	34
CHAPTER 4: AODV QoS Design and Methodology	34
4.1 Introduction	34
4.2 Traffic Reduction Methodology.....	34
4.3 Quality of Service Design	35
4.3.1 Emergency Instantiation	36
4.3.2 Intermediate Node Emergency Processing	36
4.4 Chapter Summary.....	38
CHAPTER 5: Results and Analysis	39
5.1 Introduction	39
5.2 Test Scenario.....	39
5.2.1 Simulated Emergency	39
5.2.2 Test schedule & Results.....	39
5.3 Chapter Summary.....	44
CHAPTER 6: Conclusions and Future Works	44
6.1 Introduction	44
6.2 Concluding Remarks	44
6.2.1 Java Ad-Hoc Network Test Bed.....	44
6.2.2 QoS Prioritized Emergency Route for AODV-UU.....	45
6.3 Future Works.....	45
6.3.1 Java Ad-Hoc Network Test Bed.....	45
6.3.2 QoS Prioritized Emergency Route for AODV-UU.....	46

6.4 Chapter Summary.....	47
APPENDICES	48
<i>See attached supplemental files.</i>	48
• <i>Java Wireless Ad hoc Network Testbed</i>	48
• <i>AODV-UU Source files with QoS Amendments.....</i>	48
REFERENCES	49
VITA AUCTORIS.....	51

LIST OF TABLES

Table 1 - RREQ Structure	9
Table 2 - RREP Structure.....	10
Table 3 - Test Node Preparation Commands.....	18
Table 4 - Test Node IP addressing.....	18
Table 5 - AODV-UU Install Procedure	19

LIST OF FIGURES

Figure 1 - Mesh Networking Protocols.....	5
Figure 2 - AODV Topology Example	7
Figure 3 - Neighbor RREQ.....	8
Figure 4 - RREQ Propagation	8
Figure 5 - RREP to Source	9
Figure 6- Iperf Command Example	13
Figure 7 - Iperf data sample	14
Figure 9- Test Bed System Flowchart.....	16
Figure 9 - Test Bed System Flowchart.....	16
Figure 10 - Full Test Bed screenshot	17
Figure 11 - Node Discovery Screenshot	19
Figure 12 - Node Discovery Exchange	20
Figure 13 - Node Discovery Flowchart	21
Figure 14 - Topology Setup.....	22
Figure 15 - Topology table screenshot.....	23
Figure 16 - Topology Table flowchart pt1	24
Figure 17 - Topology Table flowchart pt2	25
Figure 18 - Iperf test screenshot.....	26
Figure 19- IPERF process flowchart pt1	28
Figure 20 - IPERF Process flowchart pt2.....	29
Figure 21 - Basic Layout of Listener Software	30
Figure 22 - Listener Software main flowchart	31
Figure 23 - Listener Software flowchart pt2.....	32
Figure 24 - Listener Software flowchart pt3.....	33
Figure 25 - QoS C files	35
Figure 26 - Code Excerpt send_rreq.....	36
Figure 27 - Code Excerpt rreq_process	37
Figure 28 - Code Excerpt manage_ips.....	37
Figure 29 - Test Topology	40
Figure 30 - Emergency Route triggered in A.....	40
Figure 31 - Throughput comparison	41
Figure 32 - Latency of node A.....	42
Figure 33 - Latency drop during emergency.....	43

LIST OF ABBREVIATIONS

AODV – Ad hoc On-demand Distance Vector
AODV-UU – Ad hoc On-demand Distance Vector-Uppsala University
BATMAN – Better Approach To Mobile Ad hoc Networking
DAST – Distributed Applications Support Team
DSDV – Destination Sequenced Distance Vector
DSL – Digital Subscriber Line
DSLAM – Digital Subscriber Line Access Multiplexer
DSR – Dynamic Source Routing
IP – Internet Protocol
IPTV – Internet Protocol Television
MAC – Media Access Control
MANET – Mobile Ad hoc Network
NS-2 – Network Simulator 2
NLANR – National Laboratory for Applied Network Research
OLSR – Optimized Link State Routing
QoS – Quality of Service
RREP – Route Reply
RREQ – Route Request
SG – Smart Grid
TCP – Transmission Control Protocol
UDP – User Datagram Protocol
VOIP – Voice over Internet Protocol
WLAN – Wireless Local Area Network
ZRP – Zone Routing Protocol

CHAPTER 1: Introduction and Background Information

1.1 Introduction

The Smart Grid (SG) movement is sweeping the globe with full force as industries from appliance manufactures to power generation companies begin to define, design, and develop standards, networks, and network enabled products to fuel the Smart Grid project. IDC Energy Insights identified an average of 17% increase in spending between 2010 and 2015 for the various SG projects with an overall spending of circa 46 billion U.S. dollars, with some countries such as China viewing an increase in spending of up to 33% [1]. Nearly all subsectors of SG applications involve connected devices and systems. As these subsectors develop there is a finer definition of what is needed to accommodate the new idea's that will pave the way to the future in power grid reliability and efficiency.

There is a magnitude of research and design efforts in the area of power line monitoring for SG[2]. From hundreds of start-ups joining the market and big contenders such as General Electric and Siemens joining, there are many different definitions being derived as to how the world's power grids should be monitored in the SG movement. The ability to have two-way communication with sensors, breakers, transformers, and generation systems will make the system robust and efficient. One very broad ranged idea being investigated is the use of mesh networks to support sensor networks for monitoring purposes[3]. Already companies are utilizing mesh networking protocols for data collection such is the case with *Elster* and its smart meter that calculates usage throughout the day and transmits hydro usage to companies directly from the smart meter attached to the customer's homes[4]. Mesh networks create

redundancy in the system by adding multiple paths to destinations. Protocols that fall under the mesh network category are generally self-configuring and self-healing. This is an attractive feature for SG applications because of the sheer number of devices that will be added into each of the network systems. One particular mesh networking routing protocol has found popularity amongst researcher and it is known as Ad-Hoc On-Demand Distance Vector Routing Protocol (AODV)[5,6]. This protocol has been the focus of countless simulations and modifications for use in networks requiring mesh network-like qualities.

Testing of ad-hoc networks is a tedious time consuming activity. Networks have a few main metrics that benchmark performance such as latency, throughput, and jitter among many other things. Simulation tools such as Network Simulator 2 and OMNET++ are common tools used to calculate these metrics for given system setups with countless variables ranging from channel-type to congestion and density[7,8]. Although simulations carry much weight within the scientific research community, real-time/live testing brings to light issues that were previously over-looked in the simulated environment.

This thesis intends on making two steps forward in the SG research area. As development toward power line monitoring takes places with mesh networked systems the need for a quality of service (QoS) based prioritized route to the hydro company has been established. Emergency data such as over-current, electrical faults, or pertinent transformer sensor readings need to arrive to the control center in a nearly real-time fashion. Periodic or continuous non-real-time data congests networks and therefore hinders the decreased latency of real-time data. The proposed prioritized route solves this problem by giving notice to the network infrastructure that high priority information is being transmitted and requires more network resources than

other traversing data. The amendments to AODV that have been implemented take the information being traversed and read the priority level. Based on the level of severity of the packet being sent the revised protocol will take the appropriate steps to propagate the packet along the desired path while reducing the bandwidth of other network nodes to allow for minimal latency of the emergency packets.

With relation to the testing of ad-hoc network this thesis intends to take the tedious task of ad-hoc network testing and simplify it by introducing a newly developed java-based ad-hoc network test bed. Such features as node discovery, network topology setup, and testing between various network nodes from a centralized test center will allow the tester to obtain certain network characteristics and performance metrics that would normally facilitate steep learning curves and independent setup programming of each test node. Java has been chosen due to the swing components. A graphical user interface was desired and due to the network libraries and interface capabilities java was a very desirable candidate for the test bed development.

1.2 Smart Grid

The Smart Grid is an evolving vision of an interconnected power grid system. Green energy is leading to many localized generation centers. Wind power and solar power farms are functioning without connection to the entire power grid because society is lacking the infrastructure to tie them in reliably and efficiently. Distributed Load Generation has become a topic of interest due to added reliability and sustainability being vital qualities that we need as a part of the updated power grid. Subsectors include power line monitoring, smart meter services, network enables home appliances, distributed load generation, sensor networks, and many more[9].

1.3 Wireless Ad-hoc Network Routing Protocols

In ad-hoc networking there is no “main” protocol that is used. There is a variety of protocols that can be chosen to match the functionality of the system. Under discussion will be Mobile Ad-hoc Network (MANET) routing protocols. These protocols have been designed for nodes with mobility, thus quickly changing topologies[11]. MANET protocols have benefits on static network topologies due to the self-configurability of the network due to the design constraints put upon them with mobility as a factor. The three main protocol categories for ad-hoc networks would be Reactive, Proactive, and Hybrid. A reactive protocol reacts to routing needs of the nodes/computers in the network. Routes are established based on the need to send data to another network node and therefore less space is used storing routes to every other node in the network. A proactive protocol utilizes a full routing table on every node. This assures that every node within the network has a known path to the desired destination. This routing table can be very large and network scalability can be costly due to routing table updates during network expansion[20]. A hybrid is basically a combination of both as designed with desired functionality in mind.

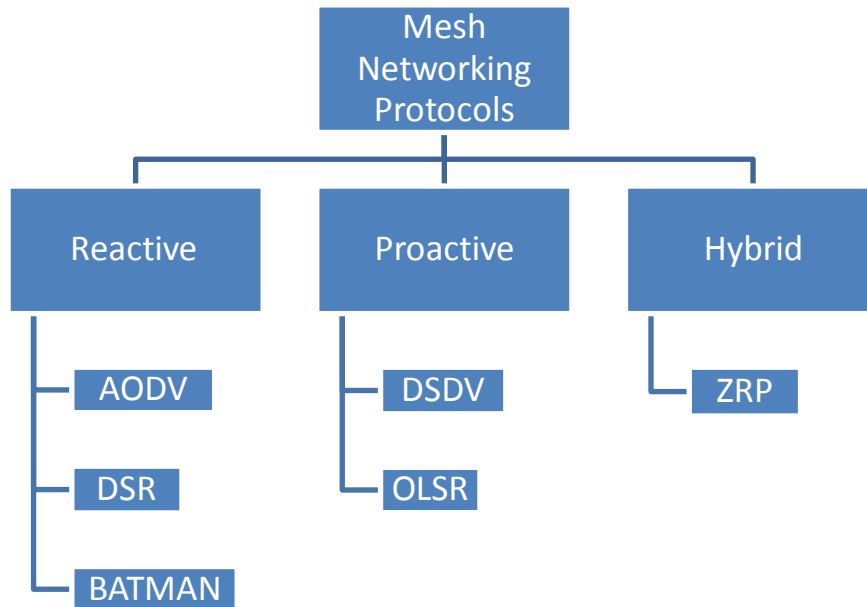


Figure 1 - Mesh Networking Protocols

1.4 Thesis Objectives

This thesis details the development of a static ad-hoc network test bed along with development and testing of a minor extension of AODV-UU to include a best possible QoS level for emergency packets.

1.4.1 Ad-hoc Network Test Bed

The java-based network testing software will allow a user to connect all nodes that need to be tested to the test center. The user can choose the network topology via an intuitive yet simple table based topology tool. The test center will allow new scripts to be written in different languages and be used for node testing. A built in throughput testing method is given as an option for users with intelligent logging and retrieval of test data from the nodes involved in the test.

1.4.2 QoS Emergency Priority Route

Through alterations and added functions, the original AODV-UU 0.9.6 was modified to create the optional ability to request a route with a higher priority. For testing purposes a simulated emergency has been programmed into the C code to mimic a trigger sent to the network layer by a sensor or device that requires an emergency best-offered route to a notifying authority such as a hydro company control center.

1.5 Thesis Structure

This thesis is organized as follows: Chapter 1 introduces the necessary topics to aid in the understanding of the motives and the reasoning for the thesis topic. Chapter 2 begins the in-depth design explanation and system functionality of the newly developed test bed. Chapter 3 explains the flow of the code and reasoning for implementing the QoS prioritized emergency route that could aid smart grid applications. Chapter 4 yields all test scenarios and methods used for testing while also displaying the findings from testing the designed protocol. Chapter 5 concludes this thesis with remarks on the improvements towards reducing latency and increasing throughput in emergency situations and also focuses on future works and ideas pertaining to both the test bed design and the QoS implementation.

1.6 Chapter Summery

This chapter began by providing an introduction to what this thesis covers. Introductions to many of the underlying principles to the motives of this research have been discussed which show need for this research and reason for development. Basic explanations have been given as to the functionality of AODV-UU which is a necessity to understanding the work completed on this thesis and the relevance of the QoS alterations to the existing code.

CHAPTER 2: Literature Review

2.1 Introduction to AODV

2.1.1 AODV Functionality

The main point of the network routing protocol is to forward and route packets to the correct destination. Due to the fact that a protocol was needed for testing and design, Uppsala University's version was used AODV-UU[6,12]. Discussion will continue to explain this functioning open source protocol. The protocol uses three main packet types: Route Request, Route Reply,

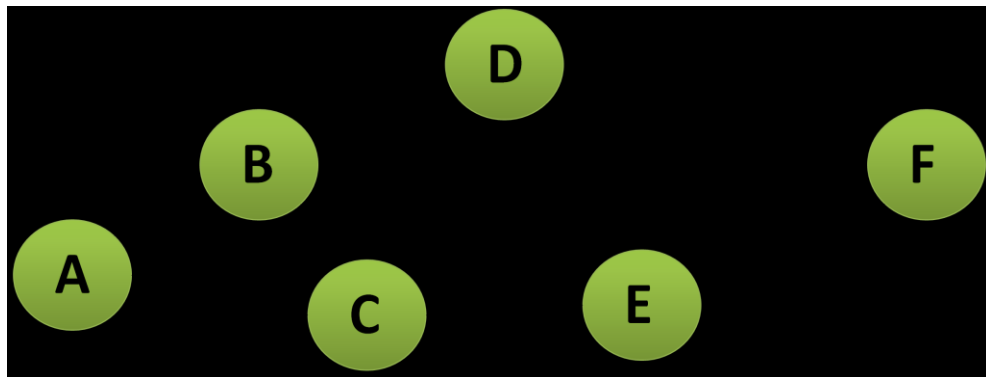


Figure 2 - AODV Topology Example

and Route Error. The initial stages of network setup occur with a broadcast "hello" message. This is used to find immediate neighbors. The system keeps these neighbors updated by periodic "hello" messages defined by a "hello interval" in the code. When data needs to be sent from a given node a Route Request (RREQ) is assembled and sent out to each of his neighbors. Assume the following topology where "A" is looking to forward data to "F" with the arrows representing node connections.

When “A” would like to communicate to “F”, it generates an RREQ and sends it out to his

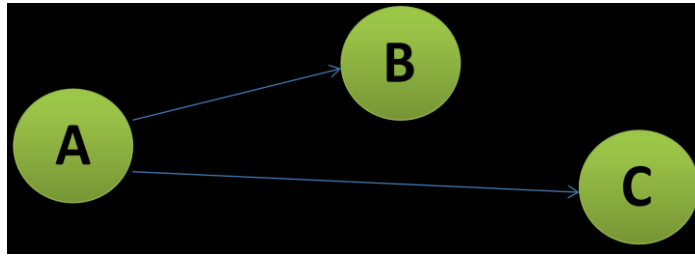


Figure 3 - Neighbor RREQ

neighbors.

As nodes “B” and “C” receive the RREQ they check to see if they have an immediate neighbor “F”, when they realize they do not they forward the RREQ to their neighbors.

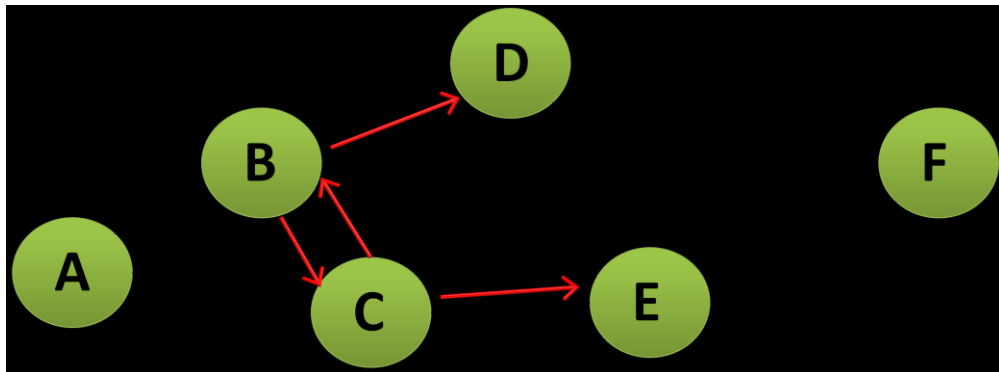


Figure 4 - RREQ Propagation

Mechanisms are built into AODV-UU that allow it to recognize already-processed RREQs. Sequence numbers matched with ID numbers and source addresses give AODV-UU a robust functionality in terms of efficiency.

As the RREQ reaches node “E” and it checks whether it has found the destination node, it realized that it has “F” as a neighbor. A Route Reply (RREP) is generated and reverse route

entries are used to send the route information back to the source node. "A" now has the route to "F". Each node only knows the next hop toward the destination and the amount of hops onward until the destination. The information carried in the packets are as follows:

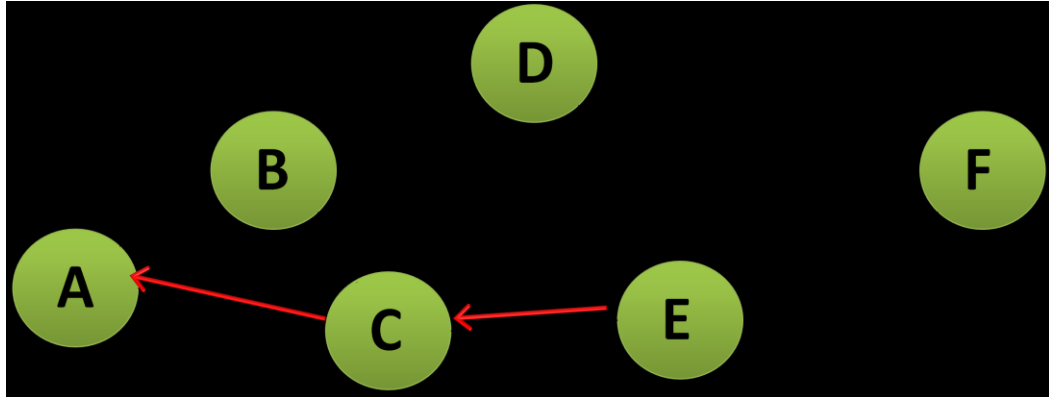


Figure 5 - RREP to Source

Table 1 - RREQ Structure

RREQ Structure			
Type	J R G D U	Reserved	Hop Count
Broadcast ID			
Destination IP Address			
Destination Sequence Number			
Source IP Address			
Source Sequence Number			

Table 2 - RREP Structure

RREP Structure				
Type	R A	Reserved	Prefix Size	Hop Count
Destination IP Address				
Destination Sequence Number				
Source IP Address				
Lifetime (TTL)				

There are more functional aspects of AODV such as a Route Error (RERR) packet that propagates the network when a link has been lost to a neighbor but for the purpose of introducing the methodology of AODV to a point where the intent of this thesis can be realized, in depth discussion will not take place for these points.

2.2 Quality of Service in Ad-Hoc Networks

There are different reasons to implement a Quality of Service (QoS) scheme. As defined, Totality of characteristics of a telecommunications service that bear on its ability to satisfy stated and implied needs of the user of the service [13]. Quality of service implementations set certain thresholds based on network metrics to assure a desired functionality or outcome of a task. The task and the level of network performance will determine the use of a QoS metric.

2.2.1 QoS Metrics

A QoS metric can be based on a number of different objectives. Prioritization values are a method of assuring that certain levels of QoS are met for a given application. An excellent example of prioritized QoS implementation would be a customer's Digital Subscriber Line (DSL) service where different packets are given different levels of service based on their priority level[21]. This example generally deals with metrics like throughput, or latency. In remote sensor nodes or battery operated networked devices QoS may define routes based on minimum transmit energy[14,15]. Other applications may rely on certain security services along a network route to define a possible route to the destination. Systems implementing a QoS such as this may or may not send data if network conditions are not met.

2.2.2 Non-Real-time and Real-Time QoS Requirements

Data is categorized on its need to be transmitted real-time or as best effort. Surfing the internet is a prime example of best effort data transfer. As a webpage is viewed the data is loaded as fast as possible though at times this is not as fast as preferred. Touching back on the example of QoS in DSL systems, network solution companies use an Access Multiplexer (DSLAM) which implements three services geared toward quality of service: queuing, shaping, and policing[16]. Companies such as Alcatel-Lucent offer services to customers where a single DSL line will provide internet browsing, Voice Over IP (VOIP), and television broadcast over the DSL (IPTV). Prioritized values are given to each type of packet to assure a certain level of QoS. When a packet carrying video (IPTV) information is delayed, the viewing for the customer suffers far more than if a millisecond of a phone conversation (VOIP) is delayed. In the same respect, a millisecond of a phone conversation being delayed is worse than a webpage taking an extra

second to load. Therefore the services implemented in the DSLAM aim to give precedence to packets marked with higher priority levels to assure a level of QoS for the provided customer product.

2.3 Ad-hoc Network Performance Testing

The performance of a network is an important metric to be understood when designing a network for a specific purpose. Performance defines service quality. Networks are hard to test due to testing infrastructure and resources. Simulation has taken a large role in assessing network characteristics and performance values provided by the system. The problem with simulation is that the mathematics and scientific principals governing the test scenarios and results are all theoretical, though accurate, they sometimes do not depict an accurate rendition of real world functionality. This is common with ad-hoc network testing. Test's including hundreds of nodes with velocity vectors and random channel interference can be utilized to try and provide network performance indices though when compared with real world results they can vary due to a number of causes. Two of the most used network simulation tools are Network Simulator 2 (NS-2) and OMNET++.

2.4 External Testing Utilities

The java interface provided circa 50% of the system functionality and functioned as a basis to tie multiple utilities together. Java development encompassed the node discovery portion as well as the Topology Table Decoder for building the desired commands and administering them to the appropriate nodes for topology setup. For the purpose of testing the nodes for network performance metrics, a program accepted amongst the network testing community known as iperf 2.0.4 was integrated with the java test bed. The java test bed incorporated a scheme to

accept modules, one being scripts written in bash or C to be loaded and test nodes executing them.

2.4.1 Iperf 2.0.4 Network Testing Tool

Iperf was developed by the Distributed Applications Support Team (DAST) at the National Laboratory for Applied Network Research (NLANR)[17]. Iperf claims a cross-platform tool that will output standardized performance measurements that can be used on any network. Iperf is set up on a node as an accepting server program. On another node iperf is set up with test parameters as a client node and contacts the server based node. The test then commences with the desired test type and duration. The tests can depict UDP or TCP as a transport layer protocol and allows for uni-direction or bi-direction testing. The commands to use iperf can be shown in the following example:

```
Iperf -c 192.168.3.2 -i 0.5 -t 60
```

Figure 6- Iperf Command Example

The “-c” sets the iperf up as a client to the following IP “192.168.3.2”. The interval of reporting is given by the “-i” flag with the argument a value in seconds. The total time of the throughput test is t=60 seconds. There are many more commands and options available and can be found via the “iperf -h” command. Iperf sends out its data in a well organized fashion for the user to read. A parsing program has been developed in Visual Basic for extracting the data fields out of the outputs of the iperf data file.

```

-----
Client connecting to 10.42.43.5, TCP port 5001
TCP window size: 16.0 KByte (default)
-----
[ 3] local 10.42.43.11 port 57689 connected with 10.42.43.5 port
5001
[ID] Interval  Transfer  Bandwidth
[ 3] 0.0- 0.5 sec  304 KBytes 4.98 Mbits/sec
[ 3] 0.5- 1.0 sec  312 KBytes 5.11 Mbits/sec
[ 3] 1.0- 1.5 sec  264 KBytes 4.33 Mbits/sec
[ 3] 1.5- 2.0 sec  232 KBytes 3.80 Mbits/sec
[ 3] 2.0- 2.5 sec  240 KBytes 3.93 Mbits/sec
[ 3] 2.5- 3.0 sec  232 KBytes 3.80 Mbits/sec
[ 3] 3.0- 3.5 sec  272 KBytes 4.46 Mbits/sec

```

Figure 7 - Iperf data sample

2.4.2 Script module

The test bed allows for scripts to be placed in a directory and loaded. Users who write bash scripts or C scripts can use this function to send out scripts to nodes and specify another node in the network to complete the script on. The script needs to be written in a method that is accepted by the java test bed, in that the script must be executable with 2 parameters:

`-i xxx.xxx.xxx.xxx` and `-o outputfilename.xx`

CHAPTER 3: Ad-Hoc Wireless Network Java Test Bed

3.1 Introduction

The need for a simple method of testing wireless ad-hoc network protocols has become evident after manually configuring test nodes during test procedures. Working with multiple network connected nodes requires much moving around being that a manually administered test requires each node (netbook to be synonymous with node for network testing in this thesis) be programmed separately for all parts of a test which include topology setup, client-server throughput test commands, data gathering, retrieval of data from each tested node, data naming schemes, etc.. The developed test bed performs these functions with minimal interaction on the test nodes.

Java by means of Netbeans IDE was chosen for the test bed due to its .NET extensions allowing for easy network functions such as socket connections and file output streaming. Java swing components provided a simple enough interface to provide the user with functionality as well adhering to a reasonable software development timeline.

3.2 Ad-Hoc Network Test Bed Development

A basic structure of the test bed is given in the following flowchart. Variations from the original pre-development design were minor. The test bed operates in a sequential manner in terms of steps to perform a network test. The steps can be visualized in the total system design flowchart.

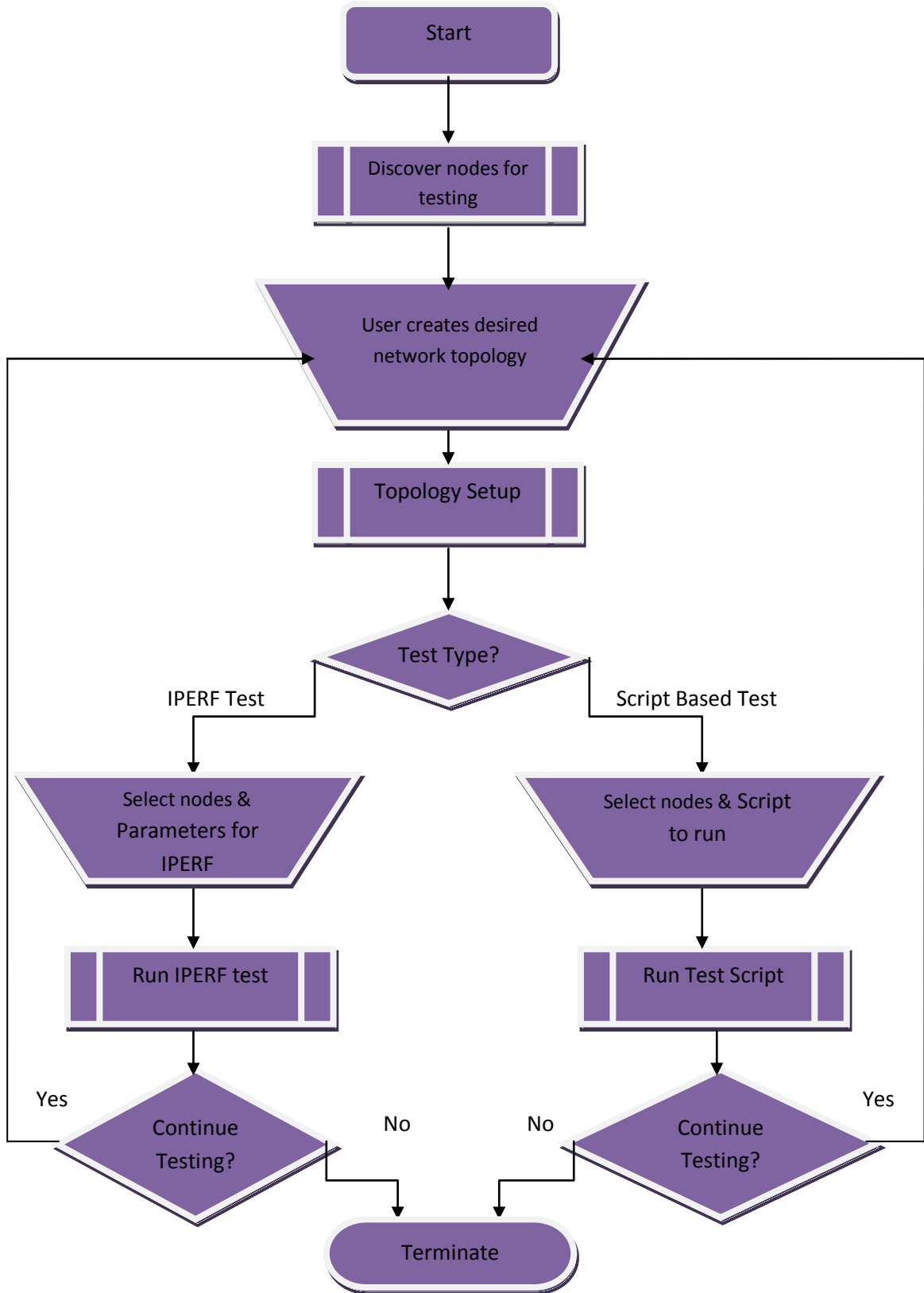


Figure 9 - Test Bed System Flowchart

There are two main parts to the Java Ad-Hoc Network Test Bed: The master and the slave/listener java software groups. The master software is the graphical user interface. This software is run on only one PC and must have a connection to every node in the test network. The main point of the testing software is minimal interaction with all of the involved test nodes, this is achieved by the Test Center. The other software is the slave, or listener software (referred to as the test node). This must be running on each of the nodes that would like to be involved in the testing. As commands are being issued from the Test Center, this software must differentiate between types of commands and different chains of action to complete. The slave software, while running, will remain indefinitely in a waiting period for commands from the master software.

The full graphical user interface is displayed in a visually pleasing manner with a step-by-step type layout. The final design is show in the below figure though changes to the positioning can be easily changed in future test bed versions.

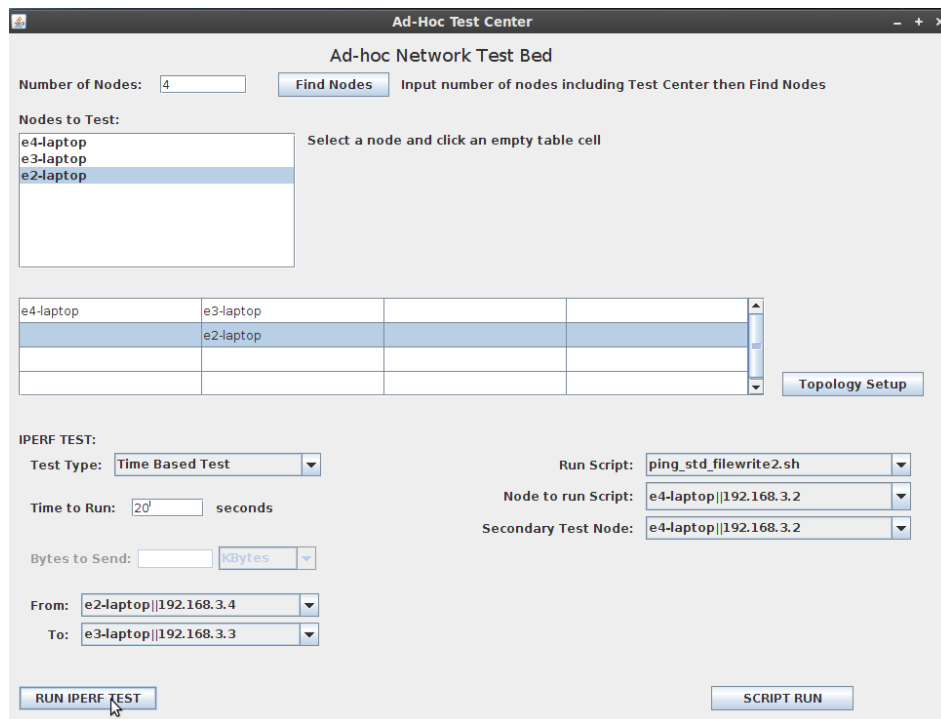


Figure 10 - Full Test Bed screenshot

3.2.1 Test Node Preparation

Prior to test setup the nodes require some manual configuration. To begin testing a network there is no way to get around putting the network into existence. On a Linux based system certain commands can be issued to create the ad-hoc test network. The IP subnet of the test center will determine the subnet on which you must attach the test nodes. The test center always must have the first IP address in the subnet.

Table 3 - Test Node Preparation Commands

Step:	Command:	Variables:
1	<code>sudo /etc/init.d/network-manager stop</code>	-
2	<code>ifconfig *wlan0 down</code>	*Interface card ID
3	<code>iwconfig *wlan0 mode ad-hoc</code>	*Interface card ID
4	<code>iwconfig *wlan0 ESSID **test_network</code>	**network name
5	<code>iwconfig *wlan0 channel ***5</code>	***channel 1-10
6	<code>ifconfig *wlan0 up</code>	*Interface card ID
7	<code>ifconfig *wlan0 xxx.xxx.xxx.xxx</code>	testcenter IP = xxx.xxx.xxx.1

The IP addresses should be arranged with additional nodes following the pattern in Table 4 on increasing the digit on the last octet as more nodes are added to the test.

Table 4 - Test Node IP addressing

NODE	IP
Test Center	192.168.3.1
Test node 1	192.168.3.2
Test node 2	192.168.3.3
Test node 3....	192.168.3.4...

All nodes must have the tested protocol installed on the system prior to starting test bed.

3.2.1.1 AODV-UU install

AODV-UU can be downloaded from a variety of websites including *www.sourceforge.com*. Once the file has been downloaded the user can enter the directory via the terminal and begin to install the protocol.

Table 5 - AODV-UU Install Procedure

Step:	Command:	Variables:
1	<code>cd ~/aodv-uu_0.9.6</code>	-
2	<code>make clean</code>	-
3	<code>make</code>	-
4	<code>make install</code>	-
5	<code>aodvd</code>	runs the protocol

3.2.2 Discovery of Test Nodes

The first step in completing a network test using the java-based Ad-Hoc Wireless Network Test Bed is to discover the test nodes involved. The user begins by inputting the number of total nodes in the small test network including the test center node.

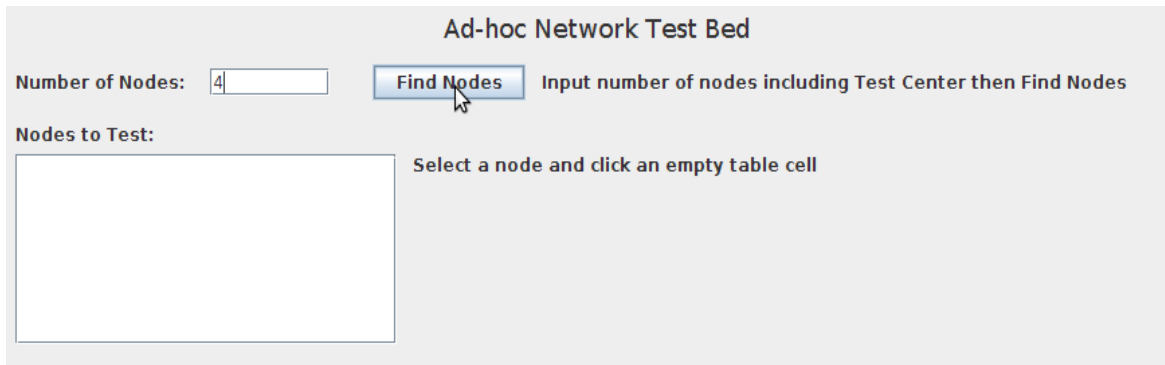


Figure 11 - Node Discovery Screenshot

The button is then pressed to begin searching and adding the nodes. The IP addresses are cycled through in the java code while attempting to open socket connections. Once a successful socket connection is made the test center contacts the node for information such as hostname and MAC address of the interface being contacted.

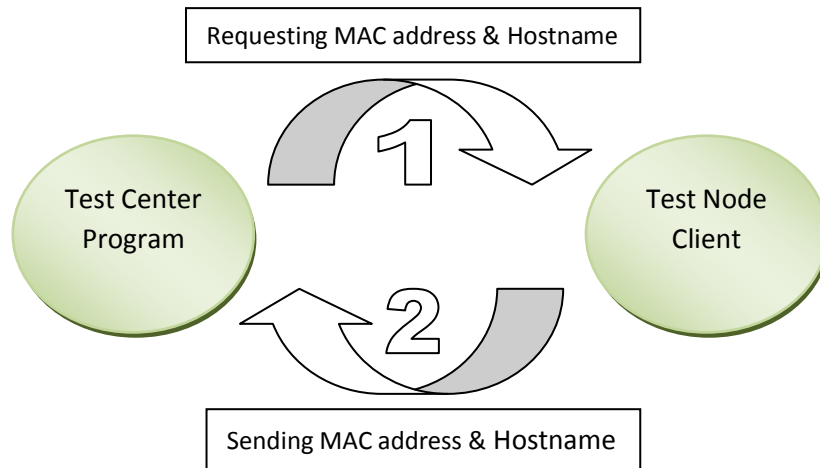


Figure 12 - Node Discovery Exchange

The test center stores the data received from the node into an array which contains the IP of the test node, the MAC address, and the hostname for visually aesthetic purposes on the GUI interface. The items are then added to combo boxes and list boxes for user interaction. The node discovery function cycles through as many IP's as specified in the initial phase where the user entered the number of nodes into the text area. This is to help cut down on time spent trying to open sockets to unused IPs in the system. When discovery is finished the system remains in an idle phase for the user to issue the next command.

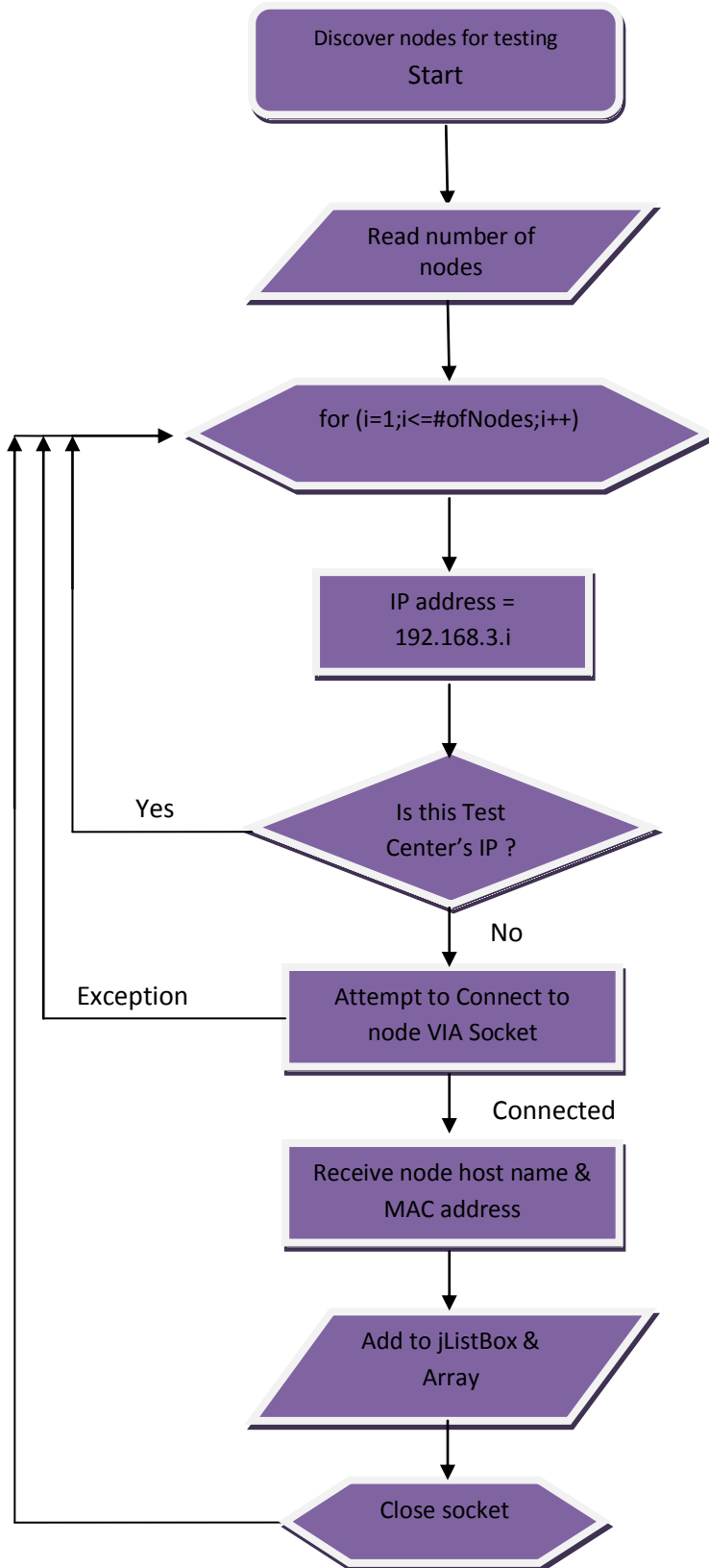


Figure 13 - Node Discovery Flowchart

3.2.3 Topology Setup

3.2.3.1 Topology Table

The graphical user interface of the test center utilizes a jTable as a method of allowing the user to enter information about the desired network topology that is needed. Using the list that has been populated during node discovery a user can select a node and then select a cell in the table. The node that was selected from the list first will then populate the cell in the table marking it as the position in the topology. The table functions on the basis that adjacent cells, whether vertical or horizontal, are interconnected neighbors in the desired network topology.

Consider the following example as a means to setup a network topology:

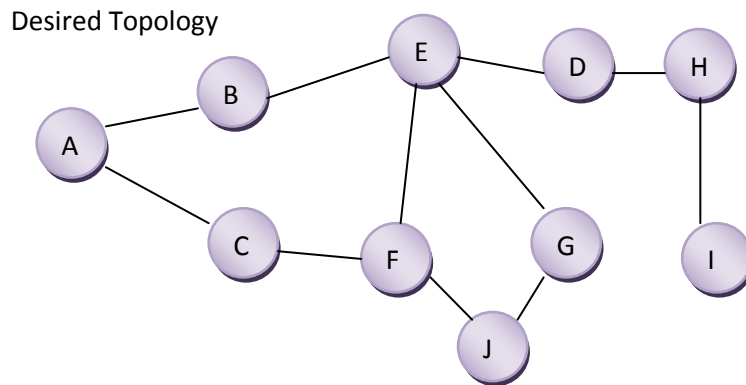


Table Input

		H	I
E		D	
B		E	G
A	C	F	J

Figure 14 - Topology Setup

The diversity of the topology arrangement is limited to the number of rows and columns in the Topology Table. For the purpose of testing such real-time mesh network protocols as AODV, the functionality has only been proven with up to 6 nodes[12], therefore the topology table need not be of extensive size.

3.2.3.2 Topology Decoder

An intelligent and functional method has been developed in java as a means to assure that each of the nodes receives the proper commands that will filter the correct MAC address to mimic the topology needed. Upon selection of the Topology Setup command button the table is scanned to view all nodes involved in the topology setup. The decoder then compares each item in the array to a duplicate array and checks for column and row values that do not fall within one increment or decrement of the current item. When the decoder finds such a case, an *iptables* command is build and sent out to the appropriate node to block incoming communication from that node[18]. After the entire array has been cycled through the topology decoder process will have successfully arranged the network into the topology as desired by the user. The following two figures display an actual rendering of the topology tables and the functional flowchart respectively.

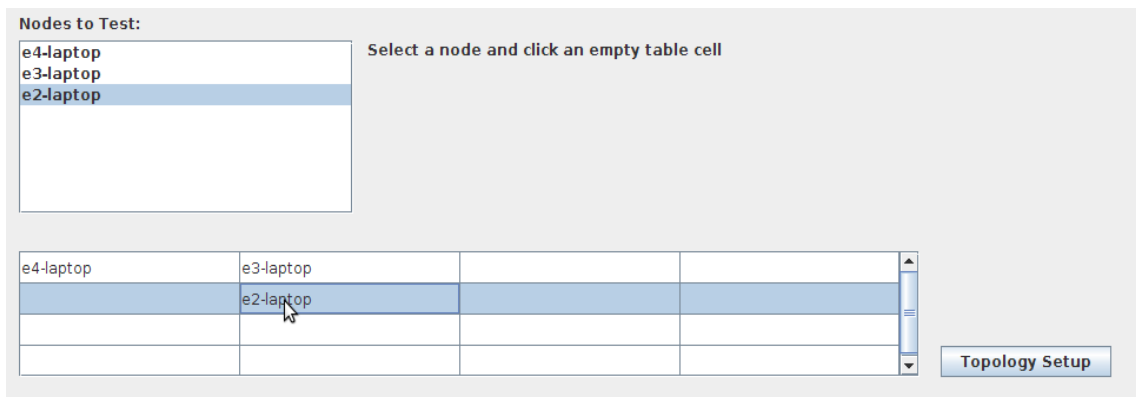


Figure 15 - Topology table screenshot

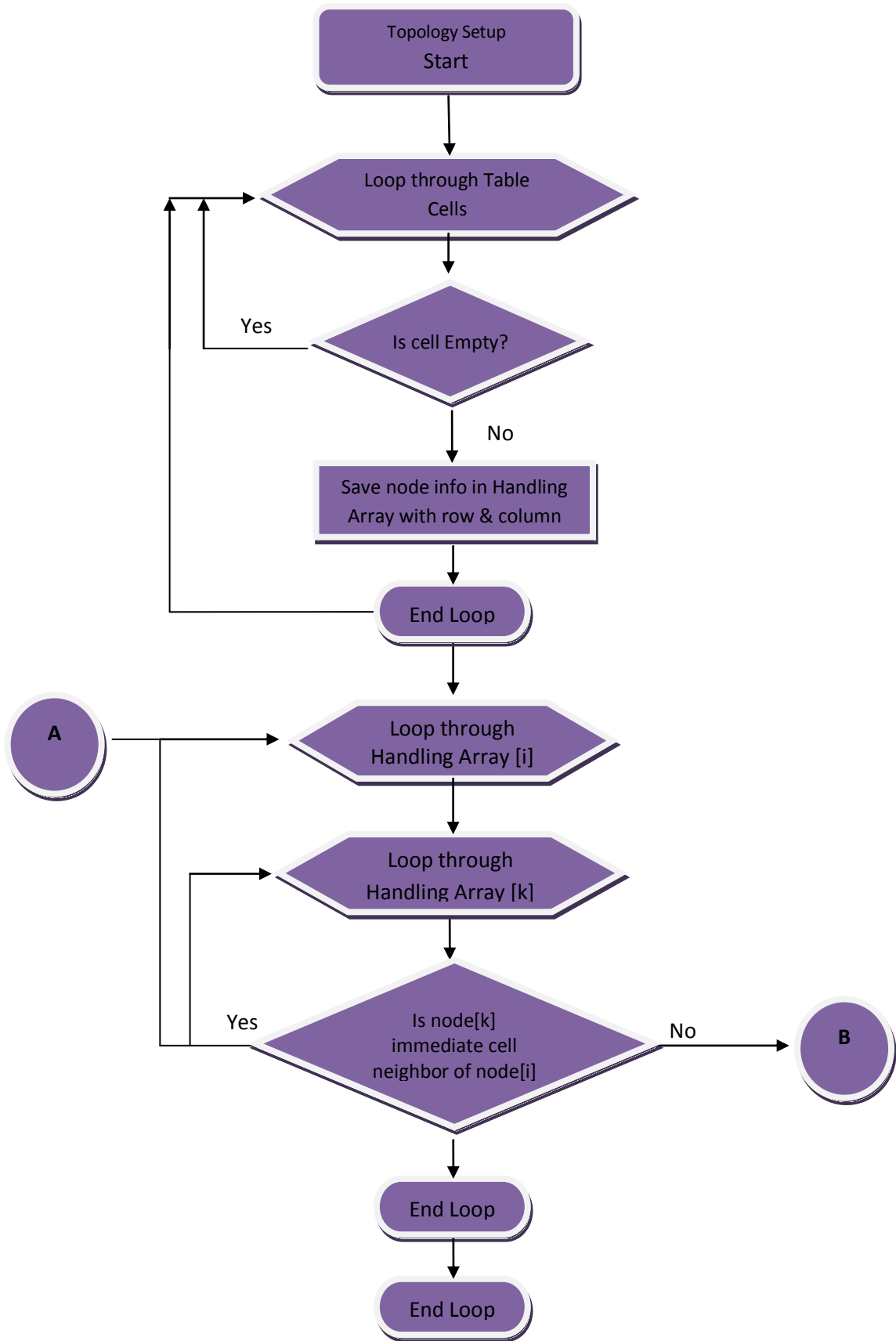


Figure 16 - Topology Table flowchart pt1

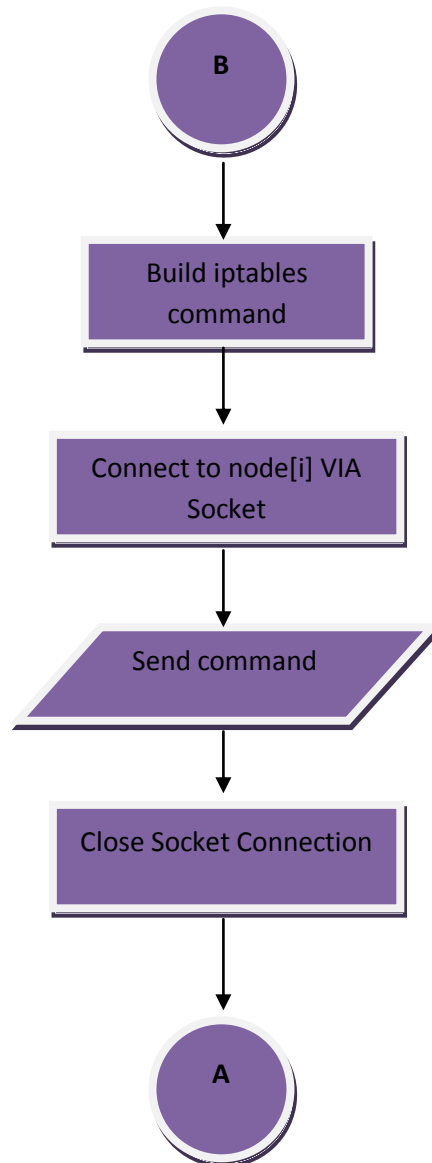


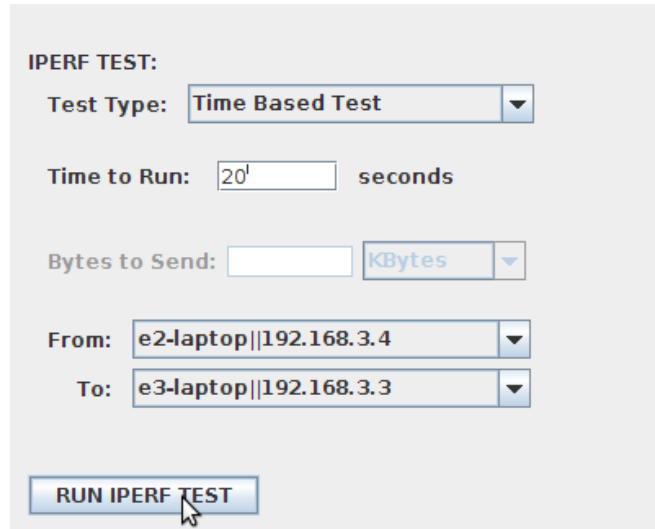
Figure 17 - Topology Table flowchart pt2

Now that this stage is complete the user has the freedom run tests on the topology. If the user decides to change the topology of the network it can be done by deleting the nodes from the tables and re-arranging the nodes in the desired manner. When the topology setup command button is selected again the test bed will flush all previous filtering rules from each of the nodes and set them up according to the new orientation.

3.2.4 Run Iperf Test

The test bed automatically fills in the drop boxes with all the available nodes at node discovery.

The section of the GUI that allows for iperf testing has already been populated with the available test nodes. The first portion of running an iperf test involves choosing the test type.



The screenshot shows a GUI titled "IPERF TEST:" with the following fields and controls:

- Test Type:** A dropdown menu set to "Time Based Test".
- Time to Run:** A text input field containing "20" followed by the label "seconds".
- Bytes to Send:** An empty text input field followed by a dropdown menu set to "KBytes".
- From:** A dropdown menu set to "e2-laptop||192.168.3.4".
- To:** A dropdown menu set to "e3-laptop||192.168.3.3".
- Run Button:** A button labeled "RUN IPERF TEST" with a mouse cursor hovering over it.

Figure 18 - Iperf test screenshot

There are two types of tests available to the user in the GUI. The user can either select a time based test or a byte based test. The time based test has a default value of 10 seconds. The user can specify a time to run the test in the text input field. This type of test will attempt to send a maximum throughput of data from one node to another for the given time period with a reporting interval of 0.5 seconds.

When the user specifies the byte based test, one node will attempt to send the desired amount of data to the other participating node and the test will finish once the entire amount has been sent through the network.

The user then selects the nodes in which the test will be carried out on. The advantage of this test is that a user can be running different mesh network routing protocols and collect data on the throughput and latency of the system between given nodes whether they are directly connected or contain a multi-hop route between the two nodes.

Once the *RUN IPERF TEST* command button is selected, the multiple variables governing the test are retrieved and a socket connection is opened to the initiating node (the node in the "From:" drop box). The IPERF command is built in a string format so that the node may execute the command directly upon retrieval. The command already contains the information about the time vs. byte test as well as the secondary test node. The secondary node will then be contacted and a specific command issued to it to await connection from the primary node. The test center will then open a socket to the primary test node and send the command. The primary node will then execute the command, thus commencing the test between itself and the selected secondary node. Upon completion of the test, the test file will be saved at the primary node. The test node will then request a socket connection to the test center on a different socket to send the data file from the test. The test center will save the test data with a name derived by the nodes involved and the test type with appendages if necessary. The test center automatically creates a directory for data files upon start up for the user. All tests are then to be saved automatically in this directory. The files are saved in the original format as verbose output to terminal from iperf for easy viewing. The files are saved with a header inserted inside the file with some important information about the test such as the nodes involved in the test and their IP addresses. A system flowchart of the iperf process is given in the following figures.

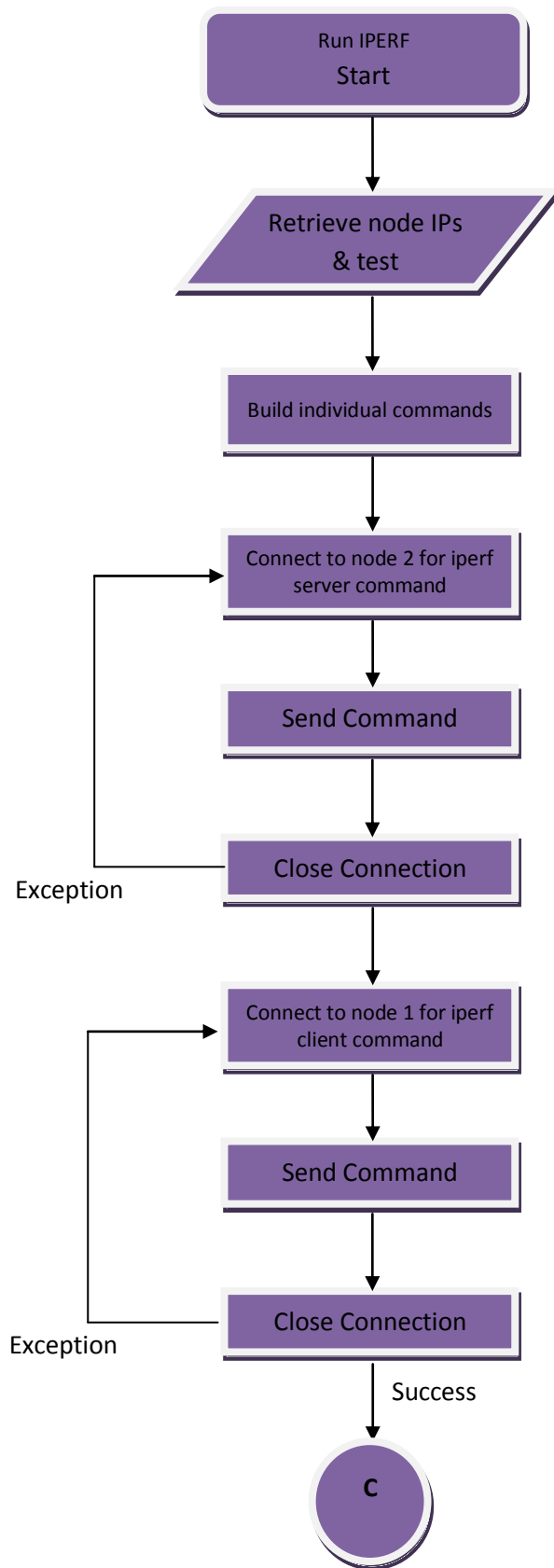


Figure 19- IPERF process flowchart pt1

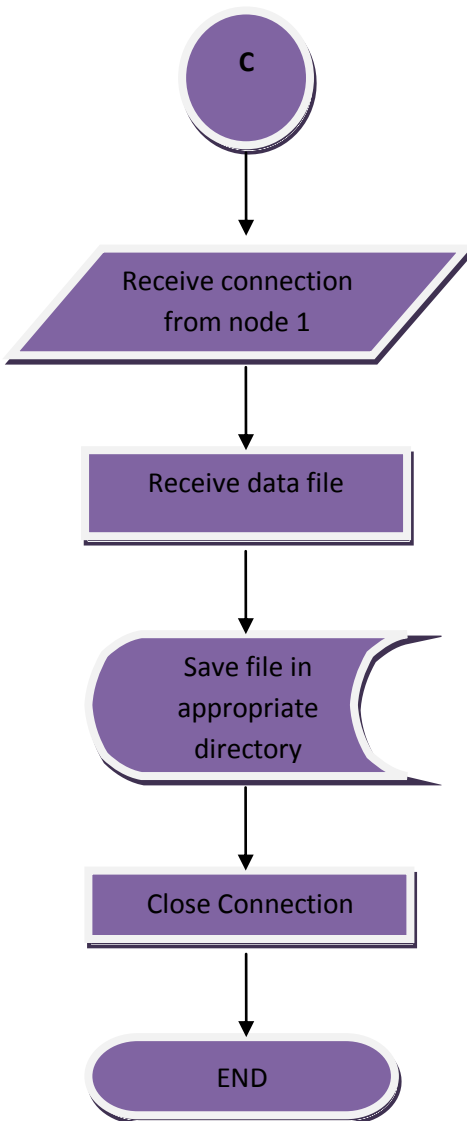


Figure 20 - IPERF Process flowchart pt2

3.2.5 Node/Listener Software

A vital member of the functionality of this system is the software that must be running on each of the nodes that may take part in testing. This software shall be denoted as the node software or listener software. This software sits in a idle state awaiting socket connection requests from the test center software. Once a socket connection has been requested, the listener software communicates with the test center and determines what the test center is asking for. A series of checks will determine what course of action is needed for the node to properly complete the

correct function. The listener software is made up of two java files, the main decision making class, and the runtime executing and file handling class. Between these two files the software can make proper decisions, execute terminal commands, receive files, send files, and request connections as well.

First the system detects the socket connection request and responds to it. The test center is then programmed to immediately send the command. The commands are handled according to what needs to be completed for each command.

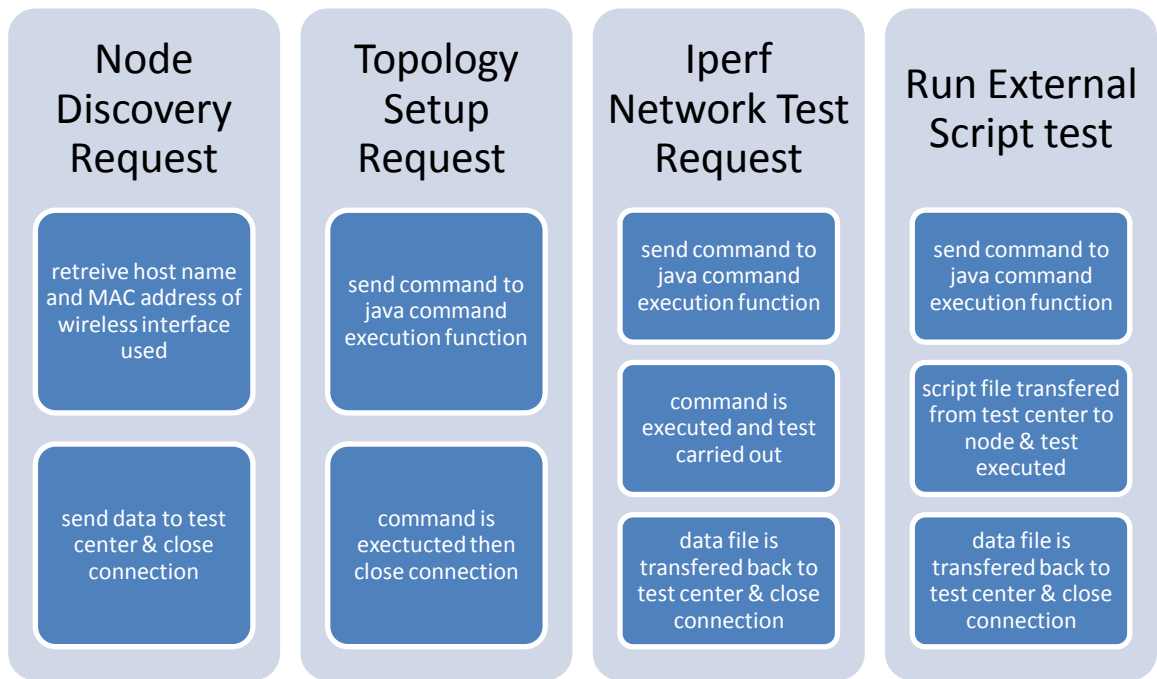


Figure 21 - Basic Layout of Listener Software

The Java Execution class also has decision making taking place. When the executable command is sent over to the execution function the command is first checked because in some cases file transfer will need to occur. This is completed in the Java Execution class as well as receiving any scripts that may need to be executed on the node. System flowcharts showing the control flow design have been designed to give the reader a better basic understanding of the processes.

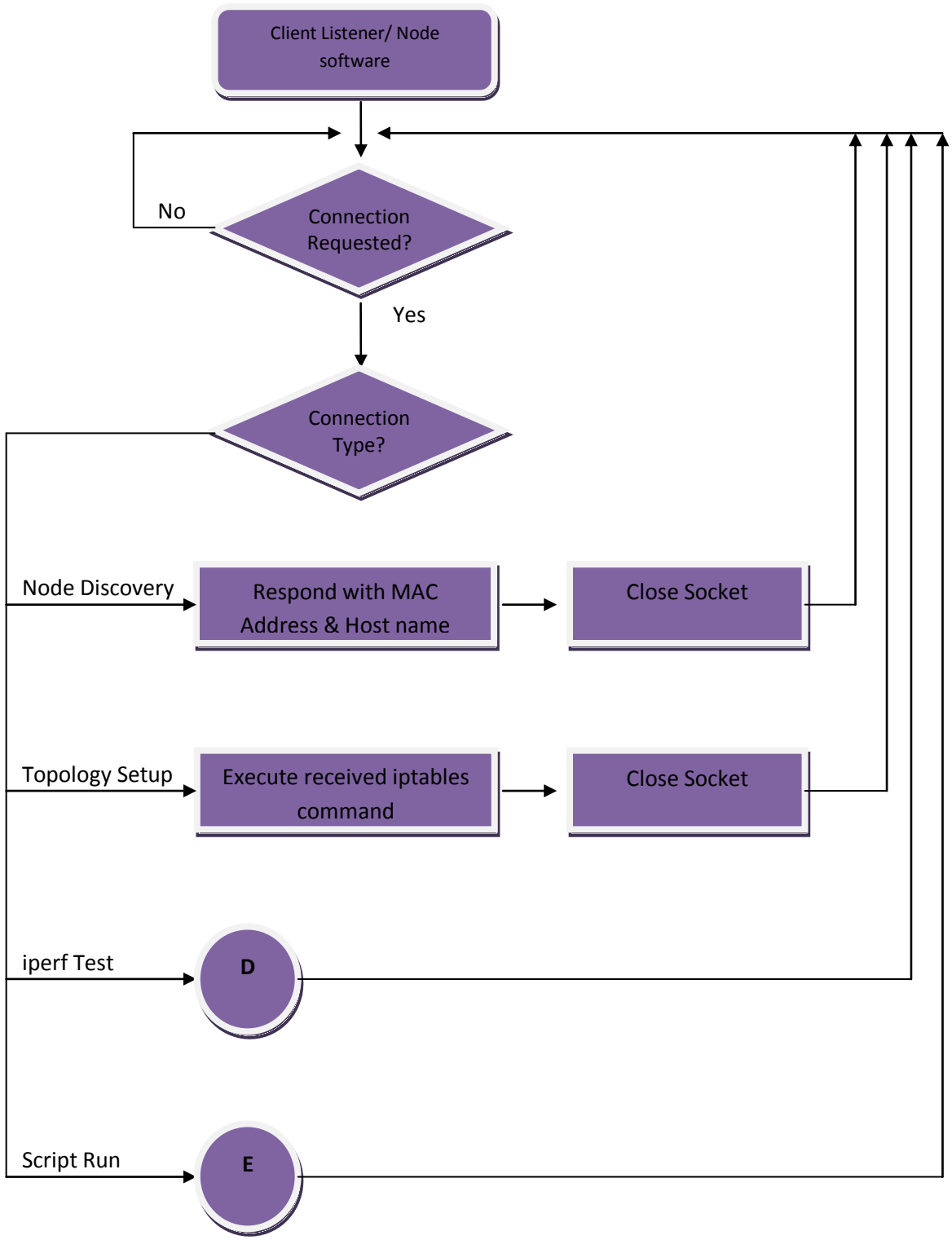


Figure 22 - Listener Software main flowchart

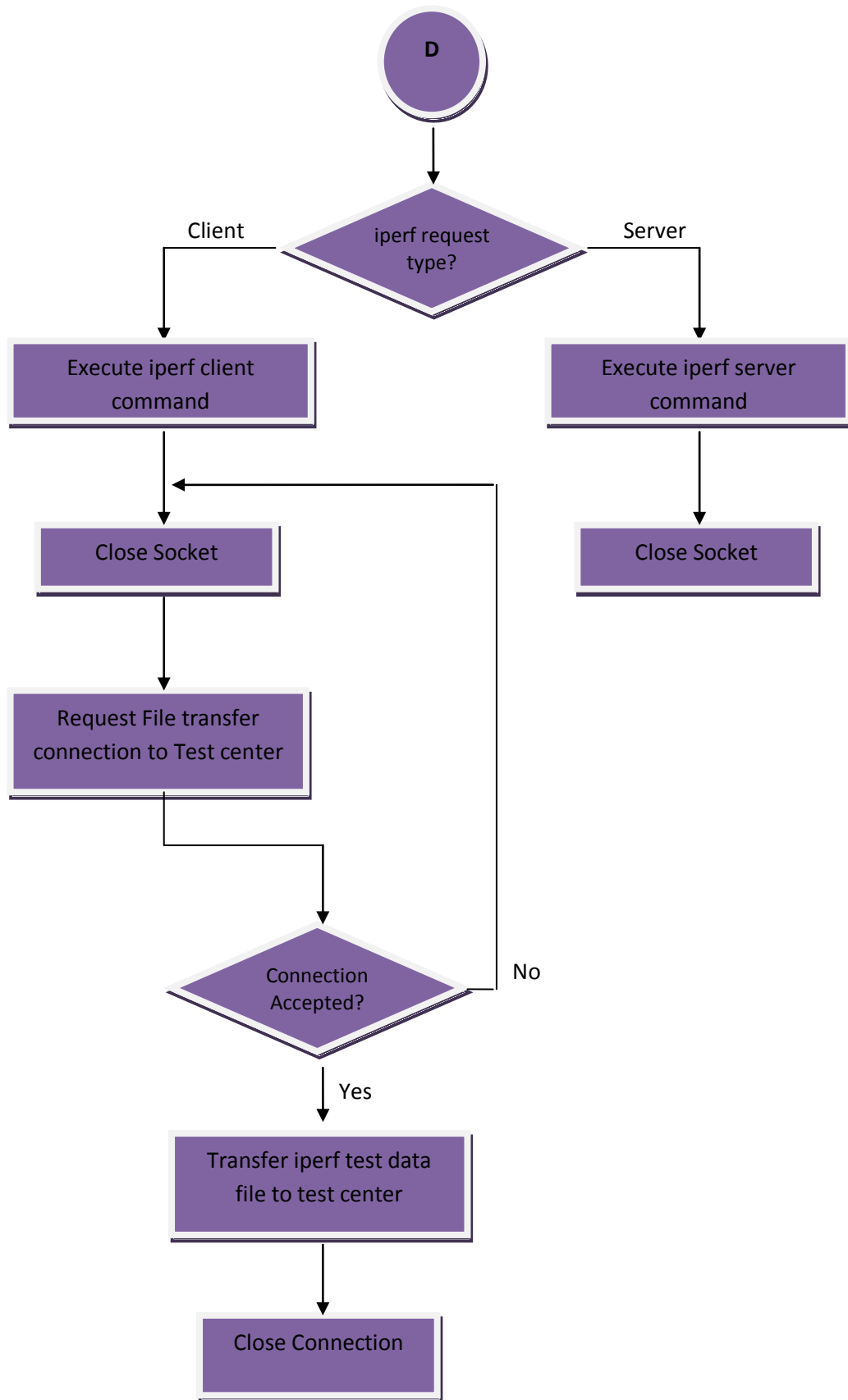


Figure 23 - Listener Software flowchart pt2

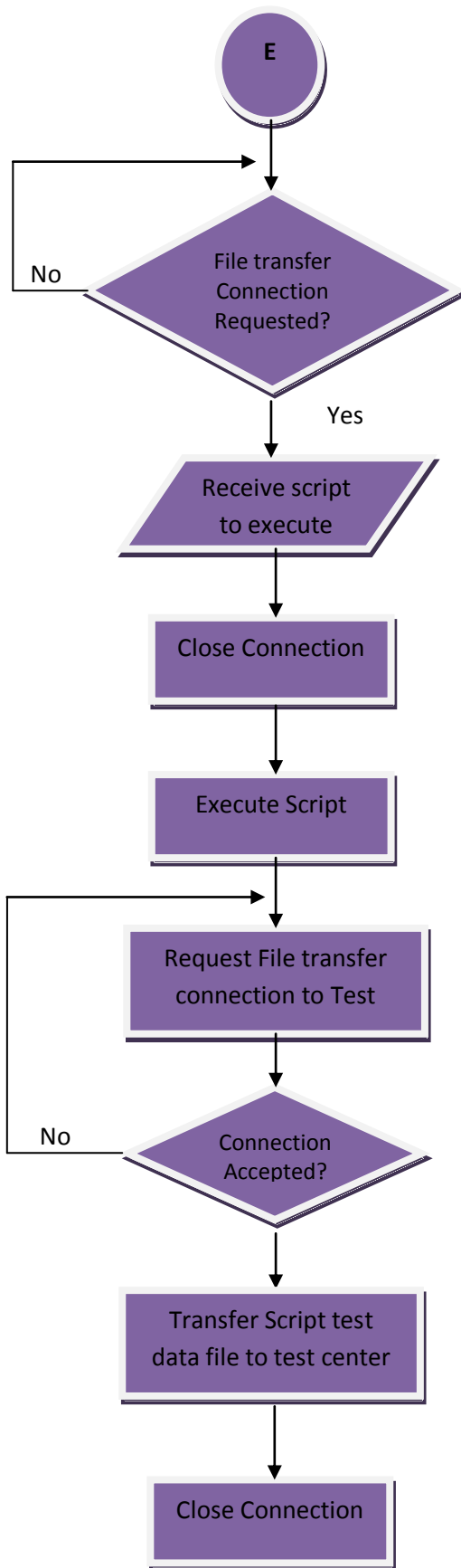


Figure 24 - Listener Software flowchart pt3

3.3 Chapter Summary

This chapter described the development and functionality of the Java Ad-Hoc Network Test Center. Explanations of functionality and reason were given for the different functions of the system. The background of the external testing utility was given as well as insight into running external scripts on test nodes. A detailed breakdown of preparation, execution, and data collection was given for the test bed. The main subjects were discussed – Node discovery, Topology Setup, and IPERF test execution.

CHAPTER 4: AODV QoS Design and Methodology

4.1 Introduction

The need for low latency, reliable emergency data transfer is evident. In the case of power line transmission monitoring, fault data or other data deemed to be of high importance must make its way as quickly as possible to control centers and/or upstream smart grid decision making devices. Current variations of QoS enabled mesh networking protocols do not include features to allow for the fastest possible route in the case of emergency but rather incorporate a variety of different strategies such as link statistics, geographical routing, etc.[19]. The following design and development assures a quick reduced traffic route to the required destination to send small amounts of emergency data for notification of a fault/failure in the system.

4.2 Traffic Reduction Methodology

Contrary to other QoS enabled designs available, this design incorporates network traffic reduction along the route from the emergency to the control center (destination). The main

aspect of this design is to request a high priority route. The nodes involved in the route setup will then give precedence to the node which has shown need for a low latency emergency route. The emergency can then be transmitted with the fastest possible delay. The route discovery structure in the base protocol is utilized and therefore the emergency route latency can be modeled by the minimum latency in a completely unloaded network for a given node transmission, dependent on topology and hop count to the destination node. The emergency information is given sufficient time to propagate through the network and make its way to the control center and then normal network operations resume.

4.3 Quality of Service Design

The design of the quality of service code was interleaved directly into the AODV-UU 0.96 C code. Various functions were developed as a means to monitor and carry out different operations necessary to moving towards a functional prioritized route. Along with multiple header files there were five main C files that viewed added code to facilitate the designed QoS scheme.

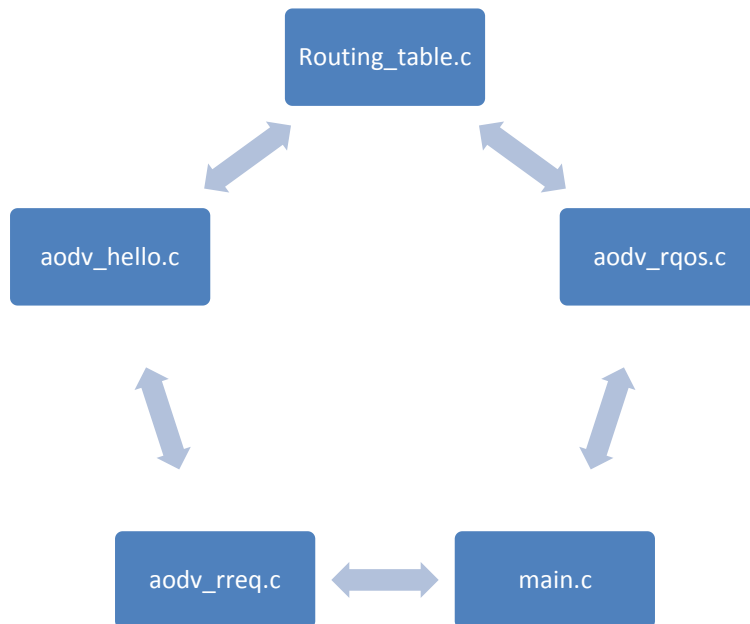


Figure 25 - QoS C files

4.3.1 Emergency Instantiation

When an emergency is sensed the AODV code calls a different function which instantiates the route request for the emergency route. The trigger is set high and then the AODV native route request discovery function is called with a destination that shall be designated emergency IP, the destination that the emergency data must traverse to. A timer (selected larger for testing purposes) is then set which will eventually end the emergency data transmission period.

```
void NS_CLASS send_rreq()
{
    DEBUG(LOG_DEBUG,0,"entered a send_rreq to %s", ip_to_str(emergency_ip));
    shawns_trigger = 1; //Trigger for emergency set high
    rreq_route_discovery (emergency_ip, 0, NULL); //calling a new route discovery process to control center
    timer_set_timeout(&iptable_delete_timer, 5000); //setting a arbitrary timeout for emergency |
    return;
}
```

Figure 26 - Code Excerpt send_rreq

Upon calling the native route discovery function the request will begin to propagate throughout the network with the knowledge of the emergency signified by the reserved header field being changed.

4.3.2 Intermediate Node Emergency Processing

The received route request function has been altered to check for the emergency field. Upon receiving a route request all nodes will begin the process with a check to acknowledge the QoS metric of the route request. The reserved flag is saved to the trigger variable and is then checked to see if an emergency route is needed. If an emergency route is needed, the originating address and the destination address are saved into variables to acknowledge them as

having key roles in the transmission of the emergency data. Another variable is set high to signify the need to clean non-emergency IPs.

```

shawns_trigger = rreq->res1; // reserved flag of QoS
rreq_new_hcnt = rreq->hcnt + 1; //incremented hopcnt for processing

if (shawns_trigger ==1) // if emergency rreq, we save important IPs
{
    DEBUG(LOG_DEBUG,0,"-----RECIEVED EMERGENCY RREQ TO %s-----",ip_to_str(rreq_dest));
    emergency_dest.s_addr = rreq_dest.s_addr;
    emergency_orig.s_addr = rreq_orig.s_addr;
    clean_ips =1; //notifier to clean IPs |
}

```

Figure 27 - Code Excerpt rreq_process

Throughout the AODV code there are checks in multiple functions which check for the value of *clean_ips*, a notifier of an emergency. These checks are based on new routing table entries into each of the nodes. The check is also located in the native functions which add precursors to the routing table. Once a new entry to a routing table has been acknowledged and the check has passed for an emergency a function called *manage_ips* is called to save the useful IPs in the route from the emergency to the emergency destination.

```

void manage_ips (struct in_addr prec, struct in_addr rte, int imp)
{
    if (rte.s_addr == emergency_dest.s_addr)
    {
        dest_prec.s_addr = prec.s_addr;
    }
    if (rte.s_addr == emergency_orig.s_addr)
    {
        orig_prec.s_addr = prec.s_addr;
    }

    manage_iptables (1);
    clean_ips = 0; // it will be reinstatiated if a new emergency rreq comes in
}

```

Figure 28 - Code Excerpt manage_ips

While managing the IPs, the precursors are saved into other variables so that they may be used later. Upon solving for the proper IPs in the path, a function denoted as *managed_iptables* is called to allow only the useful IPs in the emergency route. The native functions to invalidate and delete entries in the routing tables are used. The function cycles through the neighbors of the node checking to make sure the neighbor is not saved as the destination, source, or a useful precursor to the emergency route. Given the neighbor is found to be non-useful it is invalidated, deleted from the current routing table (to avoid congestion for rreq forwarding) and then filtered temporarily by IP address to allow for the quickest emergency data forwarding.

The emergency data transmission timer is set to call the *clean_iptables* function. This function will remove the rules which prohibit the other nodes from interrupting the emergency route data stream. The system will then function as typical AODV-UU until the next emergency is triggered.

4.4 Chapter Summary

The development of the QoS scheme has been documented. All code can be found in the appendices though useful excerpts have been shown to allow the reader to follow the design methodology and code flow. The emergency trigger will signal the emergency on a given node which forms the basis for one set of code. The majority of the prioritization scheme is fulfilled by the intermediate nodes which give precedence to any data from neighbors which are used in the emergency route, while also checking to make sure traffic is originating from the emergency source itself and not traversing through a neighbor node from an alternate source.

CHAPTER 5: Results and Analysis

5.1 Introduction

The QoS prioritization scheme has been successfully designed tested in a real world scenario using test nodes in the form of netbooks. Four netbooks with N450 Intel atom processors were the available equipment used for testing the QoS implementation. The testing was completed using ping data for latency along with topology setup. Iperf tests were utilized to show the differences in throughput with and without the prioritization scheme.

5.2 Test Scenario

5.2.1 Simulated Emergency

Whether the emergency be a transmission line fault, an overheating transformer, or burnt out lamp, the system must be triggered by a sensor attached to an AODV QoS enabled device. Even though live testing is the feature of this thesis, the emergency event must be simulated. The AODV code has been modified for testing purposes to set an emergency flag high in a node after a timer expires as to mimic a sensor reading error occurring after some time of running normal data collection and transfer. The chosen time period was 20 seconds after AODV has been running, then the emergency flag will be set high triggering a node to send the aforementioned high priority route request.

5.2.2 Test schedule & Results

The test setup was chosen to facilitate two nodes sending periodic data through a junction node toward another destination (simulated control center) node. This would allow for one of the nodes to trigger an emergency. Measurements of the pre-emergency and post-emergency

bandwidth can then be calculated. The same scenario also proved useful for showing the latency of each of the routes toward the destination address.

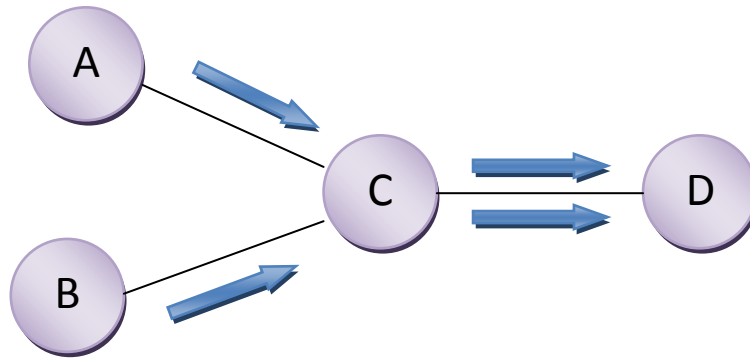


Figure 29 - Test Topology

Nodes A and B are sending data to destination D. The bottle neck can be clearly seen at node C where the link between C and D suffers due to the amount of data being double of what it is in between links A – C as well as B – C. The QoS emergency is then triggered after some time in node A. The system then changes to have the following data flow pattern.

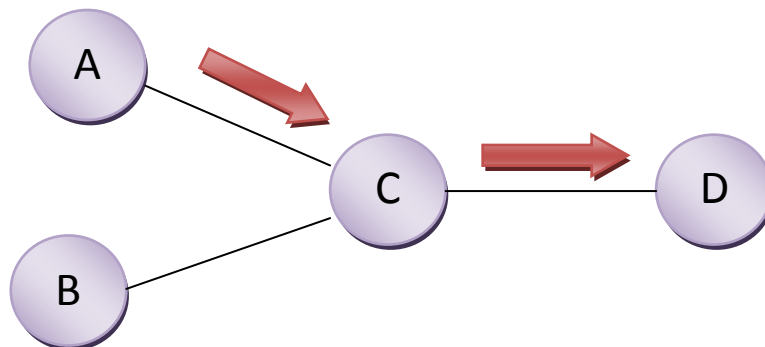


Figure 30 - Emergency Route triggered in A

The traffic in the link that was once sharing the throughput from both nodes A and B is now dedicated solely to A for the emergency route. The throughput of the data has increased to the maximum allowable throughput for the scenario to cope with the emergency.



Figure 31 - Throughput comparison

The throughput of the data has increased dramatically from the first scenario to when the emergency was triggered and traffic was halted from node B. Figure 32 displays the throughput slightly differently than just displaying the throughput as MB/s at each time interval. The iperf test was scheduled to output the amount of bytes transferred every half second. This graph shows the cumulative bytes sent through the network by node A in a second frame. There are multiple reasons that govern the dramatic increase in throughput displayed. Holding all link qualities constant and the two sending nodes having equal transmission rates, it can be seen that at C, the junction node, the bottleneck of the system occurs. In the normal route test set link C – D has clearly been overloaded in comparison to links A – C and B – C which causes the

throughput to decrease. Node C is processing data from two senders as well and AODV must appropriately continually forward data through itself. In the case of the emergency route, not only does link C – D view cleared traffic congestion, but node C is now fully dedicated to passing data from A to D. The throughput viewed in the emergency route scenario is essentially the same throughput excepted from running a three node test with A – C – D as the topology.

The latency of the system has also decreased for the emergency period. As for the following test schedule, the latency of pings from A – D have been plotted with external traffic from node B as well as without traffic from node B.

The average latency of ping test data has been plotted as seen below with traffic and without traffic. Multiple tests carried out with bursts of 10 pings have been accessed for data collection. It can be seen that with external traffic the latency is much higher and yields a more sporadic pattern due to the junction node being responsible for more data transfer and possible collisions. When the latency of A is recovered without external traffic we view a reduced latency with a more reliable data trend.

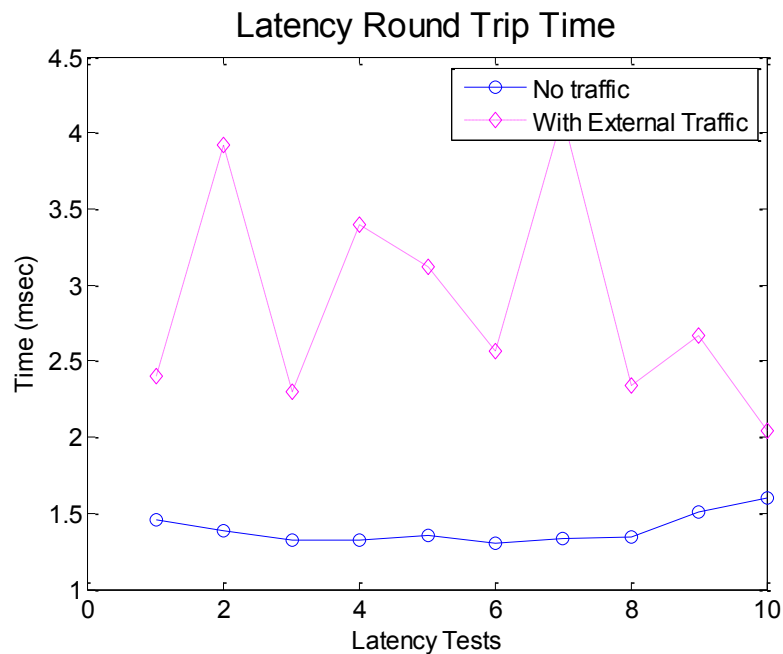


Figure 32 - Latency of node A

The system is then tested for latency with the simulated emergency occurring during the ping latency data collection period. 20 pings per set at intervals of one second are administered to the network for both routes A – D and B – D. The emergency is then triggered at the 10 second mark and the latency changes in the system can be viewed. For lack of a method to have undefined points in the Matlab plot, the traffic data set has been dropped to an indefinable zero value to better observe the graph. The diamond-point Traffic set should essentially move to infinite latency during the five second emergency period denoted between the 10th and 15th ping test. The emergency node clearly shows a reduction in latency during the emergency designated period. These values coincide with the values viewed in the previous figure for latency with traffic and without traffic.

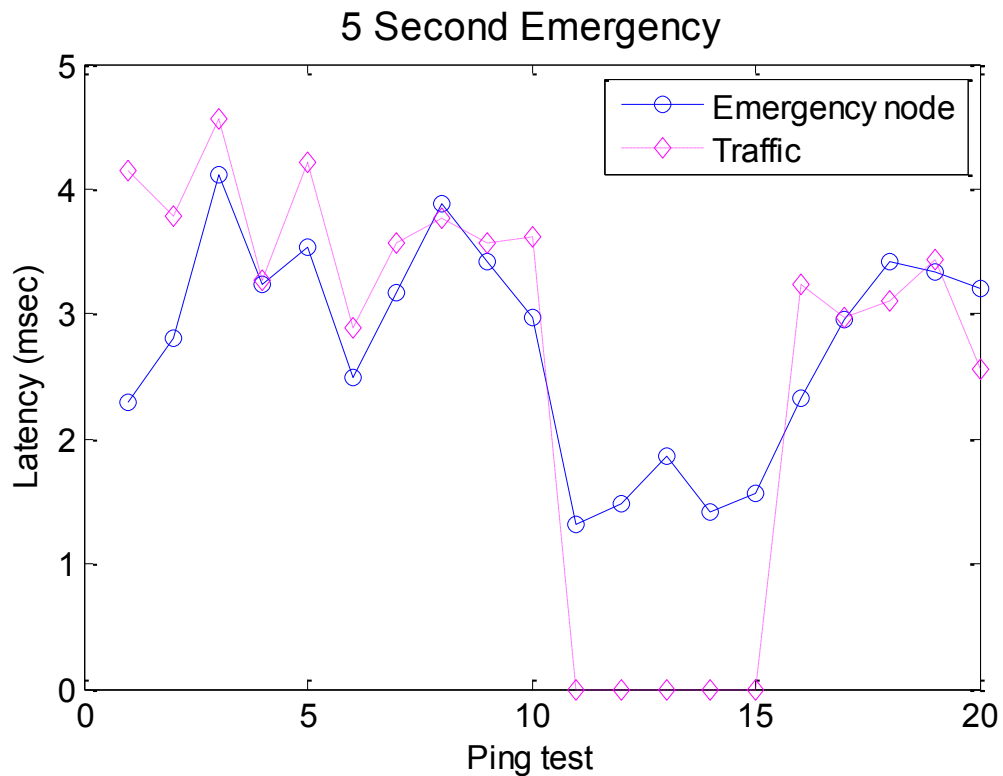


Figure 33 - Latency drop during emergency

5.3 Chapter Summary

This chapter outlined the methods of testing the new QoS system implemented into AODV-UU 0.96. The explained methodology of a simulated emergency led to the explanation of the test schedule and test topology. The throughput of the emergency node showed a large increase with coincided with a similar throughput as to when a node is transferring data at the maximum possible rate observable to its topology. The latency of the system decreased accordingly during the emergency period until the simulated emergency data had been sent and then the emergency period ended, thus returning the system to the original higher congested latency value prior to the emergency transfer.

CHAPTER 6: Conclusions and Future Works

6.1 Introduction

The concluding remarks shall be stated as to the functionality of the newly developed test bed and the improvements in AODV-UU towards low latency transfer of emergency data for smart grid applications. Future work towards continual improvement to the test bed and QoS protocol shall be discussed as well as ideas towards further efficiency of QoS enabled AODV.

6.2 Concluding Remarks

6.2.1 Java Ad-Hoc Network Test Bed

Java proved to be a useful tool in creating the test bed. Through the use of Netbeans IDE the graphical user interface was well organized and created with minimal manual coding. Each stage

of the test bed (node discovery, topology setup, etc.) proved to function according to the original plan. There are a wide range of uses for this test bed because a user can now test any different type of wireless mesh network, have topology setup, and basic network testing tools available without a sharp learning curve and the time wasted with tedious manual command input on each of the test nodes.

6.2.2 QoS Prioritized Emergency Route for AODV-UU

The QoS scheme was interleaved into the code structure of AODV-UU with the functionality goals met. The need for low-latency emergency data has fueled the need for a scheme of this type. The system used basic principles that allowed for the emergency route to be implemented with maximum efficiency in the network topology. A clear increase in throughput has been achieved during a traffic-input emergency-triggered scenario. The latency of the node which instantiates the emergency route request is seen to have a reduction immediately allowing for faster data transfer in a time when pertinent data is needed to be transferred. The system has functioned according to design specifications and previous speculations.

6.3 Future Works

6.3.1 Java Ad-Hoc Network Test Bed

The java test bed is fully functional though there are some aspects that could be improved for future developed versions. The first improvement to the system would be a more efficient method of node discovery using a multi-cast server method. Datagram packets were not used in this version, though after more investigation and interaction with java development, this method would be quicker and more efficient to finding all nodes in a certain subnet.

The iperf testing of nodes in the topology could one day include a multiple simultaneous test option. This would facilitate multiple nodes testing throughput and latency to different paths through the network. This would help users gain a better knowledge of junction node bottlenecks and the effects of the routing protocols route establishment delays.

6.3.2 QoS Prioritized Emergency Route for AODV-UU

The emergency route implemented into AODV-UU is still strictly a network-layer protocol, therefore the periodic data (“traffic”) sent during times of emergency are halted and must be managed by the application or transport layer which is feeding the data to the lower layers for routing. A future, multifunction system could also be implemented into AODV-UU that would buffer data and alleviate the responsibility from the sensor reading data collection application. This could be accomplished by each node receiving some type of acknowledgement before an emergency route node is halting their traffic. The notified node could then take action to retain the higher layer passed data for the short period that the emergency is occurring on the emergency node.

The QoS scheme could be better tested with real sensors and hardware that could trigger emergencies based on external input such as a rise in temperature. This would yield more potent data as to what the size of an emergency packet should be and thus gaining a better understanding of how the system will function in terms of latency and traffic with smaller or larger emergency data size.

6.4 Chapter Summary

The concluding remarks have been stated showing the data by means of testing. Positive results were yielded and discussed about the QoS implementation and the need for a ad-hoc mesh network test bed reiterated. Future works were discussed for both portions of the thesis and benefits of working towards further improvements have been identified.

APPENDICES

See attached supplemental files.

- **Java Wireless Ad hoc Network Testbed**
- **AODV-UU Source files with QoS Amendments**

REFERENCES

- [1] Torchia, M. (2012). *IDC Energy Insights, (2012). Worldwide Utility Smart Grid Spending Forecast, 2010-2015*. IDC Energy Insights.
- [2] U.S. Department of Energy: Office of Electricity Delivery & Energy Reliability. (2011). *Smart Grid Research & Development: Multi-Year Program Plan (MYPP)*.
- [3] Gungor, V.C.; Bin Lu; Hancke, G.P.; , "Opportunities and Challenges of Wireless Sensor Networks in Smart Grid," *Industrial Electronics, IEEE Transactions on* , vol.57, no.10, pp.3557-3564, Oct. 2010
- [4] Elster. (2012). *Residential Meters*. Retrieved 07 07, 2012, from Elster Metering: http://www.elstermetering.com/en/residential_meters.html
- [5] Perkins, C., Belding-Royer, E., & Das, S. (2003). Ad hoc On-Demand Distance Vector (AODV) Routing. *Internet Experimental RFC 3561* .
- [6] Nordstrom, E. (2002). *AODV implementation on Linux*. Retrieved from <http://sourceforge.net/projects/aodvuu/>
- [7] ISI. (n.d.). Retrieved 03 25, 2011, from Network Simulator 2: <http://www.isi.edu/nsnam/ns/>
- [8] Sommer, C., Dietrich, I., & Dressler, F. *Simulation of Ad Hoc Routing Protocols using OMNeT++*. Department of Computer Science, University of Erlangen, Germany.
- [9] Flynn, B. R. (2009). *Key Smart Grid Applications*. GE Energy.
- [10] Day, J.D.; Zimmermann, H.; , "The OSI reference model," *Proceedings of the IEEE* , vol.71, no.12, pp. 1334- 1340, Dec. 1983
- [11] Ahmed, F., & Sajjadur Rahim, M. (2011). Performance Investigation of Two-Classes of Manet Routing Protocols Across various Mobility Models with QoS Constraints. *Internation Journal of COmputer Networks & Communications (IJCNC)* , Vol.3, No.2.
- [12] *RFC3561*. (2003, 07). Retrieved 01 07, 2011, from IETF: <http://www.ietf.org/rfc/rfc3561.txt>
- [13] E.800: Terms and definitions related to quality of service and network performance including dependability. (1994, 08). *2007/2008 Ammendments* . ITU-T Recommendation.
- [14] Badis, H., & Al Agha, K. (2005, 11). Quality of Service for Ad hoc Optimized Link State Routing Protocol (QOLSR). Vancouver, BC, Canada: Internet Engineering Task Force.

- [15] Daqing Gu; Jinyun Zhang; , "QoS enhancement in IEEE 802.11 wireless local area networks," *Communications Magazine, IEEE* , vol.41, no.6, pp. 120- 124, June 2003
- [16] Beijnum, I. v. (2002). Building Reliable Networks with the Border Gateway Protocol. In *Traffic Engineering: Queuing, Traffic Shaping, and Policing*. O'Reilly Media.
- [17] Clayton, L. (2011). *Testing Network Performance*. Retrieved 07 2012, from Experts Exchange: http://www.experts-exchange.com/Networking/Network_Management/Network_Analysis/A_8010-Testing-Network-Performance.html
- [18] Russel, P. (1999-2000). *iptables Project*. Retrieved 2011, from netfilter: <http://www.netfilter.org/projects/iptables/index.html>
- [19] Djenouri, D.; Balasingham, I.; , "Power-aware QoS geographical routing for wireless sensor networks — Implementation using Contiki," *Distributed Computing in Sensor Systems Workshops (DCOSSW), 2010 6th IEEE International Conference on* , pp.1-5, 21-23 June 2010
- [20] Lee, S.-J.; Gerla, M.; Toh, C.-K.; , "A simulation study of table-driven and on-demand routing protocols for mobile ad hoc networks," *Network, IEEE* , vol.13, no.4, pp.48-54, Jul/Aug 1999
- [21] Wright, S.; Anschutz, T.; , "QoS requirements in DSL networks," *Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE* , vol.7, no., pp. 4049- 4053 vol.7, 1-5 Dec. 2003
- [22] *The Java Tutorials*. (1995, 2012). Retrieved 01 07, 2012, from Oracle: <http://docs.oracle.com/javase/tutorial/>
- [23] *C, C++ Programming Tutorials*. (1997, 2011). Retrieved 04 2012, from Cprogramming.com: <http://www.cprogramming.com/tutorial.html>

VITA AUCTORIS

Shawn A. Ruppert was born and raised in Windsor, ON in 1987. He graduated from Vincent Massey Secondary School (V.M.S.S) in 2005 and then began Electrical Engineering at the University of Windsor. Shawn was a part of the cooperative education program and gained work experience with The Corporation of the Town of Lakeshore, Lakeshore, ON; INA Schaeffler KG, Herzogenaurach, Germany; Chrysler Canada Automotive Research and Development Center, Windsor, ON; and Alcatel-Lucent, Ottawa, ON. Shawn graduated with a Bachelors of Applied Science in Honors Electrical & Computer Engineering in October 2010. He then began his Masters of Applied Science at the University of Windsor in January 2011 under the supervision of Dr. Kemal Tepe in the discipline of Wireless Communications and Computer Networks. Shawn has been a part of many projects including Zigbee enabled transmission line monitoring, Ad hoc mesh networking protocol testing, Android application development for electric vehicle proof of concept, and Java wireless network test bed development.