

ABSTRACT

Title of Document: IMPLEMENTATION, EVALUATION, AND
 APPLICATIONS OF MOBILE MESH
 NETWORKS FOR PLATFORMS IN MOTION

Jared Stanislaus Napora
Master of Science, 2009

Directed By: Professor Christopher C. Davis
 Department of Electrical and
 Computer Engineering

This thesis explores the selection, implementation, and evaluation of two mobile mesh networks, each involving a different distributed computing problem. In the forthcoming discussion, it will become apparent how system constraints affect the optimal choice of mesh networking design and implementation in these cases.

The first problem explores the design and implementation of a distributed computing mesh network that will allow a collection of autonomous land vehicles to gather, process, and exchange information in an unknown environment. This network was established by adapting standard commercial 802.11 routers and by providing a software framework that handles all communication between wireless nodes. The second problem involves the design of a network for tracking and monitoring

personnel. This network was implemented utilizing ZigBee modules due to power and custom implementation constraints. Both networks were tested with respect to their specific design constraints and they lay the foundation for additional application development and research.

IMPLEMENTATION, EVALUATION, AND APPLICATIONS OF MOBILE
MESH NETWORKS FOR PLATFORMS IN MOTION

By

Jared Stanislaus Napora

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Master of Science
2009

Advisory Committee:
Professor Christopher C. Davis, Chair
Professor Gilmer L. Blankenship
Professor Victor L. Granatstein

© Copyright by
Jared Stanislaus Napora
2009

Dedication

To my parents and brother, for their continued support and encouragement.

Acknowledgments

I would like to take this opportunity to express my thanks to those who have helped me in various stages of my academic career. At the University of Maryland, I have had the privilege of working in the Autonomous Systems Lab where I was able to design, build, and program several ground and aerial autonomous vehicles. I would like to thank Dr. Blankenship for providing the opportunity to work in this lab and to continue assisting undergraduate students as they designed autonomous ground vehicles in the University of Maryland capstone course, ENEE408I. I have had some of the most gratifying experiences while studying in this lab. Working with John Karvounis and Mike Stanley has provided memorable experiences, as we enhanced the lab over the past several years. It has also been very enjoyable working with John as we developed the ASL Framework, adding more features and overall enhancements. I am also deeply indebted to Techno-Sciences and TRX Systems where I have been able to gain real world experience in product development. Additionally, I would not have been able to achieve this academic milestone without the continued support and advice of Dr. Davis and Dr. Blankenship over the years. Lastly, I would like to thank all those individuals at the University of Maryland, who have made my academic career both stimulating and worthwhile.

Table of Contents

Dedication.....	ii
Acknowledgments.....	iii
Table of Contents.....	iv
List of Tables.....	viii
List of Figures.....	ix
Chapter 1: Introduction, Motivation, and Outline.....	1
1.1. Introduction.....	1
1.2. Motivation.....	3
1.3. Chapter Outline.....	6
Chapter 2: Mesh Networking Background.....	8
2.1. Introduction.....	8
2.2. Design Considerations.....	9
2.2.1. Power.....	9
2.2.2. Bandwidth / Operating Frequency.....	10
2.2.3. Interface.....	12
2.2.4. Size.....	12
2.3. Wi-Fi Networks.....	13
2.3.1. Wireless Mesh Networking Protocols.....	13
2.3.1.1. WDS Configuration.....	14
2.3.1.2. AODV Protocol.....	17

2.3.1.3. OLSR Protocol.....	18
2.3.1.4. B.A.T.M.A.N. Routing Protocol.....	20
2.3.2. Open Source Firmware	22
2.4. ZigBee Networks	23
2.4.1. Comparison with other Wireless Standards	24
2.4.2. Applications of ZigBee Networks.....	25
2.4.3. Organization of a ZigBee Network.....	27
2.4.3.1. ZigBee Coordinator (ZC).....	27
2.4.3.2. ZigBee Router (ZR).....	28
2.4.3.3. ZigBee End Device (ZED).....	28
2.4.3.4. ZigBee Network Topologies.....	29
2.4.4. ZigBee Stack.....	32
2.4.4.1. Physical/Data Link Level.....	33
2.4.4.2. ZigBee Stack Level.....	33
2.4.4.3. Application Level.....	34
2.4.5. Node Addressing.....	35
2.4.5.1. Binding and Binding Table.....	36
2.4.6. Network Gateways.....	37
Chapter 3: Network Selection and Implementation.....	38
3.1. Introduction.....	38
3.2. Wi-Fi Networks	38
3.2.1. Router Selection.....	39
3.2.2. Firmware Selection	40

3.2.3. Testing Platform.....	41
3.2.3.1. Components	43
3.2.4. Network Organization.....	44
3.2.5. Routing / Mesh Network Formation.....	46
3.2.6. Network Utilization	47
3.2.7. ASL Framework Structure.....	48
3.2.8. Peer-to-Peer Communication Device Specifications.....	49
3.2.8.1. Message Wrapping.....	52
3.2.8.2. IBinarifiable Interface.....	54
3.3. Wi-Fi Network Tests	54
3.3.1. Text-Based Transmission using the ASL Framework	55
3.3.2. Live Webcam Image Feed using the ASL Framework.....	56
3.3.3. Entering and Exiting the Network	60
3.4. ZigBee Networks	62
3.4.1. Device Selection	62
3.4.2. Testing Platforms	63
3.4.3. TRX Systems Custom PCBs.....	65
3.4.4. Microcontroller Pairing.....	66
3.4.4.1. Z-Stack API	66
3.4.5. Network Implementation	71
3.5. ZigBee Network Tests	73
3.5.1. Ranging Tests.....	74
3.5.2. Transmission of Data between Nodes.....	77

Chapter 4: Future Work	78
4.1. Introduction.....	78
4.2. Competing Routing Algorithms.....	78
4.3. Further Use of Ranging.....	79
4.4. More Real World Applications.....	80
Appendix A: Custom Router Firmware Installation.....	82
A.1. Introduction.....	82
A.2. Connecting to the Buffalo Router	82
A.3. Restoring Factory Default Settings.....	84
A.4. Loading Custom Firmware	84
Appendix B: Acronyms / Abbreviations.....	88
Bibliography	91

List of Tables

Table 2-1: Power Requirements for Various Wireless Technologies.....	10
Table 2-2: Comparison of Operating Frequencies and Maximum Data Rates.....	11
Table 2-3: Sizes of Several Mesh Networking Components	13
Table 2-4: Community OLSR Mesh Networks [15].....	19
Table 2-5: Common Open Source Router Firmware	23
Table 3-1: Autonomous Systems Lab Mobile Platform Components	43
Table 3-2: RFC 1918 Private Address Space [32].....	44
Table 3-3: Autonomous Systems Lab Message Wrapping.....	53
Table 3-4: Received Webcam Image Parameters	59
Table 3-5: Mesh Network Entering / Exiting Times	62
Table A-1: Default Buffalo Router Settings	83
Table A-2: Manual Computer LAN Configuration	85

List of Figures

Figure 1-1: DARPA LANdroids Mesh Networking Concept [2].....	5
Figure 2-1: Sample WDS Router Configuration	16
Figure 2-2: Comparison of Several Wireless Standards [27]	24
Figure 2-3: ZigBee Real-Life Applications [27]	26
Figure 2-4: ZigBee Network Topologies [27]	29
Figure 2-5: ZigBee Software Stack [28].....	32
Figure 2-6: Diagram Representing ZigBee Binding [27]	37
Figure 3-1: Autonomous Systems Lab Testing Platform	42
Figure 3-2: Autonomous Systems Lab Network Organization.....	45
Figure 3-3: Four Buffalo WHR-HP-G54 Routers Forming a Freifunk Mesh	46
Figure 3-4: Freifunk OLSR Mesh Network Visualizations.....	47
Figure 3-5: Autonomous Systems Lab Framework Organization	48
Figure 3-6: Autonomous Systems Lab Sample Data Flow.....	49
Figure 3-7: Peer-to-Peer Communication Device GUI (Status and Parameters)	51
Figure 3-8: ASL Framework Basic Chat Algorithm.....	56
Figure 3-9: ASL Framework Transmitting Live Webcam Images.....	57
Figure 3-10: ASL Framework Receiving Live Webcam Images	58
Figure 3-11: Live Webcam Bytes per Image	59
Figure 3-12: Texas Instruments SmartRF®04EB Development Board [34].....	64
Figure 3-13: Texas Instruments Evaluation Module [34].....	64

Figure 3-14: TRX Systems INU ZigBee Circuit	65
Figure 3-15: TRX Systems ZigBee Dongle.....	66
Figure 3-16: Z-Stack API Interface Flow Diagram	73
Figure 3-17: LQI Measurements at Three Power Levels.....	75
Figure 3-18: Smoothed LQI Measurements	76
Figure 4-1: Align T-REX 450 RC Helicopter [38].....	81
Figure A-1: Buffalo Router Configuration Web Interface [30].....	83
Figure A-2: Freifunk Web Interface [24].....	87

Chapter 1: Introduction, Motivation, and Outline

1.1. Introduction

Wireless technologies are becoming increasingly predominant in the world we know today. It is very difficult to find a laptop computer lacking a wireless networking option and the newest Netbooks, designed particularly for the wireless age are becoming increasingly more popular [1]. Nevertheless, there are some features of wireless technology which are not generally used in mainstream America, but can be extremely useful in specialized areas of engineering. This thesis aims to demonstrate some of these features that could be integrated into future wireless technologies. Specifically, this thesis explores the selection, implementation, and evaluation of two mobile mesh networks, each involving a different distributed computing problem. The selection for each network was entirely dependent upon the constraints of each problem. The forthcoming discussion will demonstrate how network constraints affect the optimal choice for network design and implementation provided in this thesis.

The first problem explores the design and implementation of a distributed computing mesh network. This will allow a collection of autonomous land vehicles to gather, process, and exchange information in an unknown environment. This

network was established through the use of modified IEEE 802.11 commercial routers. A software framework was also provided in order to allow application developers to interface seamlessly with the network infrastructure as well as the underlying robot hardware. The peer-to-peer communication device is discussed in detail, as it provides for network monitoring between nodes in addition to managing all socket connections required for data transmission across the network. Furthermore, the distribution of several types of data throughout the network is presented in several sample applications.

The second problem investigates the selection of a possible design and implementation of a distributed mobile mesh network involving the tracking and monitoring of personnel. Both power constraints as well as the requirement that this network be integrated with custom hardware, lead to the design of a low-power ZigBee network. Custom software interfaces were designed to communicate with the ZigBee modules. Moreover, applications involving signal strength measurements between nodes were developed with the intention of improving relative position estimates. By fusing inertial measurements at each node with signal strength information, this improvement is possible.

These mesh network implementations together with their respective application frameworks lay the foundation for further development into distributed computing.

1.2. Motivation

Wireless networks, in general, have become common in the workplace and are currently very affordable for home use. Traditional wireless networks rely on one or more stationary nodes to provide a backbone for mobile nodes within the network. This configuration can best be demonstrated in a corporate environment where many users are assigned laptops with Wireless Network Interface Controllers (WNICs). In this type of wireless network, typically, there is one or more access points placed strategically around the work environment. These access points are hardwired into the corporate network via Ethernet. Users with WNICs generally connect to the closest access point in or around their particular work space. Since access points are hardwired into the corporate network via Ethernet, they are stationary, not mobile. As a result, if a user strays too far away from their closest access point, he will be unable to connect to the corporate network. It is here that the positive implications of wireless mobile mesh networks become quite evident. If a wireless mobile mesh network is deployed in this workplace, then users would be able to utilize connections with one another in order to connect to a corporate network. The only requirement would be that one user of the group be in range of and connected to an access point. Packets of data would be forwarded through all interconnected WNICs and eventually sent to the corporate network wirelessly.

Working in the autonomous robotics laboratory environment for a few years now, it has become apparent that team collaboration and distributed computing are becoming increasingly sought after as primary catalysts that produce an end product.

Traditionally, it is often typical for a team of mobile robots to contain several fundamentally diverse robotic platforms, each with a specific objective in mind. For example, several platforms may be designated as servers, while others are simply drones. In order to maintain full connectivity in a network such as the wireless one mentioned previously, all of the servers must remain online consistently. This necessity raises several concerns: a server platform may become damaged while performing a task, a drone may wander too far away from a server and lose connection with the rest of the robot swarm or a lack of server platforms may not allow the network to achieve the required network topology. One solution for these problems would be to outfit every platform with a set of server hardware. This solution, while it does address the concerns mentioned above, is by no means optimal or even practical. The issue can be tackled by utilizing a wireless mobile mesh network. In this situation, each device on the network would be indistinguishable from one another from a wireless networking standpoint.

Furthermore, the Defense Advanced Research Projects Agency (DARPA) project known as LANdroids, is currently seeking to provide “small, inexpensive and smart robotic radio relay nodes” that can be used to create a wireless mobile mesh network in urban environments [2]. This network would be used to provide a temporary means for soldiers to communicate in areas where no acceptable (nonexistent or unsecure) underlying network exists. The LANdroids project calls for robots that consist of a “radio, robotic platform, battery, and a small processor” to be deployed [2]. They also require that these devices be expendable [2]. A possible network topology as provided by DARPA is provided in Figure 1-1.

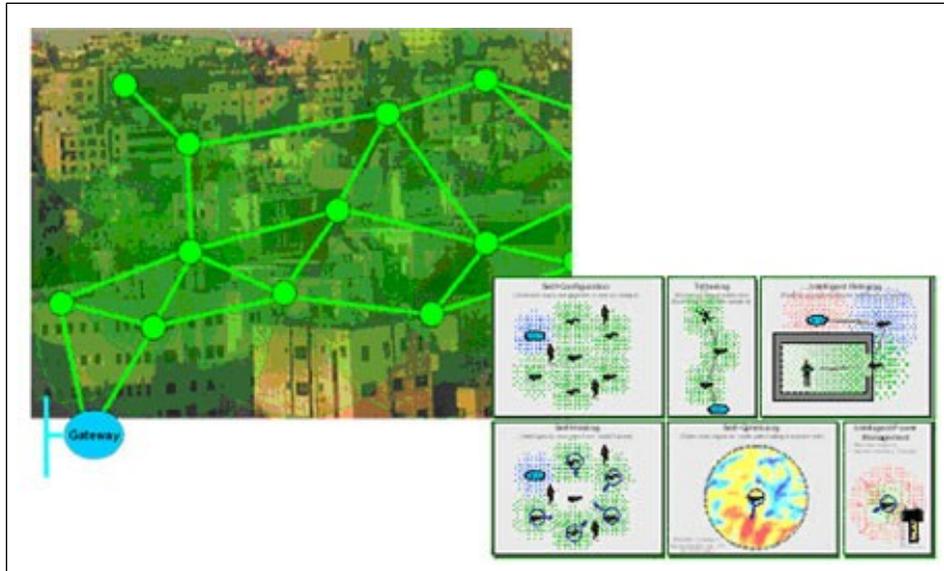


Figure 1-1: DARPA LANDroids Mesh Networking Concept [2]

The topics yet to be discussed in this thesis could be applied directly to the DARPA LANDroids project.

Another area of interest that concerns the use of wireless mobile mesh networking involves the concept of Wireless Body Area Networks (WBANs). WBANs are typically a collection of network nodes that perform different tasks, but share information via a wireless connection. In this scenario, it is often helpful for the devices to be battery-powered, low-bandwidth, and lightweight. Custom hardware is often required when implementing WBANs. These kinds of networks are often depicted being used in the healthcare industry where sensors can be deployed to monitor the vital signs of hospitalized patients [3]. Battery-powered sensors equipped with a short-range, low data rate mesh network, could be placed over the body and report back to an aggregation node on the patient. The aggregation node could then be outfitted with a higher powered network which, in turn, reports vital signs to a

hospital server. This complex and useful solution would lend itself to remote wireless monitoring of all patients in a hospital facility. Additionally, transferring a patient from one room to another would become an easier task for hospital personnel, as the traditional approach to patient care utilizing physical wires would be unnecessary. Patient WBANs could also share information amongst themselves, an example of network connectivity, to make certain that no one is ever disconnected from a hospital's monitoring system. While this configuration might raise several security/privacy concerns, it is helpful to know that many low-power wireless networks provide built-in encryption algorithms. For example, CC2431 ZigBee chips from Texas Instruments provide a 128-bit AES security coprocessor [4]. These lower-power solutions also have applications in security, and even the military [3].

Wireless mobile mesh networking technology provides a viable solution to many concerns in the world today and should be considered to make day-to-day tasks easier. Outlined below are two implementations of wireless mobile mesh networks that provide answers to some current limitations today.

1.3. Chapter Outline

Summarily, the major thoughts presented in this thesis involve the implementation, applications, and overall evaluation of two vastly different wireless mobile mesh networking infrastructures. One network is implemented using commercial consumer-grade routers found in most homes. The other network from Texas Instruments involves IEEE 802.15.4 ZigBee nodes with a customizable networking stack. Each of these mobile mesh networking implementations has

practical advantages and disadvantages with regard to overall execution and usage. Chapter 2 provides a basic background of wireless mobile mesh networking. It also details technologies currently used to create both high-speed and low-speed wireless mobile mesh networks, and addresses the pros and cons of each. Chapter 3 provides a detailed explanation of high-speed and low-speed wireless mobile mesh networks explored in this thesis. This chapter also details the specifications of both networks and provides instructions for implementing these networks. Finally, Chapter 3 details the networking tests performed on each network. The methodologies used to perform the tests along with experimental results are also presented. Chapter 4 offers some suggestions for additional research that could be performed beyond the scope of this thesis.

Chapter 2: Mesh Networking Background

2.1. Introduction

Many fundamentally diverse wireless mesh networking technologies currently exist. These technologies can be modified by adding an additional software routing layer forming a wireless mobile mesh network that eliminates all stationary nodes. This thesis details two mesh networking technologies which when altered, will produce two mobile mesh networks. The first technology involves the use of commercial, off-the-shelf routers, available generally from any consumer electronics store, and/or access points employing Wi-Fi technology. Custom firmware can be loaded onto these routers in order to provide the basis for a wireless mobile mesh network. The second technology which can be modified to provide a wireless mobile mesh network involves the use of ZigBee compliant radios. This technology provides for the simple integration of custom hardware with a low-power wireless mesh network. The remainder of this chapter illustrates the differences between both technologies and also gives a brief history of mesh networking with respect to each independent platform. Also discussed are the overall areas of application for each individual device.

2.2. Design Considerations

When implementing a wireless mesh network, several design considerations must be taken into account. For example, the two mesh networks discussed in this thesis are not necessarily used for the same purposes. Each mesh network has its own specific feature set and strengths. When choosing to adopt a mesh networking technology, it is vital to weigh all possible design considerations against the requirements of the end product. Several key design components for Wi-Fi and ZigBee mesh networks are listed below.

2.2.1. Power

When choosing hardware to create a wireless mesh network, the overall power available to the network interface is probably its most limiting factor. The power that a system can supply severely limits the overall types of mesh networks that can be utilized, especially in terms of the available bandwidth and overall speed of the network. To better illustrate the vast differences in power requirements, Table 2-1 lists the power specifications for several different mesh networking technologies.

	Transmission Current	Supply Voltage
Buffalo WHR-HP-G54 (802.11) [5]	1.1 A (Nominal)	5.0 V
Linksys WRT54G (802.11) [6]	0.5 A	12.0 V
Roving Networks RN-41 (Bluetooth) [7]	35 – 60mA RX 65 – 100mA TX	3.0 – 3.6 V
Texas Instruments CC2431 (ZigBee) [4]	27mA RX 27mA TX	2.0 – 3.6 V

Table 2-1: Power Requirements for Various Wireless Technologies

2.2.2. Bandwidth / Operating Frequency

The overall bandwidth supplied by a wireless mesh network is also a significant factor to consider when implementing a network. This component requires many design specifications such as the specific information to be transmitted and details as to how data will be encoded as it is sent over the wireless network. When the encoding scheme has been selected, it is then possible to encode some sample data and determine a rough estimate of the number of bits required to be transmitted per second. It is then crucial to consider the overhead of the underlying transmission protocol. Furthermore, it is often advisable to include an additional amount of overhead to allow for changes in protocol, encoding schemes or transmission frequencies. Overall, an overhead of approximately 15 – 25% is usually sufficient. Table 2-2 lists some of the available bandwidths for common wireless technologies.

	Operating Frequency	Maximum Data Rate
IEEE 802.11a [8]	5 GHz	54 Mb/s
IEEE 802.11b [8]	2.4 GHz	11 Mb/s
IEEE 802.11g [8]	2.4 GHz	54 Mb/s
IEEE 802.11n [8]	2.4 GHz / 5 GHz	300 Mb/s
IEEE 802.15.1 (Bluetooth) [9]	2.4 GHz	3 Mb/s
IEEE 802.15.4 (ZigBee) [10]	2.4 GHz	250 kb/s
	915 MHz	40 kb/s
	868 MHz	20 kb/s

Table 2-2: Comparison of Operating Frequencies and Maximum Data Rates

Also, consider the operating frequency of the wireless device used to create the mesh network. In order to select an operating frequency, the designer needs to weigh the benefits of choosing a higher operating frequency versus the benefits of having a large overall range. It is well-known that in wireless systems, the overall range and ability of a signal to penetrate walls as well as other obstacles decreases as the operating frequency of the system increases. Conversely, as the operating frequency of a system increases, the maximum possible data rate also generally increases. Therefore, there is an inversely proportionate relationship between the operating frequency and the overall range of a wireless system. Again, the specific requirements of the systems will dictate the final choice.

2.2.3. Interface

One additional constraint to consider when implementing a mobile mesh network is the required interface used to transmit data to a wireless node. Depending upon the end application, several different possibilities exist. Most consumer-grade commercial routers have an Ethernet (IEEE 802.3) connection with which most computers and some microcontrollers can interface. This type of network is usually best suited for applications that involve an onboard computer. For example, networks such as these could be deployed on a swarm of autonomous ground robots. Other common interfaces to wireless radios include some typical communication buses that are used in microcontrollers including UART (Universal Asynchronous Receiver/Transmitter), SPI (Serial Peripheral Interface), I²C (Inter-Integrated Circuit), and CAN (Controller-Area Network). All of these protocols are easily adapted for use with a microcontroller, and many microcontrollers already have hardware implementations for these protocols.

2.2.4. Size

One of the final criteria to consider when selecting a wireless mesh network technology is the overall size of the device. Many of the low-speed wireless mesh networking devices can be easily integrated onto custom printed circuit boards (PCBs). Wi-Fi mesh networks usually involve the use of commercial routers. These routers are typically larger in size. Common sizes of mesh networking components are listed in Table 2-3.

	Length	Width	Height
Buffalo WHR-HP-G54 (802.11) [5]	5.1 in.	5.7 in.	1.1 in.
Linksys WRT54G (802.11) [6]	6.06 in.	7.32 in.	1.89 in.
Roving Networks RN-41 (Bluetooth) [7]	13.4 mm	25.8 mm	2 mm
Texas Instruments CC2431 (ZigBee) [4]	7.0 mm	7.0 mm	0.85 mm

Table 2-3: Sizes of Several Mesh Networking Components

2.3. Wi-Fi Networks

Wi-Fi networks are generally more suited to applications involving personal computers with the ability to connect to Ethernet (IEEE 802.3) networks. These networks are especially useful in situations where large mobile systems are being deployed and there is adequate space for routers, personal computers, and usually large batteries, as power requirements are higher for devices capable of faster data transmission. This section details a few common mesh networking protocols commonly used in Wi-Fi networks.

2.3.1. Wireless Mesh Networking Protocols

Mesh networks allow for data to be transmitted in-between independent nodes or devices, contained in the mesh. Depending upon the configuration and structure of the mesh, data may flow directly from a transmitting node to a receiving node, or packetized data may be relayed through an interconnected series of nodes based upon

the underlying mesh networking protocol specifications. When extending mesh networking with the advent of wireless technology, it is often necessary to decentralize the wireless network. Essentially, as wireless devices form a mobile mesh network, it is necessary for each device that joins the network to be capable of forwarding data to other nodes without instruction from a central device. This wireless mobile mesh network, with each device capable of communicating with other devices independently (performing the roles of a router in a traditional network), and the overall ability of the network to reconfigure its topology as necessary, is sometimes referred to as a mobile ad hoc network (MANET). This section outlines several protocols and configuration methods that allow multiple routers to form a mesh network devoid of an Ethernet backbone. The OLSR protocol is most significant as it is utilized later in this paper.

2.3.1.1. WDS Configuration

The Wireless Distribution System (WDS), while not an ad hoc mesh networking protocol by definition, does provide a method to connect multiple routers to one another without a wired backbone between neighboring devices. This system is implemented in many common routers and also becomes available when many open source firmware distributions are installed such as OpenWRT and DD-WRT [11]. This type of system is typically used in an environment where it is not possible to link two access points via Ethernet, and also where an extended range is required. For example, in a multilevel building where only one access point is available on the first floor, it is possible that due to signal degradation, clients on the third and fourth

floors will no longer be able to communicate with the access point on the first floor. To remedy this situation, an Ethernet cable could be installed connecting the first floor to floors above. Then, an access point could be placed wherever wireless clients experience poor signal quality. In the situation where an Ethernet cable is unable to be connected between the first floor and floors above, the WDS methodology could be used to connect clients to the network wirelessly. It is important to note that all WDS enabled access points will still be able to allow wireless clients to connect to the network as do traditional access points [12].

When constructing a WDS network, all routers and/or access points should be configured to utilize the same encryption keys (WEP, WPA, or WPA2) as well as the same radio channel. Different Service Set Identifiers (SSIDs) may also be used, but for the initial configuration, it is usually easiest to assign all of the WDS devices the same SSID. The WDS configuration acts as both a wireless bridge and a wireless access point, once it has been organized correctly. Connections are made between all associated devices using the MAC addresses of each individual device rather than IP addresses, which in certain circumstances is preferred over other routing methods (since the MAC addresses are preserved in all packets) [11].

While the WDS configuration has its advantages in its easy setup requiring minimal configuration, there are two major disadvantages in using a WDS network. First of all, WDS does not support rotated encryption keys [11]. This limitation could create issues for larger corporations that use these advanced encryption technologies. Most likely, the home user will not be affected by this limitation due to the fact that

many sub \$100.00 routers only support static pre-shared keys supported by most WDS configurations. Secondly, WDS routers also suffer from a significant decrease in overall bandwidth availability between clients. This bandwidth available is actually halved when transmitted through a WDS node [11]. Consider the configuration displayed in Figure 2-1.

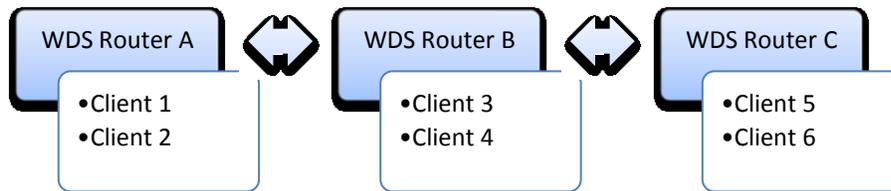


Figure 2-1: Sample WDS Router Configuration

If Client 1 establishes a connection with Client 3, then packets must be routed once through the WDS, from WDS Router A to WDS Router B. Assuming the ideal bandwidth for an IEEE 802.11g of 54 Mb/s, the maximum possible bandwidth between Client 1 and Client 3 is halved, so that only 27 Mb/s is possible. Moreover, if Client 1 establishes a connection with Client 5, then two hops through the WDS are required causing the overall bandwidth between Client 1 and Client 5 to drop to a maximum of 13.5 Mb/s. If more complex meshes are formed, this limitation can cause the network to screech to a halt. Because of this limitation, WDS is not commonly implemented in environments where more than two access points are required, and/or bandwidth requirements are strict.

2.3.1.2. AODV Protocol

The Ad hoc On-Demand Distance Vector (AODV) Routing protocol is yet another routing system that has been developed specifically for use on MANETs [13]. This protocol has been specified by the IETF as RFC 3561; however, it is not specified as a standard, rather it is listed as an experimental protocol [14]. This protocol offers “quick adaptation to dynamic link conditions, low processing and memory overhead, low network utilization, and determines unicast routes to destinations within the ad hoc network” [14]. Additionally, it allows for nodes not in constant communication to ignore routes between each other [14].

The AODV protocol differs from many of the other routing protocols currently used today especially, in that it is a reactive routing protocol as opposed to a proactive routing protocol [13]. This means that when nodes are not communicating with one another, the network will not have any traffic whatsoever. Whereas proactive routing protocols are constantly sending packets of data throughout the entire network for link discovery, the AODV routing protocol waits until a link needs to be discovered in order to transmit a packet of data to the required destination [13]. When a packet of data needs to be sent through the network, a broadcast message is transmitted to all routers that support the AODV protocol. They continue to forward the message until one AODV node responds with a route that is able to reach the destination node. This process continues until the route with the fewest number of hops is selected. The message will then travel from the sender to the destination using this route [13]. Over time, routes will become invalid, as the AODV protocol

determines the best overall route on-demand. Like any protocol, there are advantages and disadvantages to the AODV design. First and foremost, this protocol is applauded for its minimal delay in setup. Since routes are determined on demand, large computations involving the mapping of the entire network are avoided [13]. On the other hand, since routes are determined on demand and are not updated consistently, it is more likely that AODV will suffer from inconsistent routes when intermediate nodes change [13]. Also, the surge of messages required to determine routes on demand consumes bandwidth when bandwidth is required to actually send a message. This can lead to delays and unnecessary bandwidth usage [13].

2.3.1.3. OLSR Protocol

The Optimized Link State Routing (OLSR) protocol is an IP based protocol optimized for MANETs. It is currently in the development stages and new versions are being released to the public on a regular basis. OLSR, together with AODV, are two internet standards commonly used in mesh networks [15]. OLSR has also been tested in large scale environments and several large meshes are currently in use. Table 2-4 details some of the community network infrastructures currently utilizing OLSR.

Network	# Nodes
Athens Wireless Network	~2000
Berlin Freifunk Network	~600
Leipzig Freifunk Network	~600
FunkFeuer.at	~400

Table 2-4: Community OLSR Mesh Networks [15]

The OLSR protocol is an ongoing open source project released under a BSD license, and is currently available for many software platforms [15].

The OLSR protocol is known as a proactive link-state routing protocol that transmits packets of data consistently in order to discover link information [16]. These messages are referred to as Hello and Topology Control (TC) messages [16]. This flooding of messages allows for devices contained in the mesh to determine the overall network topology and build a table to utilize when routing packets between nodes [15]. OLSR triumphs over other network discovery protocols in that it relies on Multipoint Relays (MPRs). MPRs are nodes that are selected to broadcast messages when the network topology is being determined [17]. This process cuts down on the overall overhead inundated when every single node in a network broadcasts messages concerning network topology [17]. Utilizing MPRs as a means of optimization, the OLSR protocol is able to provide optimal routes (shortest number of hops) [17].

As with any protocol, OLSR has been criticized for certain design choices and applauded for others. Most noticeably, OLSR was noted for lacking the ability to determine overall link quality between nodes in its original specification. This limitation has been addressed in more recent OLSRd releases which provide a method for determining link quality [16]. OLSR has also been criticized because it is a proactive protocol and therefore maintains a routing table of all possible routes in a network when not all routes are utilized. This design creates network overhead that is avoided in reactive mesh networks [16]. Overall, OLSR mesh networks are actively maintained and this protocol is one that can be easily adapted for mobile mesh networking.

2.3.1.4. B.A.T.M.A.N. Routing Protocol

The Better Approach To Mobile Ad hoc Networking (B.A.T.M.A.N.) is yet another routing protocol for multi-hop mesh networks [18]. This custom routing protocol is currently under development by the Freifunk Community and has only recently emerged in late 2008 [18,19]. The most recent release, version 0.3.1, was announced on January 25, 2009 [18]. The B.A.T.M.A.N. routing protocol addresses common issues encountered with standard routing protocols when they are applied to a MANET; especially, in a situation where nodes are commonly entering and exiting the network causing the topology and overall routing scheme to change frequently [20].

OLSR and other link-state algorithms attempt to recalculate the entire connection topology routinely in order to maintain full connection with all nodes in

the network. Unfortunately, this is a processor intensive process and as meshes extend to contain several hundred nodes, this process has been known to take several seconds with embedded platforms such as consumer grade commercial routers [20]. OLSR has been adopted for use in several areas, and due to this limitation, the protocol has had to undergo multiple revisions to deal with scaling issues [20]. It is this major issue that the B.A.T.M.A.N. routing protocol aims to alleviate.

The B.A.T.M.A.N. routing protocol provides a more distributed approach to the routing dilemma that is introduced as more and more nodes enter the mesh. Rather than each node becoming aware of the entire topology of the mesh, nodes are only concerned with their next hop [20]. This methodology allows for each node to have knowledge of how packets are routed locally in the network. No individual node has global knowledge of the path that packets will take across the mesh [20]. Furthermore, the entire mesh is not inundated with messages concerning topology as with other proactive link-state routing protocols. This advantage allows the B.A.T.M.A.N. routing algorithm to be well suited for networks that contain one or more unreliable links [20].

A simplistic overview of the B.A.T.M.A.N. routing protocol as provided by [20] is detailed below:

1. Each node transmits Originator Messages (OGMs) in order to inform neighboring devices of its existence [20].

2. The neighbors retransmit the OGM to their neighbors according to routing regulations [20].
 - a. This causes a flood of OGM messages throughout the network; however, the messages are kept short and are approximately 52 bytes including IP and UDP overhead [20].
3. As messages are received, a routing table is built based upon the B.A.T.M.A.N. algorithm which selects the neighbor that provides the next best hop to the message originator [20].

Overall, the B.A.T.M.A.N. routing algorithm provides each node contained in the mesh with an individual routing table. This table determines which neighbor should be routed a packet in order to expedite the delivery of the message to the required destination.

2.3.2. Open Source Firmware

Most commercial routers purchased in consumer electronics stores are shipped with upgradable firmware that can be reloaded as manufacturers correct bugs and flaws. Several open source firmware solutions currently exist and are becoming increasingly more popular as they provide additional features at no cost to the end user. These open source firmware solutions are supported on many commercial routers, and most noticeably, on the Linksys WRT54G line of wireless routers. The Linksys WRT54G router supports 802.11 b/g wireless networks and has provided its open source firmware [21]. Variants of the Linksys source code and additional open source solutions are currently available for this router as well as many other

commercial routers. A few notable open source firmware solutions are listed in Table 2-5:

Firmware Name	Website	Current Version
DD-WRT [22]	http://www.dd-wrt.com/	24 SP1
OpenWRT [23]	http://openwrt.org/	Kamikaze 8.09
Freifunk [24]	http://www.freifunk.net/	1.6.25
Tomato [25]	http://polarcloud.com/tomato	1.23

Table 2-5: Common Open Source Router Firmware

The open source firmware packages listed above have active communities and well documented feature sets. Each firmware distribution has a specific feature set and various advantages. A description as to how to load the Freifunk firmware onto a Buffalo WHR-HP-G54 can be found on page 82.

2.4. ZigBee Networks

The ZigBee specification has been explicitly designed for small, low-power custom hardware implementations in wireless personal area networks (WPANs). The ZigBee specification also targets devices that require “a low data rate, long battery life, and secure networking” (using AES encryption) [10]. ZigBee has been particularly well suited for use in home entertainment and control, home awareness, mobile services, commercial buildings, and industrial plants [10]. The ZigBee specification is made available through the efforts of many companies known as the

ZigBee Alliance [26]. The remainder of this section provides insight as to a ZigBee network’s general features and implementation.

2.4.1. Comparison with other Wireless Standards

Because of all the various wireless technologies currently in the market, it is often helpful to visualize how these technologies differ from one another. The most predominant wireless technologies are currently ZigBee (IEEE 802.15.4), Bluetooth (IEEE 802.15.1), and Wi-Fi (IEEE 802.11). Figure 2-2 effectively illustrates how these devices differ in terms of range, data rate, and overall purpose or use.

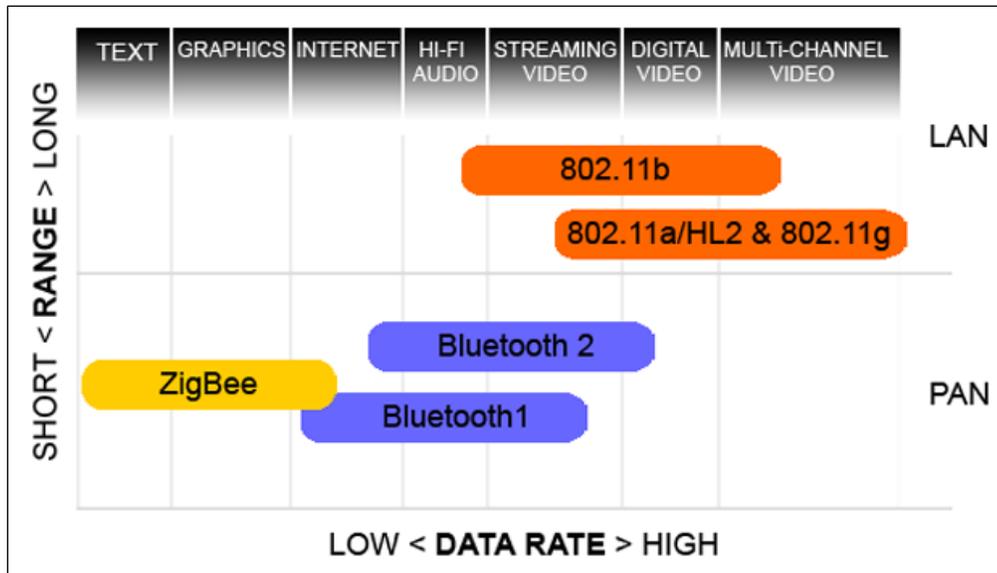


Figure 2-2: Comparison of Several Wireless Standards [27]

From the chart’s illustration, it is quite easy to observe that ZigBee has been designed for short range operations that make use of a Personal Area Network (PAN). ZigBee is most comparable to Bluetooth which currently has two versions. Bluetooth operates on the same principle of a PAN, yet Bluetooth has been designed for data

rates up to approximately 3 Mb/s which is approximately 12 times faster than ZigBee (when ZigBee is utilized at its maximum data rate). IEEE 802.11 wireless technologies are designed to be of a much higher speed and have much higher data rates at the expense of higher power requirements.

2.4.2. Applications of ZigBee Networks

The ZigBee standard lends itself well to industry applications. ZigBee networks are very common in building automation, industrial environments, and in control and monitoring applications for both medical and residential needs [27]. The overall applications are somewhat limitless as the world is becoming more and more dependent upon wireless technologies for everyday life. Figure 2-3 details many current devices that have implemented ZigBee as their chosen wireless medium.



Figure 2-3: ZigBee Real-Life Applications [27]

ZigBee also employs the use of application profiles, so that Original Equipment Manufacturers (OEMs) can create devices that are able to interoperate. Application profiles currently available include [10]:

- Home Automation
- ZigBee Smart Energy
- Telecommunication Applications
- Personal Home and Hospital Care

The ZigBee Alliance publishes these profiles [26]. For commercial uses, these profiles are only distributed to members of the ZigBee Alliance. However, the

specifications are provided free of charge to the general public for non-commercial purposes [10].

2.4.3. Organization of a ZigBee Network

ZigBee networks are organized in two ways. First, each ZigBee device alone can be classified as a ZigBee Coordinator (ZC), ZigBee Router (ZR), or a ZigBee End Device (ZED). Each of these various device types has a different set of features, ZC having all of the features, followed by the ZR having a subset of ZC's features, and lastly, ZED, having a subset of ZR's features. As each of the device types are assigned, all of the ZigBee nodes can form a mesh. The overall topology of the created mesh characterizes a second means by which ZigBee networks are organized. A more detailed description of the ZigBee network organization is provided below.

2.4.3.1. ZigBee Coordinator (ZC)

The ZigBee Coordinator device contains the most features. It is also referred to as the MAC Network Coordinator [27]. These devices form the root of a ZigBee network and are responsible for starting the network. They also allow other devices to connect to or join the network. Additionally, they select the frequency channel on which the ZigBee network will operate. The frequency selected is usually the one with the least amount of activity or energy in its channel [28]. These devices require the most amount of memory and power [27]. The coordinator is also capable of bridging between separate ZigBee networks [10]. Each ZigBee network can have only one coordinator, no matter the overall topology of the network [28]. Often, it is

important to keep the coordinator online, especially when certain network features are enabled. Because of this issue, it is usually expected that the coordinator will not be battery-powered. Lastly, the coordinator acts as the “Trust Centre and repository for security keys” [10].

2.4.3.2. ZigBee Router (ZR)

The ZigBee Router device is the second most powerful device in a ZigBee network. It contains a subset of features with respect to the coordinator. It is able to operate as a router, as well as perform an application function [10]. Routers are responsible for passing data from one device to many other devices. These devices are considered to be MAC Full Function Devices, meaning that they have all functionality as specified by the ZigBee standard [27]. Their overall role in the network is to relay messages from node to node and allow child nodes to connect to the network [28]. It is also important to note that routers are not allowed to enter a sleep or lower power mode [28]. As with coordinator devices, the routers are also expected not to be battery-powered, as they must remain online at all times.

2.4.3.3. ZigBee End Device (ZED)

The last type of ZigBee device to discuss is the ZigBee End Device (ZED). This device gets its name because it forms the ends or edges of the overall network. End devices are only used for sending and receiving messages with their parent device [28]. An end device is unable to relay messages from one node to another, and it cannot allow another ZigBee device to join the network [28]. These devices are

also commonly referred to as MAC Reduced Function Devices in that they have been limited in overall functionality in order to keep cost and complexity of the devices down [27]. However, it is still possible for these devices to be MAC Full Function Devices, if necessary [27]. End devices are commonly battery-powered and are not required to be continually online. These devices often enter a sleep or lower power mode when they are not transmitting or receiving messages in order to conserve and elongate battery life [28].

2.4.3.4. ZigBee Network Topologies

ZigBee networks can be classified into three key categories in terms of overall topology. These topologies refer to the relative location of coordinator, router, and end device nodes. They determine all of the possible routes used to send data from one node to another. A diagram of the three topologies is provided in Figure 2-4.

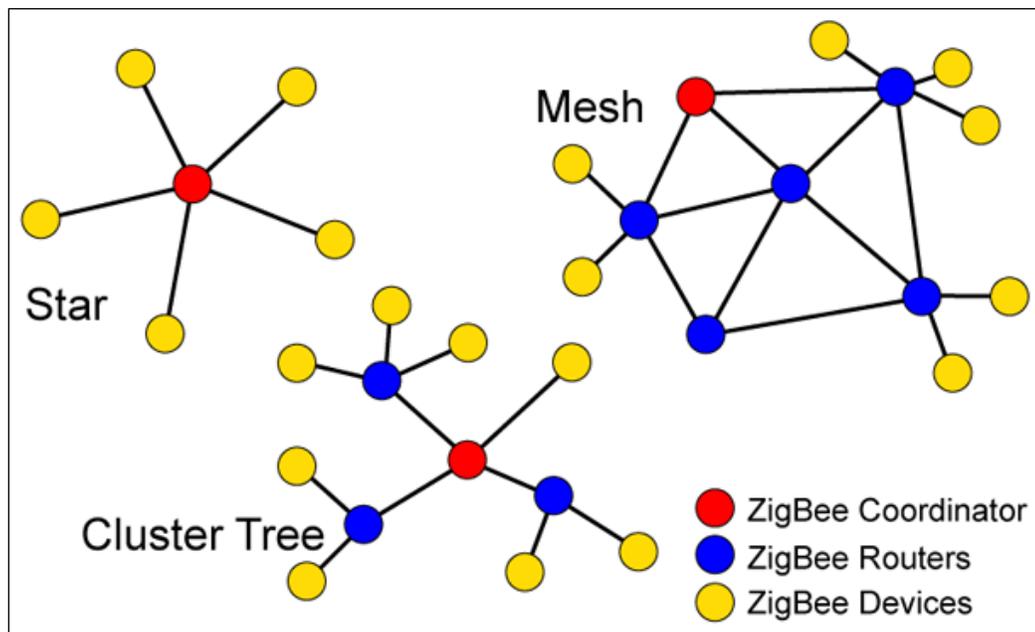


Figure 2-4: ZigBee Network Topologies [27]

The first type of network topology is the star network. In this network, the coordinator forms the central point within the network. Then, end devices and routers connect to the coordinator in order to share information. All packets are forwarded through the coordinator and sent to the correct end device. In this configuration, it is also possible for routers to be on the edge of the network since they can function as end devices as long as no end devices join the network via these routers. The routing functionality provided by the router devices is simply ignored in this type of network. The star configuration is the only configuration where only a maximum of two transmissions are required in order for a packet of data to arrive at any node in the network. However, there is one fundamental flaw in this topology. If the RF link between the coordinator and an end device fails, there is no alternate path for packets to take [28]. This flaw can cause packets to be dropped when using this network topology.

The second type of network topology is the cluster or tree style network. In this network, the coordinator forms the root of the tree. The tree continues to branch out to router nodes and end device nodes, known as the children of the root node. When traversing the tree from the root node (coordinator), every time a router node is reached, the tree can branch into additional sub-nodes at that level. Nodes below any node are known as child nodes and nodes above other nodes are known as parent nodes. When reaching an end device, that branch of the tree stops at that end device. In this topology, several rules exist as to the overall routing of packets. Child nodes can only communicate with their parent node [28]. Parent nodes can only communicate with their parent node (if there is one), and their children [28]. Lastly,

when transmitting from one node to another, packets travel up the tree until a common ancestor between both nodes is reached. This ancestor could be the root node if two entirely different branches are communicating with one another. Once the packet reaches the common ancestor between both nodes, it will then travel down the tree to the destination node [28]. As with the star topology, if one of the necessary RF links fails, there is no alternative route for packets to take [28].

The last and most versatile type of network topology is the mesh network topology. This topology is a combination of both the star and cluster or tree topologies. It, like the star and cluster/tree topologies begins with the coordinator node. From the coordinator node, many routers and end devices can join the coordinator's network. These devices are referred to as the coordinator's children. Each router can then continue to be connected to more routers and end devices as in the cluster/tree topology. The main difference between this topology and the previously mentioned is that any routers in range of one another can create links between themselves [28]. Unlike the star and cluster/tree networks, there are usually alternative routes that can be found in order to route data on the network. These alternative routes come into play when an RF link fails or if there is congestion on the RF link [28]. The ZigBee specification provides for a "route discovery" feature in order to provide the best and most efficient route for packets to travel [28]. It is also significant to mention that all routing is handled by the ZigBee software stack and therefore, a level of abstraction has been provided between application developers and the underlying RF technologies [28].

2.4.4. ZigBee Stack

ZigBee has been designed to be easily integrated using small and inexpensive microprocessors and microcontrollers. Many manufacturers are currently providing chips that include both an IEEE 802.15.4 radio as well as a microcontroller to handle the ZigBee software stack and user applications. These single chips implement all three layers of the ZigBee architecture simplifying the process for new developers to enter the ZigBee market. The ZigBee architecture is implemented in three core layers that specify the overall operation of a ZigBee device. These layers include the Physical/Data Link Level, the ZigBee Stack Level, as well as the Application Level. “Each layer performs a specific set of services for the layer above” [27]. Figure 2-5 provides a diagram depicting the ZigBee architecture.

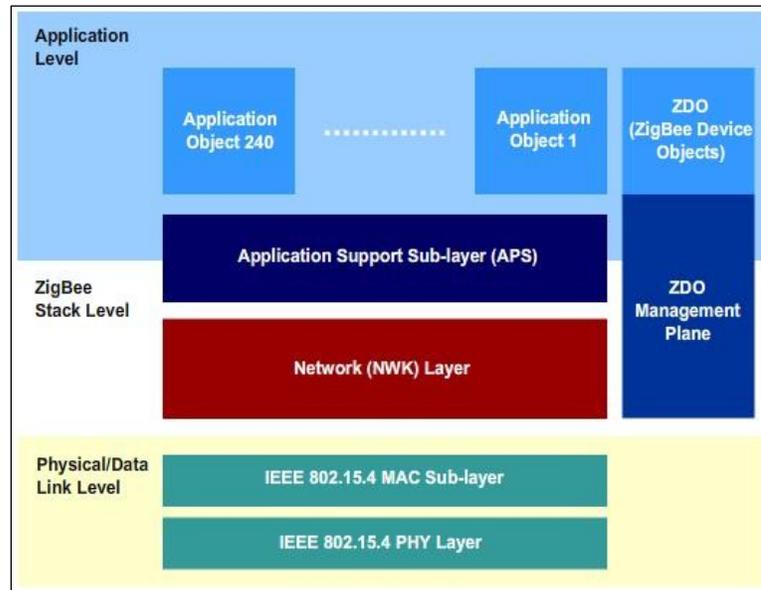


Figure 2-5: ZigBee Software Stack [28]

2.4.4.1. Physical/Data Link Level

The lowest level of the ZigBee Software Stack is referred to as the Physical/Data Link Level. This level is defined by the IEEE 802.15.4 standard which defines the Physical (PHY) Layer that specifies the RF link necessary to connect ZigBee devices wirelessly. This layer specifies the interface to the radio which is used to transmit data between nodes [28]. Additionally in this level, the IEEE 802.15.4 MAC sub-layer is defined which provides data access to the physical layer [27]. The MAC sub-layer is also responsible for the overall addressing of data packets as well as the assembling and disassembling of packets in order to determine their destination and origin [28].

2.4.4.2. ZigBee Stack Level

The next layer above is vital to the entire concept of ZigBee, in general. This layer is referred to as the ZigBee Stack Level which is responsible for the overall ZigBee functionality and for providing a common interface between applications and the Physical/Data Link Level [28]. This layer consists of the Network (NWK) layer, the Application Support (APS) sub-layer, and the ZigBee Device Object (ZDO) Management Plane. The NWK level provides for the ability to join and leave networks, supplies security for data packets, as well as determines the route that packets should take [27]. Furthermore, the NWK level is responsible for starting the network, assigning network addresses, and storing the routing table information for the overall network [28].

The ZDO Management Plane also exists at the ZigBee Stack Level. It is responsible for allowing the ZDO to communicate with the APS sub-layer and the NWK layer when completing internal jobs [28]. The ZDO Management Plane also performs security features and provides network access to the ZDO [28].

The APS sub-layer is also part of the ZigBee Stack Level. This layer crosses the boundary between the Application Level and the ZigBee Stack Level. As messages are sent throughout the ZigBee network, they are marked with endpoints. It is the responsibility of the APS sub-layer to relay incoming and outgoing messages to any of the 240 possible applications running on the device based on the relevant endpoint [28]. The APS sub-layer is also responsible for maintaining binding tables and forwarding messages between bound nodes [27]. The ZigBee Stack provides the ability for end users to bind two devices to one another after they have left the manufacturer. For example, one ZigBee device might implement a light switch and one could implement an actual light. The end user could bind the two devices together so that a specific light switch controls a specific light.

2.4.4.3. Application Level

The highest level in the ZigBee architecture is the Application Level. In this level the manufacturer defines all of the applications that can run on a single ZigBee node. “Up to 240 application objects are supported on a single ZigBee node” [28]. These applications are assigned endpoint numbers from 1 to 240, while several other endpoint identifiers are used for other routing functions [28]. Additionally, the ZDO section exists at the Application Level. This section defines specifically the type of

node that a particular device can be (coordinator, router, or end device) [28]. This section is also responsible for all device specific functions including both initialization and communication. Furthermore, the ZDO is responsible for discovering new devices and determining which applications they implement as well as performing binding and secure links between devices [27].

2.4.5. Node Addressing

In the ZigBee standard, all devices must have a unique identifier. This identifier is provided through the use of a unique MAC address. Identification is similar to the procedure currently used for all wireless and wired network cards. All of these devices are assigned a unique 64-bit address that is allocated by IEEE [28]. In ZigBee documentation, this unique identifier is sometimes referred to as the MAC address, the IEEE address, or the extended address of the device.

In addition to the 64-bit MAC addresses that are unique to all ZigBee devices, there is also a 16-bit network address that is assigned to each device as it enters the network. This address is only local to a specific ZigBee network. Because of this procedure, two different ZigBee networks can assign the same network address to two different nodes [28]. This is an important distinction because of the way in which ZigBee nodes address one another since many ZigBee functions only provide addressing using the local network address. Therefore, predetermining devices that will communicate with one another can be more difficult. Actually, when transmitting messages across the ZigBee network, the network address and endpoint number are used to specify where the message should be routed [27]. Network

addresses are assigned by the device's parent node which can be either a coordinator or a router as devices join the network [28]. In ZigBee documentation, it is very common for the network address to be referred to as the short address [28].

2.4.5.1. Binding and Binding Table

The ZigBee specification has provided for the ability of nodes to create a persistent connection between one another as necessary. Devices that have interoperable applications can choose to bind to one another, so that they can continue to communicate until the binding is removed. A binding is essentially a “point-to-point logical link” [27]. Binding exists in four main types: one-to-one, one-to-many, many-to-one, and many-to-many bindings [28]. These types can easily be illustrated using two separate ZigBee devices. One device is a light and the other, a switch. In one-to-one binding, a single light switch will be configured to control a single light. This configuration can be observed in Figure 2-6 where switch 2 has been bound to lamp 4 through an entry in the binding table. In one-to-many bindings, a single switch can be used to control several lights. This is displayed in Figure 2-6 where switch 1 is bound to lamps 1, 2, and 3. In many-to-one bindings, several switches can be used to control a single light. Lastly, in many-to-many bindings, it is possible for many switches to control an array of lamps.

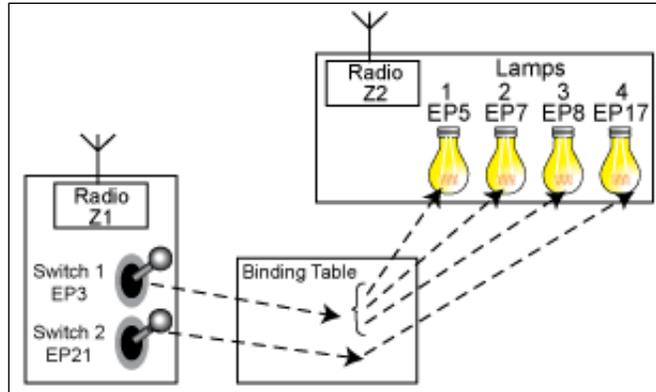


Figure 2-6: Diagram Representing ZigBee Binding [27]

2.4.6. Network Gateways

ZigBee network gateways provide the glue between a ZigBee network and existing hardware, infrastructure, and/or software. The network gateway device is responsible for interfacing an existing ZigBee network into a preexisting external system [27]. These gateway devices allow for existing devices to intercommunicate with a ZigBee network. Furthermore, network gateways provide a simple method to allow ZigBee devices to communicate with an existing IP network [27].

Chapter 3: Network Selection and Implementation

3.1. Introduction

In this thesis, two primarily different mobile mesh networking technologies have been selected for exploration. Depending upon the particular requirements of the end-user, various networking technologies may be required. Every application in need of a wireless mobile mesh network will require that network, specifically, be selected to address the overall design considerations of the end-user. The design considerations provided in the beginning of Chapter 2 assist in explaining many of the features that a wireless network can employ as well as the different constraints these elements can impose on a design.

3.2. Wi-Fi Networks

The first network selected for implementation involves a standard IP network. In this case, a network was required for robot intercommunication at the Autonomous Systems Lab at the University of Maryland. The Autonomous Systems Lab consists of several graduate students, undergraduate students, and faculty/industry advisors who are developing mobile robotic platforms for research purposes. They handle several projects involving potential swarms of ground vehicles as well as autonomous helicopters. The overall construction and refinement of these ground vehicles developed by several students in the lab required a network infrastructure that would

allow the ground vehicles to exchange information and sensor data gathered over the course of an expedition. It was expected that the robots would be deployed in many different buildings, some with and some without existing wireless networking infrastructures. Therefore, utilizing the existing network infrastructure of a building was not an option. Additionally, the Autonomous Systems Lab is interested in developing systems that are distributed; therefore, having a server that would handle all traffic was also not an option. This situation was ideal for the implementation of a wireless mobile mesh network. The following section explains the overall design considerations taken into account in order to build a wireless mobile mesh network using commercial routers. Furthermore, a description of the overall platform used in the testing of this thesis is detailed below.

3.2.1. Router Selection

The first step in developing a wireless mobile mesh network utilizing commercial off-the-shelf routers is to determine which wireless router would function best, overall. For added simplicity as well as the ability to purchase new routers as necessary, commercial off-the-shelf products were selected rather than specialized custom hardware. By choosing a commercial product, upgrades to the system become easier as new firmware patches are released.

Routers in the sub \$100.00 range were evaluated in terms of the features that they could provide for the mobile wireless mesh network. The Linksys WRT54G router was used as a starting point, as it is one of the most common commercial routers and many users have customized them in the past. Upon further examination

of the Linksys WRT54G, it was determined that the available RAM and flash memory of the newer revision routers was minimal. This issue would cause many limitations in functionality that could be provided by a custom firmware distribution [22]. The DD-WRT community has provided many suggestions as to routers that are compatible with their custom firmware. The Buffalo WHR-G125 is mentioned as the “cheapest 100% compatible wireless router” with their firmware distribution [29]. Unfortunately, this router does not have a removable antenna, so future antenna upgrades and mounting possibilities were limited. Since Buffalo Technology [30] was suggested as a compatible solution, other routers by Buffalo Technology were evaluated as well. The Buffalo WHR-HP-G54 was finally selected because of its removable high gain antenna, Broadcom chipset, and available RAM and flash memory [29]. Furthermore, this router is compatible with many major open source firmware solutions; therefore, the testing of multiple configurations would only require changing the router firmware.

3.2.2. Firmware Selection

Several firmware solutions were examined for implementation in a wireless mobile mesh network of Wi-Fi routers. The DD-WRT [22], OpenWRT [23], and Freifunk [24], router firmware solutions were the major contenders for implementation in the Autonomous Systems Lab’s mobile wireless mesh network. DD-WRT and OpenWRT were initially examined because of their support for the Wireless Distribution System (WDS). WDS was initially thought to be an ideal solution for interconnected devices since it was designed to allow access points to be

connected to one another without the need for a wired connection between individual access points [11]. This system was tested, but due to speed issues with packet transmission, overall difficulty in creating a network of multiple routers, and the delay for detecting nodes entering and exiting the network, it was not selected. Furthermore, due to the fact that as packets are forwarded through WDS nodes, the total bandwidth of the network is halved and that WDS has not been certified by the Wi-Fi Alliance, it did not possess all of the qualities required for a reliable mobile wireless mesh network [11].

In the end, the Freifunk firmware was selected for use in implementing a wireless mobile mesh network for the Autonomous Systems Lab. Freifunk is a custom firmware solution that has been built on OpenWRT and strives to provide open source free radio networks in Germany [31]. Freifunk has been customized to work on Linksys WRT54G routers; however, it supports most routers with an adequate amount of flash memory and RAM in addition to a compatible chipset. Freifunk provides a simple platform for building a fully functional mobile mesh network utilizing ad hoc WLAN communication and the OLSR routing protocol [31].

3.2.3. Testing Platform

The Wi-Fi wireless mobile mesh network demonstrated in this thesis was specifically designed for use on eight identical ground robots designed and built by students in the Autonomous Systems Lab. These robots have been constructed with the intention to perform collaborative tasks both indoors and outdoors. Each robot has the ability to function independently, but the wireless mobile mesh network

created by the combination of Buffalo routers and Freifunk firmware allows the robots to exchange large amounts of data in real time. One possible application involves the exchange of interesting data which could include camera images, sensor data (ultrasonic sonar distance measurements, inertial measurements, encoder readings), as well as positional information (possibly from a GPS module). Figure 3-1 depicts one of the fully assembled autonomous ground vehicles employed in the Autonomous Systems Lab.



Figure 3-1: Autonomous Systems Lab Testing Platform

3.2.3.1. Components

The robots in the Autonomous Systems Lab are primarily constructed from a combination of off-the-shelf components tied together with a custom software framework, custom PCBs and hardware for data acquisition, as well as some custom machined parts. The combination of parts is listed in Table 3-1.

Quantity	Component
1	Tamiya TXT-1 Monster Truck 4WD
1	Set of Custom Platform and Brackets
1	Open miniPC (Windows XP Professional)
1	Buffalo WHR-HP-G54 Wireless-G Router
1	CarNetix CNX-1900 DC-DC Regulator
1	Novak Goat Crawler ESC
1	Parallax Servo Controller
1	ASL Sensor Board V1
2	Logitech QuickCam Pro 9000 Webcam
8	Parallax Ping))) Ultrasonic Sensor
2	11.1V 6600mAh Li-Ion Battery Pack
2	8.4V 4200mAh Battery Pack

Table 3-1: Autonomous Systems Lab Mobile Platform Components

All components combined provide for a platform capable of performing tasks autonomously for approximately 2.5 – 3.5 hours on a single charge.

3.2.4. Network Organization

The overall network organization, including the policy for assigning IP addresses to each device in a wireless mobile mesh network is crucial in creating a fully functional interconnected group of computers. The mobile mesh network configuration implemented in this thesis requires that all computers and routers have unique IP addresses. Additionally, these IP addresses should be in the range specified as private network addresses (sometimes referred to as non-routable addresses) since they are not routable on the internet (only inside an organization's private internet). The Internet Engineering Task Force (IETF) has developed a standard for IPv4 private networks as published in RFC 1918 [32]. The addresses available for private internets are listed in Table 3-2.

RFC 1918 Name	Address Range	Prefix
24-bit block	10.0.0.0 – 10.255.255.255	10/8
20-bit block	172.16.0.0 – 172.31.255.255	172.16/12
16-bit block	192.168.0.0 – 192.168.255.255	192.168/16

Table 3-2: RFC 1918 Private Address Space [32]

For maximum flexibility in assigning private IP addresses to each individual router and computer in the network, the 24-bit block described in RFC 1918 was selected as the range to use for assigning IP addresses to each device. The Meraka Institute has created a guide for installing a Freifunk based network in rural areas in Africa [33]. This guide provides the basis for the initial formation of a Freifunk mesh and was utilized as a starting point for network formation in this paper. Each router was

labeled with a unique ID number (1 – 255), designated ID. With this ID number, the WLAN and LAN IP addresses of the router were assigned in the following manner:

- WLAN IP: 10.1.1.ID
- LAN IP: 10.2.ID.1

Then, each router was configured to lease DHCP addresses starting at 10.2.ID.200 with a maximum of 50 leases. This configuration allows for computers to connect to any router in the wireless mesh network via Ethernet, and receive an acceptable IP address, primarily for debugging purposes. Additionally, the primary computer on each robot was manually configured to utilize the IP address 10.2.ID.100. This organization provides for 255 unique router nodes in the mesh network which was more than adequate for this application. A diagram depicting the overall hierarchy of the mesh network is shown in Figure 3-2.

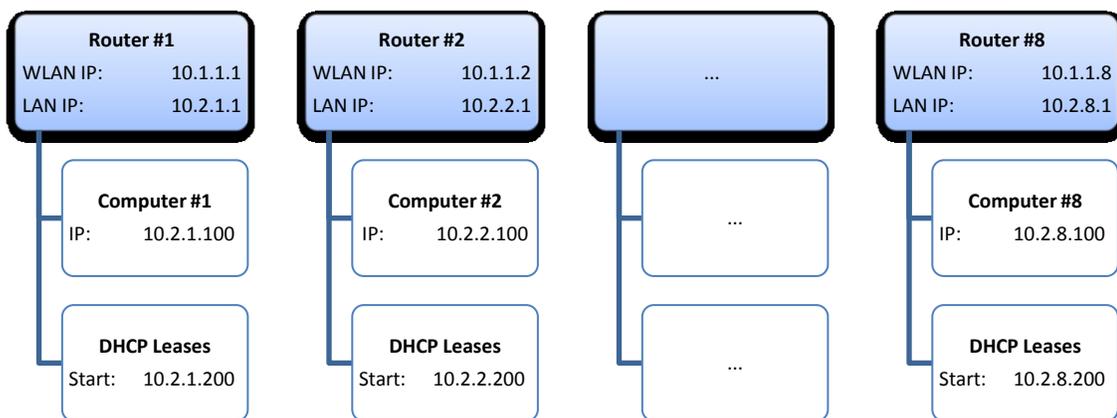


Figure 3-2: Autonomous Systems Lab Network Organization

3.2.5. Routing / Mesh Network Formation

The added benefit of utilizing a custom firmware distribution such as Freifunk is that the routing of packets between nodes and computers on the network is accomplished within each individual router. The Freifunk firmware provides an additional routing layer that connects all network nodes. If routers are within range, they will transmit packets directly to each other; otherwise, packets are routed through intermediate routers until they reach the desired destination. This routing layer allows for applications utilizing the underlying mesh networking framework to create software that ignores specific packet routes throughout the network. Figure 3-3 depicts four Buffalo WHR-HP-G54 routers that have formed a Freifunk OLSR mesh network.



Figure 3-3: Four Buffalo WHR-HP-G54 Routers Forming a Freifunk Mesh

The overall structure of the OLSR mesh network created utilizing the Freifunk firmware can be viewed by installing the freifunk-olsr-viz-en package. Two Freifunk mesh networks, one with four nodes and one with five nodes are depicted in Figure 3-4. In each of these networks, all nodes maintained full connectivity with one another.

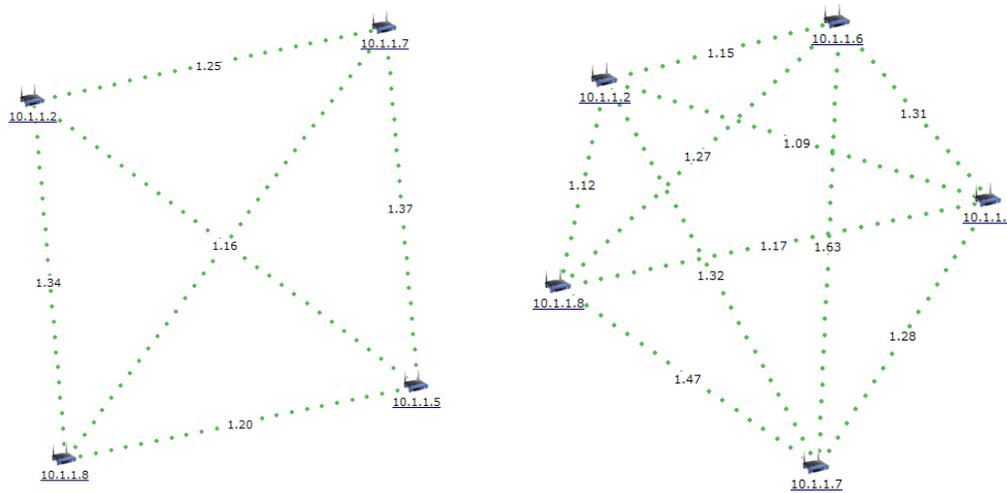


Figure 3-4: Freifunk OLSR Mesh Network Visualizations

3.2.6. Network Utilization

The Autonomous Systems Lab has developed a framework capable of executing particular algorithms on a variety of different platforms. This framework has been developed using Visual C#, an object-oriented programming language developed by Microsoft. The ASL Framework has been designed to decouple sensor acquisition and actuator control with a specific algorithm linking the two. This framework allows for ease in algorithm development since communication with external peripherals such as servos, webcams, and GPS modules is handled entirely by the framework. Data is received from and sent to devices through a simplistic

interface. As new devices are added to the framework, they become available to all existing algorithms currently present in the system. As a result, additional interfacing software is not required for communication. Furthermore, communication devices are also integrated within the ASL Framework. They keep track of any and all computers running the ASL Framework and also provide simple methods for transmitting data from and to one another.

3.2.7. ASL Framework Structure

The ASL Framework provides a hierarchical structure for application development. Every application that utilizes the ASL Framework consists of one or more algorithms as well as a platform. The platform specification consists of all sensor, actuator, and communication devices currently deployed on the testing platform. This specification could include one or more webcams, a servo controller, and/or the peer-to-peer communication network manager. Algorithms, on the other hand, consist of multiple interconnected processing components defined as functional units. A diagram illustrating the ASL Framework organization is provided in Figure 3-5.

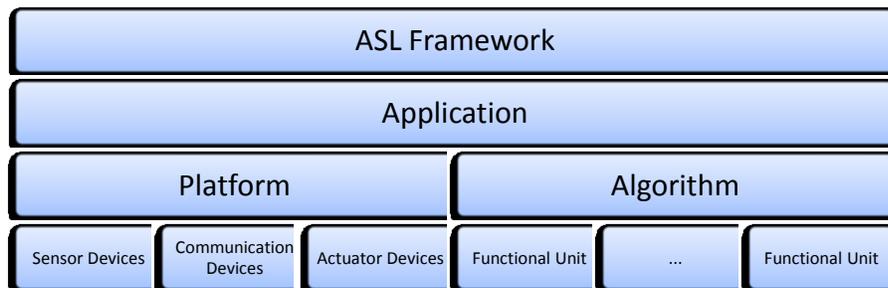


Figure 3-5: Autonomous Systems Lab Framework Organization

As sensor data is collected from devices on the testing platform, events are raised and all functional units that subscribe to these particular events are notified that new data has become available. This process continues until the algorithm chooses to control an actuator and/or communicate with another device. At this point, the algorithm will communicate with its associated platform which will, in turn, utilize the ASL Framework to communicate with the required device. A very basic platform may contain simply one webcam and the peer-to-peer communication device. A simplistic algorithm is described below. As images are captured by the webcam, a filter may be applied to the incoming image, and the application developer may choose to transmit the image to another computer on the network. The algorithm and platform association is illustrated in Figure 3-6.

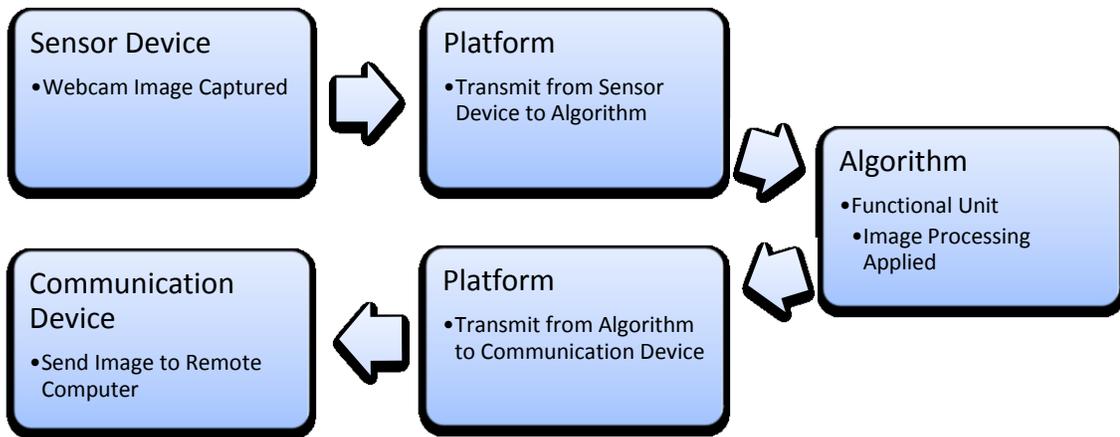


Figure 3-6: Autonomous Systems Lab Sample Data Flow

3.2.8. Peer-to-Peer Communication Device Specifications

The peer-to-peer communication device within the ASL Framework is responsible for handling all currently connected computers utilizing the ASL

Framework. Only platforms that have peer-to-peer networking features enabled will be monitored. The peer-to-peer communication devices serve four distinct purposes:

1. Monitors all devices currently connected to the network and determines if they are active.
2. Provides simple public methods for communication with other devices within the network.
3. Maintains a dictionary detailing the process for serializing and deserializing classes in order to transmit and receive data across the network. Dictionary synchronization between all connected computers is also provided by the peer-to-peer communication device.
4. Manages all network sockets required for communication purposes.

The peer-to-peer communication device is configured and monitored using the Graphical User Interface (GUI), automatically loaded when initializing the ASL Framework. The GUI allows for the specification of a list of devices and their associated IP addresses. These parameters can then be saved for future use. As soon as all devices have been configured and the ASL Framework has been loaded on each device, the status tab will display the current activity of devices in the network. Figure 3-7 provides a screenshot of the peer-to-peer communication device's status and parameters.

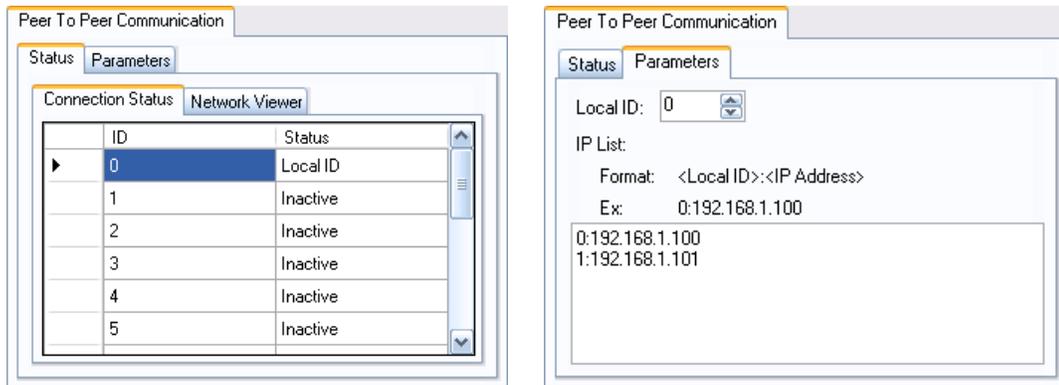


Figure 3-7: Peer-to-Peer Communication Device GUI (Status and Parameters)

The peer-to-peer communication device maintains the status of all devices connected to the network. At a selectable interval, the peer-to-peer communication device will transmit a ping message to all devices that are contained within the peer-to-peer communication device’s parameters. The structure of the ping message, together with all other message types is provided on page 52. It also provides a public method that allows messages to be sent to any other device utilizing the ASL Framework. This method takes the following form:

```
public void CommunicationToOthers(int id, IBinarifiable data)
```

To transmit a message, the addressee’s ID needs to be specified along with a C# data class implementing the `IBinarifiable` interface described on page 54. The peer-to-peer communication device monitors up to 255 different classes that implement the `IBinarifiable` interface and indexes these classes in a dictionary. When initializing the ASL Framework, a dictionary exchange takes place as messages are sent from one device to another. After synchronization, all devices are updated with

the most recent dictionary. This dictionary is then utilized to serialize and deserialize all `IBinarifiable` classes transmitted through the ASL Framework. Lastly, the peer-to-peer communication device manages all network sockets essential for communication. UDP sockets are utilized in place of TCP sockets because of the need for real-time communication with robotic systems. It is more important to receive packets in the correct order than it is to receive every single packet in a transmission. TCP packets run the risk of arriving out of order thereby causing delays if packets are dropped in transmission. Delays are undesirable in a robotic network; however, occasionally dropped packets are an acceptable reality. In the end, it is more advantageous to utilize UDP packets over TCP packets since they are not retransmitted if dropped from the network.

3.2.8.1. Message Wrapping

All messages transmitted through the ASL Framework are wrapped within a small header. This header helps to identify the message's origin, prevent duplicate messages, as well as provide the means necessary to decode a message transmitted from one device to another. The header consists of seven bytes as detailed in Table 3-3.

Data	Type	Number of Bytes
To Address	byte	8
From Address	byte	8
Message Number	ushort	16
Message Type	byte	8
Message Length	ushort	16
Message	bytes	variable

Table 3-3: Autonomous Systems Lab Message Wrapping

As messages are received, they are indexed by the sender and receiver addresses (as specified in the peer-to-peer communication device configuration). The next piece of information noted in the message header is the message number. When a MANET with an underlying routing layer such as OLSR is not available, it may be necessary for the software layer to handle the routing of packets as they are received. To prevent the possibility of duplicate messages which could drastically affect algorithms maintaining state information, a message number is included in the message header. Following the message number is the message type. This byte specifies the method used to decode the message following the header. The ASL Framework specifies an interface in C# known as `IBinarifiable` which provides the message type. This interface is discussed more thoroughly on page 54. All classes that implement this interface are able to encode and decode themselves into a sequence of bytes that can be transmitted across the network.

3.2.8.2. *IBinarifiable Interface*

The `IBinarifiable` interface provided by the ASL Framework is paramount for transmitting messages from one device to another. This interface allows for data classes to be serialized and deserialized to and from a sequence of bytes. This interface is provided below:

```
public interface IBinarifiable
{
    byte[] ToSerialData();
    void FromSerialData(byte[] dataArray);
}
```

The interface allows for data classes to specify explicitly how to serialize and deserialize themselves for transmission across the network. This serialization, since it is explicitly defined, will allow an application developer to verify that the minimum number of bytes is being transmitted across the network. Furthermore, it is possible to implement compression in the `IBinarifiable` methods in order to further minimize the total bandwidth required to transmit a data class.

3.3. Wi-Fi Network Tests

Several Wi-Fi mobile mesh network tests were performed to examine the overall functionality and operation of this network on land robots requiring autonomous control. More specifically, testing was completed in order to determine the overall effectiveness of the mobile mesh network when implemented on a swarm of ground vehicles. Three individual tests were of particular interest. In the first test, simple text-based transmissions were performed along with pings of all devices in the

network. While transmitting simple text messages may not be very labor intensive alone, it is important to note that all future tests follow the same general design principles – only the amount of data and possible uses change. Secondly, the transmission of live webcam images throughout the mesh network, while recording the number of bytes transmitted, is also of interest. Lastly, monitoring how nodes enter and exit the mesh as well as the timing involved in doing so, is a significant point for discussion.

3.3.1. Text-Based Transmission using the ASL Framework

In order to begin testing the ASL Framework peer-to-peer communication device and its overall implementation with an underlying Freifunk OLSR mesh network, it was determined that an ASCII text-based communication algorithm would provide adequate initial results. This algorithm, built upon the ASL Framework, is simply a streamlined instant messaging system that can be utilized on a local intranet and/or mesh network. Each computer or device within the network is identified in a table of configuration parameters, and is also assigned a unique ID. As each device initializes the ASL Framework, the peer-to-peer communication device opens UDP sockets and begins probing the network for active computers using the ASL Framework. As computers are detected, their status changes from Inactive to Active, and the outgoing UDP socket to the computer is enabled. Each computer in the ASL Framework is capable of sending messages to every node within the network. A screenshot of this framework utilizing the Basic Chat algorithm is provided in Figure 3-8. As illustrated, communication is taking place between computers designated as

ID 0 and 1, with the GUI of ID 0 shown. Status of the connection to ID 1 is provided as Active with the most recent ping time of 2ms.

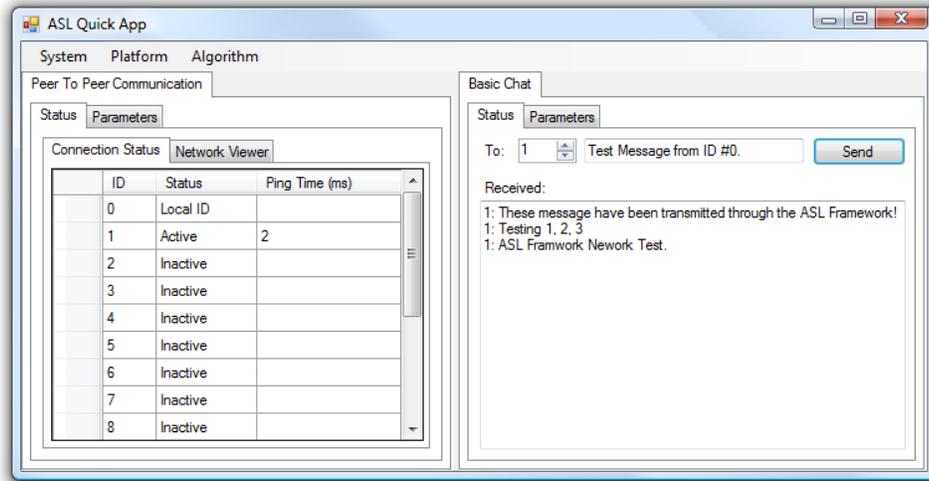


Figure 3-8: ASL Framework Basic Chat Algorithm

3.3.2. Live Webcam Image Feed using the ASL Framework

The second significant test of data transmission across the Wi-Fi mesh network involves the transmission of live webcam images from one node in the mesh to another receiving node. A data class named `Image` has been created in order to transmit live webcam images through the ASL Framework. As necessary, this class implements the `IBinarifiable` interface, and therefore contains methods to convert the image to and from an array of bytes. For image transmission it was decided to implement basic compression in order to conserve on the overall bandwidth utilized when transmitting images across the mesh. JPEG compression with the default parameters in C# was utilized to compress the image before it was transmitted. Figure 3-9 depicts the ASL Framework receiving webcam images from

an attached webcam at approximately 21.33 Hz, and transmitting each of these images to another node in the network with ID 1.

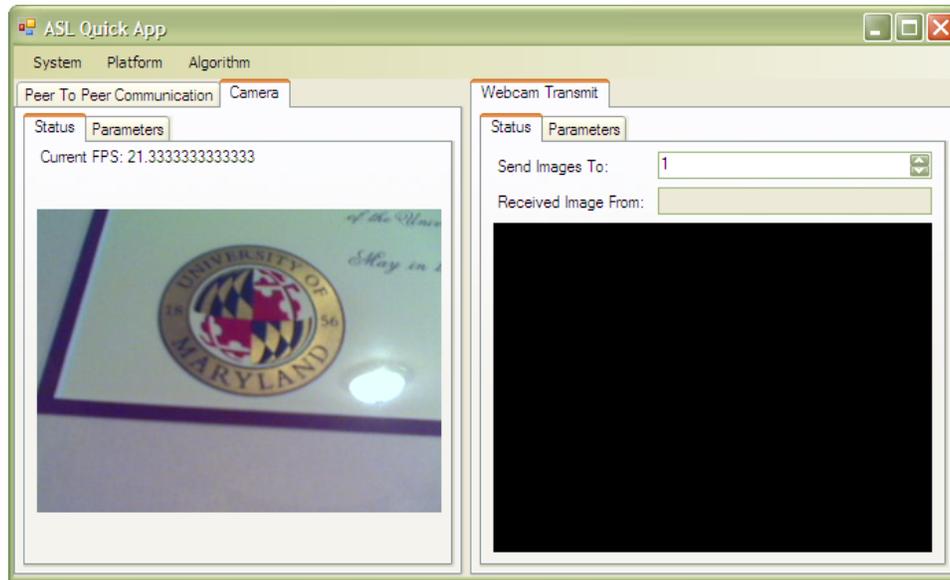


Figure 3-9: ASL Framework Transmitting Live Webcam Images

As the images are received from the transmitting nodes, the `IBinarifiable` data is converted back into its respective data class. As each image is received, an event is triggered, causing the Webcam Transmit algorithm to display the received image in the right panel of the GUI as displayed in Figure 3-10. Additionally, the node that the image was received from is noted along with the total number of bytes contained in the transmission.

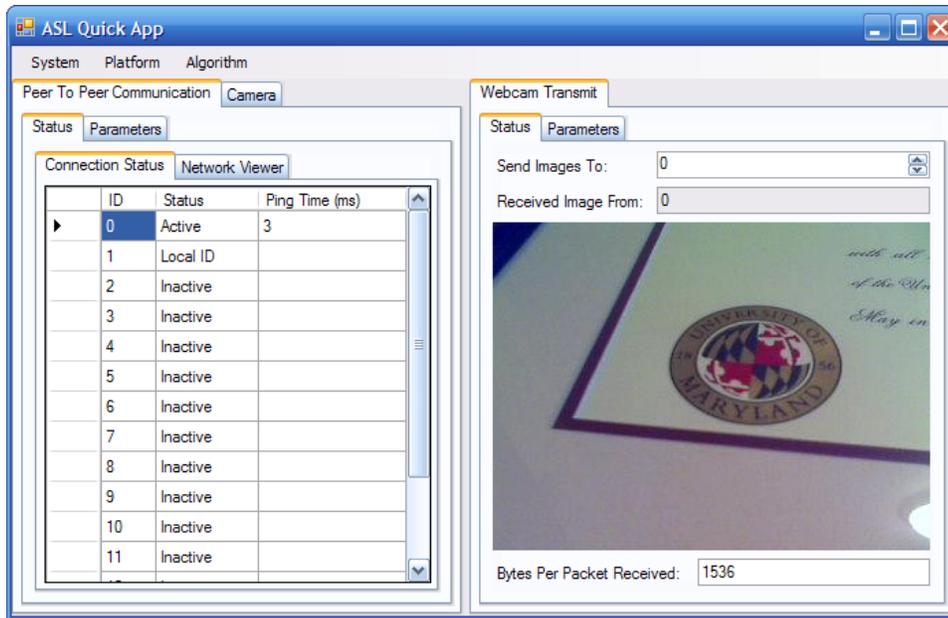


Figure 3-10: ASL Framework Receiving Live Webcam Images

Furthermore, it was interesting to note the total amount of bandwidth consumed as images were transmitted across the mesh network. As images were received, the total amount of bytes received in each `IBinarifiable` packet of data was recorded for 334 images. It is important to note that during the transmission of images, the webcam was intentionally shifted in order to simulate movement of the camera and/or the environment in a real world situation, and to take into account the varying JPEG compression. A plot of the number of bytes per image received is displayed in Figure 3-11.

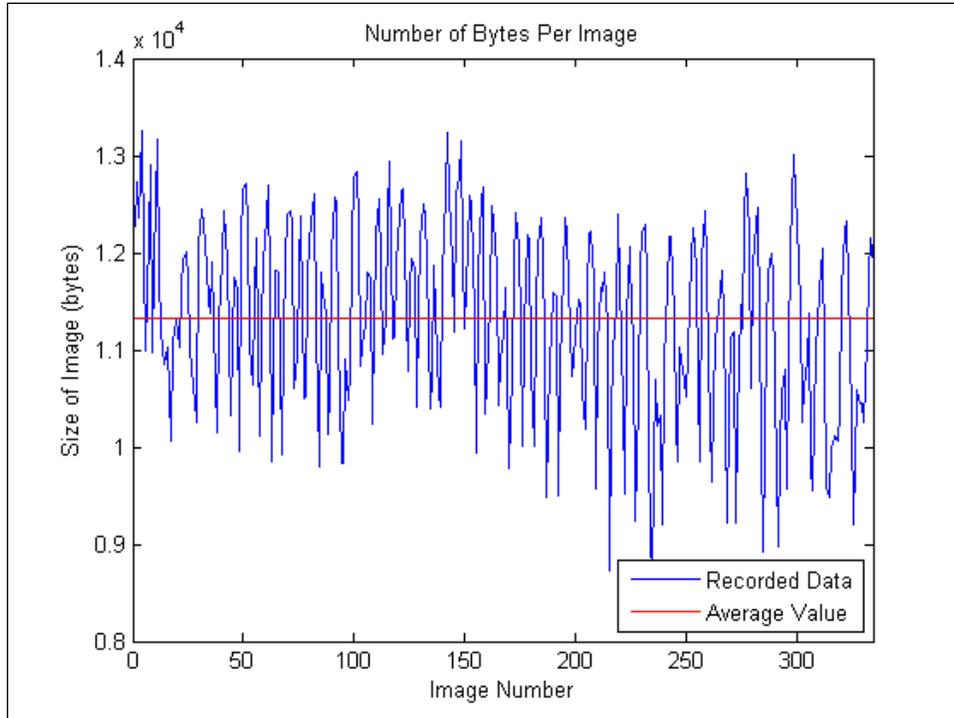


Figure 3-11: Live Webcam Bytes per Image

The number of bytes received per image varies significantly as each image is captured. Specifically, the total number of bytes varies from 8,518 to 13,253 bytes per image with an average of approximately 11,324.64 bytes per image. The statistics related to this test are provided in Table 3-4.

Image Parameters	Value
Minimum Bytes Per Image	8518.00
Maximum Bytes Per Image	13253.00
Average Bytes Per Image	11324.64

Table 3-4: Received Webcam Image Parameters

With this data, it is possible to approximate the amount of bandwidth that will be required to transmit webcam images at a full frame rate of 30 FPS. Based on the

average number of bytes transmitted, it is reasonable to estimate that approximately 340,000 bytes per second are required to be transmitted per webcam image. This is only a fraction of the total theoretical maximum of 54 Mb/s available within the OLSR mesh network. A single live webcam image would only consume approximately 2.72 Mb/s. Of course, as more devices are added to the network in addition to additional webcams per device, the amount of data to transmit live images will continue to increase.

3.3.3. Entering and Exiting the Network

As routers enter and exit the mesh network it is imperative to verify the validity of network connections before attempting to send large amounts of data. The peer-to-peer communication device described on page 49 monitors the activity of connections in the network. By default, a ping packet is sent to predefined devices in the configuration settings every second. As packets are returned, the ping time is recorded and updated on the peer-to-peer communication status GUI.

As packets travel across the network, it is possible for some to be lost since UDP sockets have been selected for communication. Due to this design choice, it was decided that the status of a remote device would not be set to inactive until five consecutive ping packets were not returned. Therefore, with default settings, the peer-to-peer communication device will wait approximately five seconds before a connection is labeled inactive.

In terms of devices entering the OLSR mesh network created with the Freifunk firmware, it is important to note two particular joining intervals. The first interval details the amount of time necessary for a device to communicate with its router via Ethernet when the router is initiated with a cold boot. In this situation, the router is disconnected from its power source completely, while the attached computer remains online. The router is then started and the amount of time before an attached computer is able to successfully ping the router is recorded. This time varies between 20 and 30 seconds. Secondly, it is important to note the amount of time required before a device connected to one router would be able to communicate with another router joining the mesh. In this situation one computer is left connected to a router that has been cold started, and is now able to accept connections from other devices in the mesh network. Another router, completely powered down, is then cold started, and the time interval between the cold start of the second router and the first ping received from the second router is recorded. In testing, this time interval varies between approximately 50 and 70 seconds. Overall, this provides an estimate of the amount of time required for a router (that has finished the boot process and is available locally) to become detected and become a part of the OLSR mesh network. From the two previous measurements, a practical estimate of the amount of time required for a remote computer to become part of the mesh network is between approximately 20 and 50 seconds. A summary of the network entering and exiting times is provided in Table 3-5.

Network Test	Time
Ping Back Not Received	~ 1 – 2 seconds
Device Set to Inactive	~ 5 – 7 seconds
Local Router Cold Boot Available Locally	~ 20 – 30 seconds
Remote Router Cold Boot Available Remotely	~ 50 – 70 seconds
Remote Router Mesh Network Detection	~ 20 – 50 seconds

Table 3-5: Mesh Network Entering / Exiting Times

3.4. ZigBee Networks

The second wireless mesh networking technology evaluated in this thesis involves a network of ZigBee devices. ZigBee was designed specifically to be used in mesh networking environments. Devices that implement the ZigBee standard are increasingly becoming more and more affordable for custom implementations and are approaching approximately \$1.00 per chip [10]. These chips have been designed especially for devices that have strict power requirements and do not require high data rates. The following section describes the ZigBee device selection, as well as the testing platform and overall network organization used with the ZigBee modules.

3.4.1. Device Selection

Several major companies are currently producing ZigBee compliant devices. Some of these devices contain only an IEEE 802.15.4 compliant ZigBee radio, while many are starting to bundle both an IEEE 802.15.4 radio together with a microcontroller. This combination of devices allows for a complete ZigBee product to be implemented with only one chip and a few discrete components. Additionally,

many of these microcontrollers and IEEE 802.15.4 radio combination devices can run a ZigBee software stack implementation directly on the device without the need for an external microcontroller. In this thesis, the Texas Instruments CC2431 module [4] was selected especially for this purpose. This chip is capable of running Texas Instrument's Z-Stack which implements the ZigBee protocol. Additionally, Texas Instruments provides an Application Programming Interface (API) that can be used to communicate with the CC2431 microcontroller via a compatible bus such as SPI or UART.

3.4.2. Testing Platforms

The CC2431 System-on-Chip solution provided by Texas Instruments was tested initially utilizing the CC2431 Development Kit [4]. This kit is comprised of two SmartRF®04EB development boards, two CC2431EM evaluation modules, two 2.4 GHz antennas, as well as required assorted cables. The development board and evaluation module are pictured in Figure 3-12 and Figure 3-13.



Figure 3-12: Texas Instruments SmartRF@04EB Development Board [34]



Figure 3-13: Texas Instruments Evaluation Module [34]

The SmartRF@04EB development board is utilized to program each evaluation module. It provides the necessary USB connection to the computer in order to load new firmware onto the CC2431 chips. Firmware for these devices was developed utilizing the IAR EW8051 C-compiler along with Texas Instruments' Z-Stack (version 1.4.3), a ZigBee compliant protocol stack for Texas Instruments devices.

Due to the limited amount of flash memory and RAM available on the CC2431 chip (128 kb and 8 kb, respectively) when the Z-Stack is utilized, it was decided to pair this ZigBee compliant device with an additional microcontroller [4]. Fortunately, Texas Instruments provides a simple API for communication with the CC2431 module's Z-Stack through a UART connection.

3.4.3. TRX Systems Custom PCBs

TRX Systems, Inc. [35] currently utilizes several of the CC2431 modules on many of their custom devices. The CC2431 module lends itself quite well to custom applications and two individual designs are illustrated in Figure 3-14 and Figure 3-15.



Figure 3-14: TRX Systems INU ZigBee Circuit

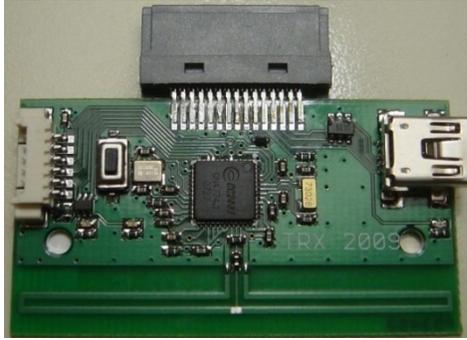


Figure 3-15: TRX Systems ZigBee Dongle

3.4.4. Microcontroller Pairing

As mentioned previously, the Texas Instruments CC2431 module was paired with an additional microcontroller for the ZigBee network implementation. This pairing allows for the ZigBee implementation to be completely abstracted from the overall application in an end product. The microcontroller selected for pairing with the CC2431 chip was Microchip's dsPIC33FJ256GP710. This microcontroller is one of Microchip's most versatile 3.3V Digital Signal Controllers (DSCs). It contains an adequate amount of onboard RAM and flash memory capable of utilizing many standard C libraries. Furthermore, it provides two UART ports, either of which is capable of communicating with the Texas Instruments' CC2431 module. Several C source files were developed for communication with the CC2431 module using the Z-Stack API. These source files are described on page 66.

3.4.4.1. Z-Stack API

Texas Instruments provides an API to interface with their Z-Stack ZigBee protocol implementation. It allows for Texas Instruments' Z-Tool software to communicate with any chip running the Z-Stack (with Z-Tool communication

enabled). The Z-Tool software allows an external device to control the underlying Z-Stack, enable and disable features of the ZigBee modules, reset the device if necessary, and query the module for information, as necessary. Texas Instruments provides detailed descriptions of all commands and callback functions available in this API, and it is through this interface that the Microchip DSC communicates with the ZigBee module. Several source files have been developed to interface seamlessly with the Z-Stack provided on the Texas Instruments' CC2431 chip. These files are written purely in ANSI C in order to make certain that all software is completely portable to any future device. The following six files provide the software basis that allows many different devices to communicate with the Z-Stack API:

- ztool.c
- ztool.h
- ztool_print.c
- ztool_print.h
- ztool_addressList.c
- ztool_addressList.h

The files ztool.c and ztool.h provide for the transmission of all commands contained in the Z-Stack Simple API. They also provide a mechanism for parsing, processing, and notifying the end user's application that a new message has arrived from the Z-Stack. It is important to note that the Z-Stack API sends and receives all messages formatted in big-endian. Unfortunately, x86 architectures along with Microchip's microcontroller architectures expect bytes to be formatted in little-

endian. Therefore, a conversion needs to take place as packets of data are transmitted to and from the Texas Instruments' ZigBee module. The files, `ztool.c` and `ztool.h` handle the endianness conversion between these two devices, and due to this point, they should only be used on devices that support little-endian.

The files, `ztool_print.c` and `ztool_print.h` provide an easy method to visualize data as it is received from the ZigBee module. When new data becomes available, and is verified as a valid message, `ztool.c` invokes a callback function (that the user's application is required to implement). For debugging and verification purposes, the `ztool_print.c` file provides the following function:

```
void ztool_print(FILE *f, uint16_t type, void *data);
```

This function allows an application developer to easily print a received message from the Z-Stack to the terminal or to a file. For example, if the response from a `SYS_GET_EXTENDED_ADDRESS` message is received and prints to `stderr`, the following would be displayed:

```
SYS_GET_EXTENDED_ADDRESS_RESPONSE
ExtendedAddress: 01 23 45 67 89 AB CD EF
```

This display informs the user that the current extended address (IEEE address) stored in the device is `01:23:45:67:89:AB:CD:EF`. Formatted output has been provided for all common responses and callbacks from the Z-Stack. These two files, while not required for standard operation, provide a simplistic display that helps to demonstrate the overall status of the device.

The files, `ztool_addressList.c` and `ztool_addressList.h` provide a method to store and retrieve pairs of ZigBee addresses. As described on page 35, ZigBee devices have two different addresses referred to as the extended address and the short address. The extended addresses are unique among all ZigBee devices, yet the short addresses are unique only within a particular ZigBee network. Rather than using the ZigBee device binding feature, it was decided to have the attached microcontroller manage all device pairings. This decision allows the compiled firmware running on all CC2431 modules to be the same, and lets the device simply perform the functions of a radio interface. Unfortunately, the Z-Stack API does not provide a method to send a message to a particular extended address in the network. Rather, it provides the following command:

```
void zb_SendDataRequest (uint16 destination, uint16 commandId,  
                        uint8 len, uint8 *pData, uint8 handle,  
                        uint8 ack, uint8 radius)
```

Only the short address can be specified. Therefore, it is necessary to query the network and determine pairs of short addresses and extended addresses. The `ztool_addressList.c` and `ztool_addressList.h` files are responsible for maintaining these associations. When the network is established, the attached microcontroller is in charge of querying the ZigBee module and determining the short address of each device with respect to its particular extended address. This allows for pairings to be maintained by the attached microcontroller.

Lastly, one final file is required to implement the interface with the Z-Stack API. Following naming conventions, it is helpful to name this file, `ztool_interface.c`;

however, it could be given any name. It is necessary for each device utilizing this interface to implement two specific functions in this file. First of all, the callback function invoked when a message arrives from the Z-Stack should be implemented and registered in this file. Additionally, the following function must be implemented with device specific UART commands:

```
void ztool_send_cmd(uint16_t cmd, uint8_t len, uint8_t *data)
```

This function is invoked whenever a packet of data needs to be sent to the ZigBee module. A pointer to a formatted array of bytes is provided, and it is necessary to send the following sequence to the ZigBee module:

1. Start of Packet (SOP) byte. Specified to be 0x02 as per the Z-Stack API.
2. Command ID. Most significant byte should be sent first.
3. Message length byte.
4. Message packet.
5. Frame Check Sequence (FCS). This value is computed as the XOR of the command ID bytes, length byte, and data bytes.

A sample implementation of this function for a Microchip DSC is provided below.

```
void ztool_send_cmd(uint16_t cmd, uint8_t len, uint8_t *data)
{
    uint8_t i = 0;

    uint8_t sop = 0x02;
    uint8_t cmd_h = (cmd >> 8) & 0xFF;
    uint8_t cmd_l = cmd & 0xFF;
    uint8_t fcs = ztool_calc_fcs(cmd, len, data);
```

```

while (BusyUART1());

WriteUART1(sop);           while (BusyUART1());
WriteUART1(cmd_h);        while (BusyUART1());
WriteUART1(cmd_l);        while (BusyUART1());
WriteUART1(len);          while (BusyUART1());

for (i = 0; i < len; i++)
{
    WriteUART1(data[i]);
    while (BusyUART1());
}

WriteUART1(fcs);          while (BusyUART1());
}

```

3.4.5. Network Implementation

To implement a mobile ZigBee mesh network, all ZigBee nodes are configured to utilize the same PAN ID. This configuration makes certain that all nodes are able to communicate with one another since they will always be part of the same network when reset. Additionally, the devices are set up to save their configuration parameters into non-volatile (NV) memory. This procedure allows the device to restart using the same parameters as initially configured. As the network is started, new devices join the network, and it is possible, at this point, to attempt communication between devices.

In order to communicate with devices based on a stored unique address (IEEE address), it is initially necessary to build a table relating extended addresses to network assigned short addresses. This process involves querying the Z-Stack through API commands provided in ztool.c. Moreover, an immediate reply is not guaranteed in response to this API command; rather, a callback is received when the command succeeds or eventually times out and fails. The overall search procedure

for building the ZigBee address table is provided in Figure 3-16. It is important to note that this ZigBee implementation has been designed for use with applications that plan to stream data over the mesh network. In the address lookup procedure, messages will be dropped until a short address resolution succeeds. It is possible to queue up messages to be sent at a later time; however, in the case where data is planned to be streamed across the mesh, a few dropped messages during device discovery is acceptable.

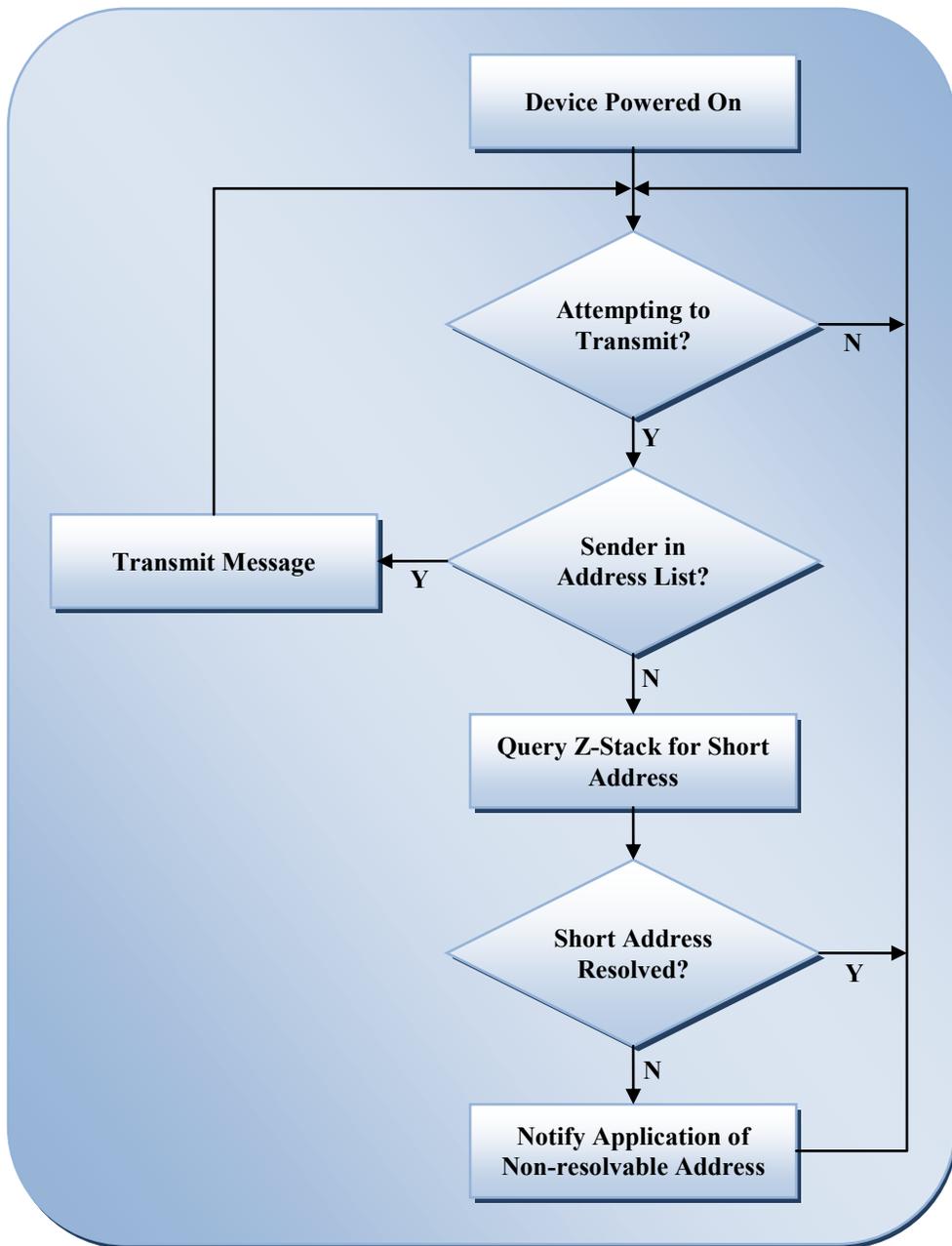


Figure 3-16: Z-Stack API Interface Flow Diagram

3.5. ZigBee Network Tests

Several tests were performed to determine the overall effectiveness of the ZigBee mesh network implemented in this thesis. First of all, ranging tests were performed in order to indicate a ZigBee device’s effectiveness in determining signal

strength between nodes. Secondly, basic data transmission tests were performed to verify overall network functionality.

3.5.1. Ranging Tests

While the CC2431 devices do have a lower data rate and smaller overall range, the Z-Stack does provide access to a value known as the Link Quality Indicator (LQI) for each message that is received. With slight modification of the method called as messages are received in the Z-Stack, it is possible to append the LQI measurement to every incoming message. Once this modification is in place, an attached device is informed of the link quality for every message received. This information is formatted as a normalized version of the Received Signal Strength Indicator (RSSI) that is scaled to fit in one byte (values from 0 to 255).

Furthermore, the Texas Instruments' CC2431 module provides the ability to vary the output power utilized when transmitting data between devices. The CC2431 module register that is responsible for this setting is called TXCTRL. The output power can be varied from -25.2dBm to 0.6dBm; however, as the power is increased, the current consumption also increases [4]. At the lowest power setting, the current consumption is approximately 18.3mA, while the highest power setting has a current consumption of approximately 32.4mA [4]. When transmitting data, it is possible to vary the output power setting in between data transmissions. This power modification allows for a receiving node to acquire packets of data with differing power levels. It is possible to use this data, especially when fused with inertial data on a fixed platform (such as robots and humans) to determine a rough position

estimate between nodes in the mesh. Figure 3-17 illustrates the data taken at multiple positions from 2 to 25 feet of separation while monitoring three differing power levels, specifically, -0.4dBm, -15.4dBm, and -25.2dBm. In this test, the devices were positioned facing each other, providing a path clear of obstructions in between.

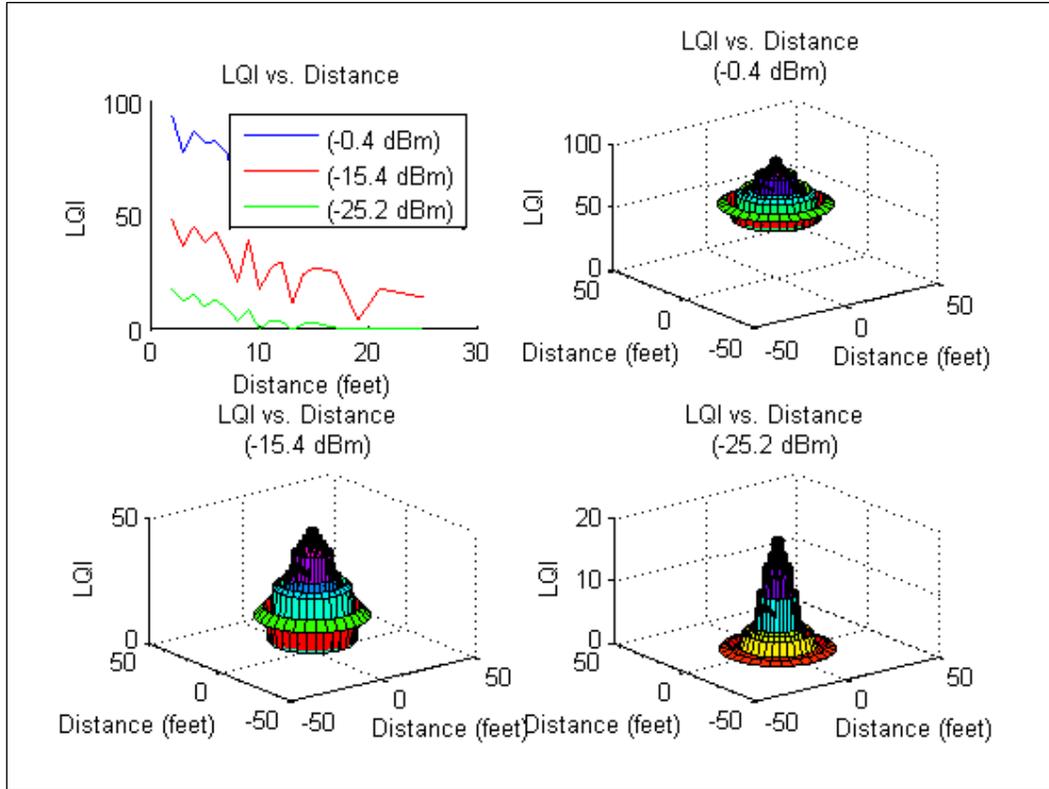


Figure 3-17: LQI Measurements at Three Power Levels

When smoothing the data, it is noteworthy to mention that the overall trend in the data remains roughly the same for all power levels. The smoothed data is depicted in Figure 3-18. When this range information is fused with data from inertial sensors placed on each node, it may be possible to improve the overall relative distance measurements between nodes.

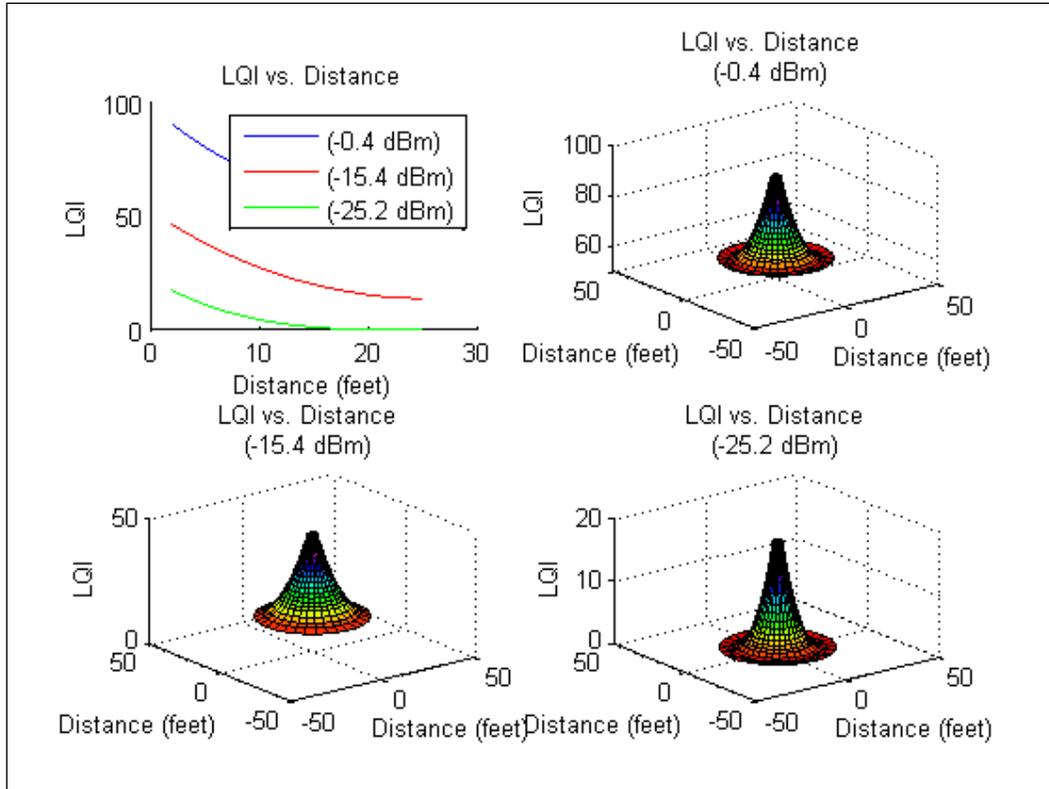


Figure 3-18: Smoothed LQI Measurements

Additionally, when the basic topology of a building is known, it is possible to incorporate equations detailing propagation losses inside buildings in order to further improve reliability of signal strength measurements. As mentioned in Physical Principles of Wireless Communications, an expression has been defined by the International Telecommunications Union (ITU) in order to estimate the mean path loss inside buildings [36]. Furthermore, it is known that for various building structures, the path loss through floors will change and is able to be estimated for office, residential, and department store buildings, if the building type is known in advance [36].

3.5.2. Transmission of Data between Nodes

In addition to utilizing the ZigBee wireless mesh network to monitor LQI information transmitted between nodes, the underlying mesh has also been utilized in order to transmit data wirelessly between nodes. For example, the TRX Systems INU utilizes the ZigBee network to share recorded data amongst other devices in the field; in essence, creating a wireless body area network (WBAN). Furthermore, it is possible to utilize the ZigBee network to stream packetized raw data from TRX Systems' INUs for development and testing purposes.

Chapter 4: Future Work

4.1. Introduction

Mobile ad hoc mesh networks are slowly becoming more popular with mainstream America. They continue to be introduced in many devices; and, as they become less expensive and more widespread, wireless networking may become more dependent upon them. This thesis touches only on two specific implementations of mobile mesh networking devices, and much additional work remains to be done. With regard to the two wireless mobile mesh networks discussed in this thesis, there are several points yet to be explored that could build upon ideas already presented here. Below are some possible topics for future discussion.

4.2. Competing Routing Algorithms

This thesis focused only on two major routing algorithms for transmitting data in a mobile wireless mesh network. The Wi-Fi mesh network implemented in this thesis makes use of the OLSR protocol for discovering and maintaining routes in a MANET. This protocol has known limitations as an increased number of nodes are added to the mesh [20]. Due to these limitations, an OLSR mesh may not be the optimum choice for creating a MANET when performing larger scale tests. Other protocols, such as B.A.T.M.A.N., which is currently in preliminary stages of development, may better address these concerns. It would be particularly significant

to measure the bandwidth variations when transmitting large amounts of data, as different underlying routing protocols are varied. Furthermore, it would be worthwhile to detail the scaling of these mesh networking protocols in order to see how it affects overall network performance.

4.3. Further Use of Ranging

When implementing MANETs, one topic only briefly discussed, was the possible use of signal strength information between nodes to better ascertain an individual node's relative position in the network. Furthermore, if a mesh network were to come in contact with a stationary node (with absolute position information), it is possible to determine absolute positions of all nodes within the network in accordance with some error bound. The CC2431 solution provided by Texas Instruments integrates a location engine that is capable of such positioning. This chip has the ability to estimate its position within a network via a hardware module that utilizes signal strength indications from nodes with known locations [4]. This information definitely has the potential for further study.

Moreover, it has been shown that sensor localization can be improved further when paired with several signal processing techniques. Additionally, sensor fusion incorporating signal strength information together with data from inertial navigation units (INUs) can be used to improve location estimates. Zhao, Wu, Mau, and Kaba address this localization improvement concept through the use of the Extended Kalman Filter (EKF) and fusion of INU measurements along with RSSI information. They have determined that an improved positional estimate can be resolved [37].

Their paper suggests that it is possible to determine a location estimate that is improved by a factor of $1/\sqrt{n}$, where n is the number of nodes in the network, when comparing individual location estimates with estimates from collaborating nodes [37]. Since ZigBee nodes provide LQI information through the Z-Stack API, it would be advantageous to complete a real world demonstration of methodology suggested in this paper.

4.4. More Real World Applications

The Wi-Fi MANET described in this thesis is currently implemented on a swarm of eight fully autonomous ground vehicles designed and built by the Autonomous Systems Lab at the University of Maryland. The Autonomous Systems Lab has also performed some initial research involving the use of autonomous helicopters paired with ground vehicles. For example, the Align T-REX 450 RC Helicopter displayed in Figure 4-1 has been utilized for research in the Autonomous Systems Lab. The ZigBee mesh network may be of particular interest for future testing as it is lightweight and could easily be attached to one of the helicopters.



Figure 4-1: Align T-REX 450 RC Helicopter [38]

Appendix A: Custom Router Firmware Installation

A.1. Introduction

Installing a custom firmware image onto a commercial router for the first time can sometimes be a daunting task. The installation does require the use of command line utilities, and does leave the possibility for permanently damaging (often referred to as bricking) the router. As long as the procedure detailed below is followed, there is a relatively low chance that the router will be damaged. The procedure provided below details the steps to install the Freifunk firmware onto the Buffalo WHR-HP-G54 router using a Windows PC.

A.2. Connecting to the Buffalo Router

Buffalo routers ship with proprietary firmware that is fairly easy to use. All settings can be changed using the web-based interface. The router is set up to be a DHCP server by default. To install a custom firmware image, a computer will need to be directly connected to the router with an Ethernet cable. This computer should be configured to obtain an IP address and DNS server automatically. Once the computer connected to the router has been leased an IP, the settings for the Buffalo router can be accessed through the web interface. The settings display in Table A-1 should be used to connect to the router.

Setting	Value
IP Address:	192.168.11.1
Username:	root
Password:	<blank>

Table A-1: Default Buffalo Router Settings

Upon keying the default IP address into a web browser, one will be greeted with the Buffalo router firmware settings page. This screen should be similar to the image in Figure A-1.



Figure A-1: Buffalo Router Configuration Web Interface [30]

A.3. Restoring Factory Default Settings

The first step to perform before loading a custom firmware image onto a router is to restore the router to factory default settings. This step is especially important if the router has been used in other environments, and passwords and/or encryption settings have been modified from their default parameters. To reset the Buffalo WHR-HP-G54 router to default settings, perform the following steps:

1. Browse to “<http://192.168.11.1/>” in any web browser.
2. Enter username and password at the prompt.
3. Click the “Advanced” tab.
4. Click “Admin Config.”
5. Click “Initialize/Restart.”
6. Click “Initialize Now.”
7. Click “Set.”

The reset procedure may take a few minutes to complete. After the router has been reset to factory default settings, the user may proceed with the firmware upgrade.

A.4. Loading Custom Firmware

To install custom firmware onto the Buffalo WHR-HP-G54 router, the new firmware must first be downloaded from the Freifunk website. As of this writing, the most recent Freifunk firmware is version 1.6.25 and the image downloaded will be

named `openwrt-freifunk-1.6.25-en.trx`. To install the new firmware, it is first necessary to make certain that the TFTP utility is installed on your computer.

The first step in loading the custom firmware is to manually configure the Local Area Network (LAN) settings of one's computer. It is also helpful to disable any other networks that may exist on the computer (such as a wireless card). They may cause installation conflicts. The installation procedure requires manually resetting the router and because of this requirement, one cannot use the router's DHCP server to assign an IP address to the computer. The following settings in Table A-2 should be used when manually configuring your computer.

Setting	Value
IP Address:	192.168.11.100
Subnet Mask:	255.255.255.0
Default Gateway:	192.168.11.1

Table A-2: Manual Computer LAN Configuration

After the computer has been configured using these settings, the following procedure will load the new firmware onto the router:

1. Open the Command Prompt.
2. Browse to the directory containing the new firmware image.
3. Key the following into the command prompt without pressing enter:

```
tftp -i 192.168.11.1 put openwrt-freifunk-1.6.25-en.trx
```

4. Unplug the router.
5. Connect the router and immediately press enter in the command prompt window.

It is possible to receive a “Timeout occurred” message if the timing is not within certain parameters when attempting to load new firmware. If this timeout occurs, repeat the procedure again. Once you receive a “Transfer successful” message, the router will begin updating the firmware, and may restart several times. After several minutes, you can then reset the computer to use DHCP to obtain an IP address and DNS servers automatically. Figure A-2 will be displayed when the Freifunk web interface is loaded in a browser window. Utilizing the Freifunk web interface, it is then possible to configure all the settings required to create an OLSR mesh network.

Home | Admin freifunk.firmware 

v1.6.25

Contents

- Status
- Contact
- Services
- OLSR Viz

Hello!

This is a Freifunk access point running the Freifunk Firmware version 1.6.25 (1.6.25) . Read about technical details regarding this access point on the [Status Page](#). Information about the Freifunk project can be found on the German internet site <http://www.freifunk.net/>.

Please note: Freifunk firmware and Freifunk webadmin are based on the outstanding Linux distribution [OpenWRT](#). OpenWRT (in contrast to other firmwares) enables you to install new software packages without the need to wait for firmware enhancements. 

Note: Instead of the usual image, the todays firmware version comes with a book recommendation. You can replace this text box by using the Admin/Software/Upload Images function as usual.

The Wealth of Networks

How Social Production Transforms Markets and Freedom

Copyright 2006 by Yochai Benkler. All rights reserved.
Excerpt citation with kind permission granted by Mr. Benkler for Freifunk Firmware.

Chapter 1 Introduction: A Moment of Opportunity and Challenge

Information, knowledge, and culture are central to human freedom and human development. How they are produced and exchanged in our society critically affects the way we see the state of the world as it is and might be; who decides these questions; and how we, as societies and polities, come to understand what can and ought to be done. For more than 150 years, modern complex democracies have depended in large measure on an industrial information economy for these basic functions. In the past decade and a half, we have begun to see a radical change in the organization of information production. Enabled by technological change, we are beginning to see a series of economic, social, and cultural adaptations that make possible a radical transformation of how we make the information environment we occupy as autonomous individuals, citizens, and members of cultural and social groups. It seems passe today to speak of "the Internet revolution." In some academic circles, it is positively naive. But it should not be. The change brought about by the networked information environment is deep. It is structural. It goes to

Some Links

- [Berlin Website, DE](#)
- [Freifunk Firmware Page of Freifunk.net \(EN\)](#)
- [Freifunk Firmware and IPK Downloads](#)
- Firmware download from this device:
[WRT54G](#) [WRT54GS](#) [WRT54GS-v4.0](#) [WRT54G3G+UMTS](#) [All0277](#) [WR850G](#) [SE505](#) [WAP54G/WL500](#)
- [Never-ending download from /dev/zero](#)

Please note: Internet access over the Freifunk network is bound to technical and organizational conditions. For this reason, the links to internet web pages may not work.

Changed: 19.9.2007 Top of page

Figure A-2: Freifunk Web Interface [24]

Appendix B: Acronyms / Abbreviations

AES	Advanced Encryption Standard
ANSI	American National Standards Institute
AODV	Ad hoc On-Demand Distance Vector
API	Application Programming Interface
APS	Application Support
ASCII	American Standard Code for Information Interchange
ASL	Autonomous Systems Lab
B.A.T.M.A.N.	Better Approach To Mobile Ad hoc Networking
BSD	Berkeley Software Distribution
CAN	Controller-Area Network
DARPA	Defense Advanced Research Projects Agency
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DSC	Digital Signal Controller
EKF	Extended Kalman Filter
ESC	Electronic Speed Control
FCS	Frame Check Sequence
FPS	Frames Per Second
GUI	Graphical User Interface

HTTP	Hypertext Transfer Protocol
I²C	Inter-Integrated Circuit
IETF	Internet Engineering Task Force
INU	Inertial Navigation Unit
IP	Internet Protocol
ITU	International Telecommunications Union
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
LQI	Link Quality Indicator
MAC	Media Access Control
MANET	Mobile Ad Hoc Network
MPR	Multipoint Relay
NIC	Network Interface Controller
NV	Non-Volatile
NWK	Network
OEM	Original Equipment Manufacturer
OGM	Originator Message
OLSR	Optimized Link State Routing
PAN	Personal Area Network
PCB	Printed Circuit Board
PHY	Physical
RAM	Random Access Memory
RC	Radio Control

RF	Radio Frequency
RFC	Request for Comments
RSSI	Received Signal Strength Indication
SOP	Start of Packet
SPI	Serial Peripheral Interface
SSID	Service Set Identifier
TC	Topology Control
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TI	Texas Instruments
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
WBAN	Wireless Body Area Network
WDS	Wireless Distribution System
WEP	Wired Equivalent Privacy
WNIC	Wireless Network Interface Controller
WPA	Wi-Fi Protected Access
WPAN	Wireless Personal Area Network
XOR	Exclusive Or
ZC	ZigBee Coordinator
ZDO	ZigBee Device Object
ZED	ZigBee End Device
ZR	ZigBee Router

Bibliography

- [1] *Technology@Intel - Thoughts on Netbooks.*
http://blogs.intel.com/technology/2008/03/thoughts_on_netbooks.php.
- [2] <http://www.darpa.mil/ipto/programs/ld/ld.asp>.
- [3] http://en.wikipedia.org/wiki/Body_Area_Network.
- [4] <http://focus.ti.com/docs/prod/folders/print/cc2431.html>.
- [5] <http://www.buffalotech.com/support/downloads/>.
- [6] <http://www.linksysbycisco.com/US/en/support/WRT54G/download>.
- [7] <http://rovingnetworks.com/bluetooth-modules.php>.
- [8] http://en.wikipedia.org/wiki/IEEE_802.11n.
- [9] <http://en.wikipedia.org/wiki/Bluetooth>.
- [10] <http://en.wikipedia.org/wiki/Zigbee>.
- [11] http://en.wikipedia.org/wiki/Wireless_Distribution_System.
- [12] http://www.dd-wrt.com/wiki/index.php?title=WDS_Linked_router_network.
- [13] http://en.wikipedia.org/wiki/Ad-hoc_On-demand_Distance_Vector.
- [14] <http://tools.ietf.org/html/rfc3561>.
- [15] <http://www.olsr.org/>.
- [16] http://en.wikipedia.org/wiki/Optimized_Link_State_Routing_protocol.
- [17] <http://tools.ietf.org/html/rfc3626>.
- [18] <http://www.open-mesh.net/>.

- [19] <http://en.wikipedia.org/wiki/B.A.T.M.A.N.>
- [20] <http://www.open-mesh.net/wiki/BATMANConcept>.
- [21] http://en.wikipedia.org/wiki/Linksys_WRT54G_series.
- [22] <http://www.dd-wrt.com/>.
- [23] <http://openwrt.org/>.
- [24] <http://www.freifunk.net/>.
- [25] <http://www.polarcloud.com/tomato>.
- [26] <http://www.zigbee.org/>.
- [27] <http://www.meshnetics.com/zigbee-faq/>.
- [28] <http://www.jennic.com/elearning/zigbee/index.htm>.
- [29] <http://www.dd-wrt.com/wiki/index.php/Index:FAQ>.
- [30] <http://www.buffalotech.com/home/>.
- [31] <http://en.wikipedia.org/wiki/Freifunk>.
- [32] <http://tools.ietf.org/html/rfc1918>.
- [33] Wireless Africa Team of the Meraka Institute. "Building a Rural Wireless Mesh Network." 2007.
- [34] <http://www.ti.com/litv/pdf/swru133>.
- [35] <http://www.trxsystems.com/>.
- [36] Granatstein, Victor L. *Physical Principles of Wireless Communications*. Boca Raton: Auerbach Publications, 2008.
- [37] Zhao, Tao, Shunguang Wu, Siun-Chuon Mau, and Jim Kaba. "Collaborative Effects of Mobile Sensor Network Localization Through Distributed Multimodal Navigation Sensor Fusion." *ION NTM*, 2008: 699-710.
- [38] <http://www.align.com.tw/>.