ADVANCED WIRELESS MESH NETWORKS: DESIGN AND IMPLEMENTATION

By

Julio Castillo-Tito

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS

2010

Approved by:

Henrick M. Ierkic, Ph.D Member, Graduate Committee	Date
Pedro I. Rivera, Ph.D Member, Graduate Committee	Date
Kejie Lu, Ph.D President, Graduate Committee	Date
Robert Acar, Ph.D Representative of Graduate Studies	Date
Hamed Parsiani, Ph.D Chairperson of the Department	Date

Abstract of Thesis Presented to the Graduate School
of the University of Puerto Rico in Partial Fulfillment of the
Requirements for the Degree of Master of Science

ADVANCED WIRELESS MESH NETWORKS: DESIGN AND IMPLEMENTATION

By

Julio Castillo-Tito

2010

Chair: Kejie Lu
Major Department: Electrical and Computer Engineering

Wireless Mesh Networks (WMNs) have emerged as a fundamental technology for the next generation of Wireless Networking. They are formed by mesh routers and mesh clients. Mesh routers have minimal mobility and usually do not have energy constraints. On the other hand, mesh clients are mobile and energy constrained. WMNs are self-forming, self-healing and self-organizing. Their easy configuration and deployment make them an economical, reliable and simple solution that can be implemented anywhere at any time. WMNs have been applied to many areas such as education, government, industry, academia and military.

The purpose of this thesis is to design and implement two WMN testbeds using the MAC Layer Routing Protocol (MACRT) and apply them in the Development of a Versatile Service-Oriented Wireless Mesh Network project (VESO-MESH). The analysis, design and implementation have been done using commercial off-the-shelf (COTS) hardware and free software. The operating systems are based on Linux distributions. The wireless driver is a Madwifi modified version. The cards used were PCI, miniPCI and PCMCIA; which are based on Atheros chipset. The mesh routers

and mesh clients used were laptops, PCs, AOpen computers, Alix 2D2 boards, and a Pocket PC.

Resumen de Tesis Presentado a Escuela Graduada
de la Universidad de Puerto Rico como requisito parcial de los
Requerimientos para el grado de Maestría en Ciencias

REDES INALÁMBRICAS EN MALLA: DISEÑO E IMPLEMENTACIÓN

Por

Julio Castillo-Tito

2010

Consejero: Kejie Lu
Departamento: Ingeniería Eléctrica y Computadoras

Las redes inalámbricas malladas (WMNs) han emergido como una tecnología fundamental para la siguiente generación de Redes Wireless. Están formadas por routers mesh y clientes mesh. Los routers mesh tienen mínima mobilidad y usualmente no tienen restricciones de energía. Por otra parte, los clientes mesh son móbiles y con restricciones de energía. Las WMNs se auto-forman, se auto-curan y se auto-organizan. Su fácil configuración y despliegue hacen de estas una solución económica, confiable y simple que puede ser implementada en cualquier lugar y momento. Las WMNs están siendo aplicadas en muchas áreas tales como educación, gobierno, industria, universidades y milicia.

El propósito de esta tésis es diseñar e implementar dos WMN testbeds utilizando el MAC Layer Routing Protocol (MACRT) y aplicarlos en el proyecto Development of a Versatile Service-Oriented Wireless Mesh Network (VESO-MESH). El análisis, diseño, e implementación ha sido realizado utilizando equipo de bajo costo comercial (COTS) y software libre. Los sistemas operativos utilizados están basados en distribuciones Linux. El manejador wireless es una version modificada de Madwifi.

Las tarjetas utilizadas fueron PCI, miniPCI y PCMCIA; las cuales están basadas en el chipset de Atheros. Los routers mesh y clientes mesh utilizados fueron laptops, PCs, computadoras AOpen, placas Alix 2D2 y una PC de bolsillo.

To God, my family, and my friends ...

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Kejie Lu for his support and guidance throughout my master studies and the development of the present thesis. I also want to thank to all my professors of the graduate committee for their support.

This work is sponsored by the US National Science Foundation (NSF) under grant DUE-0736868, "Improving the Curriculum of Electrical and Computer Engineering through A Wireless Network Testbed: A Vertical Integration Approach" and Award Number CNS-0922996, "MRI: Development of A Versatile Service-Oriented Wireless Mesh Network for Disaster Relief And Environmental Monitoring in Puerto Rico".

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ACK	Acknowledgement
AES	Advanced Encryption Standard
AODV	Ad Hoc On-Demand Vector Routing
AODV-ML	Multi-Link AODV
AP	Access Point
BSS	Basic Service Set
CCK	Complementary Code Keying
CF	Compact Flash
COTS	Commercial Off-The-Shelf
CRL	Computing Research Laboratory
CSMA/CA	Carrier Sense Multiple Access With Collision Avoidance
CTS	Clear To Send
D-AODV	Directional AODV
DCF	Distributed Coordination Function
DHCP	Dynamic Host Configuration Protocol
DRAM	Dynamic Random Access Memory
DS	Distribution System
DSDV	Destination-Sequenced Distance Vector
DSR	Dynamic Source Routing
DSSS	Direct Sequence Spread Spectrum
ESSID	Extended Service Set Identifier
ETT	Expected Transmission Time
ETX	Expected Transmission Count
GCC	GNU Compiler Collection
GNU	GNU's not UNIX
HAL	Hardware Abstraction Layer
HP	Hewlett Packard
IBSS	Independent Basic Service Set
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
INRIA	National Institute for Research in Computer Science and Control
LAN	Local Area Network
LLC	Logical Link Control
LQSR	Link Quality Source Routing
MAC	Media Access Control
MACRT	MAC Layer Routing Protocol
MANET	Mobile Ad Hoc Network
MAP	Mesh@Purdue

MCL	Mesh Connectivity Layer
MIMO	Multiple-Input Multiple-Output
MIT	Massachusetts Institute of Technology
MPDU	MAC Protocol Data Unit
MPR	Multi-Point Relay
MRI	Major Research Instrumentation
MRLQSR	Multiradio Link Quality Source Routing
MSDU	MAC Service Data Unit
MSN	Master Sensor Node
NAT	Network Address Translation
NAV	Network Allocation Vector
NSF	National Science Foundation
OFDM	Orthogonal Frequency Division Multiplexing
OLSR	Optimized Link State Routing
OS	Operating System
OSI	Open Systems Interconnection
PC	Personal Computer
PCF	Point Coordination Function
PCI	Peripheral Component Interconnect
PCMCIA	Personal Computer Memory Card International Association
PDA	Personal Digital Assistant
PHY	Physical
POE	Power over Internet
PPDU	PHY Protocol Data Unit
PSDU	PHY Service Data Unit
QoS	Quality of Service
RAM	Random Access Memory
RASSOC	Re-association
RERR	Route Error Message
RREP	Routing Reply Message
RREQ	Routing Request Message
RTS	Request To Send
SDU	Service Data Unit
SSID	Service Set Identifier
STA	Station
STAR	Source-Tree Adaptive Routing Protocol
STP	Spanning Tree Protocol
TC	Topology Control
TKIP	Temporal Key Integrity Protocol
UCSB	University of California, Santa Barbara
UPR-RUM	University of Puerto Rico at Mayaguez
USB	Universal Serial Bus
USCB	University of California, Santa Barbara
VAP	Virtual AP
VCS	Virtual Carrier Sense

VESO-MESH	Versatile Service-Oriented Wireless Mesh Network
VNC	Virtual Network Computing
WDS	Wireless Distribution System
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WRP	Wireless Routing Protocol
ZRP	Zone Routing Protocol

CHAPTER 1
INTRODUCTION

1.1 MOTIVATION

Over the past years, wireless networks have been applied in many environments such as education, government, industry, military and academia. In some cases, combining wireless networks with wired networks empowers the environment with scalability, flexibility, and mobility. Currently, this technology is becoming increasingly popular due to many important reasons, such as mobility, cost and installation. Mobility is the ability of a terminal to maintain a user session while roaming between different networks. The wireless network's cost is cheaper than that of traditional wired networks, and the installation process is relatively easy and simple opposite to wired networks that are more complicated to set up.

A particular technology called Wireless Mesh Network (WMN) is part of the wireless networks world. This technology marked the divergence from the traditional centralized wireless systems such as Wireless Local Area Networks (WLANs). WMNs have advantages such as self-forming, self-healing, and self-organizing. Wireless Mesh Networks are formed by mesh routers and mesh clients. Mesh routers have minimal mobility and do not have energy constraints, but mesh clients are mobile and energy constrained.

Currently, there are many on-going research projects making WMNs testbeds in different universities and research labs such as CitySense [1], MIT Roofnet [2], Microsoft Research [3], UCSB MeshNet [4], and Mesh@Purdue [5].

WMNs present some research issues [6], ranging from the physical layer to the application layer; algorithms and protocols in various layers need to be improved, new schemes are required for network management, and the network still lacks security. Thus, existing algorithms and protocols need to be enhanced or re-invented for WMNs.

1.2 CONTRIBUTIONS

This thesis provides the following contributions on ways to better understand how to make WMNs testbeds using Commercial Off-the-Shelf (COTS) hardware and free software.

1. We analyzed, designed, implemented, and deployed a WMN testbed based on PCMCIA and PCI cards.
    - We gave routing functionality to PC and Laptop computers, installing the MACRT routing protocol onto their wireless cards, which are based on Atheros chipset.
    - We provide all the necessary steps to configure the WMN using Ubuntu Linux v8.04.1, kernel v2.6.22.19, GNU Compiler Collection (GCC) v4.2.4, a modified Madwifi v0.9.3 and the MACRT routing protocol.
2. We analyzed, designed, implemented, and deployed a WMN testbed based on Alix 2D2 boards.
    - We provide all the necessary steps to configure this WMN testbed. For this, it was necessary to do a special configuration on a laptop and then on an Alix 2D2 board. We used Debian Linux v5.0, kernel v2.6.22.19, GCC v4.2, a modified Madwifi v0.9.3 and the MACRT routing protocol as the software.
    - We created an image file to reproduce Alix 2D2 boards configuration on miniPCI cards to expand the WMN more easily in the near future.
3. We analyzed, designed, implemented, and deployed the VESO-MESH testbed at the Jobos Bay.

- We presented two VESO-MESH testbed versions.
  - The first one was based on the Wireless Distribution System (WDS) and Linksys WRT54GL routers.
  - The second one is based on the MACRT routing protocol, AOpen computers, and Alix 2D2 boards. For this, a Wistron miniPCI CM9 802.11abg card and two 5.5 dBi omnidirectional antennas were added as the hardware. Additionally, the MACRT routing protocol was installed as the software.

1.3 APPROACH

This thesis analyzes, designs, implements, and deploys different WMN testbeds using COTS hardware and free software. We have to understand some concepts about IEEE 802.11 architecture, WMNs, routing metrics, routing protocols, Linux, and madwifi wireless driver to develop them. These testbeds are explained in detail in the following chapters.

Basically, this work presents two important testbeds that will be used later by the VESO-MESH Project.

The first WMN testbed will be built using PCs and laptop computers as mesh routers. By default, those devices cannot work as mesh routers. We will give them the routing capability inserting PCMCIA or PCI cards depending on the device we are working with. We will install Ubuntu Linux v8.04.1. Afterwards, we will compile and install kernel v2.6.22.19 and GCC v4.2.4 versions respectively. Finally, we will install the MACRT routing protocol. The WMN testbed will be deployed and tested at some strategic points in and near the Computing Research Laboratory (CRL) of the Department of Electrical and Computer Engineering at the University of Puerto Rico at Mayagüez (UPR-RUM).

The second WMN testbed will be built using Alix 2D2 boards as mesh routers. These boards will have a miniPCI card attached. Each mesh router will have Debian

Linux v5.0 installed, kernel 2.6.22.19, GCC v4.2 and the MACRT routing protocol. Furthermore, we will have two laptop computers and a Pocket PC working as mesh clients. Those mesh clients run OS Windows XP, Windows 7 and Windows Mobile v5.0 respectively. We will deploy this WMN testbed at the Stefani building of the Department of Electrical and Computer Engineering at UPR-RUM. Then, we will do some experiments to check the network connectivity using ping commands and software such as Wireshark, Jperf, SSH, Putty, and others.

Finally, we will apply the two previous testbeds for building the VESO-MESH testbed that will serve as the platform for some on-going research topics that are being investigated at the Department of Electrical and Computer Engineering at UPR-RUM. The VESO-MESH Project [7] has as its main objective the Development of a Versatile Service-Oriented WMN that can be quickly built to respond to natural disasters and that establishes a data-intensive environmental monitoring application specifically addressing hurricanes and earthquakes. Therefore, the VESO-MESH testbed will be designed, implemented, and deployed as a first step to achieve the VESO-MESH Project objectives at the Jobos Bay National Estuarine located in the more arid southeastern coast of Puerto Rico [8].

This thesis is organized as follows: Chapter 2 presents the background. Chapter 3 presents the WMN testbed based on PCMCIA and PCI cards. Chapter 4 presents the WMN testbed based on Alix 2D2 boards. Chapter 5 presents the VESO-MESH testbed. Finally, we present conclusions and future work in Chapter 6.

CHAPTER 2
BACKGROUND

In this chapter, we present the basics concepts used in this thesis to design and implement an advanced WMN. We review the IEEE 802.11 standard. Then, we explain what is a WMN and its architecture. After that, we review some routing metrics and routing protocols. Furthermore, we explain about Atheros chipsets and the madwifi driver. Finally, we briefly present related work done in the WMN testbeds area.

2.1 IEEE 802.11

IEEE 802.11 forms part of IEEE 802 family which is a series of specifications for LANs [9]. IEEE 802.11 is the first wireless standard similar to Ethernet which was standardized as IEEE 802.3. It uses the 2.4 GHz and 5 GHz frequency bands. Currently, IEEE 802.11 supports bit rates bigger than 100 Mbps.

2.1.1 IEEE 802.11 ARCHITECTURE

IEEE 802.11 consists of three layers [10]: Logical Link Control (LLC), Media Access Control (MAC), and Physical (PHY). According to the Open Systems Interconnection (OSI) reference model, the LLC and the MAC layers are placed into the Data Link layer, while the PHY layer is placed into the physical layer as shown in Figure 2.1.

PHY and MAC layers offer services to the entity in the layer immediately above it. User data is transferred between the layers as a Service Data Unit (SDU). The MAC layer receives data from the LLC layer and delivers data to the LLC layer

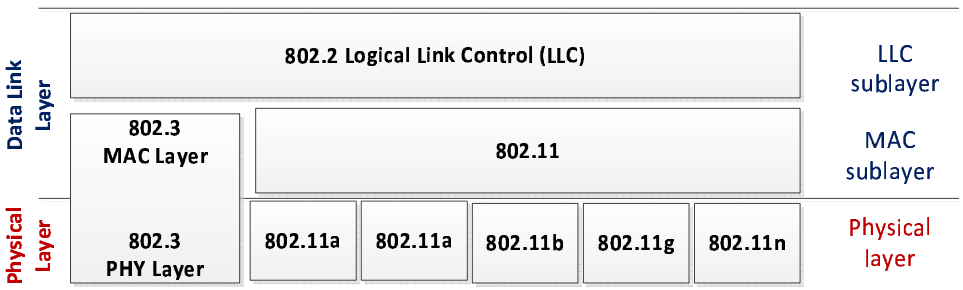


Figure 2.1: IEEE 802.11 architecture.

through the MAC SDU (MSDU). The PHY layer receives data from the MAC layer and delivers data to the MAC layer through the PHY SDU (PSDU). The MAC layer exchanges MAC Protocol Data Units (MPDUs) with its peer and the PHY layer exchanges PHY PDUs (PPDUs) with its peer. (Figure 2.2)

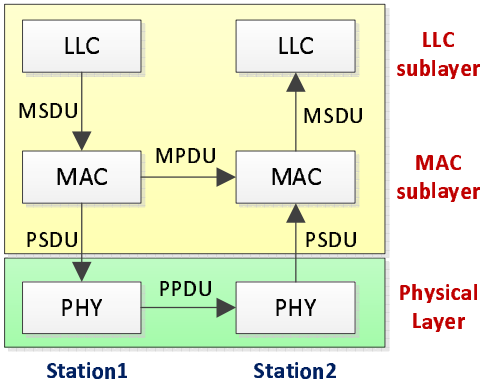


Figure 2.2: Protocol layering and messaging.

There are two important terms that we need to know: Station (STA) and Access Point (AP). The STA refers to the MAC and PHY integrated onto a Personal Computer (PC), laptop, PDA or other device. The AP is a station with additional functionality. It manages an infrastructure Basic Service Set (BSS) and provides access to the Distribution System (DS).

2.1.1.1 The Logical Link Control (LLC)

The LLC is the highest layer of the IEEE 802 Reference Model. It provides addressing and data link control. The aim of the LLC is to exchange data between

end users across a LAN. It provides an interface to higher layers and at the same time performs basic link layer functions like error and flow control. It is independent of the topology, transmission medium, and medium access control technique chosen.

Higher layers pass user data down to the LLC expecting error-free transmission across the network. Furthermore, this layer provides end-to-end link control over an IEEE 802.11-based WLAN.

2.1.1.2 The Media Access Control (MAC)

The MAC is responsible for transmitting the packets using the IEEE 802.11 Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) technique [11]. It provides addressing and channel access control, making possible multiple communication stations in a network. In addition, it ensures that all the devices cooperate into a LAN. This layer includes several services such as authentication, de-authentication, association, de-association, re-association, privacy, data transfer, distribution, integration and power management.

The MAC protocol specifies that only one station transmits at a time. The data is transmitted in blocks or MAC frames. Each frame includes user data, a source and destination address, error detection code, and MAC control bits. Furthermore, each station monitors the shared medium for frames with a destination address that matches its address, and copies frames addressed to it. The MAC layer with support of the LLC layer defines rules in order to access the shared medium. It defines two sub-layers: The Distributed Coordination Function (DCF) and the Point Coordination Function (PCF) [10].

The DCF is the fundamental access method of the IEEE 802.11 MAC. It is known as CSMA/CA. The DCF is an obligatory contention-based protocol. It can be used with both IBSS and Infrastructure network configurations. It allows STAs to access the medium in a distributed form; common asynchronous traffic is used by it. This access method begins before a STA starts a frame transmission. First, the

CSMA/CA senses the medium status and waits a specific time. If during this time the medium is free, then the STA sends the frame; otherwise, the STA defers the transmission until the medium is determined to be free. Then, a random back-off procedure is invoked. On the other hand, if the frame was received by the receiver station without errors the CSMA/CA sends an ACK frame. If no ACK is received during a time, the sender retransmits the frame again during a limited number of tries. Then, the frame is dropped. The DCF also defines two optional mechanisms: Request to Send (RTS) and Clear to Send (CTS). The sender and receiver exchange these control frames to reduce the possibility of collisions, but increase the overload decreasing the throughput.

The PCF is a priority based contention-free protocol. It can be used only on infrastructure network configurations. It provides contention-free services. Traffic with greater timing requirements makes use of the PCF. STAs called Point Coordinators are used to ensure that the medium is provided without contention. Typically, these Point Coordinators reside in the AP of the BSS and control the frame transmissions of the STAs in order to eliminate contention for a limited period of time. Furthermore, the PCF uses the Virtual Carrier Sense (VCS) mechanism that is provided by the Network Allocation Vector (NAV). The NAV is an indicator of time that shows the amount of time the medium will be reserved in microseconds by a STA. STAs set the NAV to the time for which they expect to use the medium. When the NAV is nonzero, the VCS function indicates that the medium is busy. When the NAV is 0, the virtual VCS function indicates that the medium is free.

2.1.1.3 The Physical layer (PHY)

The PHY defines the frequency band, data rate, encoding technique and other details of the actual radio transmission. There are some PHY layer extensions such as IEEE 802.11, IEEE 802.11b, IEEE 802.11a, IEEE 802.11g, and IEEE 802.11n.

They provide multiple data rates. We describe some of IEEE 802.11 standards briefly.

- IEEE 802.11

  The original version of the standard IEEE 802.11 was released in 1997. It included infrared (IR) operating at 1 Mbps, 2.4 GHz Frequency Hopped Spread Spectrum (FHSS), and 2.4 GHz Direct Sequence Spread Spectrum (DSSS) operating at 1 Mbps and 2 Mbps. Legacy IEEE 802.11 was rapidly supplemented and popularized by IEEE 802.11b due to IEEE 802.11b enhanced DSSS with Complementary Code Keying (CCK), which is a modulation scheme that increases the data rate to 11 Mbps.

- IEEE 802.11a

  IEEE 802.11a uses the same frame format as the original IEEE 802.11. It introduced Orthogonal Frequency Division Multiplexing (OFDM). It operates in the 5 GHz band with a maximum data rate of 54 Mbps.

- IEEE 802.11b

  IEEE 802.11b operates in the 2.4 GHz with a maximum raw data rate of 11 Mbps. The throughput increment led the rapid acceptance of IEEE 802.11b as the definitive wireless LAN technology. IEEE 802.11b devices suffer interference from other products operating in the same band such as microwave ovens, Bluetooth devices, and cordless telephones.

- IEEE 802.11g

  IEEE 802.11g operates in the 2.4 GHz band with a maximum data rate of 54 Mbps. It uses the same OFDM scheme as IEEE 802.11a. IEEE 802.11g hardware is fully backwards compatible with IEEE 802.11b hardware. For this reason, this standard was adopted rapidly by consumers.

- IEEE 802.11n

  IEEE 802.11n was ratified by the IEEE in September 2009. It operates in both the

2.4 GHz and 5 GHz bands. It accepts data rates of 300 Mbps in 20 MHz and 600 Mbps in 40 MHz. These data rates are achieved through the use of Multiple-Input Multiple-Output (MIMO) technology and 40 MHz operation.

2.2 WIRELESS MESH NETWORKS

WMNs have emerged as an important technology for next generation wireless networks [6]. WMN is currently under development by IEEE 802.11 Task Group. They can be seen as a type of Mobile Ad Hoc Network (MANET). WMNs are multi-hop wireless networks formed by mesh routers and mesh clients (see Figure 2.3).

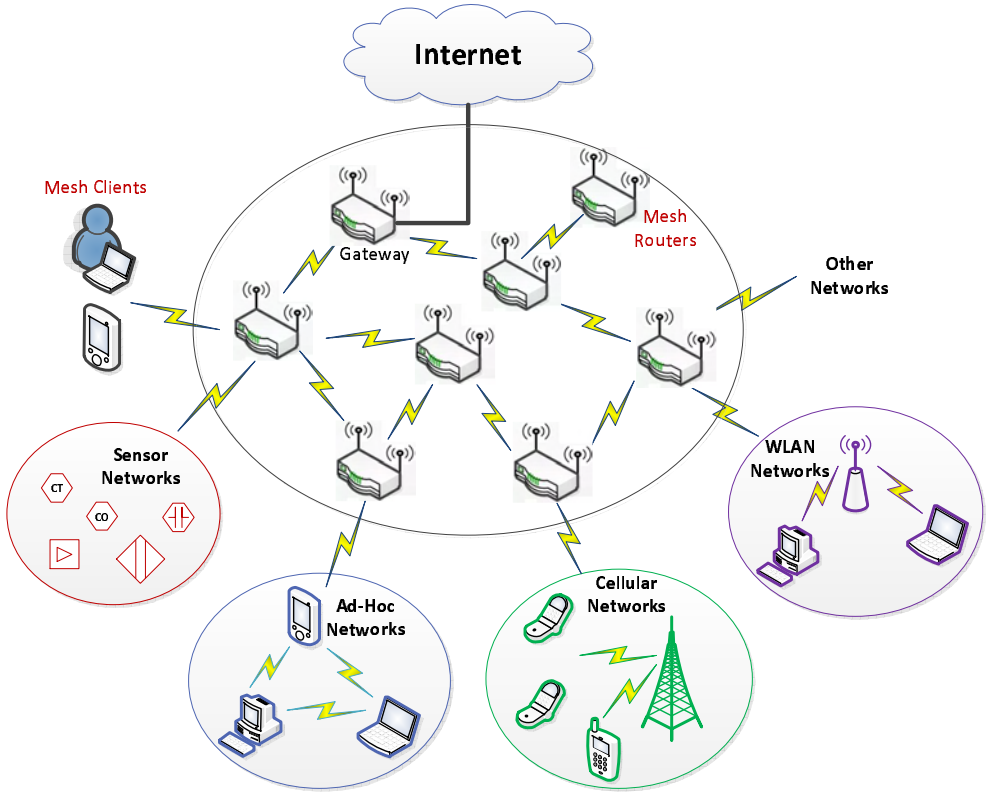


Figure 2.3: Wireless Mesh Network Architecture.

The mesh routers have minimal mobility and do not have energy constraints. They form a wireless mesh backbone among them. They forward the packets received from the mesh clients to the gateway router. The gateway router generally

is connected to the Internet through a wired backbone, and permits the interconnection of ad hoc, sensor, cellular and WLAN networks to the Internet. The mesh clients are mobile and energy constrained.

WMNs are dynamically self-forming, self-healing, and self-organizing. In addition, WMNs are great for rapid deployment, high scalability, easy management and low cost. Nowadays, WMNs are being used in many places and standards. Places like: homes, communities, municipalities, corporations, emergency response, intelligent transports, surveillance, and defense systems. Standards like: IEEE 802.11 (WLANs), IEEE 802.15 (Bluetooth and Zigbee), and IEEE 802.16 (WiMAX).

Currently, there are many on-going research projects on WMNs in academia, research labs and companies. The Massachusetts Institute of Technology (MIT) is working on the Roofnet Project, which is experimenting on IEEE 802.11bg mesh network. The University of California, Santa Barbara (UCSB) is conducting the MeshNet Project experimenting also on WMNs. The Microsoft Research is also working on a community mesh network. Those projects are building mesh platforms based on off-the-shelf products and developing demanding applications and services.

However, the WMN development deals with challenging issues in different layers of the protocol stack such as: architecture and protocol design, MAC and routing protocols, resource allocation and scheduling, network capacity, cost optimization, manageability, cross-layer design, and security. In addition, existing algorithms and protocols at each layer need to be enhanced or re-invented for WMNs.

2.3 ROUTING

WMNs combine advantages of WLAN and Mobile Ad Hoc Networks (MANETs). WMNs must deal with a highly unstable wireless medium. Hence, the designs of algorithms to choose the best routes are enabling routing metrics and protocols to be developed.

2.3.1 ROUTING METRICS

Routing metrics are values used by routing algorithms to select one route over another. Usually, they reflect communication cost in terms of required bandwidth, achievable delay, hop count, incurred load, reliability, energy consumption, etc. In literature there are many metrics for WMNs such as topology-based, signal strength-based, active probing-based, mobility-aware, and energy-aware. Some of them are Hop-Count, Expected Transmission Count (ETX) and Expected Transmission Time (ETT).

Hop-Count metric provides minimum hop-count routing. Every link (hop) counts equally as one unit, independent of the quality or other characteristics of the link. The advantage of this metric is its simplicity. This metric is used in many popular WMN routing protocols, such as Optimized Link State Routing (OLSR), Dynamic Source Routing (DSR), and Ad Hoc On-Demand Vector Routing (AODV).

Expected Transmission Count (ETX) metric proposed by [12]. It is based on active probing measurements specifically designed for MANETs. It estimates the number of transmissions including retransmissions required to send a packet over a link. The link ETX can capture the link quality and packet loss on both directions of a link. the route ETX can detect interference among links of the same route; the larger the route ETX, the less self-interference on the route [13].

Expected Transmission Time (ETT) metric proposed by [14], designed to improve the performance of ETX in multi-radio wireless networks that support different data rates. It considers the actual time incurred in using the channel. In order to measure the bandwidth of each link, each node sends two back-to-back probe packets to each of its neighbors periodically. The receiver node measures the difference between the instants of receiving the packets, and forwards the information to the sender. Then, the bandwidth is estimated by the sender node by dividing the larger

packet size by the minimum of 10 consecutive samples in order to estimate the data rate.

There are many other routing metrics such as: Weighted Cumulative ETT (WCETT) [15], Modified ETX (mETX) and Effective Number of Transmissions (ENT) [16], among others.

2.3.2 ROUTING PROTOCOLS

Routing protocols specify how mesh nodes communicate with each other to disseminate information that allows them to select routes between any two nodes on a network. Routing protocols exchange information such as: topology, load distribution, link quality among nodes, discovery and maintenance of routes to network nodes, and response when a route breaks. The major objective of a routing protocol for WMNs is to determine high-throughput routes between nodes so that the maximal end-to-end throughput can be achieved.

According to the routing information maintenance approach, there are three schemes: proactive or table-driven, reactive or on-demand, and hybrid routing protocols.

In the Proactive routing protocols, every node exchanges its routing information periodically and maintains a routing table, which contains routing information to reach every node in the network. Furthermore, the cost of overhead for maintaining the list of routes at each node would be detrimental as the network scales. Some examples of proactive routing protocols are Destination-Sequenced Distance Vector (DSDV) [17], Wireless Routing Protocol (WRP) [18], and Optimized Link State Routing (OLSR) [19].

In the Reactive routing protocols, a node requests routing information and maintains the path information only when it needs to communicate with another node. Some examples of reactive routing protocols are Ad Hoc On-Demand Vector

Routing (AODV) [20], Dynamic Source Routing (DSR) [14], Multiradio Link Quality Source Routing (MRLQSR) [15].

Finally, the Hybrid routing protocol combines the advantages of proactive and reactive protocols. The proactive approach is used for nearby nodes and/or for those routes that are used very often and the reactive approach is used for nodes that are further away and for those nodes that are seldom used for data relay. An example of such a hybrid routing protocol is the Zone Routing Protocol (ZRP) [21].

Some routing protocols along with their respective routing metrics for WMNs was summarized by [22]. Furthermore, there are some others routing protocols such as: Extremely Opportunistic Routing (ExOR) [23], Orthogonal Rendezvous Routing Protocol (ORRP) [24], among others.

2.3.2.1 Ad Hoc On-Demand Vector Routing (AODV)

AODV is a very popular routing protocol for MANETs. It is a reactive routing protocol. The routes are created only when they are needed. AODV has been standardized in the IETF as experimental RFC 3561 [20]. This protocol is loop-free and avoids the counting to infinity problem by the use of sequence numbers. AODV offers quick adaptation to mobile networks with low processing and low bandwidth utilization. It uses a simple request-reply mechanism for the discovery of routes. It can use hello messages for connectivity information and signals link breaks on active routes with error messages. Routing information has a timeout associated with the AODV as well as a sequence number. The use of sequence numbers allows detecting outdated data, so that only the most current available routing information is used. This ensures freedom of routing loops and avoids problems known from classical distance vector protocols, such as counting to infinity.

For instance, when a source node S wants to send data packets to a destination node D, but does not have a route to D in its routing table, then a route discovery

has to be done by S. The data packets are buffered during the route discovery as shown in Figure 2.4.

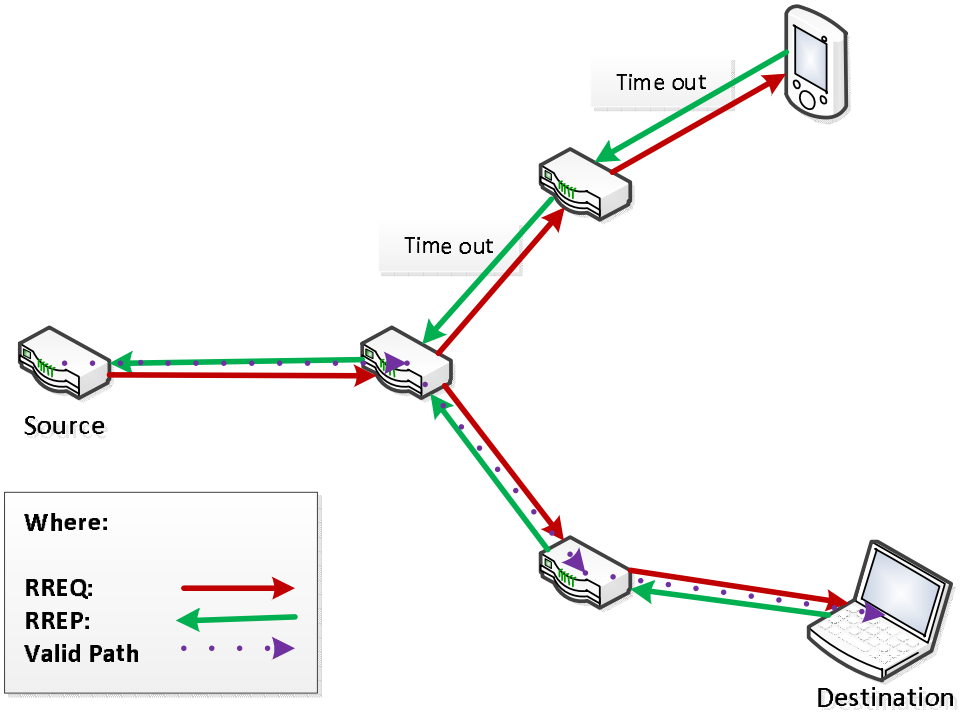


Figure 2.4: AODV route discovery.

The following control packets are used: Routing Request Message (RREQ), which is broadcasted by a node requiring a route to another node; Routing Reply Message (RREP), which is unicasted back to the source of RREQ, and Route Error Message (RERR), which is sent to notify other nodes of the loss of the link.

Additionally, AODV uses HELLO messages for detecting and monitoring links to neighbors. The weaknesses of AODV are its latency and scalability. Some implementations of this protocol are the Directional AODV (D-AODV) and the Multi-Link AODV (AODV-ML).

2.3.2.2 Optimized Link State Routing (OLSR)

OLSR is a popular proactive routing protocol and the most representative proactive routing protocol for Wireless Ad Hoc Networks. It has been developed at

INRIA and has been standardized at IETF as experimental RFC 3626 [19]. OLSR uses the classical shortest path algorithm based on the hop-count metric for the computation of the routes in the network. It is a link state routing protocol where each router keeps the topology information of the entire network. The routes to all other nodes are always available, no matter whether there is ongoing traffic or not.

OLSR uses an optimization called Multi-Point Relays (MPRs) to provide an efficient broadcast structure and to reduce the number of link advertisements (See Figure 2.5). The MPR of a node is one of its one-hop neighbors that have been selected as the next hop to reach a maximum number of its two-hop neighbors. The node that has selected its MPRs is referred to as an MPR selector. The set of MPRs, which is a subset of all one-hop neighbors of the MPR selector, can therefore reach all two-hop neighbors of the MPR selectors. The optimization of protocol overhead is achieved through the use of MPRs in a distributed network. Only MPRs with nonempty MPR selectors can generate Topology Control (TC) messages. When TC messages are received by a node, only those nodes that are MPRs will further forward TC messages to other nodes inside the same network. A MPR node may choose to report partial link state.

OLSR operates in 3 main steps:

- Neighbor sensing: This is achieved by exchanging HELLO messages between all one-hop neighbors in a network. Through periodic HELLO messages received from its one-hop neighbors, a node is able to select its MPRs. Correspondingly, a link state database and a neighborhood database are established by each node based on neighbor sensing.
- Topology control messages dissemination: Each node, through its MPRs, periodically advertises its link information to all other nodes inside the network. As a consequence, all nodes inside a network have necessary topology information for all links between any two nodes inside the same network.

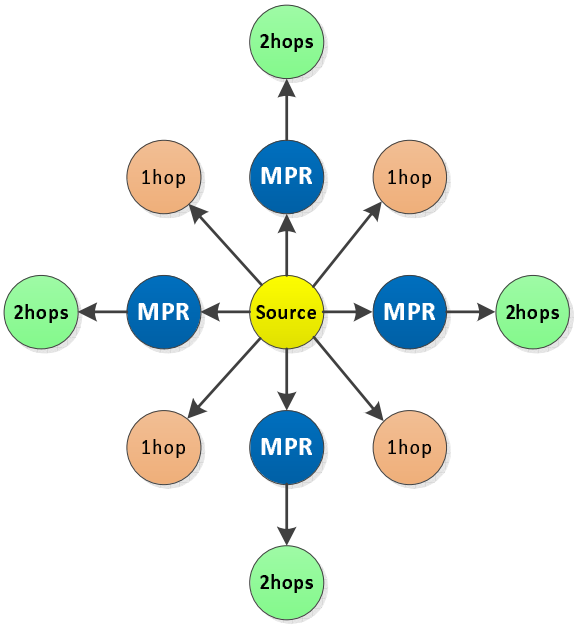


Figure 2.5: Multi-Point Relays (MPRs) in OLSR routing protocol.

- Routing table calculation: Based on TC messages received from other nodes, a node is able to compute its shortest-path routes to all reachable nodes in the network, by using an algorithm similar to the Dijkstra's algorithm.

The OLSR advantages are: provision of optimal routes, protocol suitability for large and dense networks and independency from other protocols. The OLSR disadvantages are: lack of security, overhead, and no support for multicasting.

2.3.2.3 MAC Layer Routing Protocol (MACRT)

MACRT was created for infrastructure WMNs [25]. It was built based on the idea of AODV routing protocol. MACRT was implemented on Linux distribution OS. It is deployed in the Calmesh Project [26]. MACRT uses the MAC layer instead of the IP layer, similar approaches are STP and Microsoft Research mesh network. MACRT uses the ETX routing metric to improve the throughput of the network. The MACRT advantages are: direct access to IEEE 802.11 MAC Layer messages, no need to change IP addresses during a handover, only mesh routers implement the routing protocol and mesh clients do not need to run any routing protocol. The

disadvantage of MACRT is: the Wi-Fi card driver needs to be modified to support it.

The main components of MACRT are: The Kernel Forwarding Module (kmacrt), the madwifi driver, and the User Space Routing Daemon (umacrt). Kmacrt forwards or caches data frames in the kernel space and send protocol events to Umacrt. The modified madwifi driver contents a new operation mode called mesh. This new mode is used for the communication among mesh routers. Umacrt implements the routing protocol and controls the behavior of kmacrt. (See Figure 2.6)

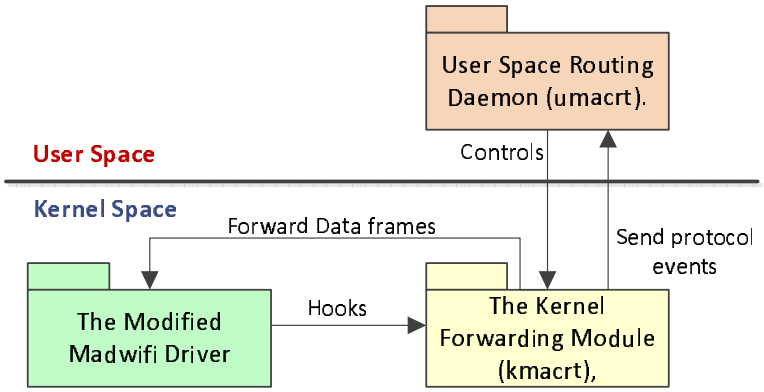


Figure 2.6: Components of MACRT routing protocol.

MACRT implements three principal operations. Those are route discovery, link break, and client roaming. On each operation, participates routing messages like: Route Request (RREQ), Route Reply (RREP), Route Error (RERR), and Re-association (RASSOC). Those routing messages are directly encapsulated in the IEEE 802.11.

MACRT creates a routing table. This routing table includes: the destination IP address, the destination MAC address, the destination sequence number, type of the destination, next hop MAC address, networks device to next hop, metric of the route, state of the route and the lifetime of the route.

2.4 MADWIFI

2.4.1 ATHEROS CHIPSETS

Multi-band Atheros Driver for Wi-Fi (Madwifi) is one of the most advanced WLAN drivers available for Linux today [27]. It is based on Atheros chipsets. There are some types of Atheros IEEE 802.11 wireless chipsets such as 5210, 5211 and 5212. The 5210 chipset supports IEEE 802.11a, the 5211 supports IEEE 802.11ab, and the 5212 supports IEEE 802.11abg. The Atheros devices security implement fixed/shared key Wired Equivalent Privacy (WEP) in hardware. Moreover, recently Atheros devices support Advanced Encryption Standard (AES) and Temporal Key Integrity Protocol (TKIP).

2.4.2 MADWIFI DRIVER

Madwifi has been developed by a team of volunteer developers. They provide three types of drivers: Madwifi, Ath5k and Ath9k [27]. The Madwifi driver is a stable version and open source. It has two components: the Hardware Abstraction Layer (HAL) and the Open Source. The HAL is available in binary format only and it is marked as proprietary to protect the hardware of improper uses. It manages the direct access to the Atheros hardware and implements basic functions of IEEE 802.11 such as CSMA/CA and processing of control frames. The Open Source implements functions such as processing of management frames, and the encapsulation/decapsulation of data frames. The current stable Madwifi driver release to date is v0.9.4. The Ath5k driver is a relatively new. It is an emerging driver and does not depend on the HAL. Ath5k is intended to replace Madwifi long-term. The Ath9k is the youngest of the three drivers. Atheros began its development after they released the complete source code to the community. Currently, the Ath9k supports all available IEEE 802.11n chipsets from Atheros.

The Madwifi driver supports the five following operation modes: STA, AP, IBSS (Ad-Hoc), Monitor and WDS (bridge). It supports PCI, miniPCI and Cardbus devices. It can be built as a module into the Linux kernel. It is divided into multiple modules such as the ath_pci module, which is loaded after the device is identified; the ath_hal module contains the HAL and it is needed by the ath_pci module.

Once we have the Madwifi driver installed, the ath_pci module must be loaded either manually or by the system. If it is load manually, we need to execute modprobe ath_pci and the remaining modules will be loaded automatically as needed. It will create two devices called wifi0 and ath0. The Wifi0 implies the existence of a physical MadWifi device. The ath0 is a Virtual AP (VAP) of the wifi0 physical device. It can be configured by networking tools like ifconfig, iwconfig, iwlist, iwpriv, etc. We can create multiple VAPs and interfaces running as an AP and station mode. The wlanconfig command is used to create and destroy VAPs with several different modes.

2.5 RELATED WORK

There is a lot of work related to WMNs testbeds. The majority of them have been developed in labs and universitites around the world. Below, we review some of these projects.

2.5.1 CITYSENSE

CitySense is an open, urban-scale wireless sensor network testbed [1]. It is developed by researchers at Harvard University and BBN technologies. CitySense's goal is to support other research groups to perform experiments in a real wireless sensor network that span an entire city. CitySense consists of about 100 single-board computers with IEEE 802.11 capability. Those single-boards are mounted on streetlights and rooftops across the city of Cambridge. They are equipped with

sensors to monitor contaminants, road traffic, air quality, noise pollution, weather, etc. CitySense uses WMNs for connectivity.

The Authors present several design challenges to design and deploy their WMN testbed including physical devices, physical environments, sensors nodes, network coverage, network security, monitoring and management. They are using the Alix 2D2 single-board computer as a mesh router. The Alix is equipped with a 4GB compact flash card as its hard drive and a Wistron CM9 IEEE 802.11abg miniPCI card, which is based on Atheros chipset, as its wireless card. The Alix has as it OS the FreeBSD Linux distributions and the Optimized Link State Routing protocol (OLSR) as its routing protocol.

2.5.2 ROOFNET

The MIT Roofnet project [2] consists of about 50 nodes based on PCs running Red Hat 9 Linux, kernel version 2.4.20. Each node has an IEEE 802.11b wireless interface and an omnidirectional antenna. All nodes run on the same channel. Each node runs a NAT, a Web server, and a DHCP Server. The goal of this project is to provide Internet access to students. Roofnet uses the SrcRR routing protocol, which is similar to Dynamic Source Routing (DSR) and the ETX routing metric.

2.5.3 COMMUNITY MESH NETWORK

Microsoft Research [3] works on a community mesh network. This mesh network allows neighbors to connect their home networks. They used a DSR modified version routing protocol and the ETX routing metric called Link Quality Source Routing (LQSR). They put the routing protocol and the routing metric in a module called Mesh Connectivity Layer (MCL). The MCL is a loadable Windows driver. They deployed testbed networks in their office buildings.

2.5.4 MESHNET

The University of California at Santa Barbara (UCSB) runs a mesh network project called MeshNet. Their vision is to design protocols and systems for the robust operation of multi-hop wireless networks [4]. They are focused on doing research in scalable routing protocols, network managements, multimedia and QoS for multi-hop WMNs. Their mesh network consists of 25 nodes equipped with IEEE 802.11abg capability. They used 2 Linksys WRT54G wireless devices strapped together as a mesh router. One Linksys is used for routing and the another for managing. MeshNet uses a modified version of the AODV routing protocol that uses the WCETT routing metric.

2.5.5 MESH@PURDUE

Purdue University has a WMN project called Mesh@Purdue (MAP) [5]. Currently, MAP consists of 32 nodes with IEEE 802.11abg capability. MAP uses commercial off-the-shelf (COTS) hardware such as HP Pentium desktops, miniPCI cards with Atheros-based chipset, Prism 2.5 Senao cards, directional and omni directional antennas. The research group works on modifications of AODV and OLSR routing protocols. Their vision is to extend MAP on their campus to study real-world issues about WMNs design and deployment.

CHAPTER 3
WMN TESTBED BASED ON PCMCIA AND PCI CARDS

This chapter discusses in detail the WMN testbed based on PCMCIA and PCI cards using the MACRT routing protocol. First, we present an overview of the WMN testbed. Second, we show the hardware and software used in this testbed. Third, we explain all the details about the mesh router and mesh client configurations. Finally, we present the deployment of the WMN testbed.

3.1 WMN TESTBED OVERVIEW

This WMN testbed was built using Commercial Off-The-Shelf (COTS) hardware and free software. We used PCs and laptop computers as mesh routers. By default, those devices cannot work as mesh routers. They need to add an extra device based on Atheros chipset. This device could be a PCMCIA or PCI card. It will depend on the computer we are using. Once we have all the hardware required, the next step is to install an operating system based on Linux distributions. In this case, we will install Ubuntu Linux v8.04.1. Then, we have to compile and install kernel v2.6.22.19 and GCC v4.2.4 respectively. This is because the MACRT routing protocol was built for those versions. After that, we will install the MACRT routing protocol on each computer as shown in Figure 3.1, where MR stands for Mesh Router and MC stands for Mesh Client. The WMN testbed was deployed and tested at some strategic points in and near the Computing Research Laboratory (CRL) of

the Department of Electrical and Computer Engineering at the University of Puerto Rico at Mayagüez.

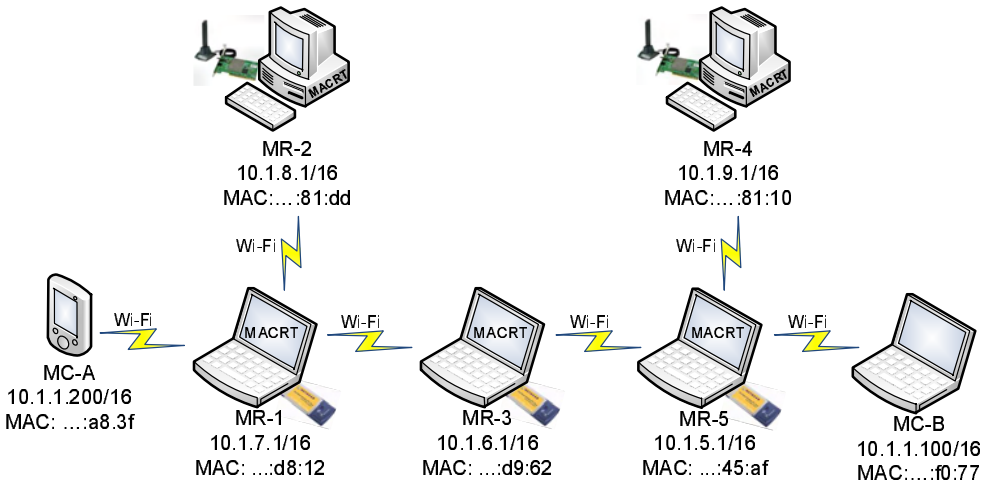


Figure 3.1: WMN testbed topology based on PCMCIA and PCI cards.

3.2 HARDWARE AND SOFTWARE

3.2.1 HARDWARE

The hardware used by this WMN testbed was:

- Dell Precision Workstation 360 (x2)
- Cisco Aironet IEEE 802.11abg Wireless PCI Adapter (x2)
- Laptop Dell Latitude D620 (x3)
- Netgear WAG511 PCMCIA card (x3)
- Pocket PC Dell Axim X51v (x1)

3.2.2 SOFTWARE

Basically, this testbed runs Ubuntu Linux v8.04.1 for the mesh routers. Mesh routers run the kernel v2.6.22.19 and the GCC v4.2.4. They also have a modified Madwifi v0.9.3 installed as the wireless driver and the MACRT routing protocol (See Table 3.1). Mesh clients could be any device running any operating system. The requirement is that the mesh client should have the IEEE 802.11abgn capability. We

Parameters	Versions
Kernel	2.6.22.19
GCC	4.2.4
Madwifi	0.9.3
MACRT	1.0

Table 3.1: Software configuration parameters.

used Microsoft Windows XP, Microsoft Windows 7, and Microsoft Windows Mobile 5 for the mesh clients. We also used many Linux commands and third party software such as Wireshark v1.2.4, Jperf v2.0.2, SSH Secure Shell v3.2.9, wicd v1.6.2, Xming v6.9, and VNC viewer v4.1.3, among others to check the connectivity and manage the network.

3.3 CONFIGURATION

3.3.1 MESH ROUTER CONFIGURATION

A mesh router configuration will be performed on the two PC and three laptop computers. Each PC computer will have a Cisco Aironet IEEE 802.11abg Wireless PCI Adapter installed, and each laptop will have a Netgear WAG511 PCMCIA card attached. The PCI and PCMCIA cards used are based on Atheros chipset. Figure 3.2 depicts the mesh router flow chart configuration.

3.3.1.1 Compiling and installing Kernel and GCC

This section explains step by step how to configure the mesh routers using the MACRT routing protocol. The first step is to have Ubuntu Linux v8.04.1 installed in the computers and the laptops. Once inside Ubuntu Linux v8.04.1, we have to check what the kernel and GCC versions are. To do this, execute the following commands:

```
// It is important to know which kernel and GCC versions are running on our computer.
    #cat /etc/issue
        Ubuntu 8.04.1
    #cat /proc/version
        Linux version 2.6.24-19-generic (root@julio-desktop) (gcc version 4.2.3 (Ubuntu 4.2.3-2ubuntu7))
```

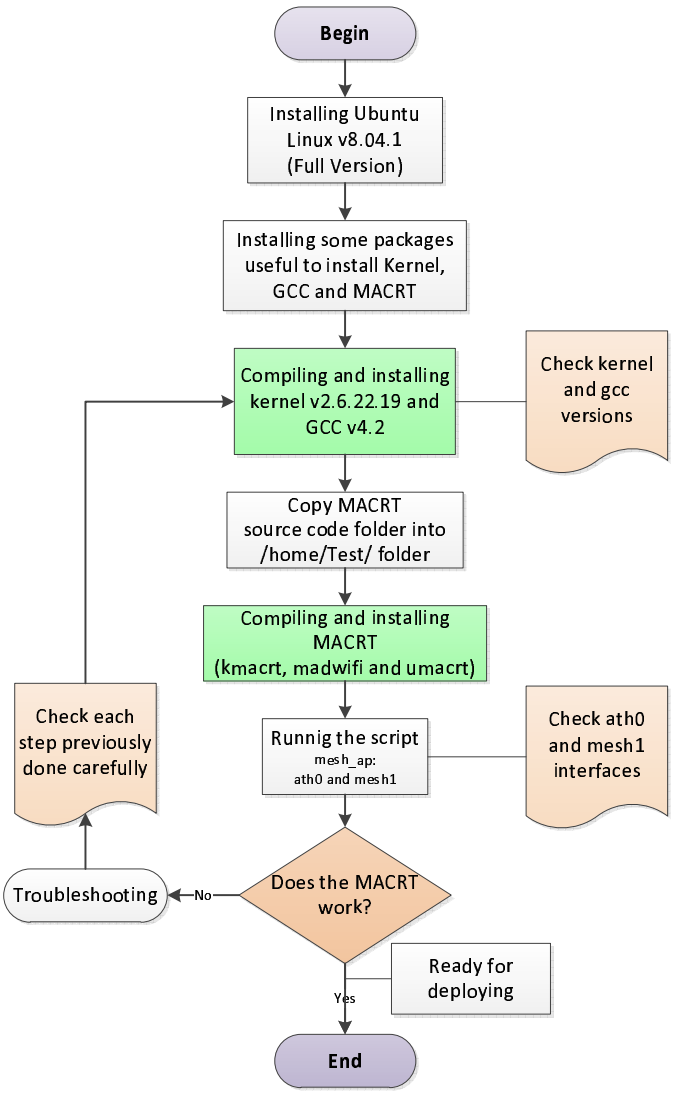


Figure 3.2: Mesh router flow chart configuration for PCs and laptops.

As we can see in the outputs, the kernel and the GCC versions are not the right ones for working with MACRT. For that reason, we have to install a new kernel v2.6.22.19 and GCC v4.2.4 as follows:

```
// Compiling the kernel v2.6.22.19 and getting the GCC v4.2.4
// Install the GCC version v4.2.4
    # apt-get update
    # apt-get dist-upgrade
    # reboot
// After rebooting the machine, we will have GCC v4.2.4
// Install some required packages for the installation
```

```
    # apt-get install build-essential
    # apt-get install kernel-package libncurses5
    # apt-get install libncurses5 libncurses-ruby libncurses5-dev libncursesw5-dev
// Compile and install the kernel v2.6.22.19
    # cd /usr/src/
// Go to /usr/src/ folder and install linux-source for the current Linux version
    # apt-get install linux-source
// Download kernel 2.6.22.19 from the web
    # wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.22.19.tar.bz2
// Extract files from the linux-2.6.22.19.tar.bz2 file
    # tar -xjvf linux-2.6.22.19.tar.bz2
// Change directory to the folder linux-2.6.22.19
    # cd linux-2.6.22.19
// Copy the boot file configuration, where the uname -r command will retrieve the current
// kernel version
    # cp /boot/config-$(uname -r) .config
// Compile the linux-2.6.22.19 configuration
    # make menuconfig
// Clean the kernel source directory
    # make-kpkg clean
// Check for dependencies
    # make dep
// Create the image
    # make bzImage
// Compile modules
    # make modules
    # make modules_install
// Copy the kernel images
    # cp ./arch/i386/boot/bzImage /boot/vmlinuz-2.6.22.19
// Generate modules
    # depmod -a
// Update the initramfs image
    # update-initramfs -c -k 2.6.22.19
// Update grub
    # update-grub
// Reboot
    # reboot
```

Verify the kernel and GCC versions again:

```
#cat /etc/issue
    Ubuntu 8.04.3 LTS \n \l
#cat /proc/version
```

```
Linux version 2.6.22.19 (root@julio-desktop) (gcc version 4.2.4 (Ubuntu 4.2.4-1ubuntu4))
```

3.3.1.2 Compiling and installing MACRT

Before compiling and installing MACRT, we have to copy the Madwifi source code into the */home/Test/* folder.

```
// Copy macrt folder to /home/Test/
    # cp macrt /home/Test/
```

Once we have the right kernel and GCC versions, and the macrt source code in */home/Test/*, the next step is to install the Madwifi driver and the MACRT routing protocol. For that, we have to run the following commands:

```
// Change directory to the /home/Test/ folder
    # cd /home/Test/
// Inside, go to kmacrt folder
    # cd macrt/kmacrt
// Compile and install kmacrt
    # make
    # make install
// If we get some errors about net folder, we have to create it.
    # mkdir /lib/modules/2.6.22.19/net
// Compile and install the modified Madwifi driver
    # cd macrt/MadWifi
    # make
    # make install
// Compile umacrt
    # cd macrt/umacrt
    # make
// If we get some errors, we have to install the next package.
    # apt-get install libpcap-dev
```

So far, we have installed MACRT routing protocol onto the mesh router. The next step is to run some scripts in order to get the MACRT routing protocol to work.

```
// Export the path where the source code is located
    # export PATH=/home/Test/macrt/umacrt/:$PATH
// Execute the mesh_ap script
    # ./mesh_ap restart
// Set the ESSID to both interfaces
    # iwconfig ath0 essid "mesh"
```

```
julio-laptop ~# ifconfig ath3
ath3      Link encap:Ethernet  HWaddr            :d9:62
          inet addr:10.1.6.1  Bcast:10.1.255.255  Mask:255.255.0.0
          inet6 addr: fe80::21b:2fff:feba:d962/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:29488 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4867 errors:0 dropped:2677 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:899501 (878.4 KB)  TX bytes:535385 (522.8 KB)

julio-laptop ~# ifconfig mesh1
mesh1     Link encap:Ethernet  HWaddr            :d9:62
          inet6 addr: fe80::41b:2fff:feba:d962/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:94199 errors:0 dropped:0 overruns:0 frame:0
          TX packets:47766 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:13844872 (13.2 MB)  TX bytes:6652271 (6.3 MB)

julio-laptop ~# ifconfig wifi0
wifi0     Link encap:UNSPEC  HWaddr            -D9-62-E0-E3-00-00-00-00-00-00-00-00
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:197277 errors:0 dropped:0 overruns:0 frame:263130
          TX packets:102754 errors:13022 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:199
          RX bytes:19844100 (18.9 MB)  TX bytes:12613915 (12.0 MB)
          Interrupt:18
```

Figure 3.3: Running ifconfig command.

```
    # iwconfig mesh1 essid "suba"

// Set the IP address to the ath0 network interface with the desired IP address

    # ifconfig ath0 10.1.7.1 netmask 255.255.0.0 broadcast 10.1.255.255 up
```

To check if the mesh router is configured correctly, we need to run ifconfig and iwconfig commands as shown in Figure 3.3 and Figure 3.4. ifconfig command shows the mesh router's IP address and iwconfig shows some parameters such as the ESSID, the frequency, the mode, and the bit rate, among others.

3.3.2 MESH CLIENTS

A mesh client should always have at least one wireless interface to associate with a mesh router. There are two ways to configure the mesh clients. We can assign them an IP address by DHCP, or we can do it by setting static IP addresses. To Assign an static IP address on Windows XP/Vista/7 go to *Start - Control Panel - Network and Internet - Network and Sharing Center*, click on *Wireless Network Connection*. It will appear the *Wireless Network Connection Status* form. Click on *Properties button*, then select *Internet Protocol Version 4 (TCP/IPv4)* option as shown in Figure 3.5. Click on *Properties button* and type *the IP address* and *Subnet mask* respectively. Finally, we have to associate the mesh client to the mesh SSID.

```
julio-laptop ~# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wifi0     no wireless extensions.

ath3      IEEE 802.11g  ESSID:"mesh"  Nickname:""
          Mode:Master  Frequency:2.457 GHz  Access Point:            :D9:62
          Bit Rate:0 kb/s   Tx-Power:18 dBm   Sensitivity=1/1
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=39/70  Signal level=-57 dBm  Noise level=-96 dBm
          Rx invalid nwid:23204  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

mesh1     IEEE 802.11g  ESSID:"suba"  Nickname:""
          Mode:Repeater  Frequency:2.457 GHz  Access Point:            :D9:62
          Bit Rate=11 Mb/s   Tx-Power=0 dBm   Sensitivity=1/1
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=39/70  Signal level=-57 dBm  Noise level=-96 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Figure 3.4: Running iwconfig command.

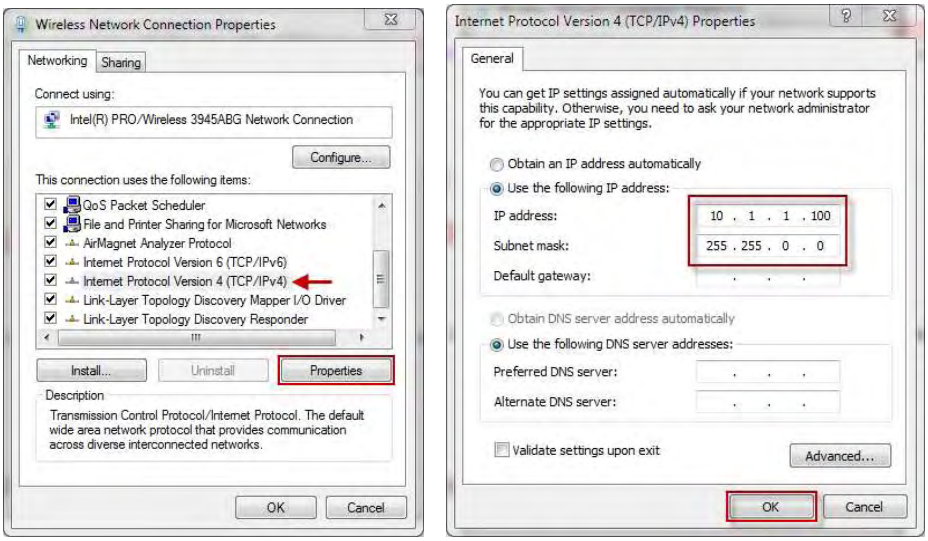
To Assign a static IP address on Linux use the next command:

```
# ifconfig ath0 10.1.70.1 netmask 255.255.0.0 broadcast 10.1.255.255 up
```

3.4 DEPLOYMENT

The five mesh routers were configured successfully in the previous section. The CRL laboratory of the Department of Electrical and Computer Engineering at UPRRUM was chosen as the place to deploy the WMN. Mesh routers are marked in red color and mesh clients in blue color, as shown in Figure 3.6.

Once we have all the mesh routers in their right places. We can see the routing table from an SSH connection. We ran the command */proc/net/kmacrt/rt_table* from MR-1. This routing table shows the mesh routers and mesh clients that are connected to the WMN. This routing table also keeps information such as: IP address, MAC address, next hop IP address, number of hops needed to reach destination, state, and network interface per each device, as shown in Figure 3.7.



(a) (b)

Figure 3.5: Setting a static IP address.

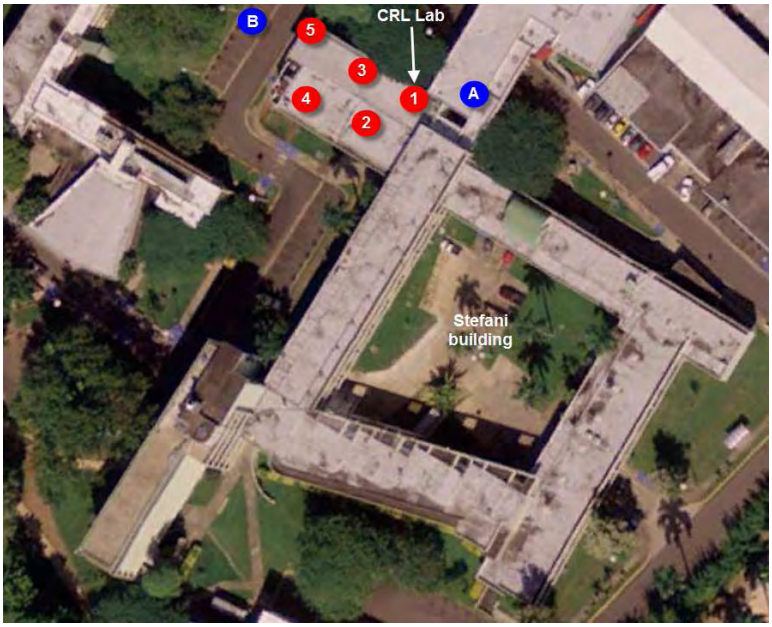


Figure 3.6: The WMN deployment at the CRL Lab.

```
julio-laptop /home/fwsl# cat /proc/net/kmacrt/rt_table
IP address       HW address        Next hop          Type   State   Device  Pktlen  Ndrop
0.0.0.0          00:03:2a:02:16:03 06:40:96:b6:81:dd 2      1       mesh1   0       0
0.0.0.0          06:14:6c:53:45:af 06:14:6c:53:45:af 0      1       mesh1   0       0
10.1.9.1         00:40:96:b6:81:10 06:1b:2f:ba:d9:62 2      1       mesh1   0       0
10.1.1.200       00:09:2d:55:a8:3f 00:09:2d:55:a8:3f 1      1       ath0    0       0
10.1.8.1         00:40:96:b6:81:dd 06:40:96:b6:81:dd 2      1       mesh1   0       0
10.1.7.1         00:1b:2f:ba:d8:12 06:1b:2f:ba:d8:12 2      1       mesh1   0       0
10.1.6.1         00:1b:2f:ba:d9:62 06:1b:2f:ba:d9:62 2      1       mesh1   0       0
10.1.5.1         00:14:6c:53:45:af 00:14:6c:53:45:af 0      1       ath0    0       0
10.1.1.100       00:1f:3b:d2:f0:77 00:1f:3b:d2:f0:77 1      1       ath0    0       0
```

Figure 3.7: The routing table from the MeshRouter01.

CHAPTER 4
WMN TESTBED BASED ON ALIX 2D2 BOARDS

This chapter discusses in detail the WMN testbed based on Alix 2D2 boards using the MACRT routing protocol. It is subdivided in four sections: WMN testbed overview, hardware and software, configuration and deployment.

4.1 WMN TESTBED OVERVIEW

This WMN testbed based on Alix 2D2 boards was inspired by [1]. Figure 4.1 depicts the configuration used for this testbed. We used three Alix 2D2 boards as the mesh routers, two Dell Latitude D620 laptops and a Pocket PC Dell Axim X51v as the mesh clients. Each mesh router has Debian Linux v5.0 installed, kernel v2.6.22.19, GCC v4.2 and the MACRT routing protocol. The two mesh clients are computers running Windows XP and Windows 7, and the Pocket PC is running Windows Mobile v5.0 as its operating system. Mesh clients must have a wireless interface based on IEEE 802.11abg to connect to the network. Once we have the mesh routers and mesh clients completely configured, the next step is to look for a place where the network will be deployed. For this, we chose the second floor of the Stefani building of the Department of Electrical and Computer Engineering at the UPR-RUM. We put the mesh routers and mesh clients at strategic points. Then, we did some experiments to check the network connectivity using ping commands and software such as Wireshark, Jperf, SSH Secure Shell, Putty, and others. Once

we have the network working, then it is possible to deploy it in the field. In this case, we will deploy this WMN in the VESO-MESH Project.

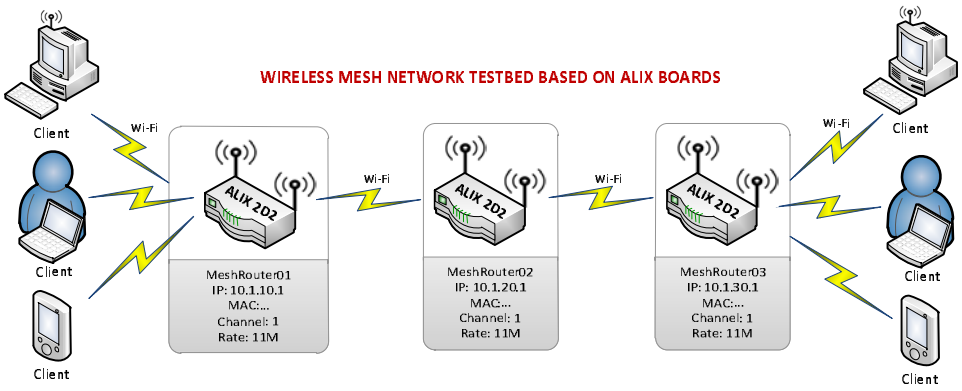


Figure 4.1: WMN testbed topology based on Alix 2D2 boards.

4.2 HARDWARE AND SOFTWARE

4.2.1 HARDWARE USED IN THE TESTBED

4.2.1.1 Mesh Routers

We chose Alix 2D2 boards because they offered a good combination of performance, low cost, and support for multiple devices based on USB interface. They are based on x86 architecture, thus there is no need to use cross-compiling.

The WMN testbed based on Alix 2D2 boards is built using commercial off-the-shelf (COTS) hardware. Alix 2D2 board is used as the main device to connect all the other devices. It consists of an AMD Geode 500 MHz (LX800) processor with 256 MB of RAM and 4 GB Compact Flash (CF) as its hard drive. It is equipped with two 100BaseT Ethernet interfaces, dual USB ports, two miniPCI slots, and a serial port. We put a Wistron miniPCI CM9 IEEE 802.11abg card, that is based on Atheros chipset inside the miniPCI slot. This miniPCI card is connected via a RP-SMA pigtail cable to a 5.5 dBi Rubber Duck Omnidirectional antenna with RP-SMA connector as shown in Figure 4.2 and Table 4.1.

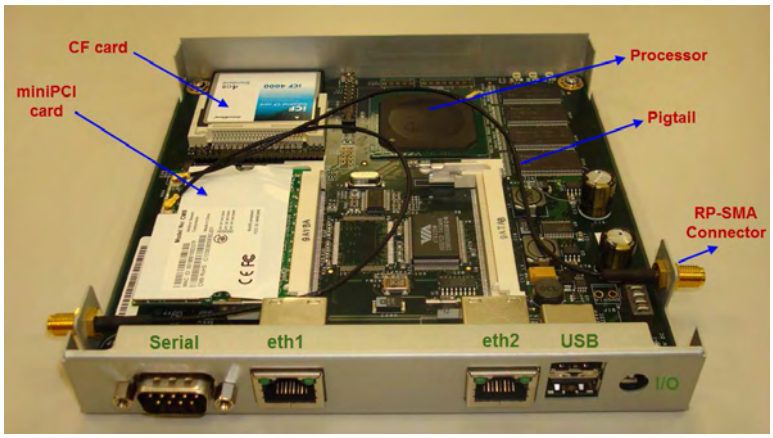


Figure 4.2: Hardware components of the Alix 2D2 board.

Specification	Alix 2D2 board
CPU	500 MHz.
USB	Dual USB port
DRAM	256 MB DDR DRAM
Storage	Compact Flash
Power	DC jack or passive POE
Expansion	2 miniPCI slots
Connectivity	2 Ethernet interfaces 10/100
I/O	DB9 serial Port
Board Size	152x152 mm
Firmware	Award tinyBIOS

Table 4.1: Hardware specifications of the Mesh Router.

4.2.1.2 Mesh Clients

Table 4.2 shows the mesh client specifications which we used for our experiments.

4.2.2 SOFTWARE USED IN THE TESTBED

Basically, this testbed runs Debian Linux v5.0, Microsoft Windows XP, and Microsoft Windows 7 as operating systems. Alix boards use the kernel v2.6.22.19 and the GCC v4.2. Additionally, they have a modified Madwifi v0.9.3 installed as the wireless driver and the MACRT routing protocol.

Specification	Laptop	Pocket PC
Brand	Dell Latitude D620	Dell Axim X51v
Processor	Intel Core 2 Duo 2.16 GHz.	Intel Xscale 624 MHz
Memory	3.25 GB.	256 MB. Flash
Hard Drive Capacity	320 GB.	Memory SD - 4 GB.
Wireless Connectivity	IEEE 802.11abg	IEEE 802.11b
Operating System	Windows and Linux	Windows Mobile 5.0

Table 4.2: hardware and software specifications of the Mesh Clients.

We used many Linux commands and third party softwares, such as Wireshark v1.2.4, Jperf v2.0.2, SSH Secure Shell v3.2.9, Putty v0.6, minicom v2.3, wicd v1.6.2, Xming v6.9, and VNC viewer v4.1.3, among others.

4.3 CONFIGURATION

4.3.1 MESH ROUTER CONFIGURATION

A mesh router configuration depends on the laptop and Alix configuration. Figure 4.3 depicts the flow chart configuration to achieve a mesh router configured. In the following sections, we will explain each of these steps.

4.3.1.1 Laptop configuration overview

The first step is to install an operating system onto our laptop computer. We chose Debian Linux v5.0 OS. Then, we have to install many others packages that will be useful in the installation of the kernel, GCC and MACRT. Afterwards, we will create a .deb kernel v2.6.22.19 package. This process will take almost 3 hours to be created onto the Dell Latitude D620 laptop. This package will be used to install the kernel v2.6.22.19 onto the laptop. Then, it is necessary to install GCC v4.2. The kernel v2.6.22.19 and GCC v4.2 are necessary because MACRT was created based on those versions. Additionally, the .deb package previously created will be used for the Alix configuration later on. The next step is to install the MACRT routing protocol. That means compiling and installing kmacrt, madwifi, and umacrt components. Each compilation will be done specifying GCC v4.2. After the MACRT

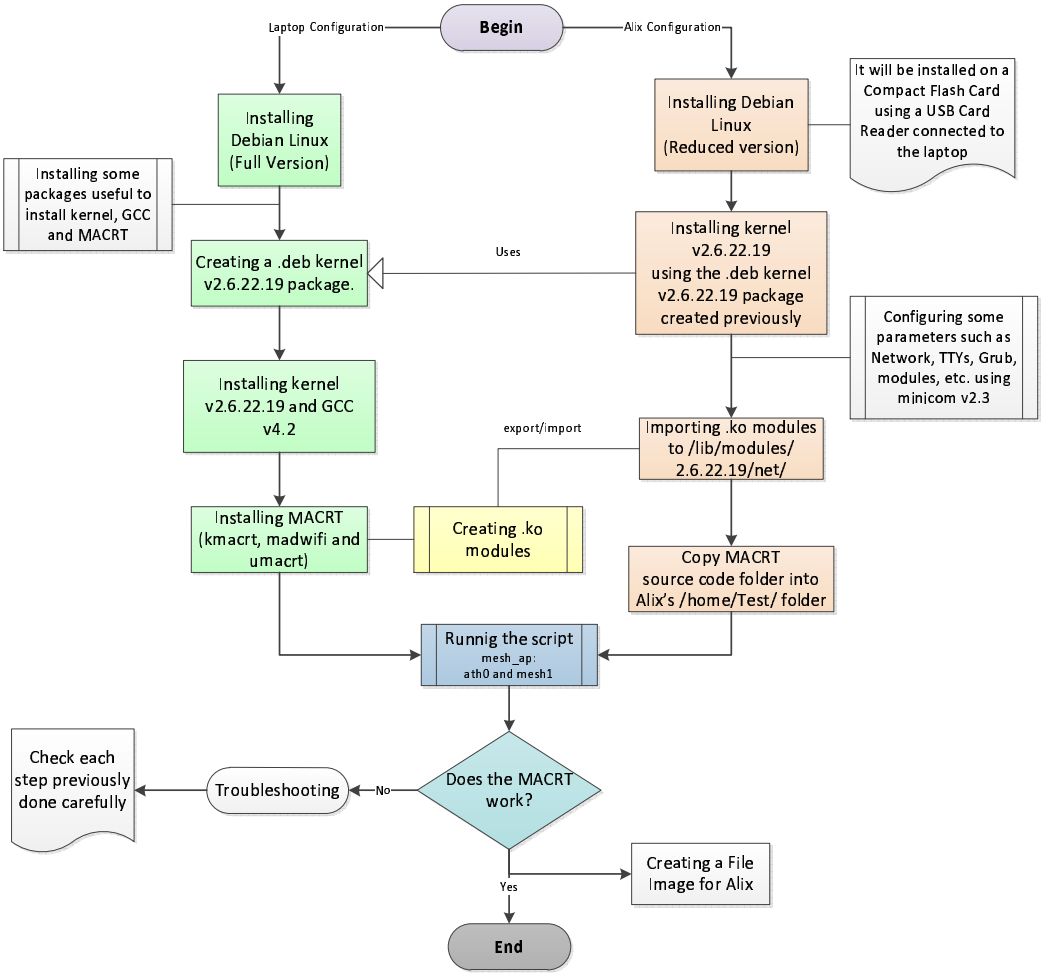


Figure 4.3: Mesh router flow chart configuration for Alix 2D2 boards.

routing protocol is installed, it will create some .ko modules that afterwards will be exported to the Alix configuration. Finally, we will run the script mesh_ap. The script will create two interfaces ath0 and mesh1 respectively, as shown in Figure 4.4. Ath0 will have an IP with the format 10.1.X.Y where X could be any value from 1 to 255, and Y from 2 to 254. That is because 1 and 255 are used for network identification and broadcast respectively. If those interfaces are created, then the MACRT routing protocol will be installed successfully. If not, then troubleshoot, carefully checking all of the previously done steps.

```
ath0      IEEE 802.11g  ESSID:"mesh"  Nickname:""
          Mode:Master  Frequency:2.457 GHz  Access Point:
          Bit Rate:0 kb/s   Tx-Power:20 dBm   Sensitivity=1/1
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=23/70  Signal level=-73 dBm  Noise level=-96 dBm
          Rx invalid nwid:61386  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0

mesh1     IEEE 802.11g  ESSID:"suba"  Nickname:""
          Mode:Repeater  Frequency:2.457 GHz  Access Point:
          Bit Rate=11 Mb/s   Tx-Power=0 dBm   Sensitivity=1/1
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=23/70  Signal level=-73 dBm  Noise level=-96 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Figure 4.4: ath0 and mesh1 interfaces.

4.3.1.1.1 Creating a .deb kernel package. The first step is to choose a Linux distribution to install in our laptop. It will depend on some factors, such as familiarity, compatibility, reliability, performance, etc. In the beginning, we chose Ubuntu Linux v9.10 OS because we were more familiar with it. Unfortunately, after doing some configurations and tests creating the .deb kernel package, we faced some setbacks. We tried to overcome them, but it was unsuccessful. For that reason, we decided to choose another Linux distribution. In this case, we chose and installed Debian Linux v5.0 onto our laptop.

Creating a .deb kernel package onto our laptop is not an easy task. For that, we need to do the following instructions:

Once we are logged onto the laptop computer, we have to do the following steps as superuser, running sudo command.

```
// It is important to know which kernel and ggc versions are running on our computer,
// as they will be used later on.
    # cat /proc/version
        Linux version 2.6.26-2-686 - gcc version 4.2.4
// Update the system running the apt package utility.
    # apt-get update
// Install some useful packages
    # apt-get install gcc-4.2 build-essential kernel-package libncurses5-dev bzip2 zlib1g-dev
```

```
// Go to /usr/src/ folder and install linux-source for the current linux version, in this case 2.6.26
    # apt-get install linux-source-2.6.26
// We need to do a .deb package of the kernel v2.6.22.19; therefore, download it from the Internet.
// We used the wget free utility to download files from the Web.
    # wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.22.19.tar.bz2
// Extract files from the linux-2.6.22.19.tar.bz2 file
    # tar xvjf linux-2.6.22.19.tar.bz2
// Change directory to the folder linux-2.6.22.19
    # cd linux-2.6.22.19
// Copy the boot file configuration, where the uname -r command will retrieve the current kernel version
    # cp /boot/config-$(uname -r) ./.config
// Compile the linux-2.6.22.19 configuration with GCC v4.2.
    # make CC=gcc-4.2 menuconfig
```

Figure 4.5 depicts the Linux kernel v2.6.22.19 configuration. From the figure, we can personalize, save, and compile the Linux kernel.

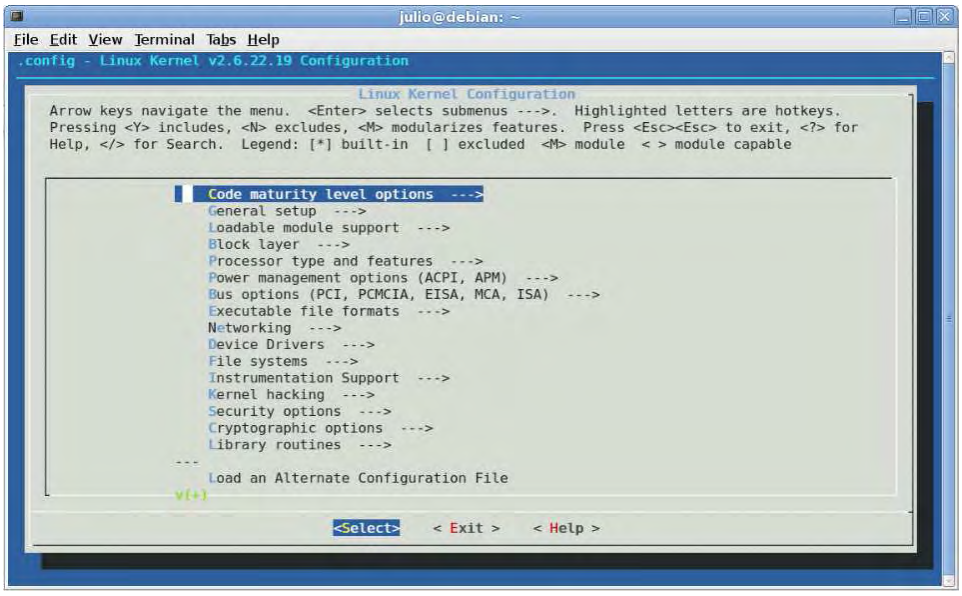


Figure 4.5: Linux kernel v2.6.22.19 configuration.

```
// Clean the kernel source directory
    # make-kpkg clean
// Set CFLAGS_KERNEL = -fno-tree-scev-cprop to avoid compiling problems in Makefile file.
    # nano Makefile
// Create the .deb kernel and headers packages with the make-kpkg command
    #make-kpkg --initrd --revision=Debian kernel_image
```

This process took around 3 hours on the Dell Latitude D620 Laptop. The time we spend on it will rely on all the options that we have chosen in Figure 4.5. Finally, the linux-image-2.6.22.19_Debian_i386.deb kernel package that we will use later will be created.

4.3.1.1.2 Installing kernel 2.6.22.19. Once we have created the *.deb linux-image-2.6.22.19_Debian_i386.deb* package, we have to install it onto our laptop. To do this, run the following command as superuser:

```
# dpkg -i linux-image-2.6.22.19_Debian_i386.deb
```

Then, check if the grub was modified with the new entry 2.6.22.19 kernel version, and restart the laptop computer with the reboot command.

```
# reboot
```

After the reboot, we will see the new option when the Laptop computer is initializing. We will see the entry Debian GNU/Linux, kernel 2.6.22.19. That represents the new kernel installed previously.

4.3.1.1.3 Installing MACRT onto the laptop. The MACRT routing protocol has three principal components: kmacrt, madwifi driver and umacrt. Each of them was compiled and installed as follows: open a new terminal and put it as superuser. Check the current kernel and GCC versions:

```
# cat /proc/version
    Linux version 2.6.22.19 (root@debian) (gcc version 4.2)
```

Then, we have to change the directory where the MACRT protocol is located. Figure 4.6 depicts the folder location, and the MACRT routing protocol source code.

The next step is installing the three components in the following order: kmarct, madwifi and umacrt. It is recommendable to execute 'make clean' before compiling a file. Additionally, we have to specify the use of GCC v4.2 for each compilation.

First, compile and install kmacrt as shown in Figure 4.7. The warnings do not affect the process at all. Second, compile and install the madwifi driver v0.9.3. Figure 4.8 and Figure 4.9 depict an excerpt of the compilation and installation

```
debian /home/Test/macrt/umacrt# make CC=gcc-4.2
gcc-4.2 -c -I.. -Wall  -g -O0 -DUMACRT_DEBUG  messages.c -o messages.o
gcc-4.2 -M -I.. -Wall  -g -O0 -DUMACRT_DEBUG  messages.c > messages.d
gcc-4.2 -c -I.. -Wall  -g -O0 -DUMACRT_DEBUG  next_hop.c -o next_hop.o
gcc-4.2 -M -I.. -Wall  -g -O0 -DUMACRT_DEBUG  next_hop.c > next_hop.d
gcc-4.2 -c -I.. -Wall  -g -O0 -DUMACRT_DEBUG  timer_queue.c -o timer_queue.o
gcc-4.2 -M -I.. -Wall  -g -O0 -DUMACRT_DEBUG  timer_queue.c > timer_queue.d
gcc-4.2 -c -I.. -Wall  -g -O0 -DUMACRT_DEBUG  main.c -o main.o
gcc-4.2 -M -I.. -Wall  -g -O0 -DUMACRT_DEBUG  main.c > main.d
gcc-4.2 -c -I.. -Wall  -g -O0 -DUMACRT_DEBUG  raw_socket.c -o raw_socket.o
gcc-4.2 -M -I.. -Wall  -g -O0 -DUMACRT_DEBUG  raw_socket.c > raw_socket.d
gcc-4.2 -c -I.. -Wall  -g -O0 -DUMACRT_DEBUG  nl_socket.c -o nl_socket.o
```

Figure 4.6: MACRT routing protocol source code.

respectively. Finally, compile the umacrt module. Figure 4.10 depicts an excerpt of the compilation.

So far, we have installed MACRT routing protocol in the laptop computer using kernel v2.6.22.19 and GCC v4.2 respectively.

4.3.1.2 Alix configuration overview

Alix 2D2 board has neither a powerful processor nor a large memory capacity. For that reason, it is not convenient to install a complete Linux distribution on it. We need to install the minimal packages to get better performance. Before installing Debian Linux v5.0 OS onto the Alix 2D2 board, we need to connect the USB card reader, the CF card and the laptop computer appropriately. The USB Card reader will be connected to the USB interface of the laptop computer. The CF card will be inserted into the USB Card reader. Figure 4.11 depicts this configuration.

Installing Debian linux on an Alix 2D2 board is different from installing it on a laptop computer. Later on, we will describe this process in detail. Once we have the Debian linux installed on the Alix, the next step is to install the kernel v2.6.22.19 based on .deb kernel v2.6.22.19 package previously created. This step is really similar to doing it on a laptop computer. Afterwards, it is necessary to configure some parameters such as network, TTYs, grub, modules, and so forth. For this, we will use minicom v2.3, which performs administrative functions on devices that do not have a monitor or keyboard attached such as the Alix board. Then, we

```
debian /home/Test/macrt/kmacrt# make CC=gcc-4.2
make -C /lib/modules/2.6.22.19/build M=/home/Test/macrt/kmacrt modules
make[1]: Entering directory `/usr/src/linux-2.6.22.19'
  CC [M]  /home/Test/macrt/kmacrt/hook.o
  CC [M]  /home/Test/macrt/kmacrt/xmit.o
  CC [M]  /home/Test/macrt/kmacrt/rt_table.o
  CC [M]  /home/Test/macrt/kmacrt/nl.o
  CC [M]  /home/Test/macrt/kmacrt/filter.o
  LD [M]  /home/Test/macrt/kmacrt/kmacrt.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: "unregister_mesh_hook" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
WARNING: "register_mesh_tx_hook" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
WARNING: "register_tx_hook" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
WARNING: "unregister_event_hook" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
WARNING: "ap_dev_list" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
WARNING: "register_event_hook" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
WARNING: "mesh_dev_num" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
WARNING: "unregister_deliver_hook" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
WARNING: "unregister_tx_hook" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
WARNING: "unregister_mesh_tx_hook" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
WARNING: "ap_dev_num" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
WARNING: "mesh_dev_list" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
WARNING: "register_mesh_hook" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
WARNING: "register_deliver_hook" [/home/Test/macrt/kmacrt/kmacrt.ko] undefined!
  CC      /home/Test/macrt/kmacrt/kmacrt.mod.o
  LD [M]  /home/Test/macrt/kmacrt/kmacrt.ko
make[1]: Leaving directory `/usr/src/linux-2.6.22.19'
debian /home/Test/macrt/kmacrt#
debian /home/Test/macrt/kmacrt# make install
test -d /lib/modules/2.6.22.19 || exit 1
cp -f kmacrt.ko /lib/modules/2.6.22.19/net/
```

Figure 4.7: Compilation and installation of the kmacrt module.

will import .ko modules from the laptop configuration done previously, and copy them to the */lib/modules/2.6.22.19/net/* Alix folder. This folder will be used for the MACRT routing protocol later on. Finally, run the script mesh_ap to create the two interfaces ath0 and mesh1. This last step is very similar to the laptop's configuration done in the previous section.

4.3.1.2.1 Installing Debian Linux. Alix's web page suggests choosing FreeBSD, pfSense or several commercial operating systems, including Linux. We installed Debian Linux v5.0 because it is the closest to Ubuntu configuration. Set the devices as Figure 4.11 and follow the next instructions as superuser:

```
// Add some useful repositories
    nano /etc/apt/sources.list
        deb http://http.us.debian.org/debian stable main contrib non-free
        deb http://security.debian.org stable/updates main contrib non-free
// Update and install some useful packages
```

```
debian /home/Test/macrt/madwifi# make CC=gcc-4.2
Checking requirements... ok.
Checking kernel configuration... ok.
/bin/sh: line 1: svnversion: command not found
make -C /lib/modules/2.6.22.19/build SUBDIRS=/home/Test/macrt/madwifi modules
make[1]: Entering directory `/usr/src/linux-2.6.22.19'
  CC [M]  /home/Test/macrt/madwifi/ath/if_ath.o
  CC [M]  /home/Test/macrt/madwifi/ath/if_ath_pci.o
  LD [M]  /home/Test/macrt/madwifi/ath/ath_pci.o
  CC [M]  /home/Test/macrt/madwifi/ath_hal/ah_os.o
  HOSTCC  /home/Test/macrt/madwifi/ath_hal/uudecode
  UUDECODE /home/Test/macrt/madwifi/ath_hal/i386-elf.hal.o
  LD [M]  /home/Test/macrt/madwifi/ath_hal/ath_hal.o
  CC [M]  /home/Test/macrt/madwifi/ath_rate/amrr/amrr.o
  LD [M]  /home/Test/macrt/madwifi/ath_rate/amrr/ath_rate_amrr.o
  CC [M]  /home/Test/macrt/madwifi/ath_rate/onoe/onoe.o
  LD [M]  /home/Test/macrt/madwifi/ath_rate/onoe/ath_rate_onoe.o
  CC [M]  /home/Test/macrt/madwifi/ath_rate/sample/sample.o
  LD [M]  /home/Test/macrt/madwifi/ath_rate/sample/ath_rate_sample.o
  CC [M]  /home/Test/macrt/madwifi/net80211/if_media.o
  CC [M]  /home/Test/macrt/madwifi/net80211/ieee80211.o
  CC [M]  /home/Test/macrt/madwifi/net80211/ieee80211_beacon.o
```

Figure 4.8: Compilation excerpt of the madwifi wireless driver v0.9.3.

```
    apt-get update
    apt-get install mbr debootstrap binutils minicom
// Check if the CF card named as /dev/sdc1/ was mounted
    mount
// If so, dismount it
    umount /dev/sdc1
// Create the partition table
    fdisk /dev/sdc
```

Figure 4.12 depicts all the menu options. Then, press 'd' to delete the partition. After that, press 'n' to add a new partition. Then type 'p' and '1' for the primary and first partition options respectively. Press the 'Enter' key to set the default values for the first and last cylinder options respectively. Then, set the bootable flag pressing 'a' and type '1' for the first partition. Press 'p' to print the partition table. Check if everything is set correctly. Finally, press 'w' to write the partition table to the disk and exit.

```
// Install a Master Boot Record into the CF card
    install-mbr /dev/sdc
// Create an ext2 Linux file system for the new partition
    mkfs.ext2 /dev/sdc1
// Create a new /mnt/cf folder
```

```
debian /home/Test/macrt/madwifi# make install
sh scripts/find-madwifi-modules.sh 2.6.22.19

WARNING:
It seems that there are modules left from previous MadWifi installations.
If you are unistalling the MadWifi modules please press "r" to remove them.
If you are installing new MadWifi modules, you should consider removing those
already installed, or else you may experience problems during operation.
Remove old modules?

[l]ist, [r]emove, [i]gnore or e[x]it (l,r,i,[x]) ?
r
for i in ./ath ./ath_hal ./ath_rate ./net80211; do \
                make -C $i install || exit 1; \
        done
make[1]: Entering directory `/home/Test/macrt/madwifi/ath'
test -d //lib/modules/2.6.22.19/net || mkdir -p //lib/modules/2.6.22.19/net
install ath_pci.ko //lib/modules/2.6.22.19/net
make[1]: Leaving directory `/home/Test/macrt/madwifi/ath'
make[1]: Entering directory `/home/Test/macrt/madwifi/ath_hal'
test -d //lib/modules/2.6.22.19/net || mkdir -p //lib/modules/2.6.22.19/net
install ath_hal.ko //lib/modules/2.6.22.19/net
make[1]: Leaving directory `/home/Test/macrt/madwifi/ath_hal'
make[1]: Entering directory `/home/Test/macrt/madwifi/ath_rate'
for i in amrr/ onoe/ sample/; do \
                make -C $i install || exit 1; \
        done
```

Figure 4.9: Installation excerpt of the madwifi wireless driver v0.9.3

```
    mkdir -p /mnt/cf
// Mount the CF card device into the /mnt/cf/ folder
    mount /dev/sdc1 /mnt/cf
// Check if it was mounted, if it has ext2 file system, and if it is marked as rw
    mount
// Bootstrap a basic Debian Linux system from the Internet and install it
    debootstrap --verbose --arch=i386 lenny /mnt/flash http://ftp.us.debian.org/debian
// Mount proc and sys systems into the chroot for communicating with the kernel
    mount --bind /proc /mnt/cf/proc
    mount --bind /sys /mnt/cf/sys
// Enter to the chroot environment. It represents the CF card system mounted.
    chroot /mnt/cf /bin/bash
// Inside chroot, we will install the .deb kernel v2.6.22.19 package created previously
// chroot section
// Add some useful repositories
    nano /etc/apt/sources.list
        deb http://http.us.debian.org/debian stable main contrib non-free
        deb http://security.debian.org stable/updates main contrib non-free
// Update and install some useful packages
    apt-get update
```

```
debian /home/Test/macrt/umacrt# make CC=gcc-4.2
gcc-4.2 -c -I.. -Wall  -g -O0 -DUMACRT_DEBUG  messages.c -o messages.o
gcc-4.2 -M -I.. -Wall  -g -O0 -DUMACRT_DEBUG  messages.c > messages.d
gcc-4.2 -c -I.. -Wall  -g -O0 -DUMACRT_DEBUG  next_hop.c -o next_hop.o
gcc-4.2 -M -I.. -Wall  -g -O0 -DUMACRT_DEBUG  next_hop.c > next_hop.d
gcc-4.2 -c -I.. -Wall  -g -O0 -DUMACRT_DEBUG  timer_queue.c -o timer_queue.o
gcc-4.2 -M -I.. -Wall  -g -O0 -DUMACRT_DEBUG  timer_queue.c > timer_queue.d
gcc-4.2 -c -I.. -Wall  -g -O0 -DUMACRT_DEBUG  main.c -o main.o
gcc-4.2 -M -I.. -Wall  -g -O0 -DUMACRT_DEBUG  main.c > main.d
gcc-4.2 -c -I.. -Wall  -g -O0 -DUMACRT_DEBUG  raw_socket.c -o raw_socket.o
gcc-4.2 -M -I.. -Wall  -g -O0 -DUMACRT_DEBUG  raw_socket.c > raw_socket.d
gcc-4.2 -c -I.. -Wall  -g -O0 -DUMACRT_DEBUG  nl_socket.c -o nl_socket.o
```

Figure 4.10: Compilation excerpt of the umacrt module.

Figure 4.11: Hardware used to configure the Alix 2D2 board.

```
    apt-get install openssh-server grub locales nano
// Copy the .deb kernel v2.6.22.19 package from the laptop to the /mnt/cf/usr/src/ folder
    cp /usr/src/*.deb /mnt/cf/usr/src/
// Change directory to the /usr/src/ and install the .deb kernel v2.6.22.19 package into
// the CF card
    cd /usr/src/
    dpkg -i *.deb
// Reconfigure locales
    dpkg-reconfigure locales
// Add some parameters to /etc/modules for loading kernel modules at boot time
    nano /etc/modules
        natsemi
        lm90
        w83627hf
```

```
debian ~# fdisk /dev/sdc

Command (m for help): m
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatibility flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   o   create a new empty DOS partition table
   p   print the partition table
   q   quit without saving changes
   s   create a new empty Sun disklabel
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)

Command (m for help):
```

Figure 4.12: fdisk Linux command menu options.

```
        scx200_acb base=0810,0820
        geodewdt
        leds-alix
// Create the /boot/grub folder
    mkdir /boot/grub
// Copy the entire grub configuration to /boot/grub/ folder
    cp /usr/lib/grub/i386-pc/* /boot/grub/
// Insert some parameters into the grub's configuration file, and enable the console on
// the serial port
    nano /boot/grub/menu.lst
        serial --unit=0 --speed=38400n8
        terminal --timeout=1 --silent serial
        timeout 1
        default 0
        title Debian-2.6.22.19
        root (hd0,0)
        kernel /vmlinuz root=/dev/hda1 console=ttyS0,38400n8
        initrd /initrd.img
// Edit /etc/inittab to enable console on serial port as follows:
    nano /etc/inittab
        #1:2345:respawn:/sbin/getty 38400 tty1
        #2:23:respawn:/sbin/getty 38400 tty2
```

```
        #3:23:respawn:/sbin/getty 38400 tty3
        #4:23:respawn:/sbin/getty 38400 tty4
        #5:23:respawn:/sbin/getty 38400 tty5
        #6:23:respawn:/sbin/getty 38400 tty6
        # Add at the final
        T0:23:respawn:/sbin/getty -L ttyS0 38400
// Configure the file system at boot time. Edit the /etc/fstab as follows:
    nano /etc/fstab
        proc /proc proc defaults 0 0
        /dev/hda1 / ext2 noatime,errors=remount-ro 0 1
        tmpfs /tmp tmpfs defaults,noatime 0 0
        tmpfs /var/tmp tmpfs defaults,noatime 0 0
        tmpfs /var/run tmpfs defaults 0 0
        tmpfs /var/log tmpfs defaults 0 0
        tmpfs /var/lock tmpfs defaults 0 0
// Configure the network interfaces
    nano /etc/network/interfaces
        auto lo
        iface lo inet loopback
        auto eth0
        iface eth0 inet dhcp
// Exit from chroot
    exit
// Install grub in the CF card
    grub-install --root-directory=/mnt/cf /dev/sdc
// Dismount the proc and sys systems from chroot
    umount /mnt/cf/proc/
    umount /mnt/cf/sys
// Dismount the CF card
    umount /mnt/cf/
```

4.3.1.2.2 Installing MACRT onto the Alix. Before installing the MACRT routing protocol, it is necessary to import all the .ko files previously built by the laptop configuration. To do this, create the script named copyKOModules, make it executable, and run it as follows:

```
    nano copyKOModules
        // Script to copy .ko files to the Alix
        #!/bin/sh
        # For every .ko file found into the /home/Test/macrt/ folder, copy it to /home/Test/net/
                for i in ‘find /home/Test/macrt/ -iname \*.ko‘ ; do
```

```
                cp $i /home/Test/net/
                done
            Press Ctrl+O to save it and then Ctrl+X to exit.
// Make it executable
    chmod 755 copyKOModules
// run the script
    ./copyKOModules
// Check if the modules were copied
    ls /home/Test/net/
// copy those files to the Alix /lib/modules/2.6.22.19/net/ folder
// In this case, we will copy them to the Alix board, which has the IP address 136.145.116.251
// The system will ask for the root's password, before copying the files.
    scp -r /home/Test/net/ root@136.145.116.251:~/
// The next step is to be inside the Alix 2D2 board
// We can do this in 2 ways: accessing it by SSH connection based on the Ethernet connection,
// or by minicom based on the serial cable.
// Once we are inside the Alix, move the net/ folder to /lib/modules/2.6.22.19/ folder
    mv net/ /lib/modules/2.6.22.19/
// Copy the /usr/src/linux-2.6.22.19/ folder to the Alix's /usr/src/ folder
    Laptop: scp -r /usr/src/linux-2.6.22.19/ root@136.145.116.251:~/
    Alix: mv /root/linux-2.6.22.19/ /usr/src/
// Copy the /home/Test/ folder to the Alix's /home/Test folder
    scp -r /home/Test/ root@136.145.116.251:~/
// Modify the paths of the scripts to run the MACRT routing protocol, specifying where each file
// is located.
    ./mesh_ap restart
```

Finally, we set up the Alix 2D2 boards to start the MACRT routing protocol as soon as they are turned on. That means, when the three Alix boards are turned on at strategic points, they will run the MACRT routing protocol and will form the WMN.

This testbed was done with three Alix 2D2 boards. If we want to add more mesh routers, we have created an image file to reproduce Alix 2D2 boards configuration more easily (See Appendix A to create and restore a CF card image backup onto the Alix 2D2 boards).

4.3.2 MESH CLIENTS CONFIGURATION

A mesh client should always have at least one wireless interface to associate with a mesh router. There are two ways to configure the mesh clients. We can assign them an IP address by DHCP, or we can do it by setting static IP Addresses. Figure 4.13 depicts that the mesh client is associated to the mesh SSID.

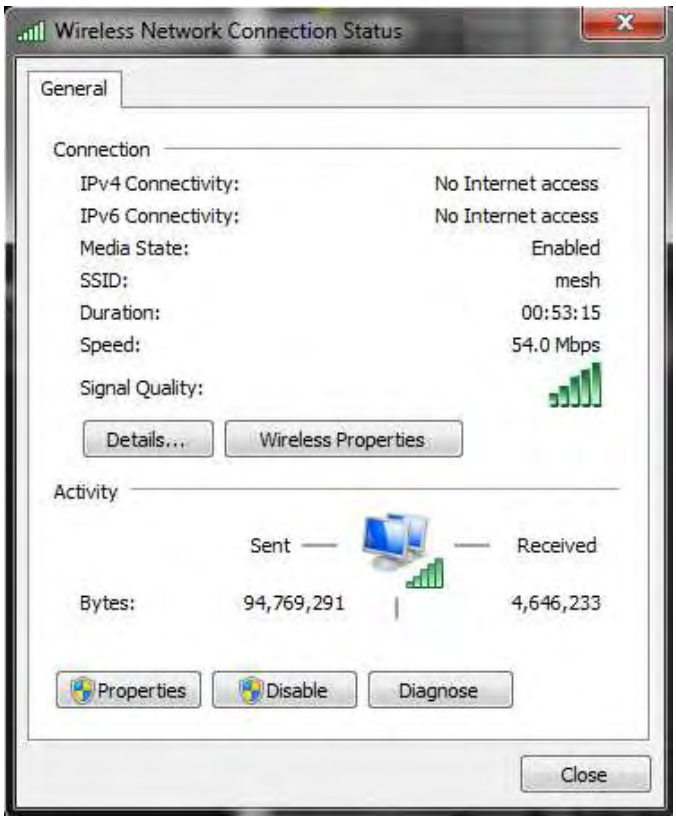


Figure 4.13: The mesh client is associated with the SSID called mesh.

On Windows XP/Vista/7 it is really intuitive to set a static IP address, but on Linux the next command will be used:

```
// Set a static IP Address.
    # ifconfig ath0 10.1.1.88 netmask 255.255.0.0 broadcast 10.1.255.255 up
```

4.4 DEPLOYMENT

The three Alix 2D2 boards were configured successfully in the previous section. The Stefani building of the Department of Electrical and Computer Engineering at

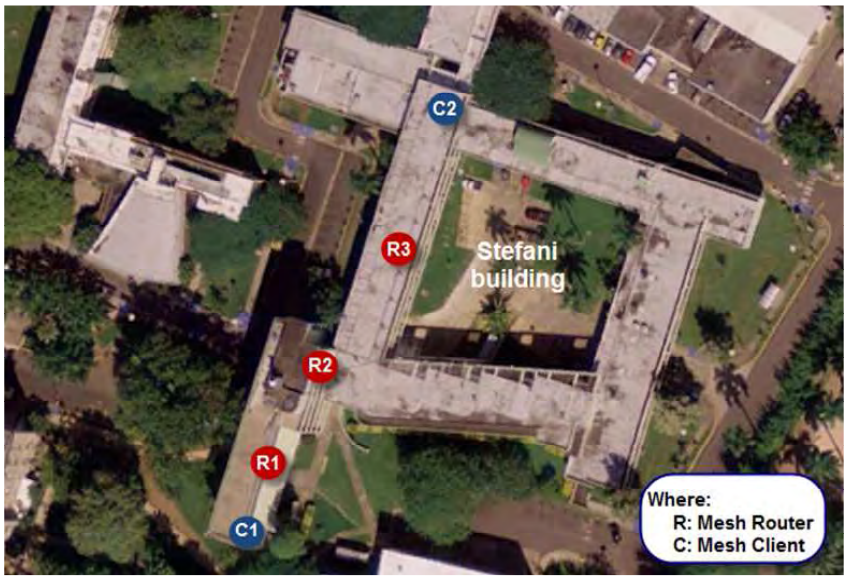


Figure 4.14: The WMN deployment at the Stefani building.

UPR-RUM was chosen as the place to deploy the WMN. The mesh routers and mesh clients were put at strategic points as shown in Figure 4.14 and Figure 4.15. Mesh routers are marked in red color and mesh clients in blue color.

MeshClient01 was connected to MeshRouter01 and Meshclient02 was connected to MeshRouter03. MeshRouter02 was turned off. When MeshClient02 was sending packages to MeshClient01, the packages were not received by MeshClient01. This was because MeshRouter03 could not see MeshRouter01 and vice versa. We turned on MeshRouter02, and the communication among the three routers was established. To check that the mesh was working, we did some tests using ping commands. Then, we captured some packets with Wireshark, and we also ran Jperf to send packets in a client/server environment. Finally, we ran some scripts to see the routing table that was created, the neighbors available for each mesh router, and the links created in the network.

After everything had been set up, we checked the network connectivity. First, we sent some pings to the three mesh routers and mesh client respectively. All the pings were received successfully as shown in Figure 4.16.

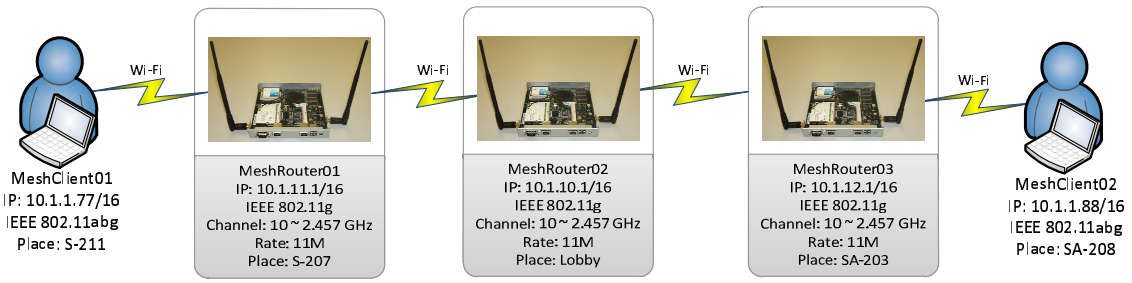


Figure 4.15: The WMN topology and parameters.

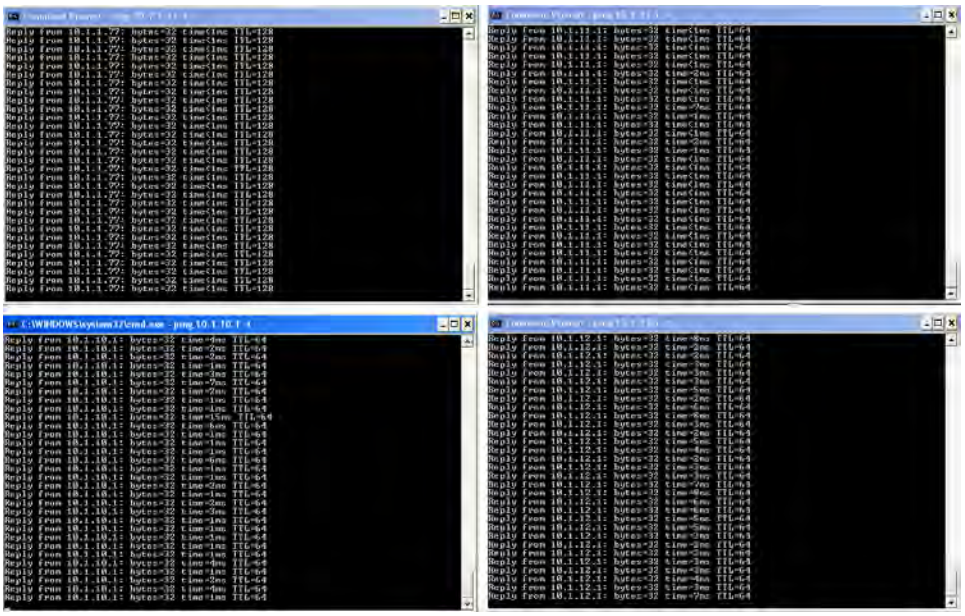


Figure 4.16: Checking the connectivity with ping command.

Second, we used Wireshark v1.2.4 to capture some packets of the mesh network. Here, we can see that the mesh routers and mesh clients are sending packets continuously as shown in Figure 4.17.

Third, we use Jperf v2.0.2 to send some packages in a client/server environment, from one mesh client to another. In this case, Client02 worked as a client and Client01 worked as a server. The client sent 1GB of packages to the server. The average transfer passing through the three routers was almost 0.7 Mbps as shown in Figure 4.18.

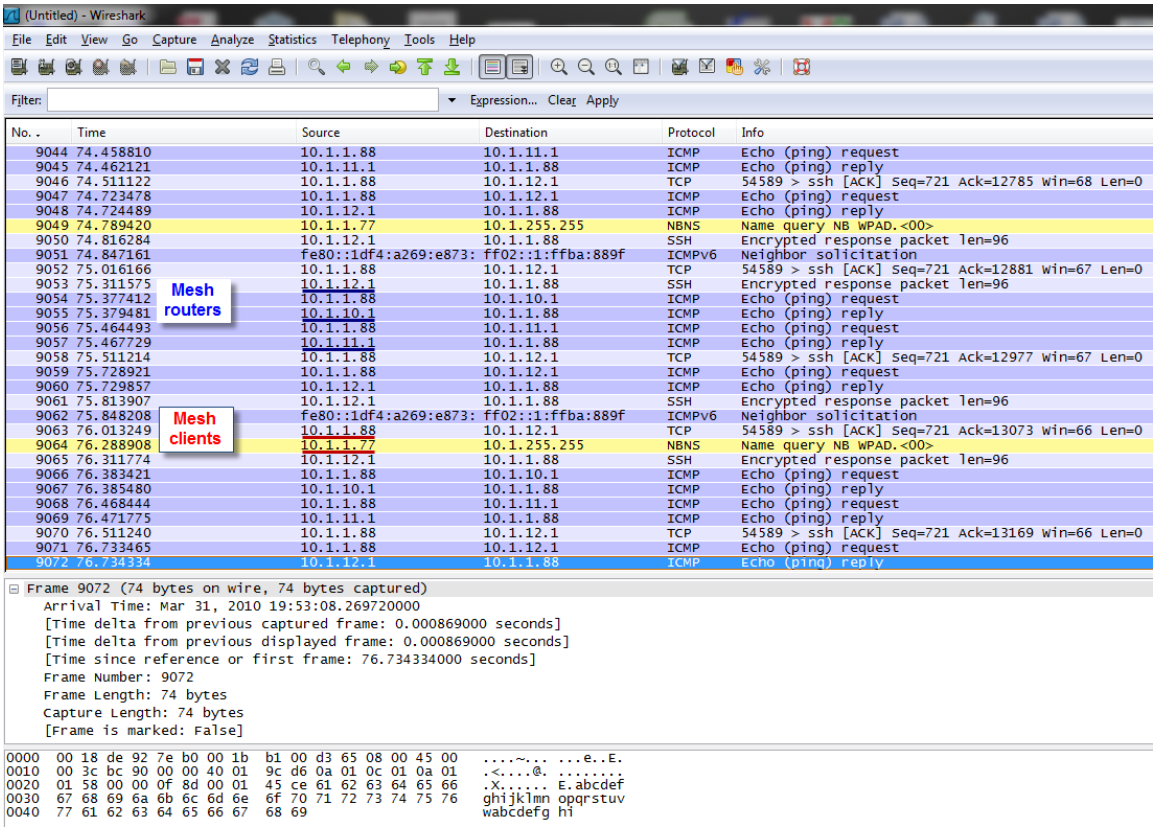


Figure 4.17: Running Wireshark to capture packets from the network.

Additionally, we ran some scripts, which shows the routing table created by each mesh router as shown in Figure 4.19 and Figure 4.20. This routing table shows the mesh routers and mesh clients that are connected to the WMN. This routing table keeps information such as: IP address, MAC address, next hop IP address, number of hops needed to reach destination, state, and network interface. Figure 4.21 depicts the available neighbors for each mesh router. Figure 4.22 depicts the MACRT log file, which shows the routes of the WMN.

Finally, we can conclude that the WMN testbed based on Alix 2D2 boards works successfully. The average data transmission sending 1GB of data was almost 0.7 Mbps. The next step is to deploy this WMN testbed in the Development of a Versatile Service-Oriented Wireless Mesh Network project (VESO-MESH).

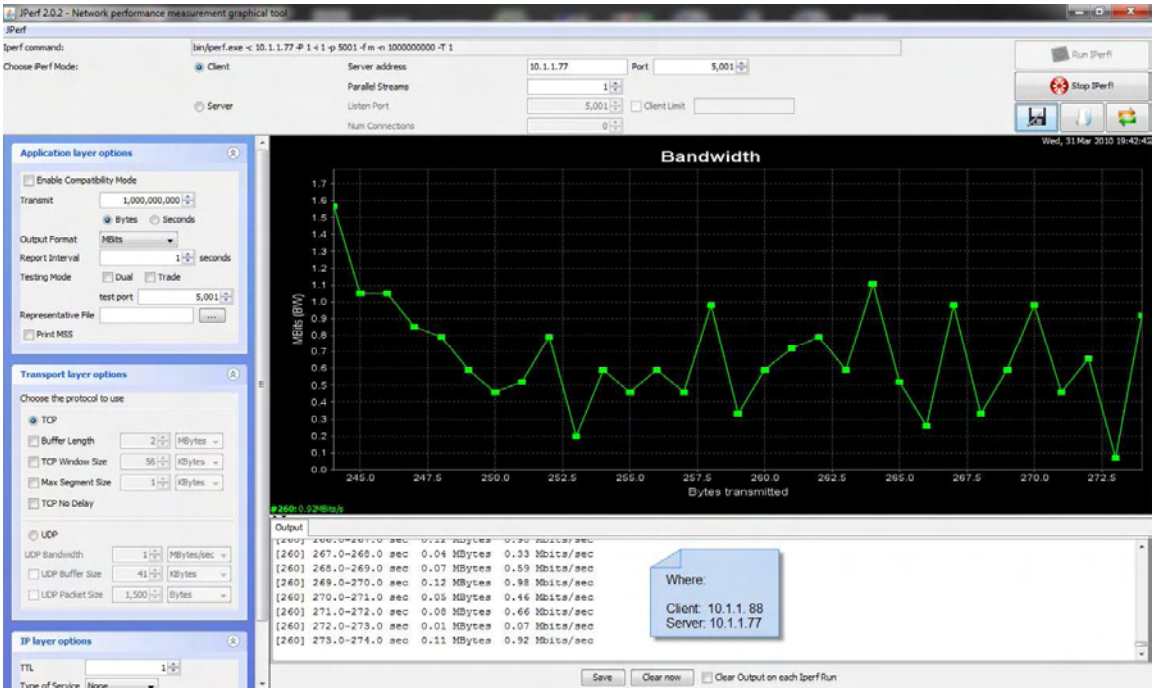


Figure 4.18: Sending packets with Jperf.

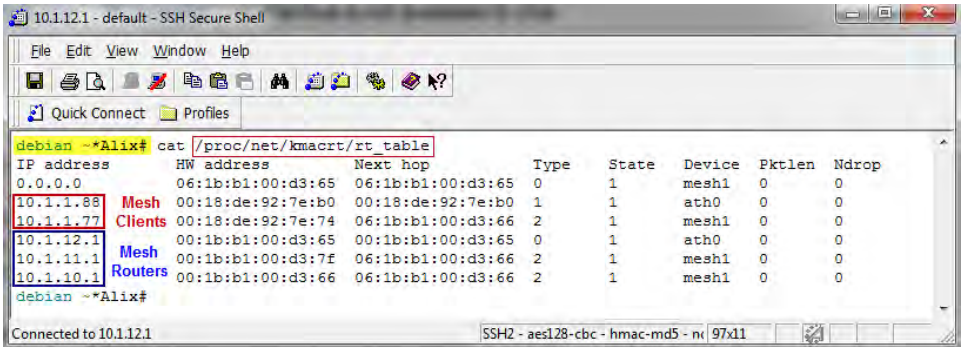


Figure 4.19: The routing table from the MeshRouter03 SSH interface.

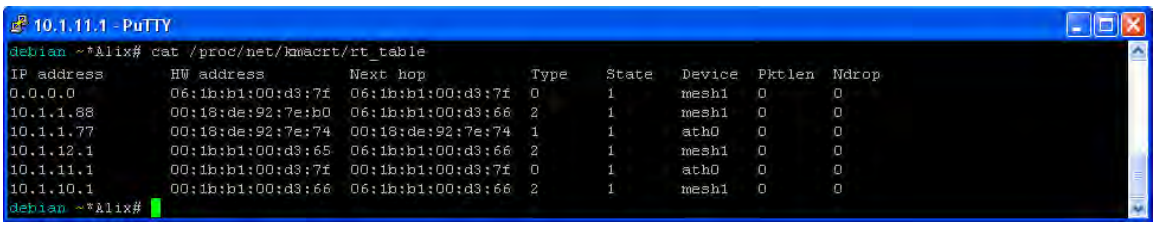


Figure 4.20: The routing table from the MeshRouter01 Putty interface.

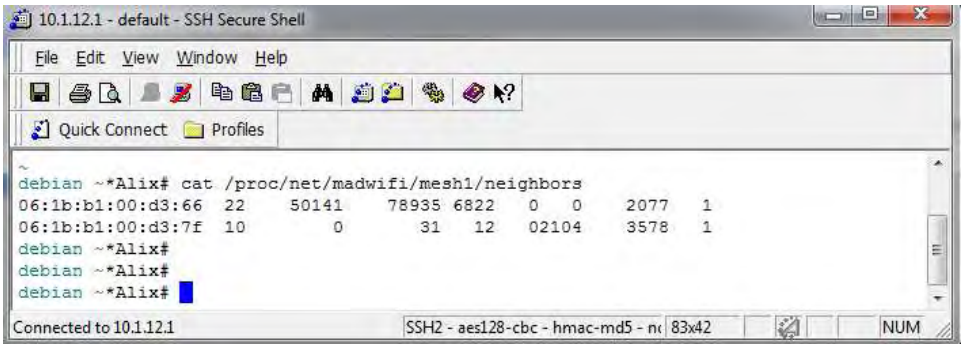


Figure 4.21: The available neighbors for the MeshRouter03.

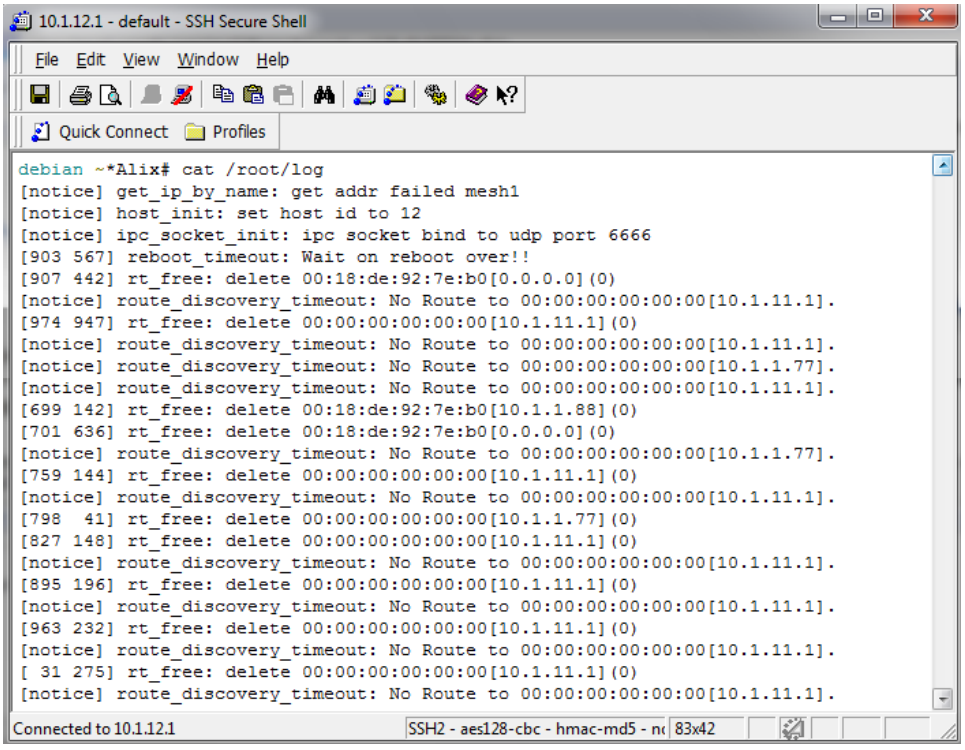


Figure 4.22: The MACRT log file.

CHAPTER 5
VESO-MESH TESTBED

This chapter presents the VESO-MESH testbed. This testbed has been designed, implemented and deployed by graduate and undergraduate students from the Department of Electrical and Computer Engineering at UPR-RUM.

The VESO-MESH Project [7] has as its main objective the Development of a Versatile Service-Oriented Wireless Mesh Network (WMN) that can be quickly built to respond to natural disasters and that establishes a data-intensive environmental monitoring application, specifically addressing hurricanes and earthquakes.

Jobos Bay is the second-largest estuarine area in Puerto Rico [8]. It is located in the more arid southeastern coast of Puerto Rico. This bay covers approximately 2,883 acres of the Jobos Bay ecosystem. It includes extensive networks of mangrove forests, upland dry forests, lagoons, seagrass beds and coral reefs. Furthermore, it is home to many kinds of species such as: the endangered brown pelican, the peregrine falcon, the hawksbill turtle and west indian manatee, among others. This estuarine is very important for marine recreation, commercial and recreational fishing, and ecotourism.

Our principal task here was to design, implement and deploy a WMN testbed at the Jobos Bay National Estuarine. This testbed will serve as the platform for the following on-going research topics that are being investigated at the Department of Electrical and Computer Engineering at the UPR-RUM:

- Time-Frequency Signal Representations for Environmental Surveillance Monitoring.

- Basic Testbed Infrastructure for Environmental Surveillance Monitoring.
- Bio-acustic Signal Analysis and Classification.
- Cyclic Short-Time Fourier Transform Algorithms for Environmental Surveillance Monitoring.
- iPhone-based Digital Streaming Data Exchange for Species Classification in a Wireless Sensor Network.

5.1 VESO-MESH V1.0

5.1.1 DESIGN OF THE TESTBED

Figure 5.1 depicts the design of the VESO-MESH v1.0 architecture. Table 5.1 shows the hardware and the operating systems used.

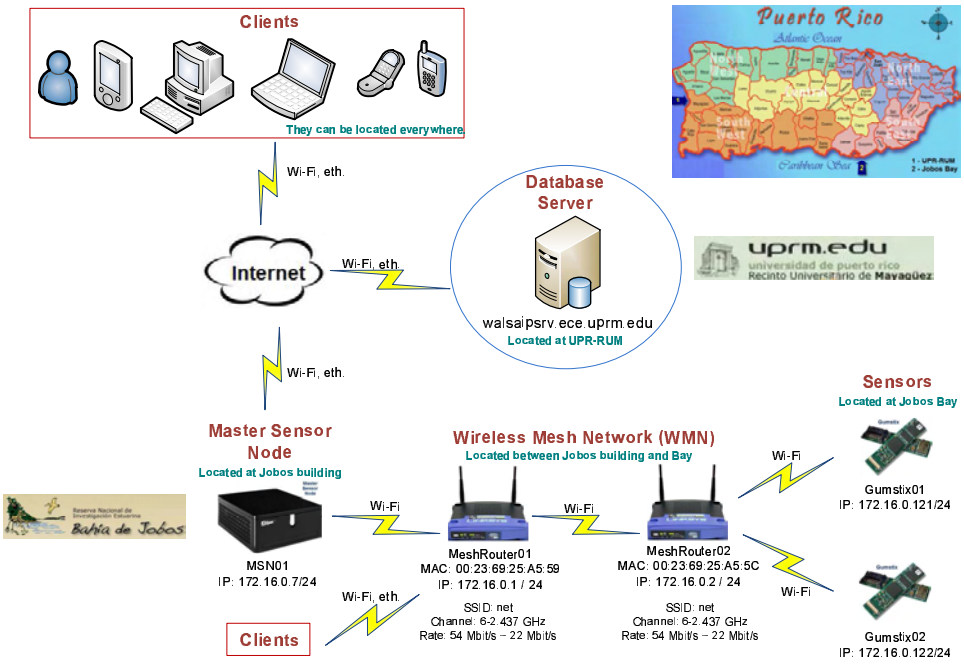


Figure 5.1: VESO-MESH v1.0 Architecture.

The two Linksys WRT54GL are working as mesh routers. These mesh routers have the Wireless Distribution System (WDS) configured, which permits a pseudo WMN to form. Each Gumstix board has Wi-Fi capability, microSD, USB, and ethernet slots. The Gumstix board has a microphone attached, which is used to

Hardware	Software
Linksys WRT54GL wireless routers	DD-WRT v24-sp2
Gumstix Boards	Based on Linux
AOpen computer (MSN)	Fedora Linux v10
DataBase Server	Ubuntu Linux v8.04
Pocket PC Dell Axim X51v	Windows Mobile v5.0
Dell Latitude D620 laptop	Windows 7 and Fedora Linux v10

Table 5.1: Hardware and OSs used in the VESO-MESH v1.0 testbed.

Specification	AOpen Computer (MSN)
Processor	Intel Core 2 T7200
Hard Drive Capacity	200 GB.
CPU	2.0 GHz.
DRAM	2.0 GB.
Expansion	1 miniPCI slot
Connectivity	1 Ethernet interface 10/100
USB	5 USB ports
I/O	1 serial port

Table 5.2: Hardware specifications of the Master Sensor Node (MSN).

record bio-acustic signals in some time intervals. These signals are stored in the microSD card of the Gumstix board. Then, this data is sent through the WMN testbed to the AOpen computer, which is known as Master Sensor Node (MSN). Table 5.2 shows the hardware specifications of the MSN. Afterwards, this data will be sent again to the walsaipsrv.ece.uprm.edu DataBase server located at UPR-RUM using internet connection.

5.1.2 MESH ROUTERS CONFIGURATION

Mesh routers have the WDS system configured as shown in Figure 5.2. WDS allows a wireless network to be expanded using multiple access points without the need for a wired backbone to link them, as is traditionally required. WDS allows a pseudo WMN to be built (See Appendix B for a detailed WMN testbed configuration based on WDS). WDS is preferably used when the number of mesh routers is less than 5. WDS can self-heal thanks to the Spanning Tree Protocol (STP). WDS

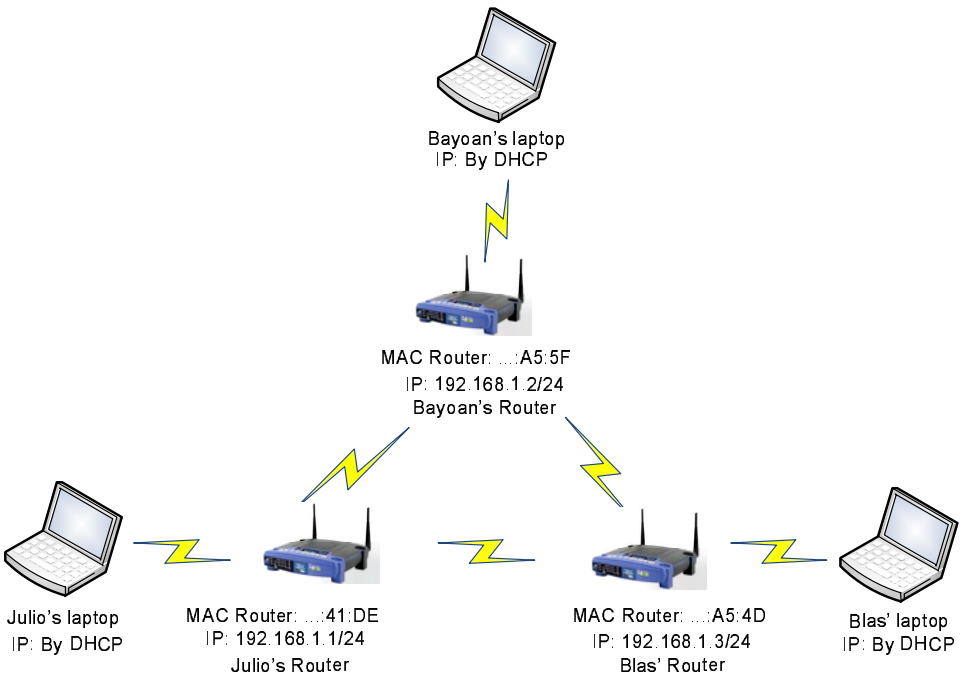


Figure 5.2: WDS Network Topology.

cannot find the best path, nor can it self-configure when adding new routers. Adding new mesh routers could be a really hard task, because we have to put all the other MAC addresses in each mesh router, greatly decreasing the performance.

5.1.3 DISCUSSIONS

- WDS is not a routing protocol for WMNs. It is a system that allows pseudo WMNs to be built. WDS was configured and tested first at the CRL Lab with the two mesh routers. Then, we deployed the pseudo WMN at the Jobos Bay Office. Due to the WDS disadvantages, it was very necessary to find a routing protocol.
- The microSD card of the Gumstix boards was not reliable in this project. After several mounts, these cards presented problems storing data.
- The Gumstix's problem could be solved by two options:
  - First, giving the MSN computer routing capability. By default, the MSN could not work as a mesh router.

Hardware	Software
Alix 2D2 board	Debian Linux v5.0
AOpen computer (MSN)	Ubuntu Linux v8.04 and Fedora Linux v10
DataBase Server	Ubuntu Linux v8.04
Pocket PC Dell Axim X51v	Windows Mobile v5.0
Dell Latitude D620 laptop	Windows 7, Ubuntu v9.04, Debian v5.0 and Fedora v10

Table 5.3: Hardware and OSs used in the VESO-MESH v2.0 testbed.

– Second, exchanging the Gumstix boards for Alix 2D2 boards. These boards have more advantages; they are cheaper and have good features to work as mesh routers, because they have USB and miniPCI interfaces.

5.2 VESO-MESH V2.0

5.2.1 DESIGN OF THE TESTBED

Figure 5.3 depicts the design of the VESO-MESH v2.0 architecture. Table 5.3 shows the hardware and the operating systems used.

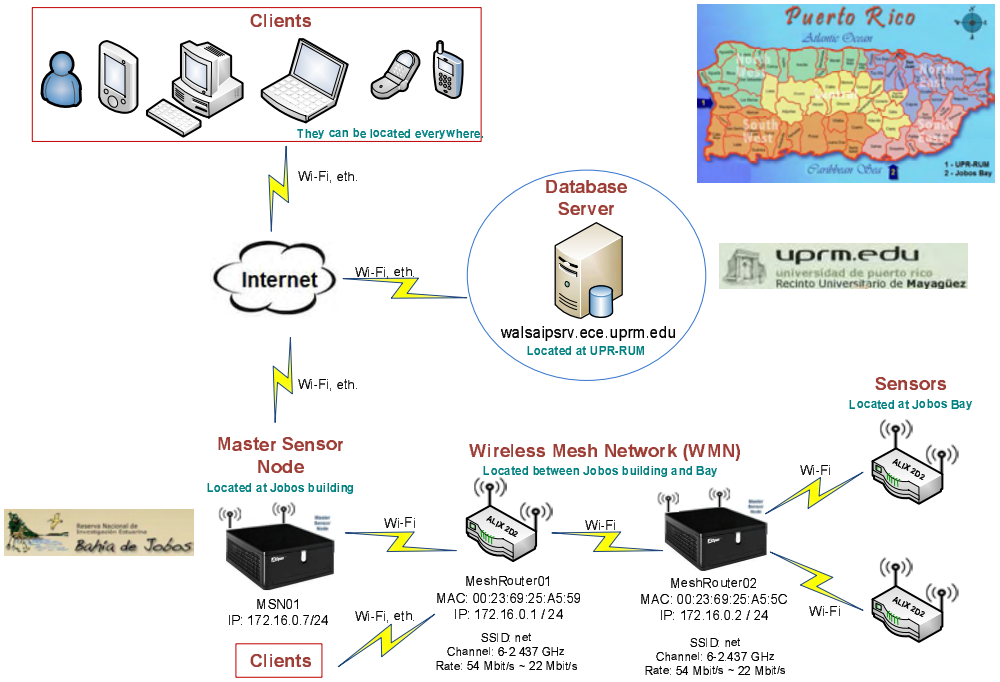


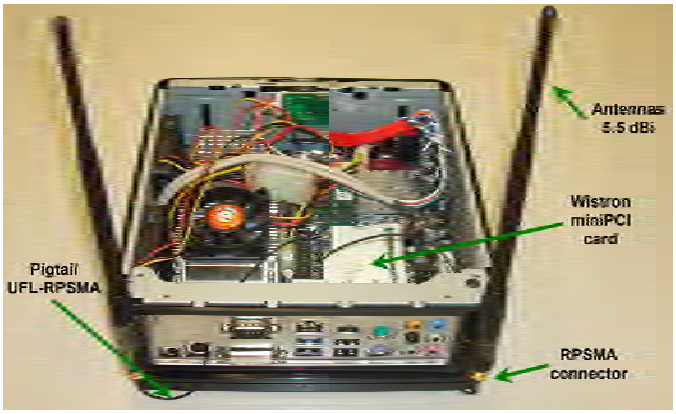
Figure 5.3: VESO-MESH v2.0 Architecture.

5.2.2 MESH ROUTERS CONFIGURATION

The mesh routers that we will use in this project are based on the AOpen computers and the Alix 2D2 boards.

The AOpen computer, known as the Master Sensor Node (MSN), by default could not work as a mesh router. For that reason, the solution we found was to insert the Wistron miniPCI CM9 IEEE 802.11abg (5004 MP Atheros 4G) card into its miniPCI slot as shown in Figure 5.4. Then, follow the mesh router configuration section in Chapter 3.

(a)



(b)

Figure 5.4: The original and modified Master Sensor Node respectively.

On the other hand, the Alix 2D2 board could be used instead of the Gumstix board in this testbed. Chapter 4 discusses in detail the WMN testbed based on Alix boards, which was deployed at the Department of Electrical and Computer Engineering at UPR-RUM.

The Alix board can perform the same functions as the Gumstix board and even much more. We can attach to Alix board any device that is used for Gumstix board. For instance, the microphone previously used for the Gumstix board can be used in the Alix board too. Furthermore, there are some very good advantages for this project that make us more inclined to use the Alix board instead of the Gumstix board, such as:

- Alix 2D2 board uses CF card with 4GB, and Gumstix board uses microSD card with 2GB. The CF cards are more robust and durable.
- Alix 2D2 board has 2 miniPCI slots
  - Having miniPCI slots, we can insert a miniPCI card based on Atheros chipset and then install the MACRT routing protocol as we have seen previously.
  - Also, it is possible to attach directional or omni-directional antennas to the miniPCI cards, using the RPSMA connector type, as we did in the testbed.

5.2.3 DISCUSSIONS

- MACRT routing protocol is used instead of WDS. This will allow us to extend our network with more mesh routers without the WDS problems. Furthermore, this routing protocol can be improved greatly in the near future.
- The CF card used by Alix 2D2 board is more reliable than the microSD card used by Gumstix board.

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

We designed and implemented an advanced WMN testbed using the MAC Layer Routing Protocol (MACRT) and applied it to the Development of a Versatile Service-Oriented Wireless Mesh Network (VESO-MESH) project. The analysis, design and implementation were done using commercial off-the-shelf (COTS) hardware and free software. The Operating Systems were based on Linux distributions. The wireless driver was a Madwifi modified version. The cards used were PCI, miniPCI and PCMCIA, which were based on Atheros chipset. The mesh routers and mesh clients used were laptops, PCs, AOpen computers and Alix 2D2 boards, and a Pocket PC respectively.

We presented a WMN testbed built using PCs and laptop computers as mesh routers. By default, those devices could not work as mesh routers, but we gave them routing capability. We installed Ubuntu Linux v8.04.1. Then, we compiled and installed kernel v2.6.22.19 and GCC v4.2.4 respectively. Finally, we installed the MACRT routing protocol. The WMN testbed was deployed and tested at some strategic points near the Computing Research Laboratory (CRL) of the Department of Electrical and Computer Engineering at UPR-RUM.

Additionally, we presented a WMN testbed built using Alix 2D2 boards as mesh routers. We installed Debian Linux v5.0 on each mesh router as its OS. Then, we compiled and installed kernel v2.6.22.19 and GCC v4.2.4 respectively. We

deployed this WMN testbed at the Stefani building of the Department of Electrical and Computer Engineering at UPR-RUM. We did some experiments to check the network connectivity using ping commands and third party free software.

Finally, we can conclude that these WMN testbeds are serving as the platform of some on-going research topics that are being investigated at the Department of Electrical and Computer Engineering at the UPR-RUM:

- Time-Frequency Signal Representations for Environmental Surveillance Monitoring.
- Basic Testbed Infrastructure for Environmental Surveillance Monitoring.
- Bio-acustic Signal Analysis and Classification.
- Cyclic Short-Time Fourier Transform Algorithms for Environmental Surveillance Monitoring.
- iPhone-based Digital Streaming Data Exchange for Species Classification in a Wireless Sensor Network.

Furthermore, these WMN testbeds will help to understand and develop new routing algorithms and protocols for WMNs in the near future, by undergraduate and graduate students of our department.

6.2 Future Work

- Develop new algorithms, routing protocols and routing metrics, and test them in our WMN testbeds.
- Expand these WMN testbeds with more mesh routers, mesh clients and new devices such as diverse sensors, and antennas, among others.
- Develop new software to analyze data and show them in many different applications, such as windows, web, iPhone, and smart mobile applications.
- Deploy a WMN testbed at our campus to do research in network coding, Quality of Service (QoS), and network management for WMNs.

- Deploy a WMN testbed to create a community mesh network in Mayagüez City, Puerto Rico.

APPENDICES

APPENDIX A
CREATE AND RESTORE A CF CARD IMAGE BACKUP ONTO THE ALIX 2D2 BOARDS

Here, we present step by step how to create and backup a CF card image onto the Alix 2D2 boards.

```
1. CREATING A BACKUP IMAGE
   Once we have the CF card already configured with Debian Linux v5.0, Kernel v2.6.22.19, GCC v4.2 and
   the MACRT routing protocol installed, the next step is to create a backup image, to later on clone
   it into the other CF cards that we will need. For this, do the following three steps:

   1.1 FORMATTING THE CF CARD
        //... Format the CF card with ext2 file system
            // Check if the CF card is already mounted
                mount
            // If so, dismount the CF card
                umount /dev/sdc1
            // Create a partition table
                fdisk /dev/sdc
                    Type 'm' to see the menu
                    Type 'd' to delete a partition
                        Select '1'
                    Type 'n' to add a new partition
                        Type 'p' to create a primary partition
                        Type '1' to specify the partition number
                        Press 'Enter key' to leave by default
                    Type 'a' to toggle a bootable flag
                        Type '1' to specify the partition number
                    Type 'p' to print the partition table
                    Type 'w' to write the partition table to disk and exit
            // Install a Master Boot Record into the CF card
                apt-get install mbr
                install-mbr /dev/sdc
```

```
        // Create an ext2 Linux file system for the new partition
            mkfs.ext2 /dev/sdc1

1.2 CREATING AN IMAGE ONTO THE LAPTOP's DISK FROM THE CF CARD
    //... Create an image on disk from the CF card
    //... First, insert the CF card to the USB card reader
    //... Attach the USB card reader to the Laptop computer
        // Check if the CF card is already mounted
            mount
        // If so, dismount the CF card
            umount /dev/sdc1
        // Create a new /mnt/cf folder
            mkdir -p /mnt/cf
        // Mount the CF card device onto the /mnt/cf/ folder
            mount /dev/sdc1 /mnt/cf
        // Mount proc and sys systems onto the chroot for communicating with the kernel
            mount --bind /proc /mnt/cf/proc
            mount --bind /sys /mnt/cf/sys
        // Create the /home/image folder. In this case, we created the number 8
            mkdir /home/image8
        // Make a secure copy from CF card to the image folder
            rsync -plarv --progress /mnt/cf/ /home/image8/
        // Make a secure copy from the image folder to the CF card
        // This step is used when we want to restore an image to the CF card.
            rsync -plarv --progress /home/image8/ /mnt/cf/
        // Dismount the proc and sys systems
            umount /mnt/cf/proc
            umount /mnt/cf/sys
        // Dismount the CF card
            umount /mnt/cf/
        // Check if some of them are still mounted
            mount

1.3 MODIFY THE GRUB INTO THE CF CARD RECENTLY CREATED.
    //... Once we have restored a CF card from the laptop's image folder, the next step is to create the
    //... GRUB into the CF card.
        // dismount the CF card
            umount /dev/sdc1
        // Create a new /mnt/cf folder
            mkdir -p /mnt/cf
        // Mount proc and sys systems onto the chroot for communicating with the kernel
```

```
    mount /dev/sdc1 /mnt/cf
    mount --bind /proc /mnt/cf/proc
    mount --bind /sys /mnt/cf/sys
// Install grub into the CF card.
    grub-install --root-directory=/mnt/cf /dev/sdc
// Dismount proc and sys systems
    umount /mnt/cf/proc
    umount /mnt/cf/sys
// Dismount the CF card
    umount /mnt/cf/
// Check if some of them are still mounted
    mount
```

APPENDIX B
WMN TESTBED BASED ON WDS

Here, we present the WMN testbed configuration based on the Wireless Distribution System (WDS) that was deployed at Jobos Bay National Estuarine. WDS enables the wireless interconnection of Access Points (AP) in an IEEE 802.11 network. It allows a wireless network to be expanded using multiple APs without the need for a wired backbone to link them, as is traditionally required. The WDS advantage over other solutions is that it preserves the MAC addresses of client packets across links between access points. WDS allows us to build a pseudo WMN.

B.1 THE WMN TOPOLOGY

Figure B.1 depicts the topology used to form the WMN. We used three Linksys WRT54GL routers as mesh routers and three laptops as mesh clients. The mesh routers run the DD-WRT v24 firmware. Each mesh router will have the MAC address of the other routers. This allows each mesh router to know its closest neighbor. The mesh clients run Windows and Linux operating systems. Mesh clients are associated with each mesh router. Mesh clients will receive an IP address by DHCP. Once we have the mesh routers and mesh clients configured, the next step is to find some strategic points to deploy the network.

B.2 SETTING PARAMETERS

1. Set up the following parameters on the router: *Wireless Tab - Basic Settings Tab* as shown in Figure B.2. Set Wireless Mode option to AP, Wireless Network Mode option to G-Only (we can use mixed too), SSID option to router-julio (we can

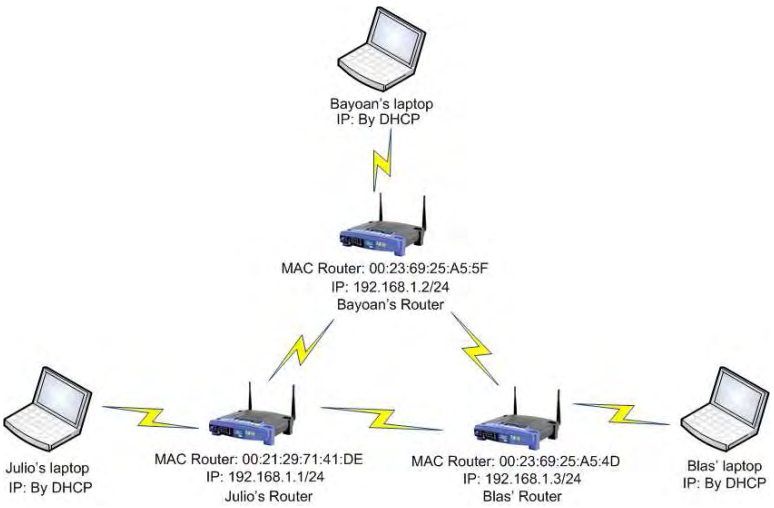


Figure B.1: The WMN topology based on WDS.

use any name), Wireless Channel option to 11. Afterward, press Save and Apply Settings buttons respectively.

Figure B.2: Wireless Basic Settings Tab.

2. Set up the following parameters: *Wireless Tab - WDS Tab*. We will enter the MAC addresses of the other two routers. We will choose LAN, insert the MAC address, and finally enter a description; in this case we have to put the other two MAC addresses (Bayoan's and Blas' routers) as shown in Figure B.3.

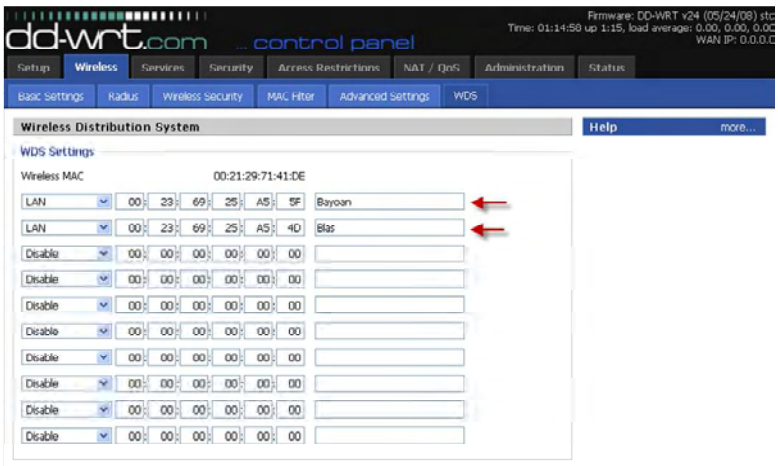


Figure B.3: Setting up the MAC addresses.

3. Set up the following parameters: *Setup Tab - Basic Setup Tab*. We have to enable STP in order to avoid loops and a lot of broadcasts that will make the network slow. We have to set the Router Name, the Local IP address and the Subnet Mask respectively as shown in Figure B.4. Afterward, press Save and Apply Settings buttons respectively.

   Note: Each router has to be a different IP address.

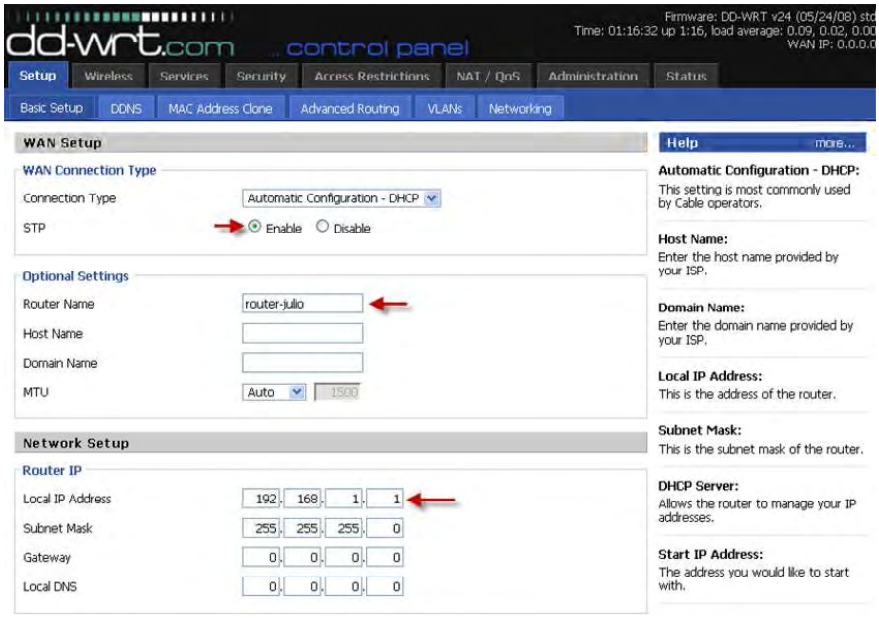


Figure B.4: Setting some parameters.

4. Set up the following parameters: *Wireless Tab - Wireless Security Tab*. Disable the Security Mode as shown in Figure B.5. Afterward, press Save and Apply Settings buttons respectively.

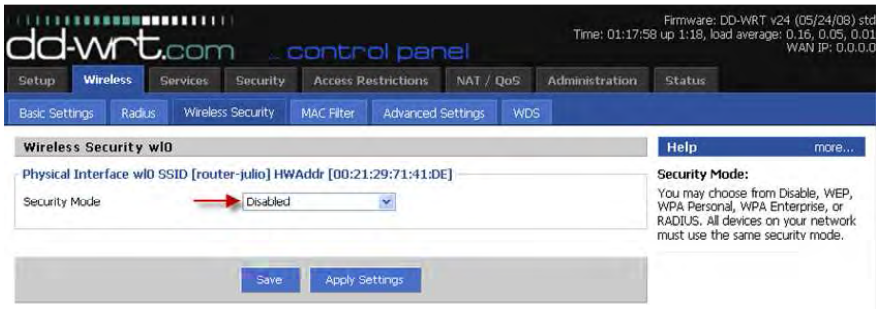


Figure B.5: Disabling Wireless Security Mode.

5. Finally, we will have a summary of the configuration (Figure B.6).

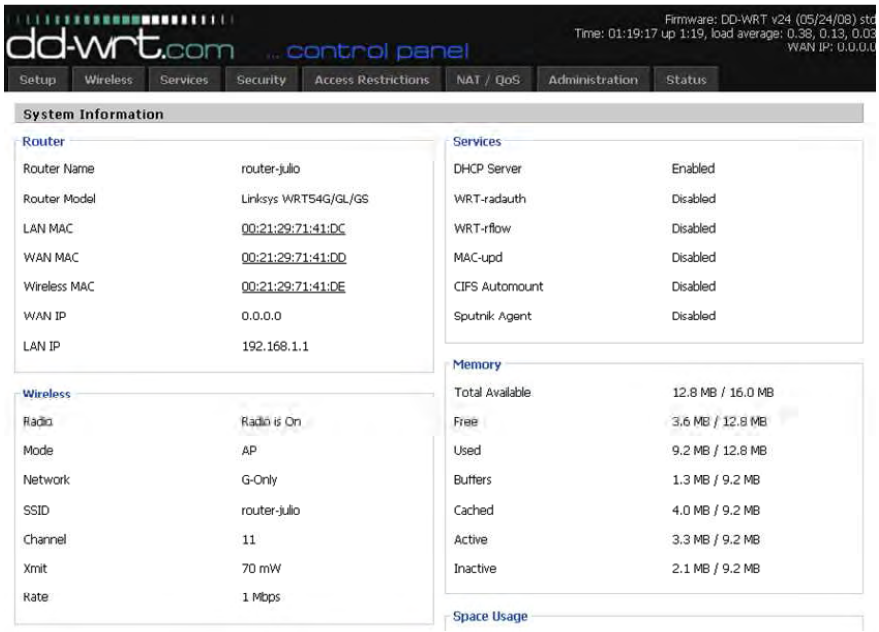


Figure B.6: Summary of the configuration.

Note: Do the similar steps in the other two routers using Figure B.1 as a guide. Finally, find strategic points to deploy and test the WMN.

GLOSSARY

- Commercial Off-The-Shelf (COTS)

  It defines a technology that is ready to use and is available for sale, lease, or license to the general public. *Page 2.*

- Data rate

  It communicates the maximum possible rate at which a device or interface can transmit data. *Page 8.*

- Direct Sequence Spread Spectrum (DSSS)

  This modulation technique spreads a signal over a wide frequency band for transmission. IEEE 802.11 and IEEE 802.11b are based on DSSS. *Page 9.*

- Frequency Hopped Spread Spectrum (FHSS)

  This modulation technique transmits radio signals by rapidly switching a carrier among many frequency channels. It uses a time-varying narrowband signal to spread radio frequency energy over a wide band. IEEE 802.11 is based on FHSS. *Page 9.*

- GNU Compiler Collection (GCC)

  It is a compiler system produced by the GNU Project. It is used on Linux systems and compiles programs written in C, C++, Java, Ada and Fortran. *Page 2.*

- Grub

  It is a boot loader package from the GNU Project. It enables a computer to have multiple operating systems, and allows for the choice of which one to run when the computer starts. *Page 27.*

- Kernel

  It is the core component of most computer operating systems that starts up when you boot your computer. It is a bridge between applications and the actual data processing done at the hardware level. *Page 2.*

- Linux

  It is a Unix-like operating system that uses the Linux kernel. It is a free and open source software. The source code can be used, modified, and redistributed by anyone under certain licenses. *Page 2.*

- Orthogonal Frequency Division Multiplexing (OFDM)

  This new modulation technique uses a large number of small overlapping channels to transmit the data. IEEE 802.11a and IEEE 802.11g are based on OFDM. IEEE 802.11n uses MIMO to transmit multiple OFDM data streams. *Page 9.*

- Protocol

  It is the set of rules governing the syntax, semantics, and synchronization of communication that can be implemented by hardware and/or software. *Page 2.*

- Standard

  It is an established norm or requirement. It is generally a formal document that establishes uniform engineering or technical criteria, methods, processes and practices. *Page 5.*

- Testbed

  It is a platform for experimentation of development projects. Testbeds allow for rigorous, transparent, and replicable testing of scientific theories, computational tools, and new technologies. *Page 1.*

- Throughput

  It is the amount of data that provides a real-world measure of performance. It is always lower than the data rate and is usually measured in bits per second (bps). *Page 8.*

REFERENCE LIST

[1] Rohan Narayana Murty, Geoffrey Mainland, Ian Rose, Atanu Roy Chowdhury, Abhimanyu Gosain, Josh Bers, and Matt Welsh. CitySense: an Urban-Scale wireless sensor network and testbed. In *2008 IEEE Conference on Technologies for Homeland Security*, pages 583–588, Waltham, MA, USA, 2008.

[2] John Bicket, Daniel Aguayo, Sanjit Biswas, and Robert Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 31–42, Cologne, Germany, 2005. ACM.

[3] Self Organizing Wireless Mesh Networks - Microsoft Research. http://research.microsoft.com/en-us/projects/mesh/, May 2009.

[4] Elizabeth M. Belding. UCSB MeshNet. http://moment.cs.ucsb.edu/meshnet/, December 2009.

[5] Purdue University. https://engineering.purdue.edu/MESH, October 2009.

[6] I.F. Akyildiz and Xudong Wang. A survey on wireless mesh networks. *IEEE Communications Magazine*, (9), 2005.

[7] Award#0922996 - MRI: development of a versatile Service-Oriented wireless mesh network for disaster relief and environmental monitoring in puerto rico. http://www.nsf.gov/awardsearch/showAward.do?AwardNumber=0922996.

[8] nerrs.noaa.gov - NERRS reserves. http://nerrs.noaa.gov/Reserve.aspx?ResID=JOB.

[9] Matthew Gast. *802.11 Wireless Networks: The Definitive Guide, Second Edition*. O'Reilly Media, 2 edition, April 2005.

[10] W. Stallings. IEEE 8O2.11: wireless LANs from a to n. *IT Professional*, 6(5):32–37, 2004.

[11] IEEE standard for information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, 2007.

[12] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A high-throughput path metric for multi-hop wireless routing. *Wirel. Netw.*, 11(4):419–434, 2005.

[13] Ian Akyildiz and Xudong Wang. *Wireless Mesh Networks*. Wiley, March 2009.

[14] David B Johnson, David A Maltz, and Josh Broch. DSR: the dynamic source routing protocol for Multi-Hop wireless ad hoc networks. *In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5*, pages 139—172, 2001.

[15] Richard Draves, Jitendra Padhye, and Brian Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking*, pages 114–128, Philadelphia, PA, USA, 2004. ACM.

[16] C.E. Koksal and H. Balakrishnan. Quality-Aware routing metrics for Time-Varying wireless mesh networks. *IEEE Journal on Selected Areas in Communications*, 24(11):1984–1994, 2006.

[17] Charles E. Perkins and Pravin Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In *Proceedings of the conference on Communications architectures, protocols and applications*, pages 234–244, London, United Kingdom, 1994. ACM.

[18] Shree Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mob. Netw. Appl.*, 1(2):183–197, 1996.

[19] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). http://www.ietf.org/rfc/rfc3626.txt, October 2003.

[20] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand distance vector (AODV) routing. http://www.ietf.org/rfc/rfc3561.txt, April 2003.

[21] Prince Samar, Zygmunt J Haas, and Marc R Pearlman. draft-ietf-manet-zone-zrp-00 - the zone routing protocol (ZRP) for ad hoc networks. http://tools.ietf.org/html/draft-ietf-manet-zone-zrp-00, August 2009.

[22] M.E.M. Campista, P.M. Esposito, I.M. Moraes, L.H.M. Costa, O.C.M. Duarte, D.G. Passos, C.V.N. de Albuquerque, D.C.M. Saade, and M.G. Rubinstein. Routing metrics and protocols for wireless mesh networks. *IEEE Network*, 22(1):6–12, 2008.

[23] Sanjit Biswas and Robert Morris. Opportunistic routing in multi-hop wireless networks. *SIGCOMM Comput. Commun. Rev.*, 34(1):69–74, 2004.

[24] Richard Draves, Jitendra Padhye, and Brian Zill. Comparison of routing metrics for static multi-hop wireless networks. In *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 133–144, Portland, Oregon, USA, 2004. ACM.

[25] Wenhua Zhao. A new MAC layer routing protocol for infrastructure wireless mesh networks. Master's thesis, Technical University of Denmark, DTU, 2008.

[26] CalMesh. http://calmesh.calit2.net/, January 2010.

[27] madwifi-project.org - trac. http://madwifi-project.org/, February 2010.

[28] Eldad Perahia and Robert Stacey. *Next Generation Wireless LANs: Throughput, Robustness, and Reliability in 802.11n.* Cambridge University Press, 1 edition, September 2008.

[29] J.J. Garcia-Luna-Aceves and M. Spohn. Source-tree routing in wireless networks. In *Network Protocols, 1999. (ICNP '99) Proceedings. Seventh International Conference on*, pages 273–282, 1999.

BIOGRAPHICAL SKETCH

Julio Castillo Tito received the B.S. degree in Informatics and Systems Engineering from the San Antonio Abad University, Cusco, Peru, in 2004.

Since August 2007, he was studying the master degree in Computer Science in the Department of Mathematics, University of Puerto Rico at Mayagüez. After a year, he translated to the Department of Electrical and Computer Engineering to pursue the master degree in Computer Engineering. His research interests include computer and wireless networks, network management and security.