

**WIRELESS MESH NETWORKS: SERVICE ORIENTED DESIGN
AND CHANNEL OPTIMIZATION**

By

Héctor M Lugo-Cordero

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

December, 2008

Approved by:

Henrick Ierkic, Ph.D
Member, Graduate Committee

Date

Domingo Rodriguez, Ph.D
Member, Graduate Committee

Date

Kejie Lu, Ph.D
President, Graduate Committee

Date

Omar Colón, Ph.D
Representative of Graduate Studies

Date

Isidoro Couvertier, Ph.D
Chairperson of the Department

Date

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

**WIRELESS MESH NETWORKS: SERVICE ORIENTED DESIGN
AND CHANNEL OPTIMIZATION**

By

Héctor M Lugo-Cordero

December 2008

Chair: Kejie Lu

Major Department: Electrical and Computer Engineering

In recent years, IEEE 802.11 based wireless local area network (WLANs) have been widely deployed and the cost of IEEE 802.11-enabled routers and adapters are very low. To effectively establish wireless networks in various scenarios, it is important to exploit such inexpensive commercial off-the-shelf (COTS) products. In this thesis, we address this issue and propose a novel medium access control (MAC) scheme based on the idea of network coding and intelligent channel selection. Specifically, we designed a new sub-layer between the network layer and the legacy IEEE 802.11 MAC layer. In this manner, the cost for updating the system can be minimized, while the performance can be improved efficiently. We also address the scalability of the network using a new architecture scheme known as the Service Oriented Architecture (SOA). By implementing a Service Oriented Routing Algorithm (SORA), a network layer protocol that chooses the best route based on the service that is being requested rather than the traditional source and destination approach, one can add this architecture to any existing network. Through

the implementation of some testbeds we show how this new scheme improves the performance of the Wireless Mesh Networks (WMNs).

Resumen de Disertación 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 MALLAS: DISEÑO ORIENTADO A SERVICIOS Y OPTIMIZACIÓN DE CANALES

Por

Héctor M Lugo-Cordero

Diciembre 2008

Consejero: Kejie Lu

Departamento: Ingeniería Eléctrica y Computadoras

Durante los últimos años, las redes inalámbricas de área local basadas en el IEEE 802.11 han ganado popularidad causando una disminución en costo de los equipos de la misma. Para establecer efectivamente redes inalámbricas en múltiples escenarios, es importante utilizar equipo de bajo costo comercial (COTS). En esta tesis, nos enfocamos en este asunto y proponemos un esquema innovador de Control de Acceso al Medio (MAC) basado en la idea de codificación de redes y la selección inteligente de canales. Específicamente, diseñamos una nueva sub-capa entre la capa de red y el legado del IEEE 802.11 en la capa del MAC. De esta manera, el costo para actualizar el sistema puede ser minimizado, mientras que el desempeño puede ser mejorado eficazmente. También nos enfocamos en la escalabilidad de la red utilizando un nuevo esquema de arquitectura conocido como Arquitectura Orientada a Servicios (SOA). Implementando un Algoritmo de Enrutamiento Orientado a Servicios (SORA), un protocolo en la capa de la red que escoge la mejor ruta basado en el servicio que se solicita, en lugar del esquema tradicional de fuente y destino,

se puede añadir dicha arquitectura a cualquier red existente. A través de la implementación de testbeds demostramos como este nuevo esquema mejora el desempeño de las Redes Inalámbricas en Mallas (WMNs).

Un résumé de Dissertation Présenté à l'École Élevée au grade
de l'Université de Puerto Rico comme condition partielle des
Requêtes pour le degré de Maîtrise dans les Sciences

**DES RÉSEAUX SANS CÂBLES EN MAILLES: DESSIN ORIENTÉE
AUX SERVICES ET OPTIMISATION DE CANAUX**

Par

Héctor M Lugo-Cordero

Decembre 2008

Conseiller: Kejie Lu

Département: Ingénierie Électrique et de l'Informatique

Depuis quelques années les réseaux sans câbles d'aire locale construits par le IEEE 802.11 ont gagné en popularité ce qui a abouti à une réduction de leur prix. Créer des réseaux sans câbles convenables, la plus part du temps exige l'utilisation des matériels à bas prix commercial (COTS).

Dans cette thèse, nous analysons le problème de création à bas prix et nous proposons un nouveau schéma du Contrôle d'Accès au Milieu (MAC) fondé sur l'idée de codage des réseaux ainsi que de la sélection intelligente de canaux. Spécifiquement, nous créons une nouvelle sous-niveau entre le niveau de réseaux et le niveau du MAC (IEEE 802.11).

De cette façon, le prix d'actualisation du système peut être minimisé, alors que la performance peut effectivement être améliorée. Nous travaillons aussi dans la modularité du réseau à l'aide d'un nouveau schéma d'architecture connu comme Architecture Orientée aux Services (SOA).

Si l'on utilise un Algorithme de Diriger Orienté aux Services (SORA), un protocole dans le niveau du réseau qui choisit la meilleure route basé sur le service

qui est sollicité, au lieu du schéma traditionnel de source et de destination, on peut ajouter cette architecture à n'importe quel réseau existant.

À l'aide de l'implémentation de testbeds nous démontrons comment ce nouveau schéma améliore la performance des réseaux sans câbles en mailles (WMNs).

Copyright © 2008

by

Héctor M Lugo-Cordero

This thesis is dedicated first of all to God for giving me the most precious of all gifts, life. Secondly, I want to dedicate this thesis to my family for being always there for me and making me the person I am today. To my advisor, Kejie Lu, and my friends that have been there during my best days and my worst days when I needed them the most. Also I would like to dedicate this thesis to Prof. Baldomero Llorens for being like a second father to me during my BS studies and inspiring in me the desire of continuing my graduate studies.

Also I want to congratulate my brother on his 18th birthday, which is today October 22 2008, the day that I finished the writting of my thesis.

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, helping me to develop into a greater professional. I also want to thank to all my professors of the graduate committee, Dr. Domingo Rodriguez for helping me in the DSP area and Dr. Henrick Ierkic for helping with the physical medium issues.

Many thanks also to the chairman of the department, Dr. Isidoro Couvertier. His teachings during the beginning of my master studies gave me the idea to develop the scheme proposed in this thesis. Dr. Raul Torres also contributed in providing me the necessary background for developing the scheme.

This work was supported in part by the US National Science Foundation (NSF) under Award Number CNS-0424546 and Award Number DUE-0736868.

TABLE OF CONTENTS

	<u>page</u>
ABSTRACT ENGLISH	ii
ABSTRACT SPANISH	iv
ABSTRACT FRENCH	vi
ACKNOWLEDGMENTS	x
LIST OF TABLES	xiv
LIST OF FIGURES	xv
LIST OF ABBREVIATIONS	xviii
 1 INTRODUCTION	 1
1.1 MOTIVATIONS	1
1.2 CONTRIBUTIONS	2
1.3 APPROACH	3
 2 BACKGROUND	 5
2.1 WIRELESS MESH NETWORK	5
2.2 LAYERED MODEL FOR WMN	5
2.2.1 PHYSICAL LAYER	6
2.2.2 DATA LINK LAYER	10
MEDIUM ACCESS CONTROL SUB-LAYER	10
2.2.3 NETWORK LAYER	14
PRE-ROUTING	14
POST-ROUTING	19
2.2.4 TRANSPORT LAYER	22
TCP	22
UDP	25
2.2.5 APPLICATION LAYER	25
RTP AND RTCP	25
2.3 SERVICE-ORIENTED ARCHITECTURE (SOA)	26
2.4 FUZZY LOGIC	28
2.4.1 FUZZIFICATION	29
2.4.2 IF-THEN RELATIONS	29
2.4.3 DEFUZZIFICATION	31
2.5 NETWORK CODING	31

2.6	SECURITY	34
3	RELATED WORK	35
3.1	WIRELESS MESH NETWORKS	35
3.2	SERVICE-ORIENTED ARCHITECTURE	38
3.3	CHANNEL SELECTION	45
3.4	NETWORK CODING	48
4	THE TESTBED	52
4.1	HARDWARE USED	52
4.2	SOFTWARE USED	53
4.3	OPENWRT DISTRIBUTIONS	53
4.3.1	WHITE RUSSIAN	54
4.3.2	KAMIKAZE	54
4.3.3	DD-WRT	55
4.4	CONFIGURATION OF THE LINKSYS WRT54GL	55
4.5	CONFIGURATION OF THE LAPTOP COMPUTER	63
5	SERVICE ORIENTED ROUTING ALGORITHM	67
5.1	INTRODUCTION	68
5.2	RELATED WORK	69
5.3	DESIGN OF SORA	70
5.3.1	CLIENT SIDE PROCEDURE	70
5.3.2	INTERMEDIATE NODE PROCEDURE	71
5.3.3	SERVER SIDE PROCEDURE	74
5.3.4	SORA AGENT	74
5.3.5	PACKET FORMAT	74
5.4	EXPERIMENT AND DISCUSSION	76
5.5	CONCLUSION	78
6	CHANNEL SELECTION NETWORK CODING	79
6.1	INTRODUCTION	80
6.2	RELATED WORK	81
6.3	DESIGN OF CHANET	82
6.3.1	CHANNEL SELECTION	82
6.3.2	NETWORK CODING	84
6.4	EXPERIMENT AND DISCUSSION	89
6.5	CONCLUSION	91
7	CONCLUSIONS AND FUTURE WORK	93
7.1	CONCLUSIONS	93
7.2	FUTURE WORK	94

APPENDICES	96
A SORA SOURCE CODE FRAGMENTS	97
A.1 SORA PACKET STRUCTURE IN C	97
A.2 SORA SERVICE SESSION STRUCTURE IN C	97
A.3 SORA ROUTE TABLE STRUCTURE IN C	97
A.4 SORA ROUTER FORWARD REPLY PROCEDURE FRAGMENT IN C	98
A.5 SORA ROUTER FORWARD REQUEST SENDING PROCE- DURE FRAGMENT IN C	99
A.6 SORA SERVER PROCEDURE FRAGMENT IN C	99
A.7 SORA AGENT FRAGMENT IN JAVA	100
B CHANET SIMULATION	102
B.1 FUZZY.H	102
B.2 FUZZY.CC	104
B.3 CHANNEL UTILAZATION AGENT CODE IN C++	122
BIOGRAPHICAL SKETCH	138

LIST OF TABLES

<u>Table</u>	<u>page</u>
2-1 Name of data at each layer	6
2-2 IEEE 802.11 Protocol Family	7
2-3 DSDV Routing Table Example	18
2-4 Examples of network applications	26
2-5 Fundamental Fuzzy Sets	30
2-6 Definition of Logical Operators	31
2-7 Defuzzification Methods	31
3-1 Actions for fault tolerant environment	45
4-1 Dell Latitude D620 Specifications	53
4-2 Linksys WRT54GL Specifications	53
5-1 An example of Service Table	73
5-2 Time to Download a File (3MB)	78
6-1 Connection Flows	91
6-2 Performance Comparison	91

LIST OF FIGURES

<u>Figure</u>	<u>page</u>
1-1 Protocol Stack.	4
2-1 Types of Node Configurations	8
2-2 Ideal OFDM Transmitter	9
2-3 Ideal OFDM Receiver	10
2-4 Hidden Terminal Problem	13
2-5 Virtual Carrier Sense Timeline	14
2-6 OLSR Hello Messages	15
2-7 OLSR Topology Control Messages	15
2-8 OSPF Hello Messages	17
2-9 IP Header	21
2-10 Mobile IP Agent	22
2-11 IP Agent Example	23
2-12 TCP triple handshake	23
2-13 Example of a service-oriented wireless mesh network	27
2-14 Boolean Logic Curve	28
2-15 Typical S Shape Fuzzy Curve	29
2-16 Order of Fuzzy Logic Process	30
2-17 Network coding on a butterfly network	33
3-1 Examples of wireless mesh network.	36
3-2 Example of service-oriented network	39
3-3 JTang components	43
3-4 Database approach to SOA	43
3-5 Web approach to SOA	44

3-6	Single-radio WMN	46
3-7	Multi-radio WMN	47
3-8	2.4GHz frequency band	48
3-9	Network coding example	49
3-10	MORE header	50
3-11	Physical network coding example	51
4-1	Testbed	52
4-2	White Russian web interface	54
4-3	Kamikaze web interface	54
4-4	DD-Wrt web interface	55
4-5	Main download page for kamikaze 7.09	56
4-6	Brcm-2.4 download page	56
4-7	Original Linksys web interface	57
4-8	Web interface for wireless configuration	59
4-9	Web interface for networks configuration	60
4-10	Web interface for DHCP configuration	61
4-11	Network connections on control panel	64
4-12	Dropdown menu	65
4-13	Network properties window	65
4-14	Preferred networks tab	66
4-15	Advanced window	66
5-1	Client procedure	71
5-2	Intermediate-node procedure	72
5-3	Server procedure	75
5-4	Agent TELNET connection example	76
5-5	Packet header format	77
5-6	SORA testbed	77

6-1	Collision sets definition	83
6-2	Utilization sets definition	84
6-3	Graphical representation of the channel selection process	86
6-4	Channel selection flow chart	87
6-5	Network coding send flow chart	88
6-6	Network coding receive flow chart	89
6-7	CHANET example scenario	90
6-8	CHANET simulation	90

LIST OF ABBREVIATIONS

ACK	Acknowledgement
AOA	Access-Oriented Architecture
AODV	Ad-Hoc On-Demand Distance Vector Routing Protocol
AP	Access Point
BATMAN	Better Approach To Mobile Ad-Hoc Networking Routing Protocol
CCK	Complementary Code Keying
COTS	Commercial Off-The-Shelf
CHANET	Channel Selection Network Coding Protocol
CSMA/CA	Carrier Sense Multiple Access/Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
CTS	Clear To Send
DCF	Distributed Coordination Function
DFS	Dynamic Frequency Selection
DFT	Discrete Fourier Transform
DHCP	Dynamic Host Configuration Protocol
DLL	Data Link Layer
DNS	Domain Name Service
DSDV	Destination-Sequenced Distance Vector Routing Protocol
DSP	Digital Signal Processing
DSR	Dynamic Source Routing
FFT	Fast Fourier Transform
FTP	File Transfer Protocol
ICMP	Internet Control Message Protocol
IDFT	Inverse Discrete Fourier Transform
IEEE	Institute of Electrical and Electronics Engineers
IFFT	Inverse Fast Fourier Transform
IGMP	Internet Group Management Protocol
IP	Internet Protocol
LAN	Local Area Network
LLC	Logical Link Control
MAC	Medium Access Control
MAN	Metropolitan Area Network
MANET	Mobile Ad-Hoc Network
MAP	Mesh Access Point
MCMR	Multi-Channel Multi-Radio
MCSR	Multi-Channel Single-Radio
MIMO	Multiple Input - Multiple Output
MISO	Multiple Input - Single Output

MP	Mesh Point
MR-WMN	Multi-Radio Wireless Mesh Network
MRS	Mobile Relay Station
MS	Mobile Station
MSDU	MAC Service Data Unit
MSPU	MAC Service Protocol Unit
NetIF	Network Interface
NS2	Network Simulator 2
OFDM	Orthogonal Frequency-Division Multiplexing
OLSR	Optimized Link State Routing
OSI	Open Systems Interconnection Basic Reference Model
OSPF	Open Shortest Path First
PBCC	Packet Binary Convolutional Coding
PHY	Physical Layer
QoS	Quality of Service
RIP	Routing Information Protocol
RS	Relay Station
RTP	Realtime Transport Protocol
RTS	Request To Send
SIMO	Single Input - Multiple Output
SISO	Single Input - Single Output
SSH	Secure Shell
SSID	Service Set Identifier
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SORA	Service-Oriented Routing Algorithm
TCP	Transmission Control Protocol
TCL	Tool Command Language
TELNET	Telecommunication Network
TPC	Transmission Power Control
UDP	User Datagram Protocol
WDS	Wireless Distribution System
WEP	Wired Equivalent Privacy
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WPAN	Wireless Personal Area Network
WRN	Wireless Relay Network
WS	Work Station
WSN	Wireless Sensor Network

CHAPTER 1

INTRODUCTION

1.1 MOTIVATIONS

Over the past decade wireless networks have won great popularity. Now these days people tend to like more wireless networks over wired networks for several reasons:

1. Cost: Wireless networks are cheap to develop since the equipment does not use cable installation to work, thus scaling the network is really simple.
2. Installation: Wireless can travel anywhere, on the other hand wires may not be suitable to install at certain locations.
3. Mobility: Nodes can move freely over a coverage area.
4. Transparency: Users work with wireless networks the same way they do with wired ones.

However such popularity brings some drawbacks. As the number of users increases in a wireless network its performance reduces for several reasons. Among them one of the most important, since it causes the other ones, is an increase of traffic causing congestion in the medium. Since a wireless node is half-duplex, meaning it cannot receive and send information at the same time interval, it has to wait to complete one task in order to do the other. Also if two nodes send at the same time a collision will occur causing a retransmission later. Another thing that this causes is a reduction in bandwidth, because all users along with some equipment (e.g. microwaves) share the same bandwidth.

Sometimes when a node does manage to receive a packet, it may find that an error occurred and also asks for a retransmission. This can happen a lot because of the interference of other nodes and the noise caused by the environment.

To make things worse the current architecture does not provide a mechanism to reduce the transmissions in a medium. Users right now have to specify the server to which they want to connect and then the network finds the best path to get to it, but there is no guarantee that indeed this server is the best option for the host to begin with. It could have happened that there were some servers closer to the host that could have provided the same service.

This is a great issue since most of the traffic in wireless networks is peer to peer, and because of the nature of the medium, broadcast. It is, without any doubt, possible to benefit from this broadcast nature to transmit application to multiple users with a performance better than the wired network. But first some changes need to be done.

1.2 CONTRIBUTIONS

In this thesis we managed to develop some schemes to better use the wireless medium and benefit from its natural features:

1. We investigated the current protocol stack architecture and what it provides to the current network, analyzing its weakness as well.
2. We developed a scheme that is focused on searching for a service from the best server, rather than a service from a certain server using the best route.
3. We developed a scheme to send multiple packets through the same channel at one time interval, which is great for multicast messages and broadcasts, not much difference for unicasts although it provides more security as well.
4. We developed a scheme that gives intelligence to the wireless network as well as the potential to become a full-duplex link, thus sending more information in less time.

5. We created a testbed based on a wireless mesh network to test our service-oriented scheme and some simulations for the channel selection and network coding part using the Network Simulator 2 (NS2).
6. We provide the steps to configure a wireless mesh network using a LinksysWRT54GL router.

1.3 APPROACH

This thesis studies different techniques for optimizing the channel utilization of a wireless mesh network (WMN). The first one is the development of a routing protocol using a cross-layer design. Using information from the network layer and the transport layer to route along with the knowledge of the application layer as one of the routing metrics. The routing protocol is service-oriented, thus we called it Service-Oriented Routing Algorithm (SORA). The concept of service oriented is explained in details in the next chapter, but for now let us refer to it as an architecture that focuses on the services that will be provided rather than the connectivity from a node to another.

The second technique involves actually two things, and is focused more on exploiting the channel capacity to send the most possible information in less time. Using network coding we achieved to send more data through one channel at a given time. Also by implementing a channel selection algorithm, based on a fuzzy logic approach, we were able to give some sense of intelligence to the WMN and make it capable of making dynamic decisions for the best channel at the current environment. This makes the network not only choose the best channel, but also have multiple radios so that it could transform the WMN from a half-duplex environment into full-duplex, since there are 11 channels in the 802.11 standards out of which we have 3 non-overlapped channels. To test the algorithms simulations were used as well as a testbed.

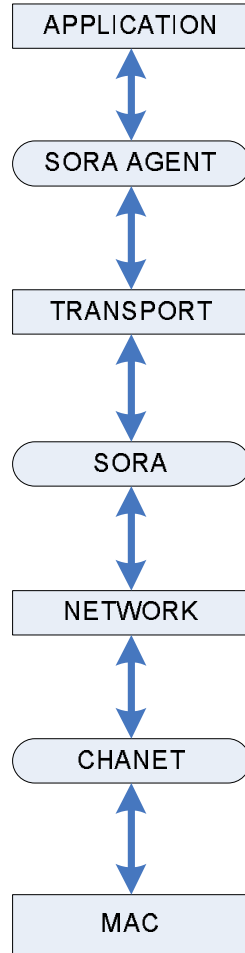


Figure 1–1: Protocol Stack.

Figure 1–1 shows where each technique is located in the current network architecture.

The organization of the rest of the document goes as follows. First chapter 2 describes what is a WMN and different technologies in the literature that support this kind of network along with some background information that will allow the reader to understand how the techniques that are applied work. Chapter 3 provides a brief description of previous work in the field of study. Then in chapter 4 we describe the testbed and how to configure it followed by the implementation details of the work in chapters 5 and 6. Finally chapter 7 presents some conclusions based on the results and future work to improve the work.

CHAPTER 2

BACKGROUND

In this chapter you will find the basics of the three techniques used in this thesis to improve the wireless network performance. The first section explains what is a Wireless Mesh Network and its advantages over other types of networks. Then the second section presents the current protocol stack for WMNs followed by its architecture in the third section. Section four contrasts the current architecture with the Service Oriented Architecture. Afterwards in section five we find the usage of Fuzzy Logic and how it differs from Boolean Logic. Finally in section six we present the alternative of network coding and how it works.

2.1 WIRELESS MESH NETWORK

Wireless Mesh Network (WMN) is a rising technology composed of radio nodes. Just like an Ad-Hoc network, WMN operate using multi-hop wireless transmissions to communicate from one point to another. WMN nodes can also self arrange when necessary if some node in the mesh crashes in some manner. The main difference between Ad-Hoc and WMN is that WMNs force the intermediate nodes to have an infrastructure.

2.2 LAYERED MODEL FOR WMN

This section will introduce the details of the current protocol stack for WMNs, using the TCP/IP reference model, except that for better understanding we have divided the link layer into two: data link and physical just like in the OSI reference model. Table [2-1](#) shows the terms use to refer to the data at each layer.

Table 2-1: Name of data at each layer

Layer	Data Name
Physical	Bytes
Data Link	Frames
Network	Packets
Transport	Segments
Application	Data

2.2.1 PHYSICAL LAYER

In this section we talk about the physical medium used to transmit in WMN. Data is transmitted through air, which unlike Ethernet cables in wired networks, is a shared medium meaning that anybody can intercept or receive the data bytes. This brings a great security issue that typically is handled at upper layers although there are some ways of dealing these at the physical layer such as network coding or any other kind of modulation.

The biggest problem with the physical layer in WMN is however performance rather than security. The fact that the medium is shared also introduces a great number in collisions, thus a high probability of error and greater number of retransmissions and larger delays. As a matter of fact even a microwave oven can cause problems to the WMN because it works at the same frequency band (2.4 GHz).

As a solution to this problem people have developed multiple technologies for better using the wireless medium. Among these technologies one can find channel allocation schemes, antenna design, signal modulation and multiple-input multiple-output (MIMO) systems. The Institute of Electrical and Electronics Engineers (IEEE) has also created several standards known as the IEEE802.11 family to support different aspects. Table 2-2 has a comparison of the different IEEE802.11 standards.

Now we will discuss the advantages of the technologies previously mentioned. First of all in the antenna design we have two types that are very popular; the first is the omni directional antenna which sends data in a radius to all directions,

Table 2–2: IEEE 802.11 Protocol Family

Standard	Comments
802.11a	Operates in a 5GHz radio band 8 available channels 54 Mbps maximum link rate per channel
802.11b	Operates in a 2.4GHz radio band At most 3 non-overlapped channels 11 Mbps maximum link rate per channel
802.11d	Allows APs to communicate permissible radio channels Allows WLANs to be used worldwide
802.11e	Supports QoS for 802.11 a, b, and g Provides QoS for data, voice and video applications
802.11f	Defines the registration of APs in a network Defines the information interchange between APs
802.11g	Operates in a 2.4GHz radio band At most 3 non-overlapped channels 54 Mbps maximum link rate per channel Modulations used: OFDM, CCK, PBCC
802.11h	To comply with European regulations for 5GHz WLANs Products have TPC and DFS 54 Mbps maximum link rate per channel Modulations used: OFDM, CCK, PBCC
802.11i	Supplementary for security in 802.11 a, b, and g Provides an alternative to WEP with encryption methods and authentication procedures
802.11n	Coming standard to provide high throughput
802.11s	Created to specify how nodes interact to form a mesh

which is perfect for broadcasting environments. The second one is called beam-forming. This kind of antenna concentrates its signal into one direction, making it stronger, thus able of reaching farther places than the omni directional antenna. Using these types of antennas one can create a radio node that typically has one of them. However people have added more than one antenna to the radio nodes to increase the transmission capacity of them. Different configurations can be achieved as seen in figure 2–1. If the sender has multiple antennas and the receiver just one then the system is multiple-input single-output (MISO). This type of system is good for sending redundancy or multicasting. If on the other the sender has one antenna and the receiver has multiple antennas the system is called a single-input

multiple-output (SIMO). A SIMO system is good for receiving information from many sources at the same time. Now finally if the system uses multiple antennas at both sender and receiver it will be called a multiple-input multiple-output (MIMO) system which has all the previously mentioned advantages.

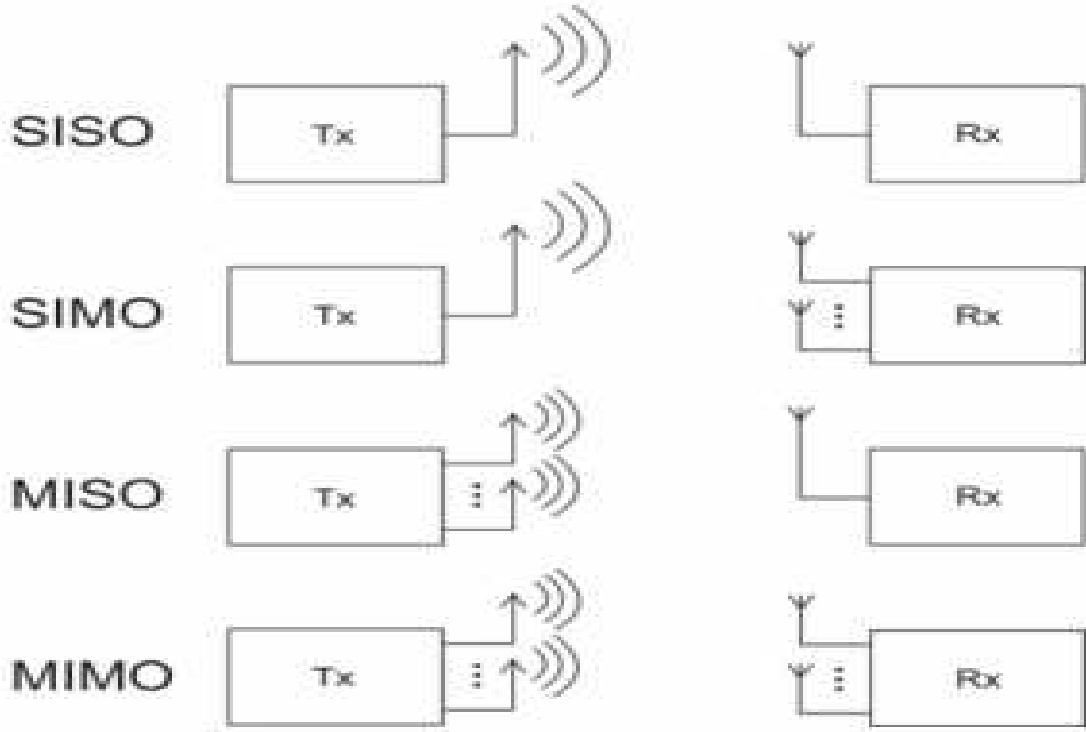


Figure 2-1: Types of Node Configurations

The next step for understanding the physical layer in WMN is to study the different signal modulations that exist at it. One of the most important is OFDM which uses orthogonal sub-carriers to send data, thus eliminating the cross-talk between sub-channels of the transmission.

Figure 2-2 presents the ideal transmitter of an OFDM system. As seen the first step is get the serial sequence of bits and separate them into individual bits. Then each bit is mapped to its corresponding OFDM code and an input signal $X[k]$ is created, that is the same original signal after being mapped. This $X[k]$ is then applied the Inverse Discrete Fourier Transform using the inverse of the Fast Fourier Transform algorithm. Then the real part and the imaginary part of the signal are

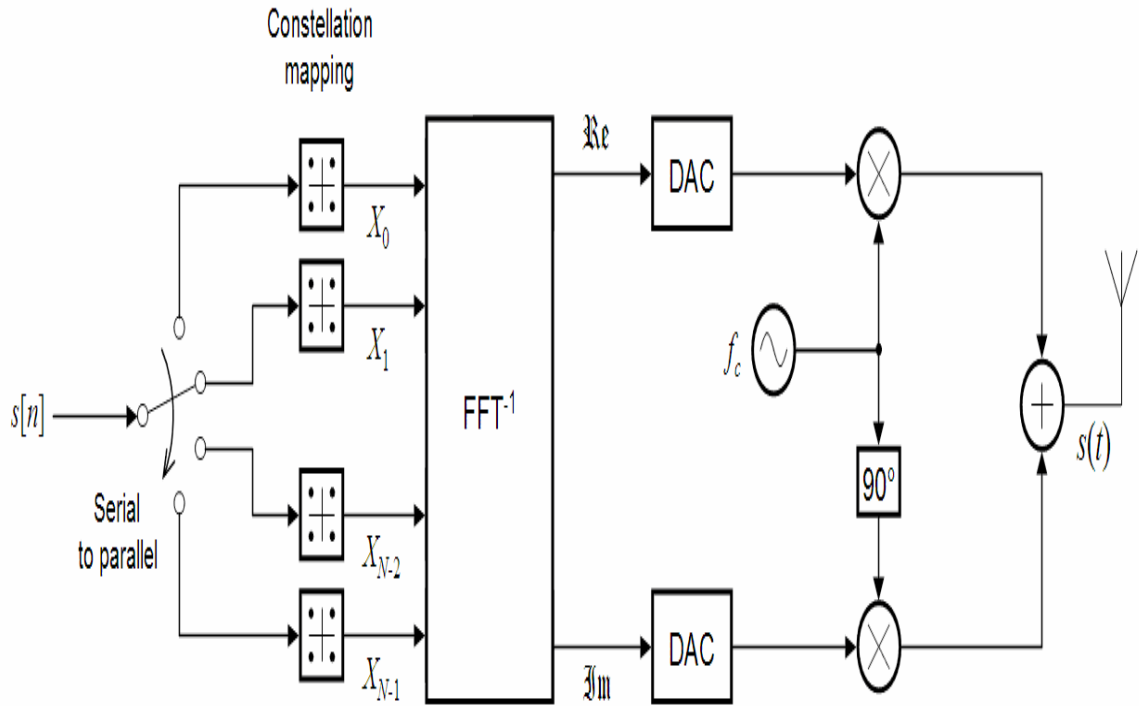


Figure 2-2: Ideal OFDM Transmitter

individually converted into analog form namely $\text{real}(t)$ and $\text{imag}(t)$. Now $\text{real}(t)$ is multiplied by a cosine with frequency f_C and $\text{imag}(t)$ is multiplied by the same cosine except that is first shifted by 90 degrees transforming it into a sine wave, thus giving $\text{real}(t)\cos(2\pi f_C t)$ and $\text{imag}(t)\sin(2\pi f_C t)$ respectively. These two can now be added together and then be send out to the medium to be received by another node.

On the other hand figure 2-3 shows the idea receiver of an OFDM system. The procedure is pretty natural, is basically undo the changes done at the transmitter. First of all the signals are multiplied by the same cosine and sine waves. This part is actually one of the problems in OFDM because both the transmitter and the receiver have to know this frequency so that the decoding can be done successfully at the receiver. Now the two signals are passed through a low pass filter and entered as one signal $x(t)$ into the Discrete Fourier Transform, using the Fast Fourier Transform

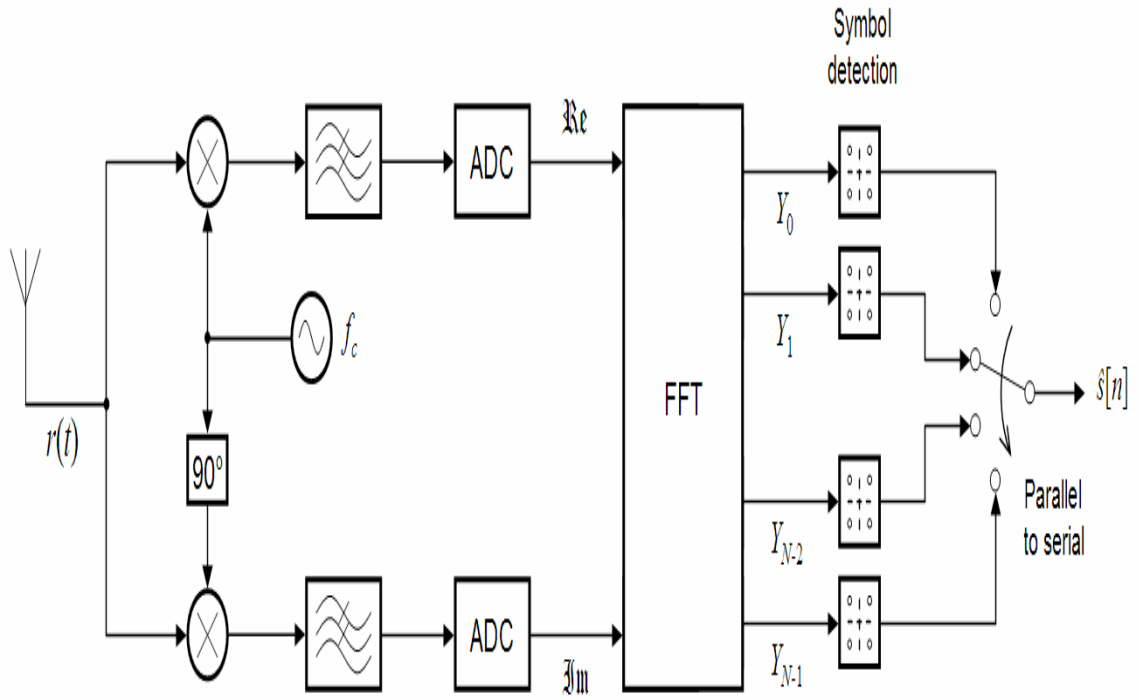


Figure 2–3: Ideal OFDM Receiver

algorithm. The output of the FFT is then correctly mapped and returned into a sequence of bits, ending the receiving process.

2.2.2 DATA LINK LAYER

The next layer to be passed in the protocol stack is the Data Link Layer (DLL). This layer is in charge of ensuring the ability to transmit and receive. It also changes information from signals to data frames and vice-versa. It is mainly divided into two sub-layers, namely the Medium Access Control (MAC) and the Logical Link Layer (LLC) which provides all functionality required for transmissions and independent of topology, medium and medium access control. The MAC is far more complex and will be discussed in the following sub-section.

MEDIUM ACCESS CONTROL SUB-LAYER

The MAC sub-layer is responsible for scheduling transmissions as well as verifying if it is possible to send or not. This includes several services, among which one

can find authentication, association, privacy, data transfer, integration and power management.

The authentication service is divided into two parts: authentication and de-authentication. The authentication step is not really needed, since any node can associate to the mesh normally. However it is recommended, for security and performance issues, to make mandatory the specification of a password to be able to connect to the mesh. This password is typically known as Wired Equivalent Privacy (WEP) and only nodes that specify this WEP key correctly can have access to the mesh. If either the password is incorrect or the mesh has a filter that does not allow access to a the certain computer trying to access the mesh then the process of de-authentication is performed at the mesh, denying then the access of that certain computer to the mesh.

Next we have the association services, which include: association, de-association and re-association. First we have the enabling and disabling of wireless links between clients and the rest of the mesh, in association and de-association respectively. The association process itself is composed by three states: unauthenticated and unassociated, authenticated and unassociated, and authenticated and associated. The process of re-association occurs at the moment that a node moves from one default gateway to another, thus has access to the mesh using a different default gateway. However although the process of re-association is specified in the 802.11 standard it does not specify how the coordination between gateways is performed.

Now we discuss briefly the privacy service. This service uses again the WEP key to encrypt data. WEP uses a 40 bit encryption algorithm known as RC4 to encrypt and decrypt the data, meaning that only clients that have been authenticated can correctly decipher the data. Another service that could be mentioned along with privacy is the integration service. This a function that enables logical integration between wired LANs and 802.11 LANs. Finally before passing to data transfer

service we should mention the power management service. This service defines basically two power modes: active and power save. The active mode is when a node is transmitting and receiving information. In the other mode, power save mode, one could have different energy consumptions depending on the task that is being performed, e.g. consume less energy when the node is only waiting for a transmission. It is important to notice that during this mode the node cannot receive or send information, to do so it has to get out from power save mode into active mode. The power consumption itself is not specified by the standard and it depends on the implementation. Several MAC protocols have been designed to specialize in low power consumption, among them we can mention SEA-MAC (Simple Energy Aware Medium Access Control)

Next we will discuss the primary service in the MAC layer, the data transfer service. This service is in charge of the exchange of data frames between nodes. The nodes use a media access scheme known as Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). Wired networks on the other hand use CSMA/CD where CD stands for collision detection, this however cannot be done for wireless networks for two reasons. The first reason is that a node is unable to send and receive data at the same time, since the medium is half-duplex, for reasons explained in the physical layer section. The other reason is that in CSMA/CD all nodes can hear each other, this cannot be guaranteed in wireless networks because of the hidden terminal problem. In figure 2-4 nodes B and C are in communication range with each other. However A and C cannot hear each other, now if A transmits to B and C transmits to D a collision will occur at B due to the two transmissions at the same time in the same range.

Collision avoidance mechanisms are related to deterministic networks, where response time is predictable. The 802.11 standard specifies two different access methods: the Distributed Coordination Function (DCF) and the Optional Point

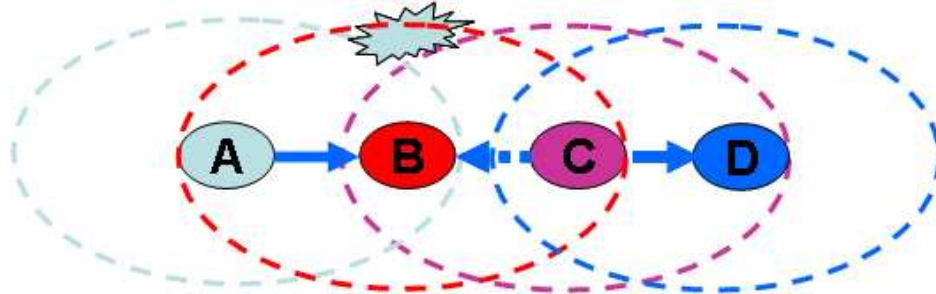


Figure 2-4: Hidden Terminal Problem

Coordination Function (PCF). In DCF a node that wishes to transmit has to sense the medium first and wait a specific time known as the Distributed Inter Frame Space (DIFS). If during this time the medium remains free then it sends the frame else it reschedules the sending of the frame for later. The receiver node then computes the Cyclic Redundancy Check (CRC) to verify if there were errors and sends an ACK if everything is correct. This lets know the sender that there were no collisions. If no ACK is received in a period of time then the sender sends the frame again. This mechanism introduces a delay that is unacceptable in some applications such as VoIP or video streaming. For these cases PCF comes in, with this function there is a central node in charge of scheduling when the nodes can transmit. Typically this node is called an Access Point (AP) for normal WLANs and the default gateway for WMNs. Nodes cannot transmit unless the default gateways says so. Still the hidden node can cause collisions at any time during transmission. To prevent this from happening a mechanism called Virtual Carrier Sense (VCS) is used. As seen in figure 2-5 the mechanism is divided into a Request To Send (RTS) RTS request and a Clear To Send (CTS) response. The way it works is as follows a node that wishes to transmit sends an RTS to reserve the space in a given time. The RTS contains the source address, the destination address, and the time that the transmission will take. The destination then replies if possible with a CTS indicating that the medium is free.

OLSR: First we start with Optimized Link State Routing (OLSR) protocol. OLSR is optimized for mobile ad-hoc networks but can also be used on other wireless ad-hoc networks. It uses Hello (figure 2–6) and Topology Control (figure 2–7) messages to discover and then discriminate link state information throughout the mobile ad-hoc network. Individual nodes use this topology information to compute next hop destinations for all nodes in the network using shortest hop forwarding paths.

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Reserved										Htime										Willigness											
Link Code					Reserved					Link Message Size																					
Neighbor Interface Address																															
Neighbor Interface Address																															
..																															
Link Code					Reserved					Link Message Size																					
Neighbor Interface Address																															
Neighbor Interface Address																															

Figure 2–6: OLSR Hello Messages

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
ANSN															Reserved																
Advertised Neighbor Main Address																															
Advertised Neighbor Main Address																															

Figure 2–7: OLSR Topology Control Messages

OLSR also has a mechanism to inject default routes on the system called Host Network Address (HNA). This allows a node to bypass the neighbor discovering stage, thus speeding up the routing process.

OSPF: Next we discuss Open Shortest Path First (OSPF). OSPF is a successor of Routing Information Protocol (RIP). It has been developed for Ad-Hoc networks but it can also be used in wired networks just like RIP can. OSPF divides the networks into areas, labeled with numbers. By convention area 0 (0.0.0.0) is reserved for the core of the network and all OSPF areas must be connected directly or virtually to it to be able to communicate among them. Other areas have as id a numeric 32 bit number which it is advised to be used in dotted form (x.x.x.x). Although it seems to be an IP address it is just a convention and not an IP address itself, addresses inside of the areas can be the same as long as they do not belong to the same area. It will occur that at some point two routers form adjacencies and one of is selected as the designates router. This happens when the two routers are sending Hello messages (figure 2–8) and they see themselves inside of the message. The other as the backup designated router to reduce the traffic between routers. OSPF has reserved two multicast addresses 224.0.0.5 for all SPF routers and 224.0.0.6 for all designated routers. Also OSPF has its own error detection and correction code, so it does not depend on TCP, thus it works directly with IP.

OSPF denotes several type of routers depending on their responsibility in the network. First we have the area border router (ABR) which is the one that is connected to more than one area. Another to be mentioned is the autonomous system boundary router (ASBR) that is the same as an ABR except that is connected to autonomous systems. The internal routers (IR) are those routers that are local to an area and backbone routers are the ones inside of the core of OSPF.

Finally the designated routers (DR) are the routers that each area selects to be the core of it. There is only one DR per area and this one creates the spanning tree

+	Bits 0–7	8–15	16–18	19–31
0	Version	Type	Packet Length	
32	Router ID			
64	Area ID			
96	Checksum		Authentication Type	
128	Authentication			
160	Authentication			
192	Network Mask			
224	Hello Interval		Options	Router Priority
256	Router Dead Interval			
288	Designated Router			
320	Backup Designated Router			
352	Neighbor ID			
384	...			

Figure 2–8: OSPF Hello Messages

of the area using the Dijkstra’s algorithm taking the cost (e.g. interface bandwidth) as routing metric. It is responsibility of the DR also to flood the spanning tree to the rest of the routers in the area so they can have the updated version of the routes. The DR is elected using the router with highest id (priority) at the moment of election, that is neither a router with priority zero nor a router that is down can be elected as DR. If there is a tie in priorities then the router with the highest IP address in one of its interfaces is elected as DR. The second option is then elected as backup designated router (BDR), which will become DR if at any time the DR fails. If this happens then the process of electing a new BDR is repeated.

BATMAN Now we mention the Better Approach To Mobile Ad-Hoc Networking (BATMAN) protocol. BATMAN is rather simple, it does not work in the traditional manner that routing protocols work, there is no centralized data in any node. Instead BATMAN learns information about the direction from which data was received and broadcasts the learned information to all of the neighbors. Thus when sending

Table 2–3: DSDV Routing Table Example

Destination	Next Hop	#hops	Seq #	Install Time
A	A	0	A 46	001000
B	B	1	B 36	001200
C	B	2	B 28	001500

information from one node to another, the intermediate node locally analyzes which is the best next hop to be sent rather than the whole route. It then sends the packet to that next hop where the BATMAN process will be redone.

DSDV The Destination-Sequenced Distance Vector (DSDV) routing protocol is one of the oldest, if not the oldest, ad-hoc protocols and is not very used today. However it is import to know it since protocols used in todays ad-hocs, such as AODV and DSR, are based on the sample principle.

DSDV is a table-driven protocol that uses the Bellman-Ford algorithm to compute its route. Each entry in the routing table (table 2–3) contains the destination address, next hop, number of hops remaining to reach the destination and a sequence number, which is generally even if a link is present or odd if a link is not present. This number is generated by the destination, and the emitter needs to send out the next update with this number. Routing information is distributed between nodes by sending full dumps infrequently and smaller incremental updates more frequently.

DSDV selects the routes based on these sequence numbers, if there is a tie then the option with better routing metric is chosen. Routes that are not updated in a long time are deleted from the routing table. The only problems with DSDV are that, due to the constant updates to the routing table, DSDV is only well suited for networks with few nodes also these updates make the nodes to consume energy even when the network is idle.

AODV Ad-Hoc On-Demand Distance Vector protocol is a successor of DSDV. Just like DSDV it used sequence numbers in its routing table, however in contrast to DSDV it only calculates routes when is necessary. Is is able of working with both

unicast and multicast messages and it has a time to live (TTL) field, indicating how many times can the route messages be retransmitted. In AODV route request contain the source identifier, the destination identifier, the source sequence number, the destination sequence number (just like DSDV), the broadcast identifier, and the TTL field. One disadvantage of AODV is that it may lead to inconsistency in routes if there is too much difference between the sequence numbers in the source node and the sequence numbers in the intermediate nodes.

DSR The last protocol to be discussed is Dynamic Source Routing (DSR). This protocol routes using the source addresses rather destination. When a source wants to communicate with a destination and it does not know the route to take it sends out a route request. This request is retransmitted, as long as it does not exceed the TTL counter, by every other node except the destination. When the destination receives this request it then sends back the reply along the same path where the route is constructed. To avoid creating overhead of storing all the IP addresses of the sources along the way, DSR uses a flow id instead of the addresses.

The main advantage of this protocol is that it reduces the network flooding to learn the routes, however this introduces a greater setup delay and if a link is locally broken it cannot be easily repaired.

POST-ROUTING

On the other hand in Post-Routing protocols the responsibility is the delivery of the packet from one node to another independently of the route that the packet is going to take to reach it. Here we can mention protocols such as IP, ICMP, IGMP which used on WMNs as well as for traditional LANs.

ICMP and IGMP: ICMP or Internet Control Message Protocol is an extension of the Internet Protocol (IP). It travels using a datagram inside of the IP header, however ICMP packets are treated as a special IP packet and not as normal data packets. ICMP as the name suggests, is used to send error messages if something

occurred that did not allowed to finish a request (e.g. Destination Unreachable). The ICMP header starts typically at the bit number 160 of the IP header (figure 2-9) unless other IP options are used. The organization goes as follows: bits 160-167 is for the type of the message, bits 168-175 is for the code which is an extension of the type. Following these two bytes is the checksum to verify if an error occurred and is 2 bytes long (bits 176-191). Afterwards the next four bytes are used for id and sequence two bytes each in case that an echo reply is needed.

The Internet Group Management Protocol (IGMP) on the other hand can be used to configure the memberships of muticast groups. It is analogous to ICMP with unicast, perfect for broadcasting applications (e.g. video streaming).

IP: IP is an essential part of the protocol stack. Whenever a node wants to access resources outside of the LAN is going to need an IP address to identify itself and another to identify the destination to which it wants to communicate. The way a node works is that it does not know the destination's physical address it proceeds to make an Address Resolution Protocol (ARP) request which will find the physical address of the machine that has the specified address, thus allowing the node to create the whole packet to be send. In an analogous manner RARP (Reverse ARP) finds the IP address of a known physical address, this one is typically used by routers that might not have knowledge of MAC addresses, still IP is the heart of all the communication process. As a matter of fact one could change the MAC protocol, as well as the transport protocol and still it would use IP in the current architecture.

Now traditional IP is not enough in WMN because of the mobility of the nodes. There comes a time where a node changes its default gateway or mesh point (MP) because is no longer in range of it. Now normally what nodes do is use Dynamic Host Configuration Protocol (DHCP) to obtain the configuration to communicate automatically. This is not exactly what we want to happen if we change MP. If we use

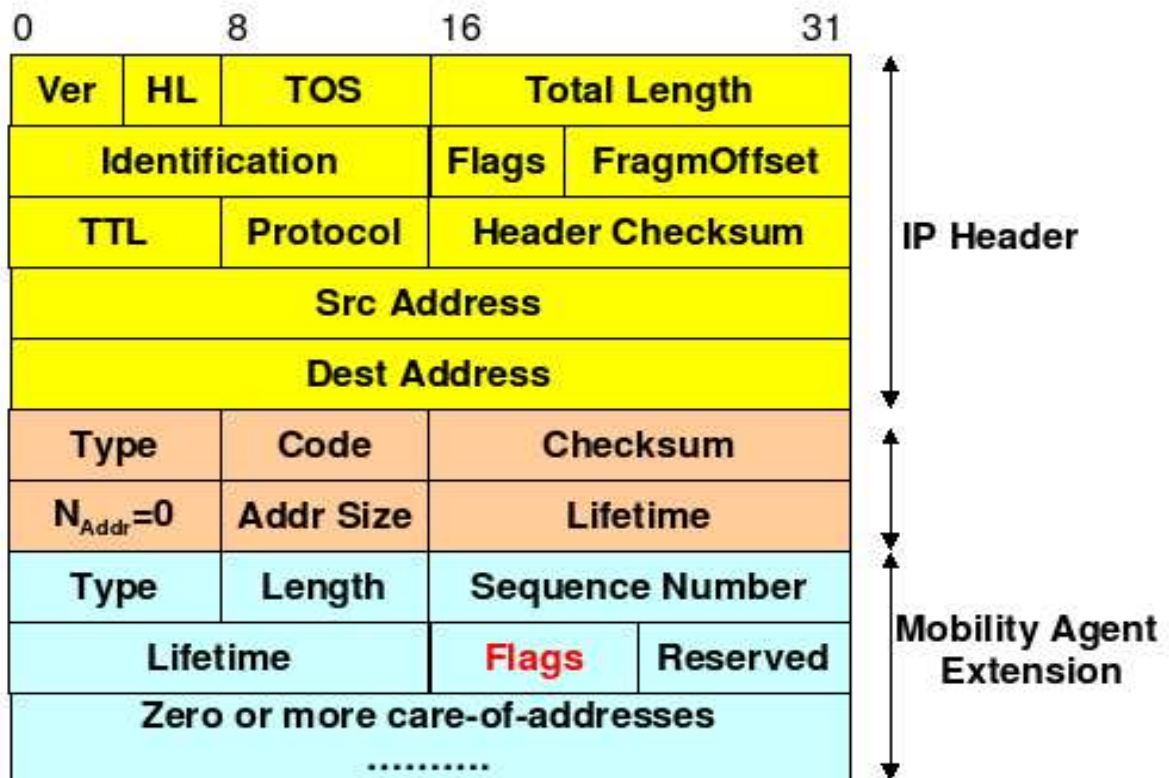


Figure 2–9: IP Header

DHCP to configure the node what will happen is that we will be able to communicate with the outside world normally, however other nodes that are expecting our node to have the old configuration will not be able to reach it. Thus another mechanism is needed, one that allows nodes to keep the old IP address even when the MP is changed. For this purpose an agent is proposed (figure 2–10). The way the agent works is that each node has a permanent address known as home address and an IP address that changes depending on the location of the node.

For example let us consider the scenario in figure 2–11 where C wants to communicate with M. Now what happens is that C creates the IP packet with destination address equal to the home address of M and sends the packet to the network. Then the home agent H receives this packet and redirects it to the foreign agent F using normal routing, assuming that M was previously registered in H. When F receives the packet it then realizes that the packet is meant for M and sends it to its original destination M.

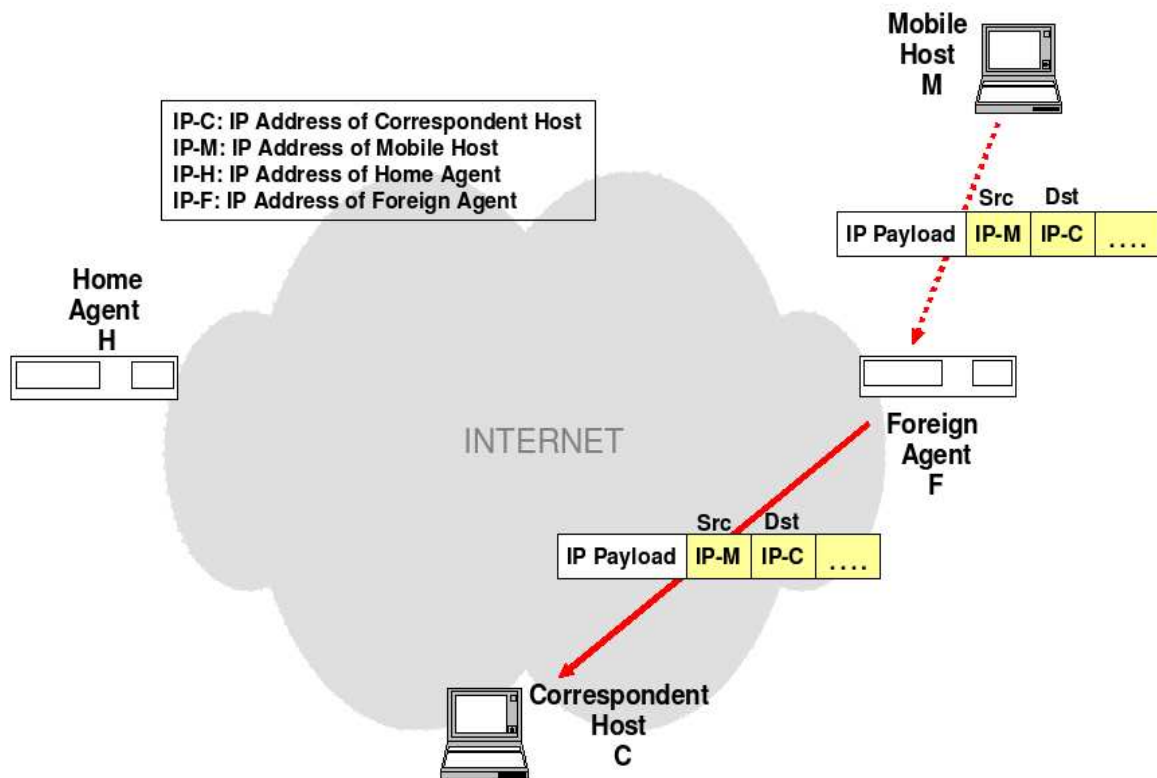


Figure 2–10: Mobile IP Agent

2.2.4 TRANSPORT LAYER

Passing on now to the transport layer. This layer is in charge of providing reliable connection from the source node to the destination, without caring on how many intermediate nodes are in the middle. Several protocols work at this layer. Among them one can find the Transport Control Protocol (TCP), and User Datagram Protocol (UDP).

TCP

TCP is a transport protocol that always guarantees the delivery of segments. However it uses IP to do its job, which does not guarantee at all the delivery of packets. To correct this issue then it is the job of TCP to control the flow of segments and schedule them for optimal performance. TCP has several tasks to achieve this:

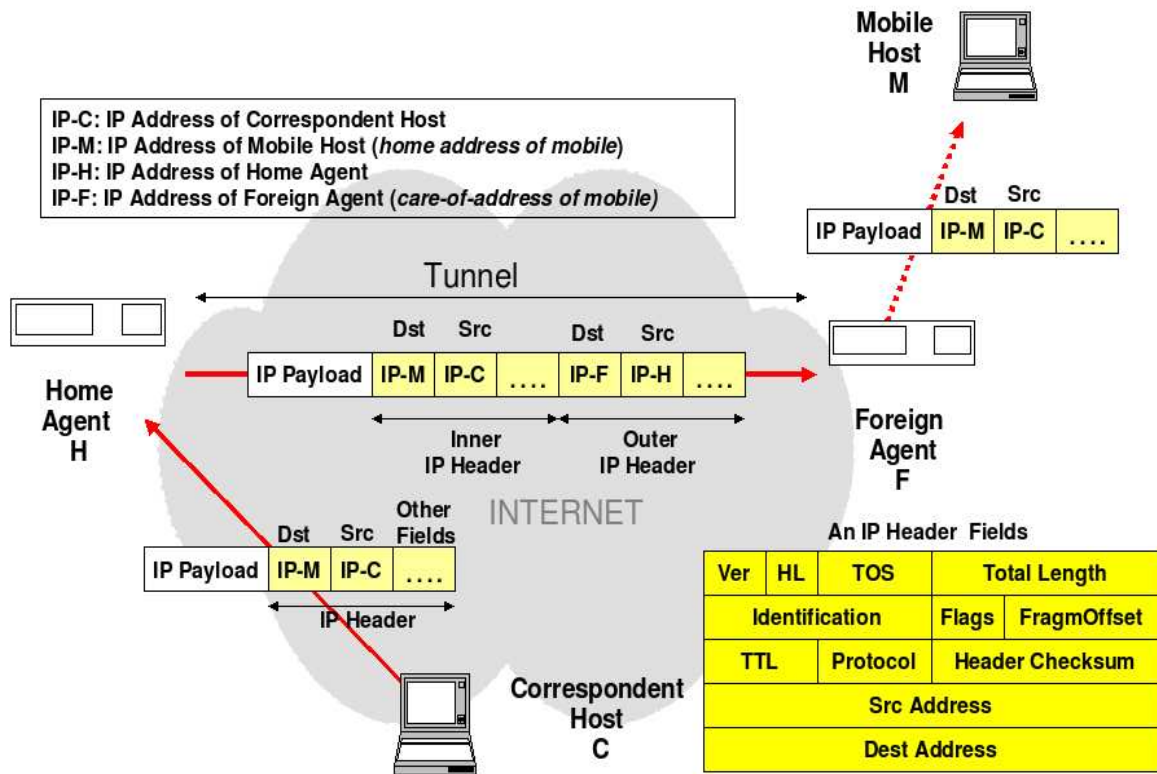


Figure 2-11: IP Agent Example

CONNECTION: Connection establishment and termination: TCP establishes a connection using a triple handshake (figure 2-12). This procedure involves three steps to establish a connection. The first is a synchronization segment used to ask for a connection. Then the destination acknowledges the SYN message to the source. When this occurs the client sends an ACK for the SYN ACK and then starts sending the data. This procedure makes sure that indeed both ends are ready to communicate before starting the actual communication.

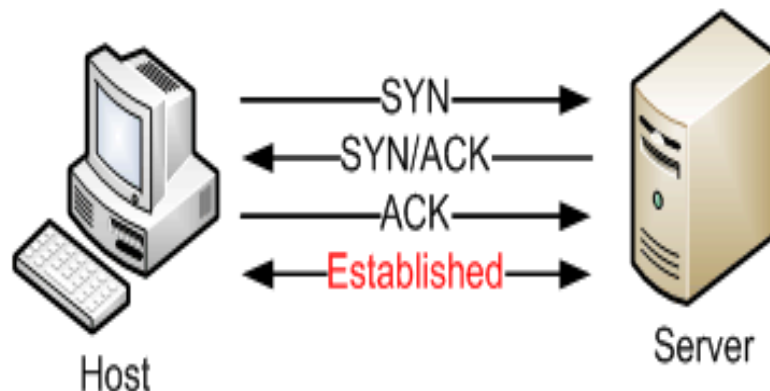


Figure 2-12: TCP triple handshake

To end a communication a similar process is done, only that sometimes it takes four steps instead of just three. Both ends must agree to end the connection, this is done in the following manner:

1. FIN:source \longrightarrow destination
2. FIN ACK:destination \longrightarrow source
3. FIN:destination \longrightarrow source
4. FIN ACK:source \longrightarrow destination

SEGMENT SIZE: When sending a segment from one end of the network to another we have to consider that at some point during the transmission the medium could change. Physically each type of medium has a Maximum Transmission Unit (MTU) size that if this is exceeded then the segment is too high and has to be fragmented. For this reason TCP first sends a discovery to find the MTU of the path that will be used and adjusts the segment to that size by fragmenting it into smaller segments. Then at the receiving end TCP re-organizes the segments to recover the original segment. The typical MTU is 1500 bytes.

DATA TRANSFER: TCP delivers segments using a sliding window mechanism. This also allows TCP to control the congestion of the network and minimize the collisions occurring. When a segment is delivered successfully the receiver sends an ACK to indicate that it received the segment. If this ACK is not received in a period of time the sender retransmits assuming that a collision occurred, also this causes a decrease in the sliding window, which indicates how many segments can be sent at the same time. ACKs are sent with a sequence number one unit greater than what it wants to be acknowledge to tell the sender that everything was successfully received and that the next sequence can be sent.

On the receiver's side TCP is in charge of re-ordering the segments in case of fragmentation and discarding duplicates of segments that may have been received.

UDP

UDP is a much simpler protocol. It does not guarantee the delivery of segments but is faster than TCP. This speed is due in fact to the lack of checks that UDP performs. Segments in UDP are sent using datagrams, meaning that there is no guarantee that all segments of a sequence will travel through the same path nor that they will arrive in the same order. However UDP is frequently used for applications that require real-time reactions where speed is a must (e.g. voice, video, and audio). If the applications using UDP need transmissions to be reliable they have to take care of it by themselves. UDP only checks that its header is arrived correctly, if this happens then it proceeds to decode the segment and pass it on to the Application Layer.

2.2.5 APPLICATION LAYER

The final one in the layered model is the Application Layer. This layer is equivalent to three layers in the OSI reference model: session, presentation and application. Thus the layer has to be in charge of tasks done by those three layers. Network applications that use multiple users have to manage the sessions to schedule when users will have the chance of using the network. Application must also change the user input to the bytes that will be processed by the network and later on changed to signals to be transmitted. It must also change the bytes received from the network stack into information that users can understand. Table 2-4 shows some examples of network applications and indicate if they use TCP or UDP as transport protocol.

RTP AND RTCP

Previously when we discussed UDP it was mentioned that it does not guarantee the delivery of segments from node to another and that it is responsibility of the application layer to make sure that the delivery is done correctly. As seen in table 2-4 UDP is still widely used for many important and promising applications. For this

Table 2–4: Examples of network applications

Application	Protocol	Port
FTP	TCP	20, 21
SSH	TCP	22
TELNET	TCP	23
SMTP	TCP	25
DNS	UDP	53
DHCP	UDP	67, 68
TFTP	UDP	69
HTTP	TCP	80
MYSQL	TCP	3306
RTP	UDP	5004
RTCP	UDP	5005
VoIP	UDP	≥ 1024
IPTV	UDP	≥ 1024
Online Games	UDP	≥ 1024

reason the Real-time Transport Protocol (RTP) and Real-time Transport Control Protocol (RTCP) were developed. These two are small layers built on top of UDP that help to make sure that data is delivered correctly. One thing they do for example in video streaming, they create two sessions, in one they send the audio and in the other they send the video. Using this approach makes it lighter to transmit, plus users do not get the idea that the application is frozen and it could happen that make errors tolerated.

2.3 SERVICE-ORIENTED ARCHITECTURE (SOA)

This section introduces the concept of SOA. Typically any transmission requires a source and a destination. These two parameters introduce mostly two facts that can be seen easily 1) the source needs to find out the IP Address of the destination, and 2) this introduces ARP requests that may result in broadcasts in the network. Another thing that we have to consider is the case when the user has an extensive list of hosts to select from (mirrors), the user might naively choose an option that may be too far away from the source. This will introduce an unnecessary delay that could also create a bottle neck at the mirror due to other connections that may exist in it.

With these issues a new approach comes to be. An architecture that focusses on the services being provided (FTP, SMTP, TELNET, etc.) rather than the traditional source and destination approach. This is a cross-layer design that benefits from the information that is known to other layers in order to improve their own layer functionality; i.e. a routing protocol might take some decisions if the physical layer is wired and others if it is a wireless environment, thus improving the performance of the network. Figure 2-13 shows an example of a service-oriented mesh with several servers providing services such as FTP and a client who wants to find the best option.

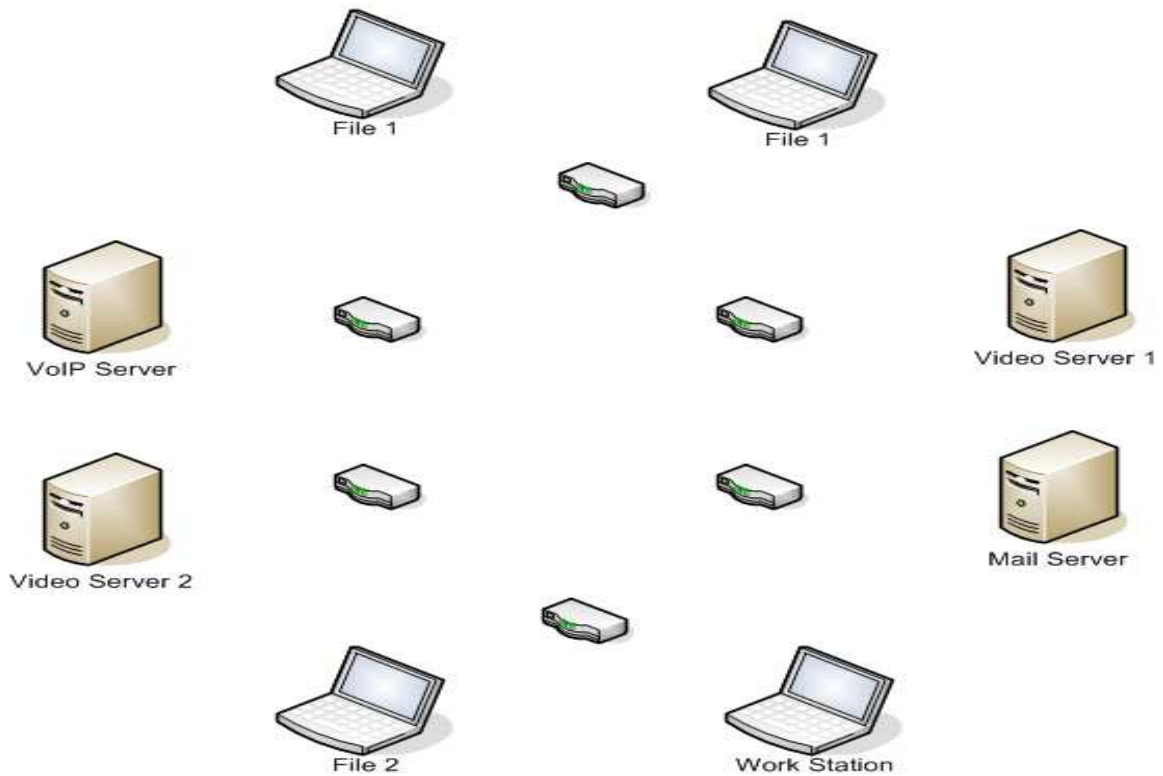


Figure 2-13: Example of a service-oriented wireless mesh network

In this thesis we proposed a routing algorithm namely SORA (Service Oriented Routing Protocol). SORA will be discussed in full details in the Implementation section.

2.4 FUZZY LOGIC

In the beginning any application involving electronic commutation used Boolean Logic in order to make decisions. With it if an input sensor read a value lower than a certain threshold then the value is considerate as low (false) otherwise high (true).

Figure 2-14 :

Boolean Logic.

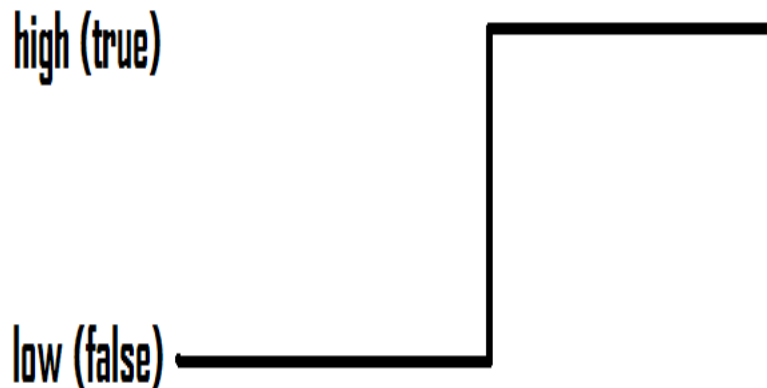


Figure 2-14: Boolean Logic Curve

Sometimes however it is hard to determine whether or not something is false or true, because it may be partially false or true. Boolean Logic doesn't allow us to make distinguish of these events, thus Fuzzy Logic is born as an attempt to model in electronic circuits the human ability to make partial decisions.

For example let us consider the scenario where we want to control the temperature of a room using an air conditioner. We define then a hot temperature as any value above 85 degrees Fahrenheit. Now with boolean logic anything below that value would be cold, however the fuzzy logic allows us to say that from 80 degrees to 85 degrees the temperature is partially high. This fuzzy set definition then can be seen as a S shape figure having $\mu(x)$ equal to 0 (completely not hot) when x is 80 degrees and $\mu(x)$ equal to 1 (completely hot) when x is 85 degrees. Any value of x between these two values will give a $\mu(x)$ equal to a value between 0 and 1. In this scenario $\mu(x)$ is the degree of truth in the temperature being hot. Formally a fuzzy set F can be defined as the pair (A, μ) where A is a set and μ is a function that moves from A to $[0, 1]$.

2.4.1 FUZZIFICATION

In figure 2-15 we appreciate the transition between a logical low to a high. In the y-axis we have the degree of truth of an event, in this case high event, -4 corresponds to a low while 4 corresponds to a high.

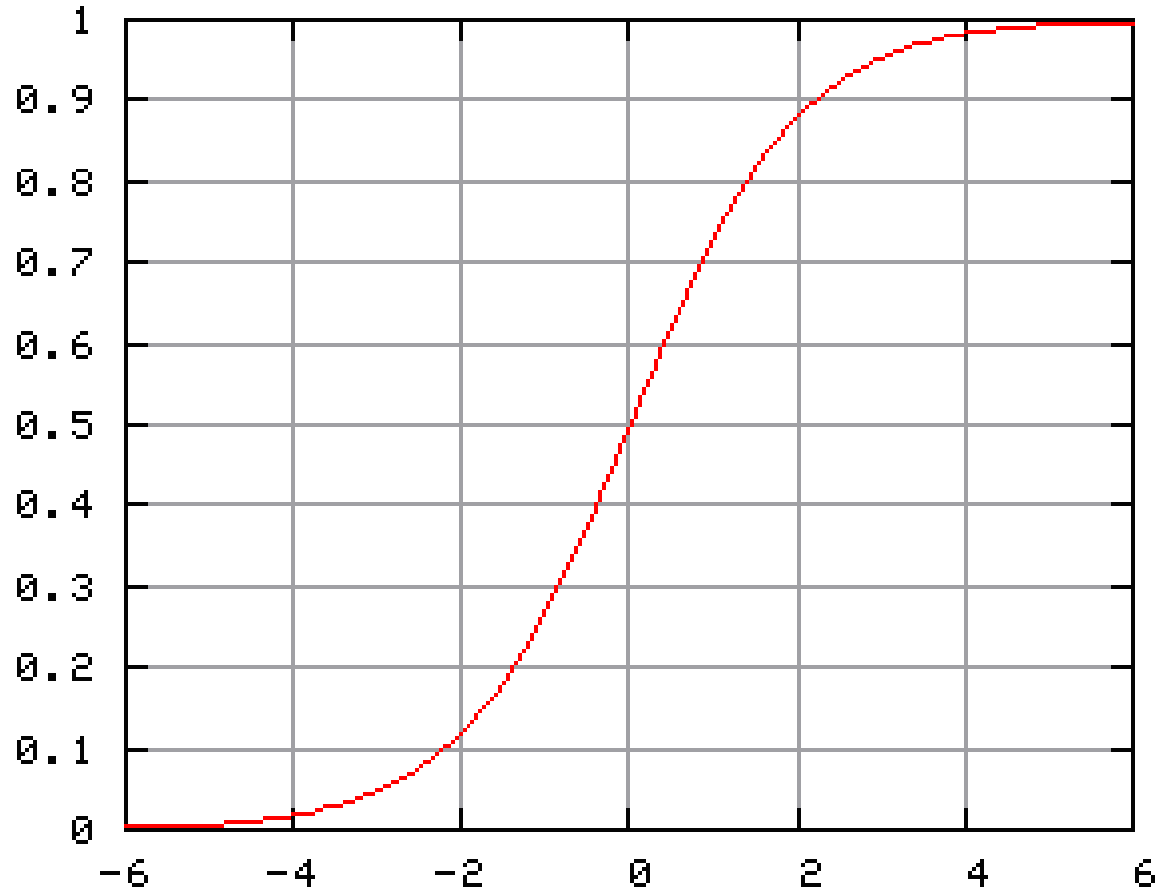


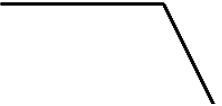
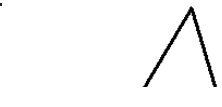
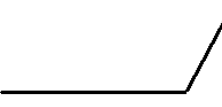
Figure 2-15: Typical S Shape Fuzzy Curve

Anything between is considered as partially truth. In the x-axis we find the values of inputs which are mapped to a value between 0 and 1 to indicate its truth in an event. This mapping is called fuzzification of the data and the curve is referred to as a fuzzy set.

2.4.2 IF-THEN RELATIONS

The next step in fuzzy logic is based on the fuzzy sets to make an inference on the output. For example if the collisions over a wireless channel are too high one would want to limit the number of transmissions occurring through that channel.

Table 2–5: Fundamental Fuzzy Sets

Type	Shape	Frequent Usage
Z Shape		Low
Triangular Shape		Medium
S Shape		High

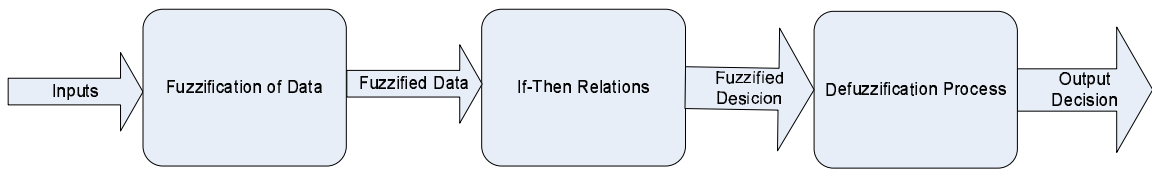


Figure 2–16: Order of Fuzzy Logic Process

More complex relations can be achieved by using the logical operators and/or/not. Thus we can then consider if the number of collisions is high or if the utilization is high and the throughput is low in our decisions of channel selection. The same rules of De Morgan in Boolean Logic apply to Fuzzy Logic thus:

$$\text{not}(A \text{ or } B) = \text{not}(A) \text{ and } \text{not}(B) \quad \text{not}(A \text{ and } B) = \text{not}(A) \text{ or } \text{not}(B)$$

In table 2–6 we can observe how the logical operators are defined for Fuzzy Logic, if we use this definition in Boolean Logic we can verify that these definitions are correct for all kind of logic. Given a logical if-then relation one can find the result by finding the minimum or maximum of the memberships in the fuzzy sets depending on the logic operator joining the expression (see table 2–6). After this value is found is used as a limit, do to the fact that an effect cant be more true than its causes, to find the membership of the output. Lets call this limit L then the new output membership is defined as:

$\mu_{new}(x) = \min\{\mu_{old}(x), L\}$ $\mu(x)$: value of x in the fuzzification. It ranges from 0 to 1 included.

Table 2–6: Definition of Logical Operators

Operation	Boolean Truth Table			Fuzzy Definition
Not	Input		Output	$1 - \mu(Input)$
	0		1	
	1		0	
Or	A	B	Output	$\max\{\mu(Input1), \mu(Input2)\}$
	0	0	0	
	0	1	1	
	1	0	1	
	1	1	1	
And	A	B	Output	$\min\{\mu(Input1), \mu(Input2)\}$
	0	0	0	
	0	1	0	
	1	0	0	
	1	1	1	

Table 2–7: Defuzzification Methods

Type	Description	Formula
Max Criterion (MAX)	The value of x such that the membership achieves its maximum.	First $arg_x max(\mu)$ or Last $arg_x max(\mu)$
Mean of Maxima (MOM)	Mean of all x's where the membership achieves its maximum.	$\sum(arg_x max(\mu))/N$ where N is $count(arg_x max(\mu))$
Center of Area (COA)	Calculates the center of area of all the maximums	$\sum(x \cdot \mu(x)) / \sum(\mu(x))$

2.4.3 DEFUZZIFICATION

The final phase of a fuzzy process is to return from the fuzzy set of the if-then result to the real world, in other words the decision. Once we have this output set there are several ways of getting the value needed.

Table 2–7 has a list of methods to get the value needed as decision of the if-then operation. In terms of performance the results are as follows: MAX ; MOM ; COA.

2.5 NETWORK CODING

As seen up to this point data is individually encapsulated between headers of the different layers to send them from node to node. Depending on the amount of information in headers overhead is created and sometimes for applications like those

dealing with voice, data is too small and the packet being send is mostly composed of headers rather than data. For these cases and for the cases when one wishes to multicast multiple packets it is necessary to reduce the number of transmissions required to send them. Thus network coding is proposed. Network coding is a technique that mixes data at intermediate nodes using mathematical operations then the destination is able to decode it to its original packets by applying the inverse operation. Figure 2-17 shows an example of how network coding acts in a basic network. This configuration is known as the butterfly network. Here the two upper nodes send A and B respectively. These two packets are received by the upper intermediate node. Also the left lower node received A and the right lower node received B. On the next transmission, the intermediate node sends $A + B$ to the next intermediate node which will forward it to the lower end nodes. Now each one had either A or B and received $A + B$, hence they can subtract the packet they already knew from the sum and decode the other packet. This technique can be applied with software at the network layer or with hardware at the physical layer. Physical layer is faster and perfect for communications to end nodes, however if one is searching for communicating with the intermediate nodes network layer network coding is more suitable since it allows intermediate nodes to decode the data as well.

Examples of type of network coding algorithms are:

- Linear Coding: Uses addition to mix N packets. The destination needs to have stored N - 1 packets to decode the new one.
- Random Linear Coding: Uses coefficients to multiply the bytes of the packets and then add them together into one. It is able to mix N packets, where each packet is a linear equation of the form $\sum(c_i \cdot p_i)$. The destination needs, to decode the original packets, receive at most N encoded packets and multiply them by the inverse of the matrix of chosen coefficients. However it can happen that the node already posses some of the original packets, thus reducing the size of the matrix.

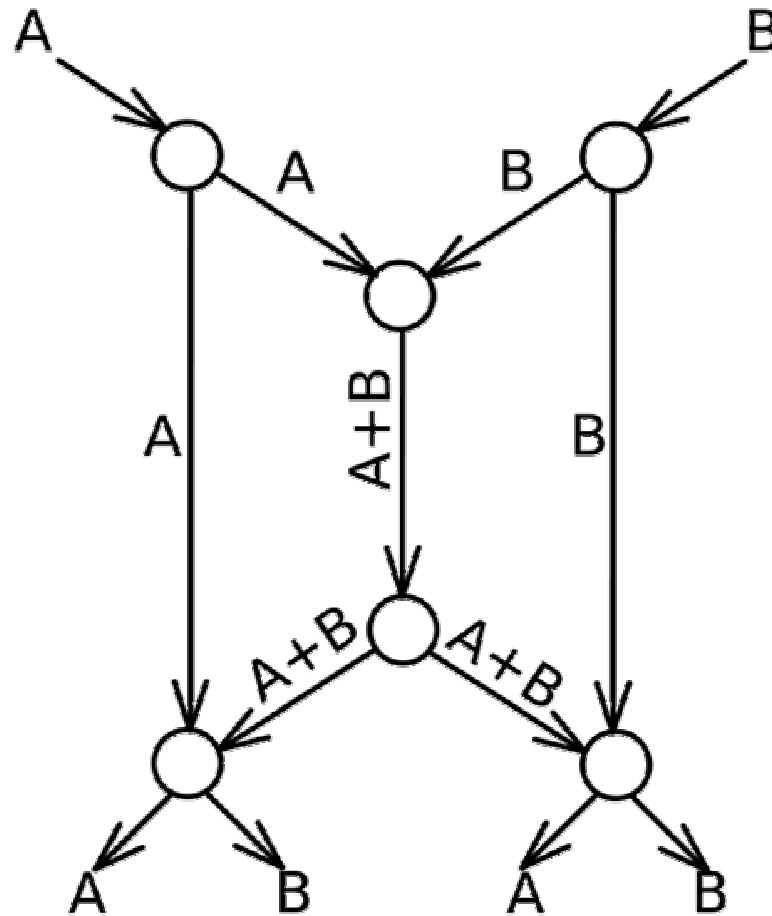


Figure 2–17: Network coding on a butterfly network

Previously linear coding has been used on other applications in networking such as error detection or correction codes (e.g. cyclic codes, hamming codes, parity codes, etc.). Now with network coding even more applications are available for linear codes. Linear coding theory is a branch in mathematics that deals with the issue of ensuring communication over a noisy channel. An important property of a linear code is that if we choose two values from the the space we can add them togheter and the result will still be inside of the same space. The same goes for multiplying the value by a scalar, since this can be seen as a multiple addition. Thus the linear coding is used to encode the data to be sent then at the other end the decoding is performed once again staying in the same space and recovering the data.

2.6 SECURITY

Security in wireless networks is an issue of great importance. The main problem with the wireless medium is that its broadcast nature prevents the signal from being controlled to a specific location. Being the medium shared makes then the signals able to be received by anyone nearby. Thus practices to make the network more secure are needed.

One of these is to change the default Service Set Identifier (SSID) and the default password, since hackers tend to use these to gain access to the equipment. Another good thing to do is to disable the broadcast of the SSID since the users of the network already know it and there is no need to public it into outsiders. Finally encryption of data using either WEP or even better WPA; and adding firewall along with mac address filtering services is highly recommended to keep intruders out of the network.

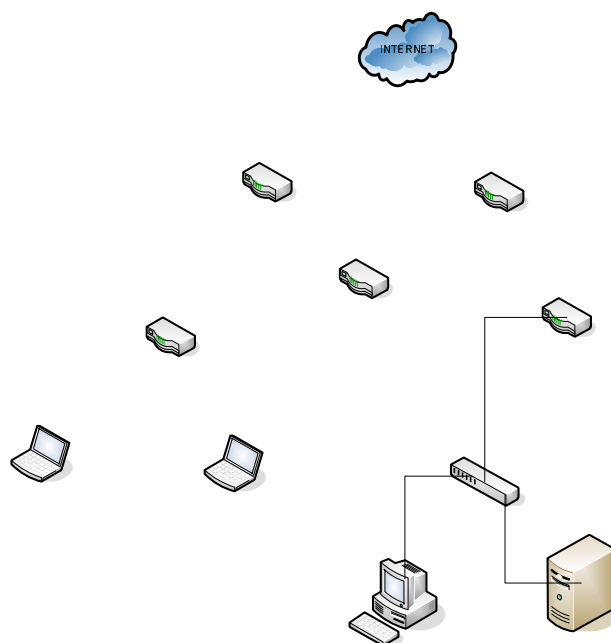
CHAPTER 3

RELATED WORK

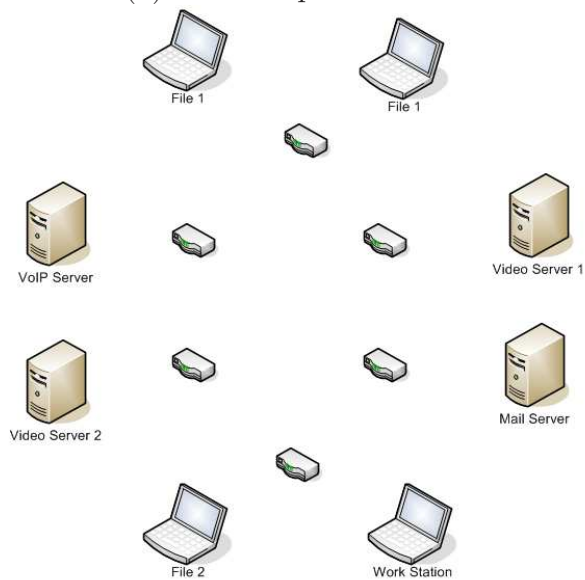
This chapter discusses related work in the area of wireless mesh networks, service-oriented architecture, channel selection, and network coding. Each is briefly described and analyzed its pros and cons, while contrasting with what we have done in this thesis. The chapter itself is organized into sections, one for each area of this thesis.

3.1 WIRELESS MESH NETWORKS

According to [1], WMN can be considered as a type of wireless ad-hoc network because nodes can automatically establish network and maintain the connectivity in an ad-hoc manner. In addition, the WMN can also be dynamically self-organized and self-configured. Compared to many previous ad-hoc network designs, the main feature of WMN is that mesh routers inside the network can provide a backbone for routing packets in wireless channels. An example of WMN is illustrated in Fig. 3-1(a), in which mesh routers are set up on top of big buildings and mesh clients are deployed at the rooftop of residential houses.



(a) An example of WMN



(b) Services in WMN

Figure 3–1: Examples of wireless mesh network.

Authors also classified the architecture of WMN into three types:

- Infrastructure/Backbone WMNs where mesh routers form an infrastructure of self-configuring, self-healing links. If mesh routers have gateway functionality, then they can connect to the Internet which is named as infrastructure meshing because it provides a backbone for mesh clients and common clients with Ethernet interfaces that connect to mesh routers via Ethernet links.
- Client WMNs where nodes create peer-to-peer networks. In this architecture, clients perform configuration and routing functionalities as well as end-user applications to customers.
- Hybrid WMNs. In this type, we combine the two previous architectures, infrastructure and client meshing[Figure 3]. Hybrid WMNs allow mesh clients to connect directly to other clients and to mesh routers through which they could have access to other networks such as Internet among others.

In addition, this work states the importance of the MAC layer in the sense that this layer is concerned with more than one-hop communication, need to be collaborative, and work for multipoint-to-multipoint communication.

In the near future, WMN is expected to support diverse services such as voIP, content distribution, etc. Fig. 3-1(b) illustrated such a scenario, in which we can observe that different devices can be connected to the mesh network and provide a variety of services. Now the next interesting issue is how can a user of WMN efficiently obtain necessary services.

For example, in Fig. 3-1(b), suppose a user of work station (WS) is attempting to download a video clip, which is available in video servers 1 and 2. With the existing IP protocol stack, the user shall have known the IP addresses of these two servers and must specify which one it shall be connecting to. Apparently, such an approach could be inefficient because the user does not know neither the state information of the wireless network, nor the operational conditions of the servers.

3.2 SERVICE-ORIENTED ARCHITECTURE

In recent years, the concept of service-oriented design and service-oriented architecture have been discussed in previous work. The Organization for the Advancement of Structured Information Standards (OASIS) [2] defines SOA as a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. Specifically, the SOA shall provide a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

There has been several work done in which SOA is proposed as an alternative to today's demand diversity. Most of these works however, have approached this issue at the application level. The problem with this approach is that since we do not have certain information that it is known at the network layer, such as the routing metrics, we have to make some steps to find out that information, thus introducing a delay to the system.

In [3] an Autonomic Service Delivery Platform is proposed. The author presents service-oriented networking (SON) as an emerging middle-ware and telecommunications architecture. The author also discusses some challenges in building service-oriented and networks (figure 3-2) and integrating them into the current architecture. The platform presented routed requests from service consumers to providers. The author states that SON provides exciting new multidisciplinary research opportunities in service-oriented computing, hardware, software, and networking; stating as well that in order to fulfill the requirements for migrating into SOA, the creation of new routing protocols using SOA is of great need.

A number of studies have been focused on the architecture itself. In [4], the authors suggested the *Model Driven Service Oriented Architecture* (MDSOA) as a

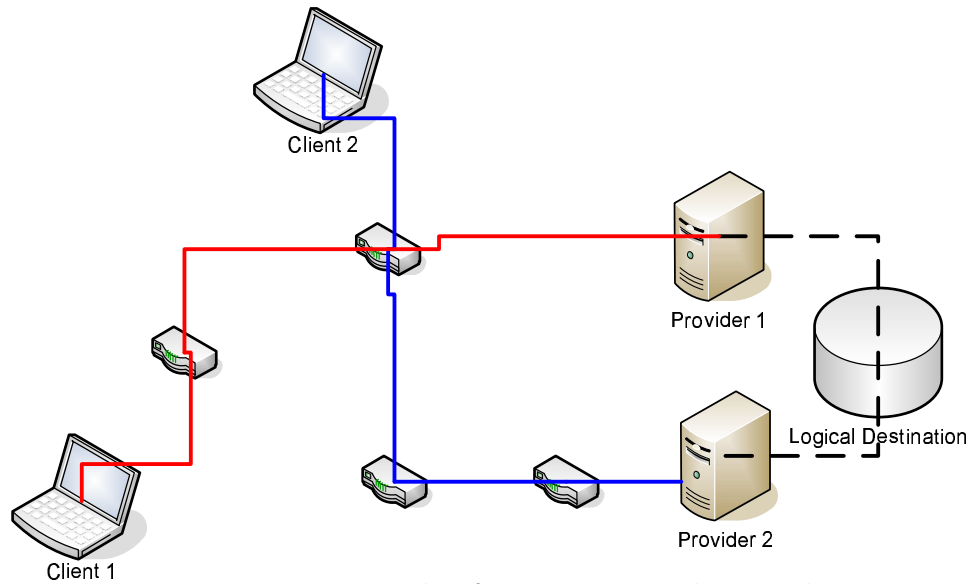


Figure 3-2: Example of service-oriented network

solution for integrating new technology, such as ad-hoc communications and protocols, along with the opportunity to unify solutions for organizations, while having in mind the scalability of the network.

In [5], the author presented SOA fundamentals using J2EE and a router as a proxy server to support the needs of telecommunications of today and next generation networks. Thus helping service providers to deploy and manage services faster and with lower cost. The authors clarified that without SOA it is not possible to provide to the content providers a common framework, hence solving issues just from specific applications without having in mind the integration of new ones.

The authors in [6] made an analysis of the impact of SOA in the current business and how is important to integrate SOA in the current world. There were two things that the author says are promised by web services: a drastically reduction in the time-to-market of new products and more value for consumers due to the constant evolution of web services.

In [7], the authors studied the possibility of using legacy software to help with this integration. Several options are presented: applying reverse engineering to the code to move it into another language and taking advantages of the current languages

features. However this is time consuming for which a second approach is presented, wrapping the code using an executable code to access it through an interface, but this has its limitations in performance since some code may no be worth reusing. Finally the third approach merges the previous ones by making the service available as a web service, thus creating a transformation over the code. This approach is preferred by the authors. For this three steps were suggested:

1. Analyzing the legacy code to determine if it is worthy of reusing it. This can be done with the actual code or with reverse engineering tools.
2. One the code to be reused has been selected the next step is creating a wrapper code using existing interfaces. A tool named SoftWrap was developed to automate the interfacing for languages like PL/I, COBOL, and C/C++. Using SOAP [8] the parameters for the functions written in these programming languages are seen as XML objects.
3. The final step is integrating the code to a web service. This is done by using a proxy that forwards request to the legacy code.

Many other research work focused on the service perspective of SOA, in particular as a technology with great scalability and easy control of services in large networks.

With the service-oriented network layer design [9], the network layer will dynamically determine the destination of the request, with certain quality-of-service (QoS), security, and reliability considerations. In the next subsection, we will discuss the service-oriented design from a broader view, instead of the network layer.

Multiple applications to SOA have been presented. First we can mention air transportation [10], among the services provided one can find Flight Plan Evaluation, Weather Forecasting, Traffic Congestion Prediction, and many more. The benefits of integration using SOA found were: efficient creation of new or expanded services, cost reduction for both development and operation; as well as improved maintenance

since services are built for adaptable changing both in terms of additional servers as for faster or larger servers. Authors also developed a five layer SOA architecture, where they included:

- Users: the most top layer it includes humans as well as other systems.
- Application Services: these span or combine basic data to produce new kinds by using peer or lower layer services.
- Information Services: these are responsible for collecting and delivering fundamental data types used through the NAS. The key data types were Flight and Flow Data, Surveillance Data, Weather Data, Aeronautical Data, and NAS Status Information.
- NAS Infrastructure Services: is the bottom layer, responsible for core middle-ware services. This includes registry services, message brokers, security infrastructure, infrastructure management, and access to telecommunications and transport services.

Following we have E-Healthcare in [11]. Healthcare applications and professionals, even now these days, deny the usage of computers in their field. This makes even harder to centralize the services. For this reason authors proposed SOA as a way of integrating and centralizing services for physicians, nurses, pharmacists and other health care professionals, as well as by patients and medical devices used to monitor patients. With SOA it is possible to reduce errors cause by misunderstanding of prescriptions and at the same time increases the speed and effectiveness of the service, because communication between physicians and pharmacists can be done instantly by electronic ways.

In [12] the smart home application is presented with the analogy of a robot with sensors to learn about the environment. Four layers are taken into account when designing the SOA smart home:

1. Perception Layer: obtain necessary information about the context.

2. Context Layer: gives sense to the perceived data.
3. Inference Layer: gives artificial intelligence to the house allowing it to reason over the contextual data.
4. Action Layer: acts accordantly to the decisions taken by the inference layer.

Our society has given in majority two meanings to smart homes. The first providing health care for incapacitated and the second focusing more on the multimedia and comfort parts. No matter what scope we take into account in a smart home, the authors showed that SOA is a viable solution because of the ease when adding or removing services.

Another useful application of SOA is business integration. The authors of [13] introduces a Java Business Integration (JBI) based application named JTang. This application utilizes SOA, because it is independent of the context and state of services. It contains four components (figure 3-3): Basic Information Platform based on SOA and referencing to JBI, Supported Framework used to develop and assemble service, Administration Tools use to administrate the whole system, and Development Tools to provide tools to assist in the application integration development. [14] shows a SOA design using a monitor (3-4) and transactions principles of *begin* to start services, *commit* to end successfully the services, and *abort* to cancel the services transactions.

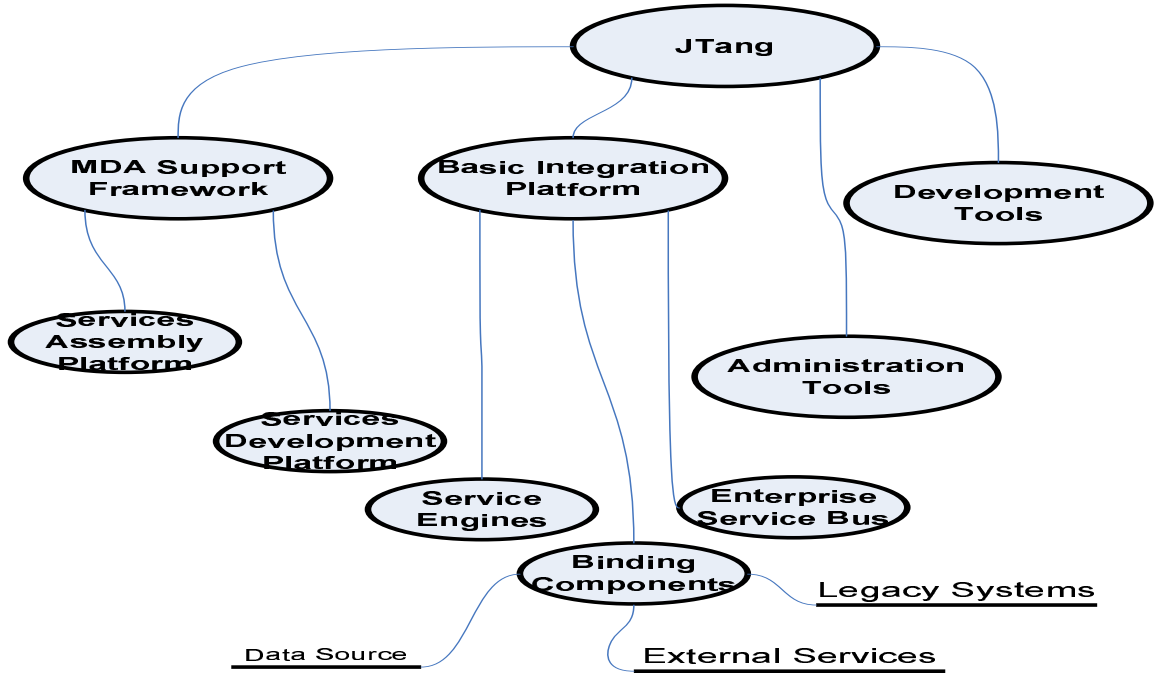


Figure 3-3: JTang components

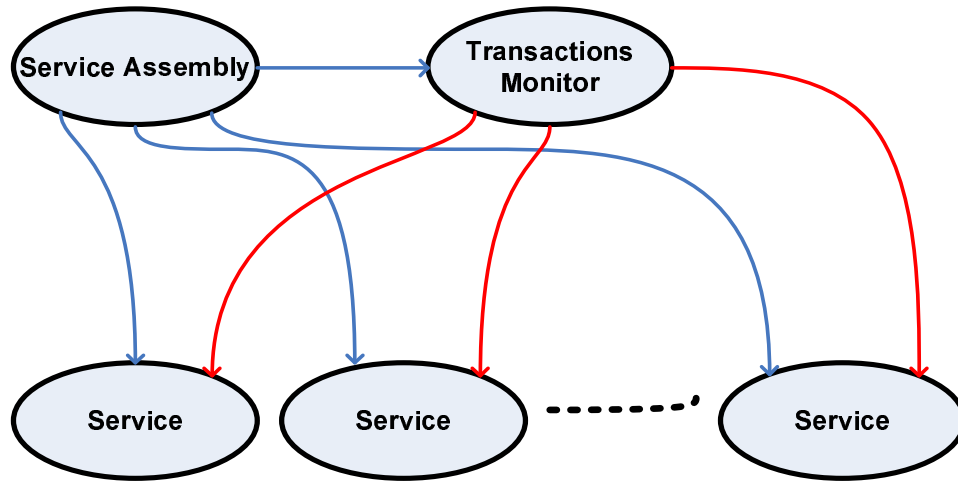


Figure 3-4: Database approach to SOA

Other types of applications that have been discussed as possible applications for SOA are network monitoring, information exchange and quality of service (QoS). [15] worked in the development of web services for military communications networks using the principles in web applications as seen in figure 3-5. [16] worked in SOA as a case of study for network management. In [17] a network monitoring application is presented using SOA, hence taking benefit of it to bypass the diversity of networking tools that exist and centralize them in one manner.

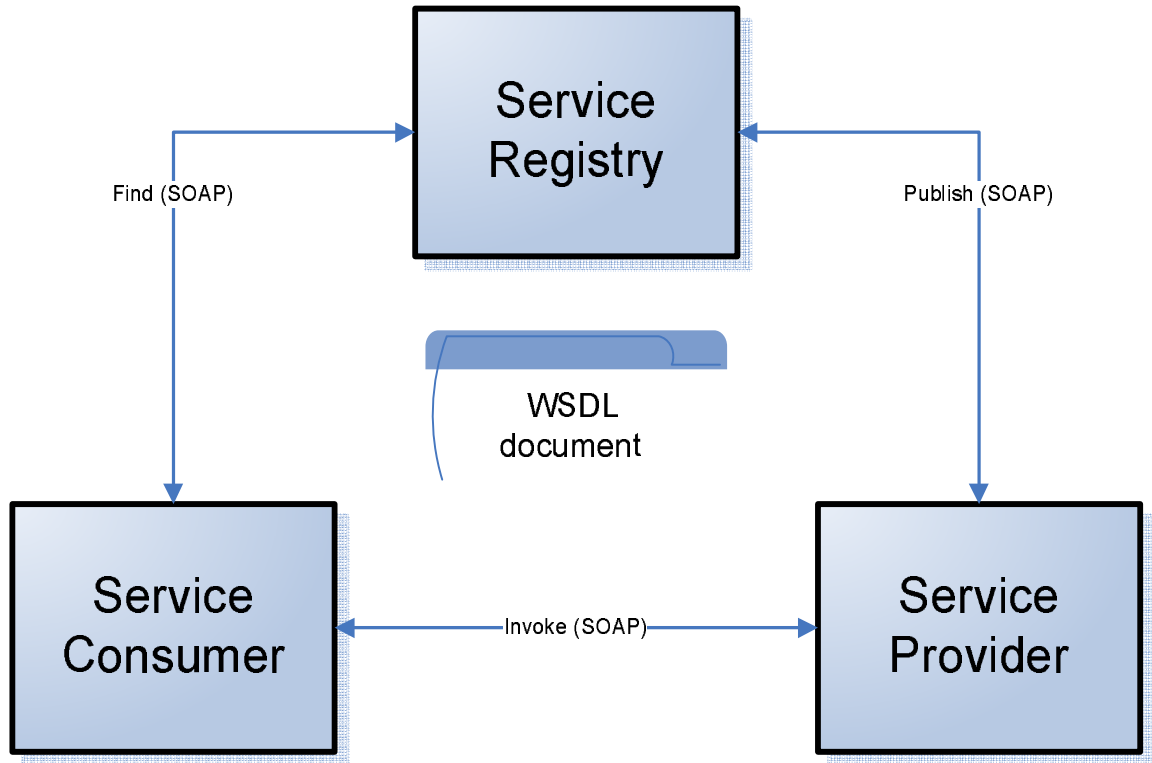


Figure 3–5: Web approach to SOA

In [18] the authors aimed their studies into three issues: dependability requirements of SOA in pervasive computing, how these may change during their life cycle, and suitability of Service Discovery technologies. Authors also summarize the typical SOA scenario into the following phases:

- Infrastructure Discovery: end-user gets into the environment and using some devices discovers the available points to enter the infrastructure.
- Client Registration: user connects to the infrastructure by identifying its own device and preferences.
- Service Registration: the service provider publishes the services it has.
- Service Discovery: user discovers services that best fit its needs.
- Service Selection: user analyzes the services and selects one.
- System Configuration: the infrastructure and the service configure themselves to satisfy the client.
- Service Delivery: the requested service is delivered to the client.

Table 3–1: Actions for fault tolerant environment

Event	Action
Unavailable Infrastructure	Detection
Unavailable Service	Change service object
Session Failure	Session recovery
Server Crash	Switch to another server
Client Crash	Session abort

The authors also developed a failure model for SOA, under which they defined the following failures: Unavailable Infrastructure, Unavailable Service, Session Failure, Server Crash, Service Failure, and Client Crash. For these failures the authors defined the actions seen in table 3–1 to create a fault tolerant environment.

In [19] a reliability study of SOA is done by focusing on the quality of service. Finally, some work have addressed the design of SOA using tools like UML to make an analysis of material control in [20]. [21] created a supply management architecture named MIDAS. MIDAS is based on SOA and it provides services for information flow on multiple supplies in a low cost environment for both client/supplier and supplier/manufacturer sides. The authors of MIDAS also plan to design support for product and financial flows.

3.3 CHANNEL SELECTION

Typically a wireless node has one radio interface, allowing it to use one channel at one given time (figure 3–6). It is a fact that wireless nodes with more than one radio interface are capable of achieving greater performance (figure 3–7), although the computational power required may be greater. Figure 3–8 shows the 2.4GHz frequency band for most WMNs. At most 3 channels can be used without overlapping with others.

There has been some work related to the topics used in this work, but frequently the multiple channel assignment is done in a static way, because it has been shown that even with a static assignment performance is greatly improved and not much energy is consumed by the protocol. In a WMN this static assignment

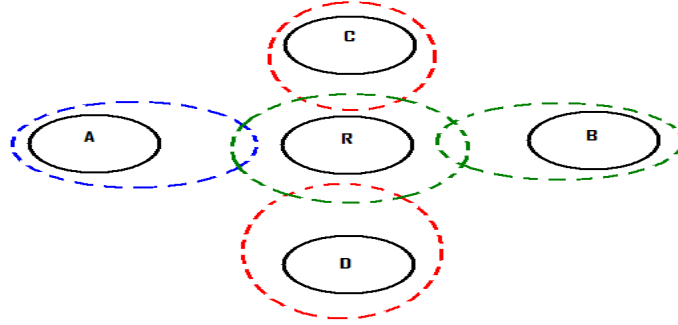


Figure 3-6: Single-radio WMN

causes problems when new nodes are added because the configuration of channels was made without having them into account. For this reason a dynamic channel assignment is suggested.

[22] discusses a way of deploying sensor nodes in a distributed system using a fuzzy optimization algorithm (FOA). In the FOA method were taken into account the number of adjacent nodes in a given radio and the distance between them.

Meanwhile [23] uses a multiple channel environment to test the IEEE802.11n using packet aggregation which is adding redundancy data to a packet at the source node in order facilitate network coding and decoding at the destination node. However this work focuses more on the aggregation of packets, while uses the available channels in a static way with 25 multi-hop nodes topology.

[24] presents a static assignment of channels in a multi-hop multi-radio environment. The focus of the work is based on minimizing the size of collision domains using a mesh network relying on adjacent nodes to make the delivery of data.

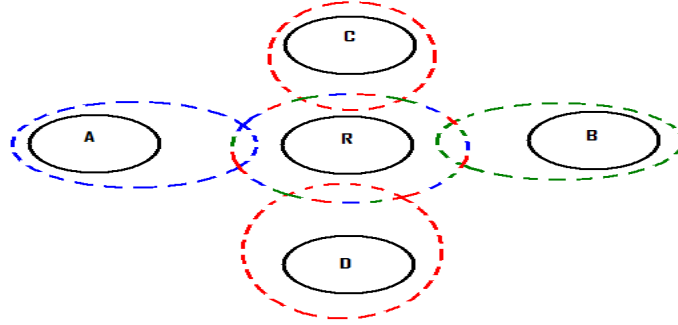


Figure 3–7: Multi-radio WMN

A description of multi-hop wireless architecture that employs multiple-radio channels simultaneously is presented in [25]. Here, the authors focus on the design of channel assignment and routing algorithms. They show that even with only two network interface cards in each node, it is possible to improve the network throughput.

On the other hand, [26] presents optimization models for fixed channel assignment in wireless mesh networks with multiple radios. The authors consider the interference constraints and try to maximize the number of bidirectional links that can be activated simultaneously. In [24] however they take into account the interference of adjacent channels and make the fixed assignment using intelligent decisions. In [27] a similar but non static solution is presented.

[28] presents how competitive networks although they are non-cooperative their channel allocation achieves load-balancing in other words nodes occupy the channels almost evenly.

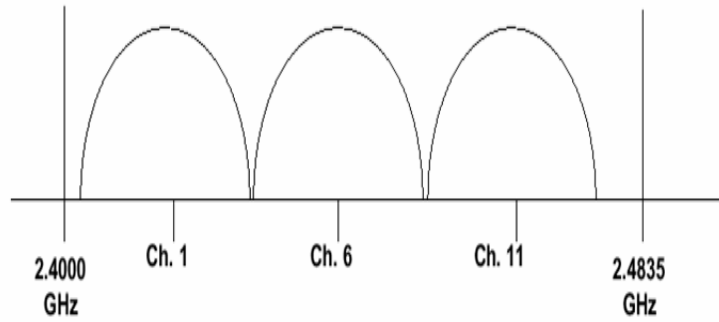


Figure 3-8: 2.4GHz frequency band

In [29] the authors created a testbed with 14 nodes to test their distributed, self-stabilizing protocol, which improved the capacity of the network by 50% over homogeneous channel assignment and 20% over random channel assignment. Meanwhile in [30] a 9-node testbed was developed, taking into account a channel cost metric defined as the sum of the expected transmission time weighted by the channel utilization over all interference channels. The protocol was written between the MAC layer and the routing layer.

Two protocols were developed in [31] without having into consideration the delay when making channel switch but performed better than MMAC and DCA via extensive ns2 simulations. Finally in [32] different protocols were classified in 4 categories: Dedicated Control Channel, Common Hopping, Split Phase, and McMAC. Here the authors examined the performance according to number of channels and devices, channel switching times, and traffic patterns on throughput and delay. The throughput, delay, and jitter improve for all types of packets, using any of the 4 protocols.

3.4 NETWORK CODING

As other wireless networks WMN are also susceptible to data collision or limited to their communication range. In [33] network coding was used in a multicast environment. Here authors discovered that the bandwidth used in multicasts using network coding is reduced in comparison with the multicasts used in IP protocols.

Furthermore in [34] network coding was combined with multi-rate transmission in order to work when there are different loads in a node which need more rate transmission than others.

Not only the use of resources is reduced by using network coding but in [35] it was shown that network coding can successfully increased robustness to Byzantine attacks and to distributed authentication peer-to-peer downloads. Also in [36] linear coding was used, which is a network coding technique that uses matrixes to encode the data, thus a new method was given in order to prevent less encrypted data to go out through the network.

As presented in [37] network coding can also help to control errors in a network yielding comparable results to those of the rateless coding and link-by-link ARQ, with this increasing performance by 65% with less overhead than conventional multicast as shown in [38]. Finally in [39] a study of the challenges encountered when using network coding was done. These challenges include the coding speed along with the fact that as seen in also in [33] if some data is lost during transmission then the coded data cannot be decoded thus it is counted as a lost as well.

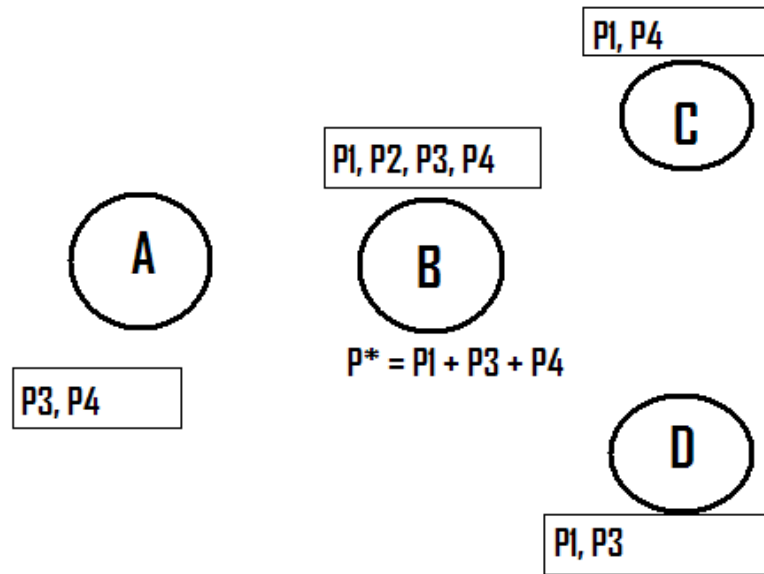


Figure 3-9: Network coding example

Following we have [40] presents COPE, a simple opportunistic network coding protocol applied to wireless networks, using probabilities as the base to decide when to apply network coding or not and packet queues to store the packets that will be mixed (figure 3–9). Also in [41] MORE is proposed as an alternative using random linear coding to create linear equations composed of packets multiplied by random coefficients defined as the coding vector (figure 3–10). MORE does not need any probabilities analysis because it sends as many equations as needed for the destination to decode the packets. However the computational power needed is greater than that of COPE because the destination node has to perform matrices multiplication.

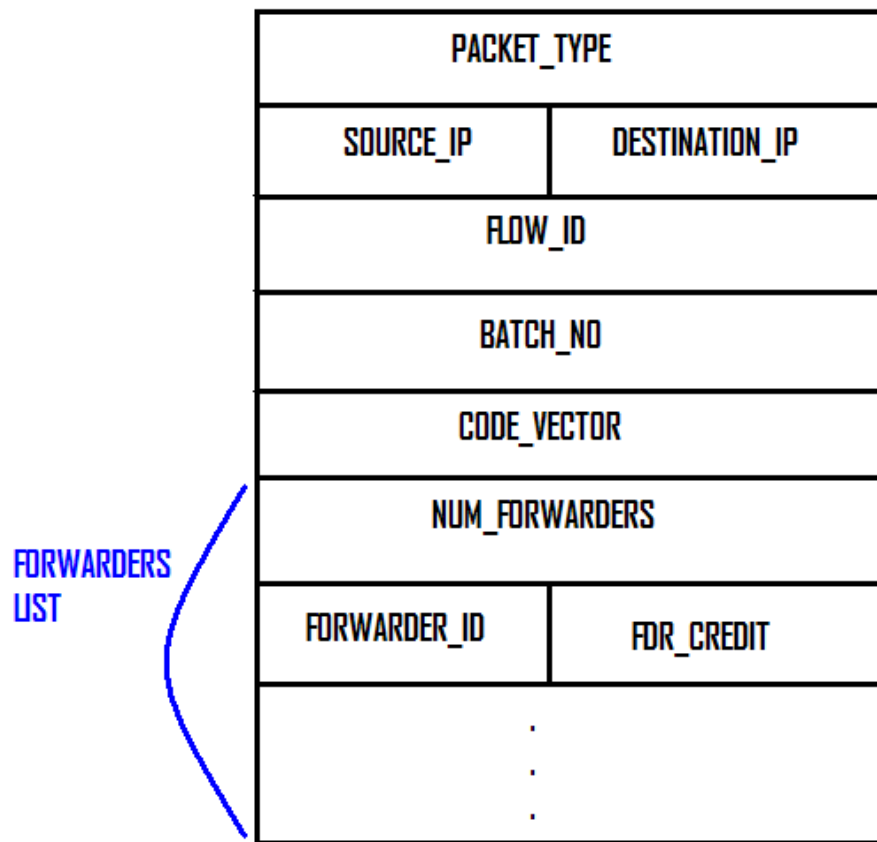


Figure 3–10: MORE header

On the other hand [42] shows another alternative of network coding, which is done at the physical layer (figure 3–11), this mechanism is faster than the one presented in [40] it does not allow intermediate nodes to decode the packets, making

it harder to use for broadcast and multicast applications. Also in [43] physical network coding was applied but this time combining information of both physical and network layers. Meanwhile in [41] SOFT is introduced as a coding used to send confidence intervals to neighbors so that recovery from errors can be done, thus making the network more reliable.

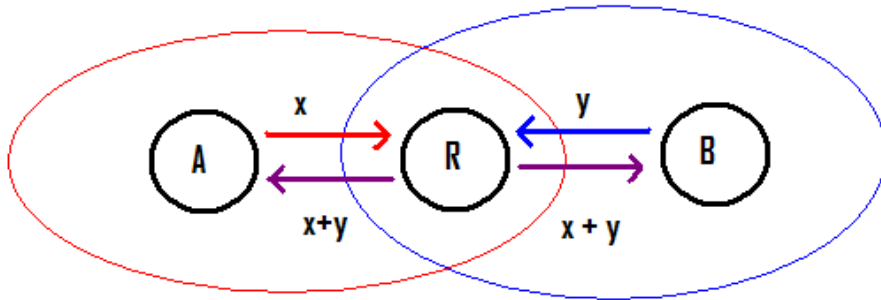


Figure 3–11: Physical network coding example

CHAPTER 4

THE TESTBED

This chapter discusses the details of the testbed used to test SORA. It is subdivided into five sections 1) hardware used, 2) software used, 3) openwrt distributions, 4) configuration of the Linksys WRT54GL, and 5) configuration of the laptop computer running WinXP. It is important to remember that the testbed was configured to work as a Wireless Mesh Network and all configuration discussed will be with that goal. Figure 4-1 contains the configuration used for the testbed. S1 and WS are connected to R1, S2 to R2, and finally S3 to R3.

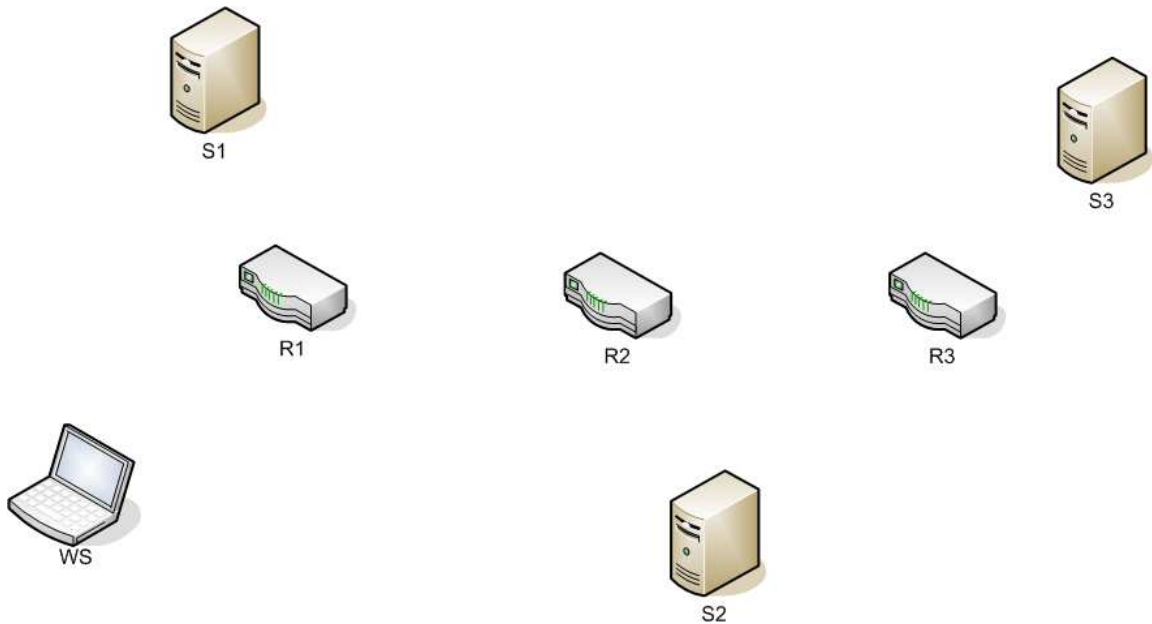


Figure 4-1: Testbed

4.1 HARDWARE USED

- Laptop Computers (x4)
- Linksys WRT54GL (x3)

Table 4-1: Dell Latitude D620 Specifications

Spec	Description
CPU	Intel Core 2
Speed	2.16GHz
RAM	1GB
NetIF	Intel PRO/Wireless 3945ABG
Architecture	Intel
OS	WinXP Professional

Table 4-2: Linksys WRT54GL Specifications

Spec	Description
CPU	BCM5352EL
Speed	200MHz
RAM	16MB
NetIF	Broadcom (integrated)
Architecture	Mipsel
OS	Kamikaze 7.09

4.2 SOFTWARE USED

- Kamikaze 7.09: An OpenWrt distribution. You can download the source code and its binaries from <http://downloads.openwrt.org/kamikaze/7.09/>.
- OLSR: An routing algorithm for Ad-Hoc networks (2.2.3).
- X-Wrt: A web interface to configure the openwrt systems.
- SSH Client: A secure shell file transfer client to configure files and add our code to it.

4.3 OPENWRT DISTRIBUTIONS

In this section we study some of the different versions of the OpenWrt operating system. Openwrt is a Linux based operating system for the WRT Linksys series. OpenWrt primarily uses a command-line interface, but also features an optional web-based graphical user interface for configuration of the system. In their website <http://www.openwrt.org> one can find a wiki with help and forums full of technical support for users. External third packages can be managed using the `ipkg` command.

4.3.1 WHITE RUSSIAN

Configuration is made directly to the nvram. The web interface is shown in figure 4-2



Figure 4-2: White Russian web interface

4.3.2 KAMIKAZE

A modified version of white russian. The configuration was moved from the nvram to configuration files under the /etc/config directory. The web interface is shown in figure 4-3

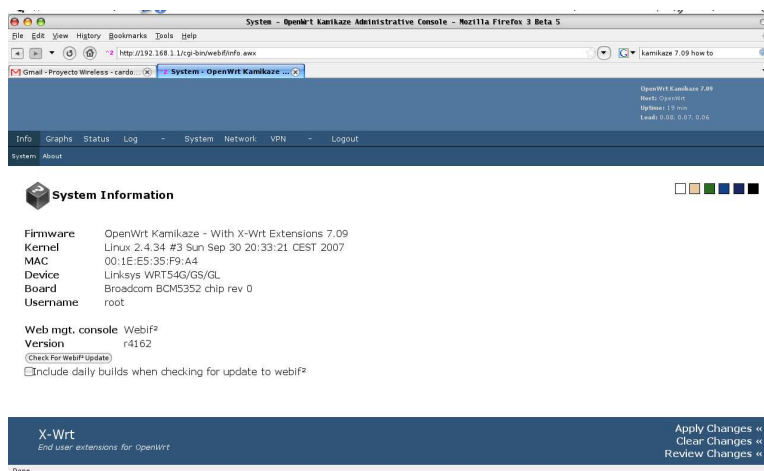


Figure 4-3: Kamikaze web interface

4.3.3 DD-WRT

A more advanced version of openwrt. This distribution provides support for IPv6, QoS, WDS and more. Due to the fact that provides more services, more disk space is used by this distribution in order to access its full potential. The web interface is shown in figure 4-4

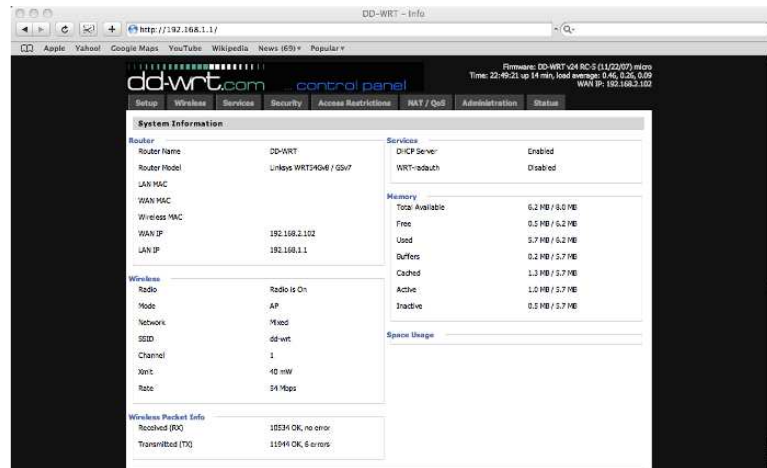


Figure 4-4: DD-Wrt web interface

4.4 CONFIGURATION OF THE LINKSYS WRT54GL

This section explains step by step how to configure the Linksys WRT54GL to work as a wireless mesh router using the OLSR routing protocol. The first step is changing the original firmware to Kamikaze, the binary for the Kamikaze distribution can be found in <http://downloads.openwrt.org/kamikaze/7.09/> (figure 4-5). In case of the Linksys router, which has a Broadcom wireless card, one can either download the binary from the `brcm-2.4/` or `brcm47xx-2.6/` directory. However the version inside of `brcm47xx-2.6/` is not stable with the wireless card, so `brcm-2.4/` is recommended.

Once inside of `brcm-2.4/` there will be a list of downloads and a folder named `packages`. Inside of `packages/` lies a list of extra components that can be installed in the router. We will cover these packages later with the command `ipkg`, but for now just download the file `openwrt-wrt54g-2.4-squashfs.bin` (figure 4-6).

OpenWrt

Wireless Freedom

Development Documentation Downloads Wiki Forum

Name	Last modified	Size
Parent Directory	-	-
packages/	20-Sep-2007 20:23	-
OpenWrt-ImageBuilder-brom-2.4-for-linux-i686.tar.gz	17-May-2008 13:02	29M
OpenWrt-ImageBuilder-brom-2.4-for-linux-x86_64.tar.gz	20-Sep-2007 20:26	26M
OpenWrt-SDK-brom-2.4-for-linux-i686.tar.bz2	17-May-2008 13:02	33M
OpenWrt-SDK-brom-2.4-for-linux-x86_64.tar.bz2	20-Sep-2007 20:26	38M
modules	20-Sep-2007 22:11	95B
openwrt-brom-2.4-squashfs.tgz	20-Sep-2007 20:23	1.8M
openwrt-usb1401-squashfs.bin	20-Sep-2007 20:23	1.8M
openwrt-w8040g-squashfs.bin	20-Sep-2007 20:23	1.8M
openwrt-w8050g-squashfs.bin	20-Sep-2007 20:23	1.8M
openwrt-w8050g-squashfs.bin	20-Sep-2007 20:23	1.8M
openwrt-wr100g-squashfs.bin	20-Sep-2007 20:23	1.8M
openwrt-wr150n.2.4-squashfs.bin	20-Sep-2007 20:23	1.8M
openwrt-wr150n_v1-2.4-squashfs.bin	20-Sep-2007 20:23	1.8M
openwrt-wr150g-2.4-squashfs.bin	20-Sep-2007 20:23	1.8M
openwrt-wr150g-2.4-squashfs.bin	20-Sep-2007 20:23	1.8M
openwrt-wr150g-2.4-squashfs.bin	20-Sep-2007 20:23	1.8M
openwrt-wr150g_v4-2.4-squashfs.bin	20-Sep-2007 20:23	1.8M
openwrt-wr150g_v4-2.4-squashfs.bin	20-Sep-2007 20:23	1.8M
openwrt-wr150g-2.4-squashfs.bin	20-Sep-2007 20:23	1.8M

First to recover the web interface we will need to run a few `ipkg` commands once more, so do not remove the internet access of the router. Type `wget http :`

`//downloads.x-wrt.org/xwrt/kamikaze/7.09/brcm-2.4/webif_latest.ipk` to download the latest package of the web interface, this will also add this repository to the `ipkg` list and could make more packages available as well. Now type `ipkg install webif_latest.ipk` in the same directory where `webif_latest.ipk` is to install the web interface. Once it is installed the one will be able to configure the router using the command line or the web interface, please note that not all things can be configured through the web interface.



Figure 4-7: Original Linksys web interface

Now is time to set up the working environment for configuring the WMN. Login to the router by establishing a TELNET session by typing the command `telnet 192.168.1.1` (no password is required) in the computer. Before continuing make sure the router has internet access, then type the command `ipkg update` to download the list of all available packages for the router. Now proceed to install the packages for configuration of the mesh by typing `ipkg install openssh-sftp-server` for the secure ftp server and `ipkg install olsrd` for the OLSR daemon. Finally type the command `passwd` to enter the password that will be used to connect using SSH in the future and type `exit` to finish the TELNET session. Once the TELNET session is finished ti

will be impossible to return to the router terminal using TELNET unless everything is reset, now the router will be access through SSH only.

The next step in the configuration is login to the router using SSH, the username is root and the password is whatever was written when the command passwd was typed.

The first file to be configured is the `/etc/config/wireless` (figure 4-8). This file is used to configure the wireless card of the router.

```
config wifi-device wlo
option type      broadcom
option channel '6' #Channel to be used
option disabled '0'      #0 for enable wifi, 1 to disable
config wifi-iface
option device wlo      #The network interface that will be used
option network 'wlan'  #The network configuration that will be used,
                        #must be created later
option mode 'adhoc'    #To be used in adhoc mode, can be access also
option ssid 'OLSR'     #Name of the mesh network
option encryption none #Encryption used none, wep, wpa
option hidden '0'      #0 for broadcast ssid, 1 to stay hidden
```

Next file in configuration is `/etc/olsrd.conf`. This file configures the OLSR daemon that will create the routing table. The following lines are the most important part in the configuration of OLSR.

```
Hna4 #This block contains the IP addresses of the networks that
      #don't run OLSR and are directly connected to the router
{
192.168.1.0 255.255.255.0
}
```

Wireless Configuration

Wireless Adapter wl0 Configuration	
Radio	<input checked="" type="radio"/> On <input type="radio"/> Off
Channel	6 <input type="button" value="v"/>
Max Associated Clients (Default 128)	<input type="text"/>
Wireless Distance (In Meters)	<input type="text"/>
Add Virtual Interface	

Wireless Virtual Adaptor Configuration for Wireless Card wl0	
Network	wlan <input type="button" value="v"/>
Mode	Ad-Hoc <input type="button" value="v"/>
ESSID Broadcast	<input checked="" type="radio"/> On <input type="radio"/> Off
ESSID	OLSR <input type="text"/>
Encryption Type	Disabled <input type="button" value="v"/>
Remove Virtual Interface	

Figure 4–8: Web interface for wireless configuration

Interface "wl0

```
{
AutoDetectChanges yes #This line will make any changes
#required to the interface automatically
}
```

Now we configure `/etc/config/networks` (figure 4–9). In this file we can add and configure networks. Initially there are two networks LAN and WAN. We need to create two more networks WLAN, which will break the bridge between the Ethernet and wireless interfaces. This is the same name of network that was typed previously in `/etc/config/wireless`. The other network is named OLSR and it will be the heart of the WMN.

```
config "interface" "wlan" #This is the same name typed in /etc/config/wireless
option proto 'static #Configure the interface ip static
option ipaddr '104.3.2.1'#IP address of the interface,
#it is a subnet of the virtual network OLSR
option netmask '255.0.0.0 #The subnet mask of this network
config "interface" "OLSR" #Name of the virtual network
```

```
option proto 'dhcp'           #Will use dhcp
option ipaddr '104.3.2.0'     #Network address
option netmask '255.255.255.192' #Subnet mask
```

wlan Configuration

Connection Type

Type

IP Address

Netmask

Default Gateway

wlan DNS Servers

[remove Network wlan](#)

OLSR Configuration

Connection Type

Type

IP Address

Netmask

Figure 4–9: Web interface for networks configuration

We are almost done, the next configuration file to be changed is `/etc/config/dhcp` (figure 4–10). In this file one can find the current DHCP configuration. We will add a DHCP pool for the wireless network. One can have in a mesh as many DHCP servers as wanted, as long as the IP addresses do not overlap in each other.

```
config "dhcp" ""
option start '100' #Start giving addresses after 100
option limit '150' #Maximum number of ip addresses to be leased
option leasetime '720m' #How much time will last the ip address
option ignore '0' #0 for enabled, 1 for disabled
option interface 'wlan' #The network that the pool will be applied to
```

wlan DHCP	
DHCP	<input checked="" type="radio"/> On <input type="radio"/> Off
Start	<input type="text" value="100"/>
Limit	<input type="text" value="150"/>
Lease Time (in minutes)	<input type="text" value="720"/>

Figure 4–10: Web interface for DHCP configuration

Now for the final step in the configuration of files for WMN the `/etc/firewall.user`.

In this file we will need to open the ports used by OLSR so that the routers can configure their routes, thus create the mesh.

```
#!/bin/sh #Copyright (C) 2006 OpenWrt.org
```

```
iptables -F input_rule
```

```
iptables -F output_rule
```

```
iptables -F forwarding_rule
```

```
iptables -t nat -F prerouting_rule
```

```
iptables -t nat -F postrouting_rule
```

```
#The following chains are for traffic directed at the IP of the
```

```
#WAN interface
```

```
iptables -F input_wan
```

```
iptables -F forwarding_wan
```

```
iptables -t nat -F prerouting_wan
```

```
WIFI=wlan0
```

```
#This allows port 22 to be answered by (dropbear on) the router
```

```
iptables -A input_wan -p tcp --dport 22 -j ACCEPT
```

```
#Allow connections to olsr info port.
```



```
iptables -A input_wan -p tcp --dport 1979 -j ACCEPT
```

#OLSR needs port 698 to transmit state messages.

```
iptables -A input_rule -p udp --dport 698 -j ACCEPT
```

```
iptables -A forwarding_rule -i $WAN -o $WIFI -j ACCEPT
```

```
iptables -A forwarding_rule -i $WIFI -o $WAN -j ACCEPT
```

#For forwarding LAN and WIFI in nodes

```
iptables -A forwarding_rule -i $LAN -o $WIFI -j ACCEPT
```

#For WIFI clients to connect to nodes.

```
iptables -A forwarding_rule -i $WIFI -o $WIFI -j ACCEPT
```

#For connecting a wired lan client of node 1 to wired lan client of node 2

```
iptables -A forwarding_rule -i $LAN -o $LAN -j ACCEPT
```

#WIFI needs to go to LAN ports, too!

```
iptables -A forwarding_rule -i $WIFI -o $LAN -j ACCEPT
```

Now everything is set to run the WMN. However one could want to start OLSR automatically when the router turns on instead of typing `/etc/init.d/olsrd start`, to do so create a symbolic link by typing `ln -s /etc/init.d/olsrd /etc/rc.d/S60olsrd`.

Finally let us consider the case were something crashes the system and there is no way to fix it. The following steps will guide the user to reset everything back to its defaults (just after installing Kamikaze). The mode is called Failsafe mode. During this mode the router does not have any other interface except Ethernet with IP 192.168.1.1 and the DHCPs are turn off. The only way to connect to it is giving the computer a static IP in the range of 192.168.1.1 (e.g. 192.168.1.2),

subnet mask 255.255.255.0 and default gateway 192.168.1.1. Then one can establish a TELNET connection and type the following commands to erase everything back to its defaults and start all over again. The commands to be run are: *firstboot* (will load the configuration as if it were the first time that the system boots), *sync* (to synchronize the changes) and */sbin/mount_root* (to commit the changes to the main partition).

Another way of recovering from errors is setting up an alternative boot method, more specifically TFTP. To do this it is necessary to set some nvram variables first.

1. *nvram setboot_wait = on*: tells the router that has to wait for a tftp image to be sent.
2. *nvram setwait_time =< secs >*: wait *secs* seconds for an tftp transfer.
3. *nvram commit*: commits changes to nvram.

This will cause the router to wait *secs* seconds for a tftp image when booting. The image can be sent using *tftp put filename ipaddr*, where *filename* is the name of the image file and *ipaddr* is the IP address of the router. This must be done during the first *secs* seconds of boot. Otherwise the router will boot normally, if possible.

4.5 CONFIGURATION OF THE LAPTOP COMPUTER

It may be that your computer is set to connect to Access point (infrastructure) networks only. If your computer is set as mentioned then you won't be able to connect to an ad-hoc network. To change these settings go to Start → Control Panel → Network Connections (figure 4-11).

Follow the following steps to allow your computer to connect to ad-hoc networks:

1. Right-click on Wireless Network Connection
2. Select Properties on the displayed menu (figure 4-12).
3. A properties window should open (figure 4-13).
4. Click on the tab named Wireless Networks.
5. Click on the button named Advanced on the Preferred Networks (figure 4-14).

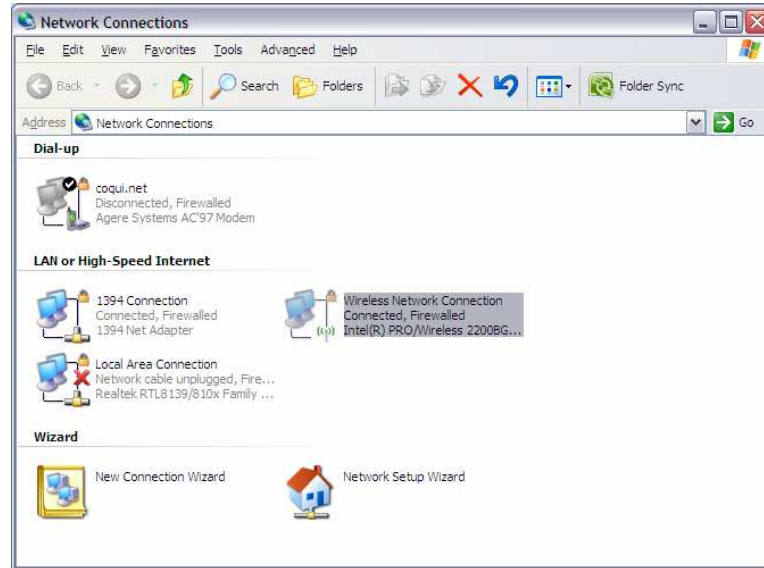


Figure 4-11: Network connections on control panel

6. The following window should open (figure 4-15).

Select the Any available network option. This will enable your computer to connect to infrastructure as well as ad-hoc networks. If this does not work then the option can be forced to ad-hoc by selecting Computer-to-computer (ad-hoc) networks only.



Figure 4-12: Dropdown menu

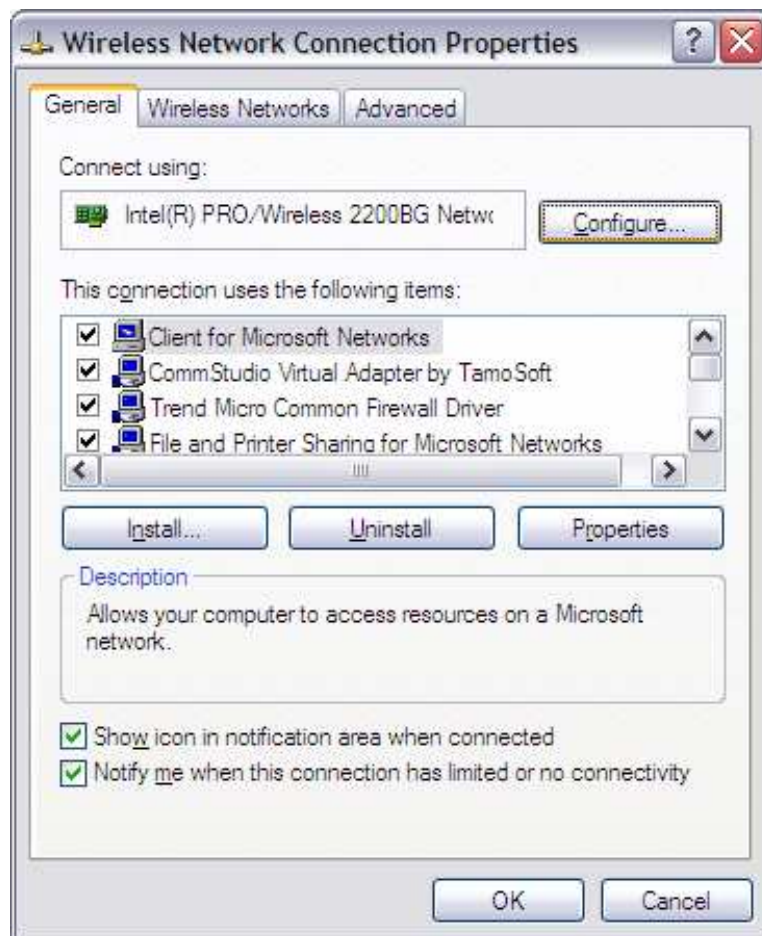


Figure 4-13: Network properties window

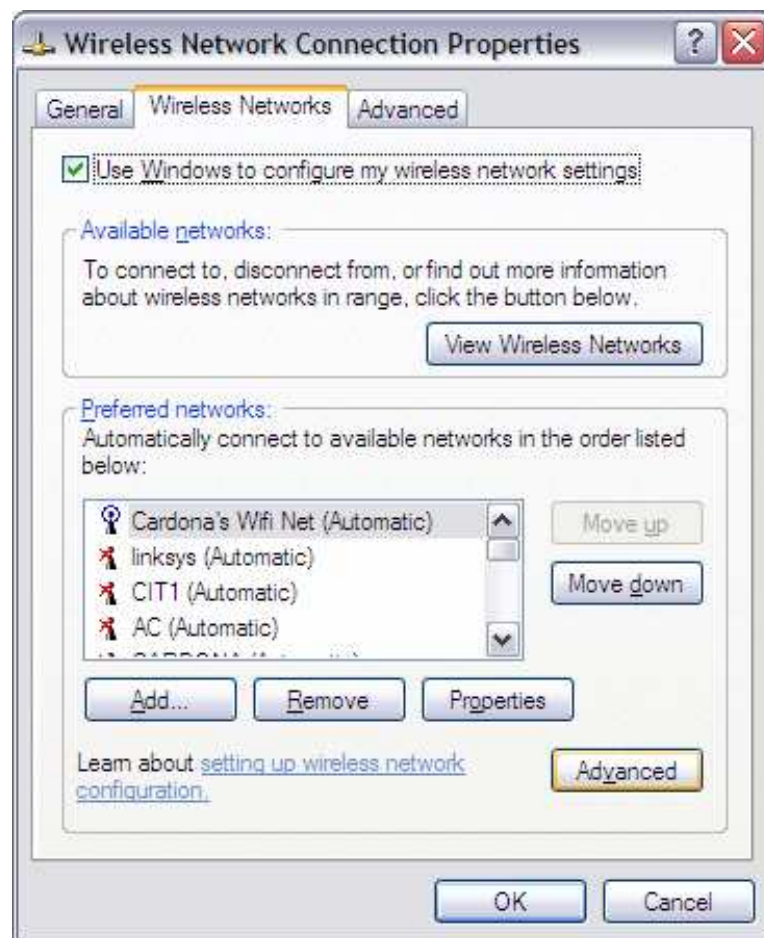


Figure 4-14: Preferred networks tab

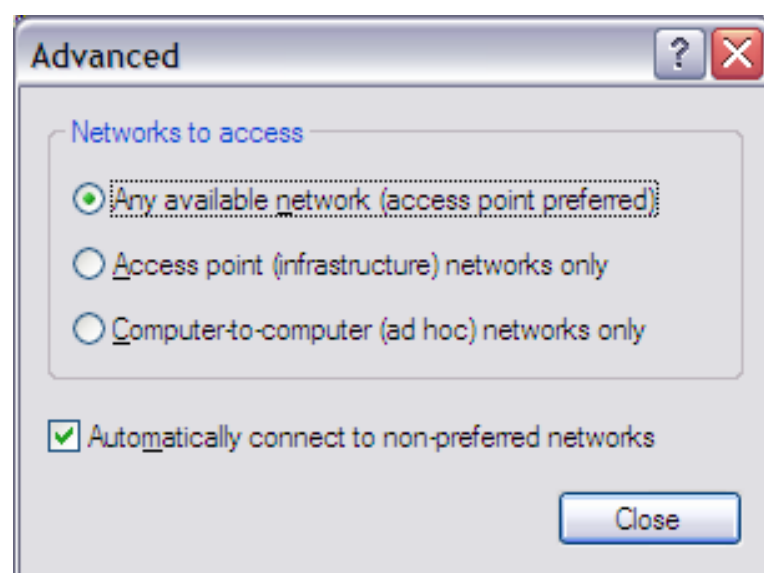


Figure 4-15: Advanced window

CHAPTER 5

SERVICE ORIENTED ROUTING ALGORITHM

Wireless mesh network (WMN) is a promising technology that can provide cost-effective solutions to cover a rather large area. Despite this important feature, the future generation of WMNs needs to support more and more applications, such as voice over internet protocol (VoIP) and multimedia content distribution. To efficiently support these diverse demands and to effectively utilize the wireless network, service-oriented network layer design has been proposed recently. A key idea of such a service-oriented design is that a user only needs to specify the service to the network, instead of the destination address in a traditional manner. In this chapter, we present a simple solution, namely, the *service-oriented routing algorithm* (SORA), to enable service orientation in WMNs. The proposed scheme works in a similar manner as the Domain Name System (DNS). Specifically, SORA associates IP addresses to services just like the DNS associates IP addresses to domain names. SORA calculates the best route using the number of hops from a source to an arbitrary destination. This destination is chosen to be the one closest to the source and that has the service that it requests. This chapter discusses in details the different steps that are taken to select this destination.

To demonstrate the viability of the proposed scheme, we have developed simple but efficient programs that are running in *Commercial off-the-shelf* (COTS) devices, in particular, workstations and wireless mesh routers. Our experiments show that

the proposed algorithm can efficiently provide distributed services in wireless mesh networks.

We present in this chapter an introduction to the problem and to SOA architecture. The rest of the chapter is organized as follows: client side procedure in which the client request is explained, the intermediate node procedure is then discussed, followed by the server side procedure. A SORA agent that runs inside of the server is also explained afterwards. Finally the packet format is presented and explained in details. At the end we present the results and conclusions.

5.1 INTRODUCTION

In recent years, *wireless mesh networks* (WMN) are becoming more and more important because it can provide cost-effective solutions to cover a rather large area [1]. Despite such a salient feature, the future generation of WMNs needs to support more and more applications, such as voice over internet protocol (VoIP), multimedia content distribution, etc.

In our recent studies [9, 44], we have investigated the features of existing and potential services in future wireless networks, and we have concluded that these services may express diverse requirements, in terms of 1) communication pattern (one-to-one, one-to-many, many-to-one, and many-to-many), 2) delay (real-time, non-real-time, and delay-tolerant), 3) service availability (centralized, distributed, and location-aware), 4) security, and 5) reliability.

To fulfill such demands, we proposed to develop “service-oriented” network layer for future WMN [9]. Compared to existing approaches, the new framework has the following major features:

- Availability of service: The network layer is aware of the availability of different services, such as Internet access, real-time communications, content distribution, interactive gaming, medical applications, and vehicular safety applications.

- Communication pattern: A service can have a specific communication pattern, including one-to-one, one-to-many, many-to-one, and many-to-many.
- Service requirements: A service can also be associated with a variety of service requirements, including bandwidth, delay (real-time, non-real-time, and delay-tolerant), security, and reliability.
- Service-orientation: To obtain a service, a customer only needs to request the service (with certain specifications) from the network layer. The network layer will determine the availability of service and will route the traffic accordingly.

With the service-oriented network layer design [9], the network layer will dynamically determine the destination of the request, with certain quality-of-service (QoS), security, and reliability considerations. In the next section, we will discuss the service-oriented design from a broader view, instead of the network layer.

5.2 RELATED WORK

In recent years, the concept of service-oriented design and service-oriented architecture have been discussed in related work. The Organization for the Advancement of Structured Information Standards (OASIS) [2] defines SOA as a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. Specifically, the SOA shall provide a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

In the literature, most existing studies consider the SOA and service-oriented design from the application layer perspective.

A number of studies have been focused on the architecture itself. In [4], the authors suggested the *Model Driven Service Oriented Architecture* (MDSOA) as a solution for integrating new technology, such as ad-hoc communications and protocols, along with the opportunity to unify solutions for organizations. In [5], the author presented SOA fundamentals using J2EE and a router as a proxy server to support

the needs of telecommunications of today and next generation networks. The authors in [6] made an analysis of the impact of SOA in the current business and how is important to integrate SOA in the current world. In [7], the authors studied the possibility of using legacy software to help with this integration. Three approaches were suggested: 1) disassembly on the code to write the program in another language, 2) creating a wrapper code using existing interfaces, and 3) transforming the code, which is recommended by the authors.

Many other research work focused on the service perspective of SOA, in particular as a technology with great scalability and easy control of services in large networks. Applications like air transportation [10], e-healthcare [11], smart home [12], business integration [13, 14]. Other types of applications that have been discussed as possible applications for SOA are network monitoring, information exchange and quality of service (QoS) [15–19, 45].

Finally, a few work have addressed the design of SOA using tools like UML [20], XML [46], and cost analysis [21]. For more details on the related work please refer to the corresponding section (3.2) in chapter 3. Additional information can be also found at [46–51].

5.3 DESIGN OF SORA

In the previous discussion, we have demonstrated the importance of service-oriented design. In this section, we elaborate on the design of Service-Oriented routing algorithm (SORA). Fundamentally, SORA is a routing concept, and thus it can be combined with any existing routing protocols. The main idea of SORA is rather similar to the principle of the DNS protocol. In particular, we need three types of procedures, 1) server, 2) client, and 3) intermediate-node.

5.3.1 CLIENT SIDE PROCEDURE

If an end-user wants to obtain a service, it will follow the procedure defined in Fig. 5–1, which is rather straightforward. Basically all that it has to be done is to

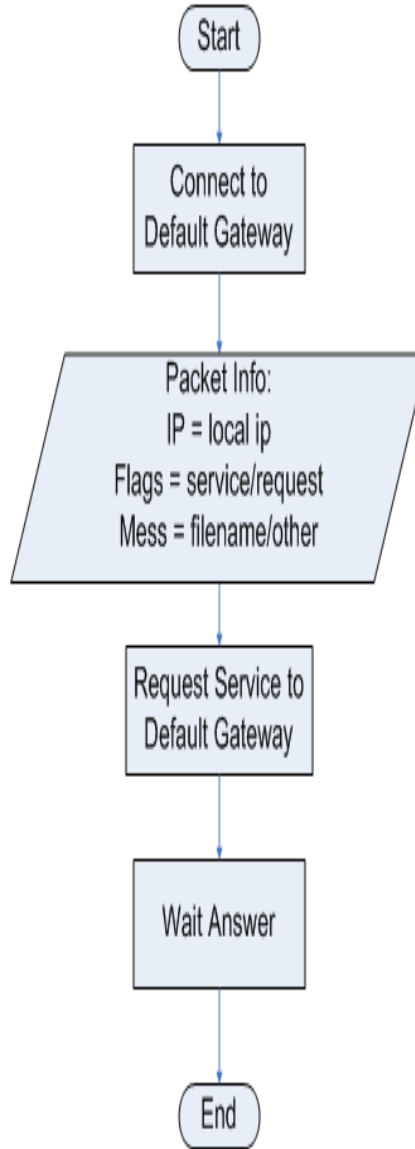


Figure 5–1: Client procedure

create the request by telling the default gateway what type of service it wants and what it wishes to ask from that service if necessary. Afterwards it awaits a reply from the default gateway containing either the service requested or an error.

5.3.2 INTERMEDIATE NODE PROCEDURE

The main difficulty in our design is the intermediate node procedure.

First, upon receiving the notification message from a server or other intermediate node, an intermediate node will follow the procedure shown in Fig. 5–2. For instance, the intermediate node will increase the number of hops in the packet, and

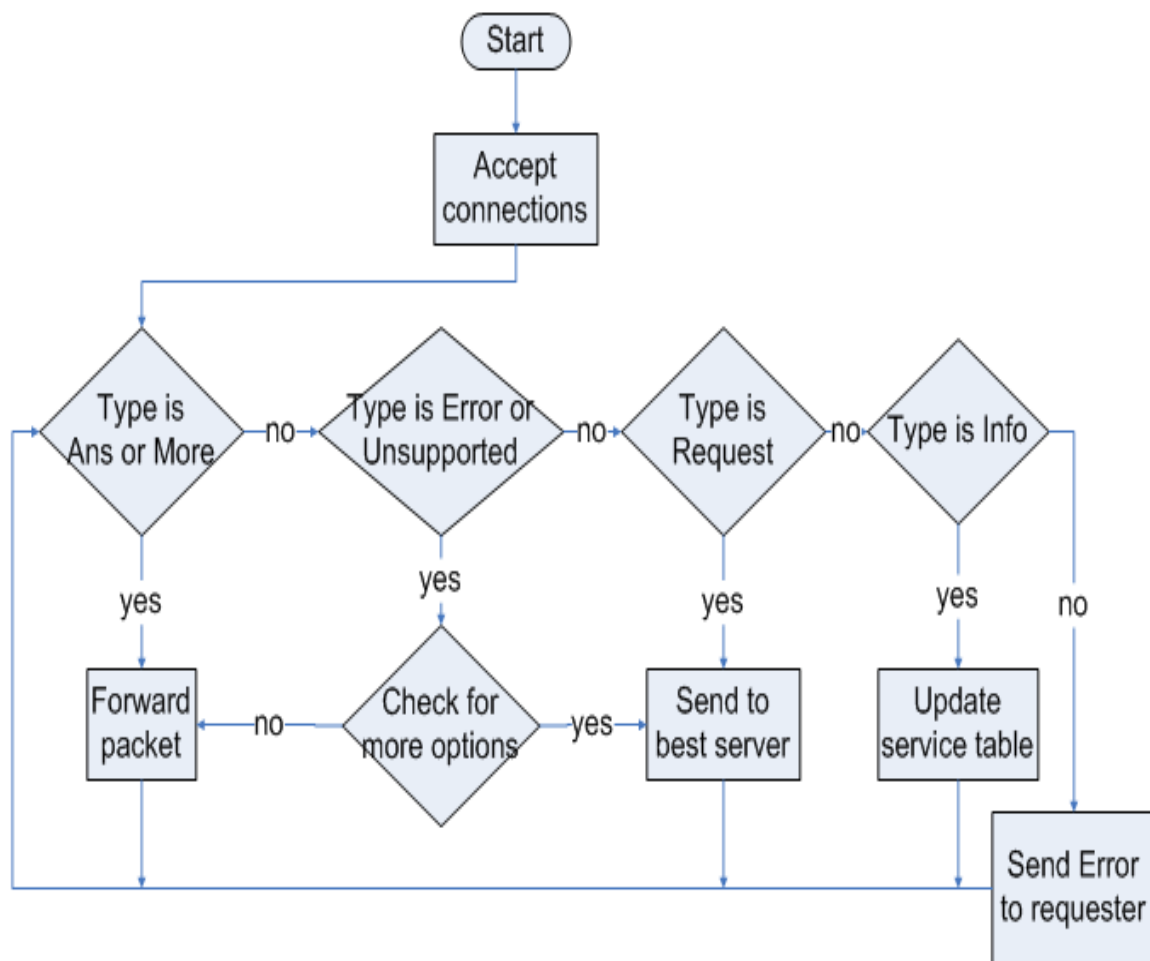


Figure 5-2: Intermediate-node procedure

Table 5–1: An example of Service Table

Service	IP Address	Next Hop	# Hops
25 (SMTP)	172.16.3.1	172.16.3.5	2
21 (FTP)	192.168.0.1	192.168.0.7	1
21 (FTP)	192.168.0.2	172.16.3.1	4
25 (SMTP)	192.168.1.1	192.168.0.15	2

then forward it to the next-hop neighboring nodes. In addition, if the router does not have the service information previously, it will create a service table entry with (a) the service type, (b) the interface where the machine that provides this service is connected to, (c) the IP address of the server, and (d) the number of hops to the service. An example of the service table can be found in Table 5–1.

Notice that, this procedure will be performed by each node if the number of hops in the packet is smaller than a threshold and if the hop distance is smaller than the one already stored at the intermediate node. Eventually, the notification information will be broadcasted to all the nodes in the network with a certain range to the service.

Secondly, if a router received the request from a user, asking for a certain service, the router must first check the service table to see on which interface is the device that provides the service, and will then forward the request through it. To avoid bottle necks in the network, the routers verify if the service is provided by more than one station so that the work load can be uniformly distributed.

For example, in Table 5–1, if a new request for an FTP session is received, then that router should send the request to the IP address 192.168.0.1 since it is connected in less hops than 192.168.0.2. Now let us consider the case where an SMTP request is made. As we can observe from the service table, both SMTP servers are at 2 hops distance from the intermediate node. In this scenario the server is chosen randomly to avoid bottle necks.

In case that the selected machine does not or cannot accept for some reason, the request can be forwarded to the other service providers, until either it successfully

finds a provider or fails to find one within a predefined range, in which case an error will be sent back to the work station (WS) and notify it.

5.3.3 SERVER SIDE PROCEDURE

Clearly, SORA needs service registration or notification at the beginning of the algorithm and every time when there is a change in the availability of services inside a server. In Fig. 5-3, we illustrate the flow-chart for the server side procedure.

To implement the server side procedure, a simple way is to notify the neighbors by sending a 1 in the position of the service in the data field if the service is offered and a 0 if not. Discussion on how the mesh deals with this type of packet will be presented shortly in the intermediate node procedure section.

5.3.4 SORA AGENT

The SORA agent is a code that runs along side in the server and acts as a translator from SORA packets to IP packets and vice-versa. The agent itself works using port forwarding. It scans on the port used by SORA (3550) and then redirects to the desired port. For example let us consider a scenario where a client wishes to establish a telnet session (port 23). In this case the SORA packet contains the 23 inside of the flags, which is extracted by the SORA agent. Then the agent establishes a socket connection to localhost using port number 23. When the TELNET server replies the SORA agent gets the bytes from the reply and sends it back to the client using the 3550 port (figure 5-4).

5.3.5 PACKET FORMAT

Fig. 5-5 shows the packet header structure for the SORA protocol. We can see that the header consists of Source IP Address (4 bytes), and followed by some flags (4 bytes). These flags are divided into Type (2 bytes) and Service (2 bytes). In our current design, we consider the following types:

1. Request — telling the mesh that the packet is a request from a source.

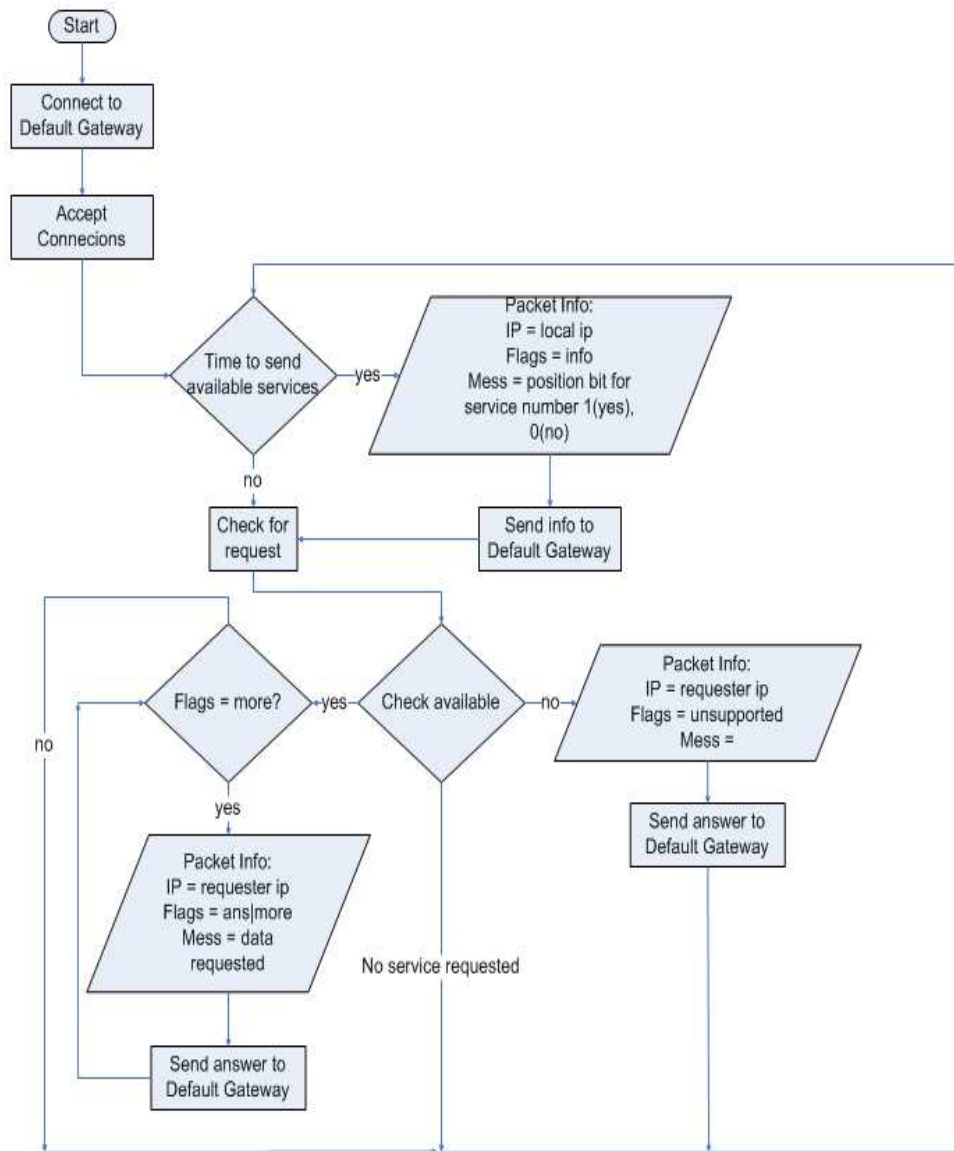


Figure 5–3: Server procedure

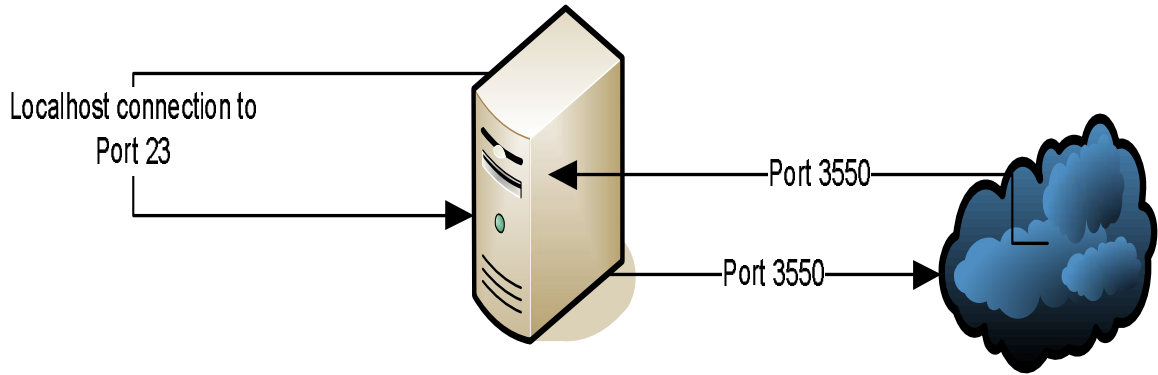


Figure 5-4: Agent TELNET connection example

2. Answer — saying that the packet is meant to be sent back to the source and is a response from a server
3. More — an extended case of Answer saying that more packets have to be sent in order to end the transmission.
4. Error
5. Unsupported

In both Error and Unsupported, the mesh will search for more options until either finds a service or it has no more options. If no option is found then the packet is forwarded to the client. The difference between these two types is that Error means that an internal error occurred or that no more connections can be accepted; while Unsupported means that the requested service is either not know or unimplemented by the mesh.

6. Info — which is meant for the intermediate nodes, telling them that the packet is an information packet containing the services provided by the source of the packet.

The last two fields of the packet is the length of the data followed by the data itself.

5.4 EXPERIMENT AND DISCUSSION

To validate the proposed algorithm, we have develop a testbed, illustrated in Fig. 5-6, using COTS devices such as laptop computers and wireless routers. We have implemented programs running on these COTS devices. In our experiment,

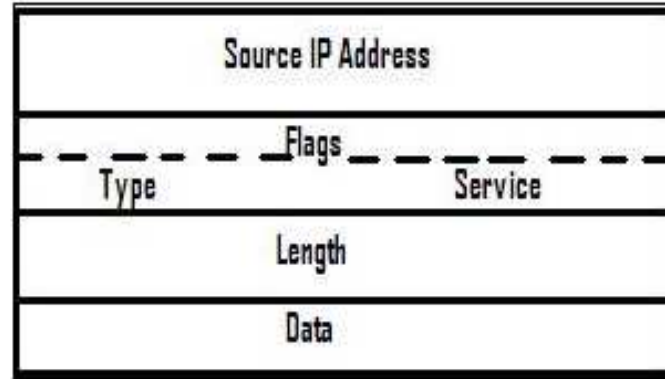


Figure 5-5: Packet header format

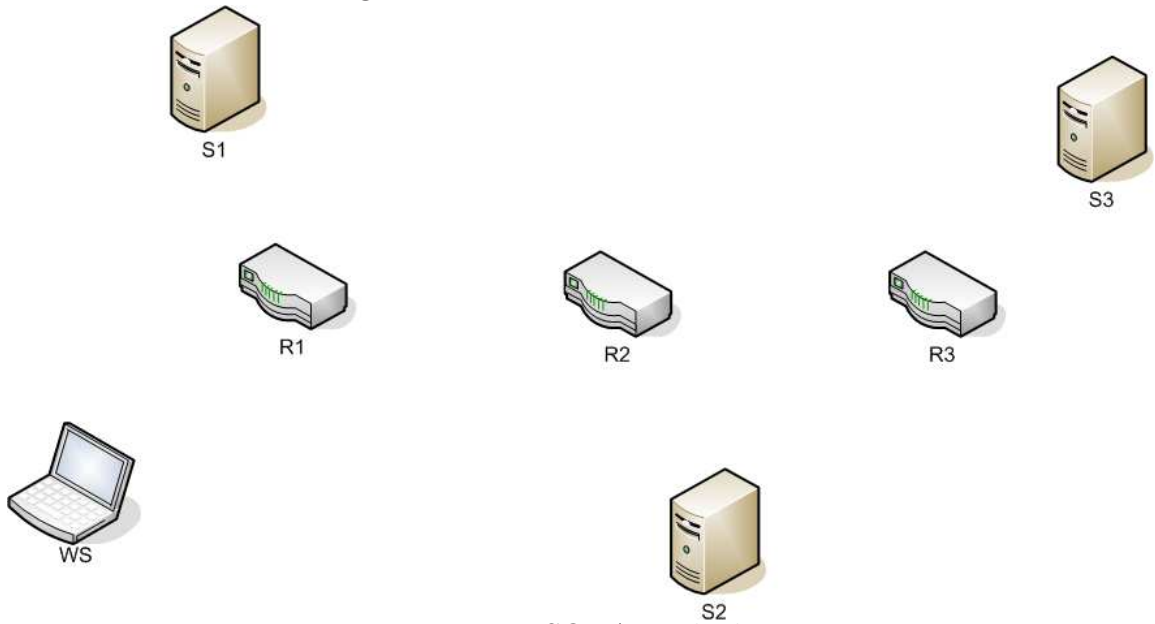


Figure 5-6: SORA testbed

the testbed consists of a laptop acting as the client (workstation, denoted as WS), three laptop computers (S1, S2, and S3) running the server code, and three Linksys WRT54GL wireless mesh routers (R1, R2, R3) running the intermediate node procedure. S1, S2, and S3 are connected to R1, R2 and R3, respectively. The operating system we apply on the wireless router is OpenWRT Kamikaze [52]. And we enable OLSR as the routing protocol.

In the testbed, we first let the WS request a file of size approximately to 3MB from a particular server. As shown in Table 5-2, our experiments show that the performance of file downloading depends on the location of server. We then enable our SORA procedure, and we have observed that the transmission time can be

Table 5-2: Time to Download a File (3MB)

Server	# Intermediate Nodes	Time (mins)
S1 (fixed)	1	00:05
S2 (fixed)	2	00:32
S3 (fixed)	3	00:55
SORA	-	00:05

minimized because a request will always be forwarded to the closest server. For the node deployment scheme in Fig. 5-6, the performance of SORA is the same as that of choosing S1. As we move the position of WS, we have observed that the server been accessed is also changing, which proves the correctness of our design.

5.5 CONCLUSION

In this work, we have presented a simple but efficient *service-oriented routing algorithm* (SORA), to enable service orientation in wireless mesh networks. The proposed scheme is similar to the Domain Name System (DNS) in that it associates IP addresses to services just like the DNS associates IP addresses to domain names. In our design, we consider three categories of procedures, 1) server, 2) client or user, and 3) intermediate node. To demonstrate the viability of the proposed scheme, we have developed programs that are running in commercial off-the-shelf devices, including workstations and wireless mesh routers. Our experiments show that the proposed algorithm can efficiently provide distributed services in wireless mesh networks.

CHAPTER 6

CHANNEL SELECTION NETWORK CODING

In recent years, IEEE 802.11 based wireless local area network (WLANs) have been widely deployed and the cost of IEEE 802.11-enabled routers and adapters are very low. To effectively establish wireless networks in various scenarios, it is important to exploit such inexpensive commercial off-the-shelf (COTS) products. In this chapter, we address this issue and propose a novel medium access control (MAC) scheme based on the idea of network coding and intelligent channel selection. Specifically, we will design a new sub-layer, named CHANET, between the network layer and the legacy IEEE 802.11 MAC layer. In this manner, the cost for updating the system can be minimized, while the performance can be improved efficiently. The first technique discussed is intelligent channel selection using Fuzzy Logic; with it the mesh can be transformed into a full-duplex connection, meaning that can receive and transmit at the same time. Of course that in order to achieve this the nodes need to have at least two radios, if this is not the case it can still send the information using the channel that is being used the less. The second technique involves network coding at the network layer. Using DSP algorithms, the nodes in the mesh can create linearly independent equations that can mix N unknown packet to be decoded at the destination. This allows more data to be sent through the mesh utilizing even better the channel capacity. The encoding of the packets is done while the system is scanning for the best channel to be use to transmit. Our simulation results show that the proposed scheme can substantially improve the throughput and delay performance of the wireless network.

We present in this chapter an introduction to the problem and some related work. The rest of the chapter is organized as follows: channel selection is discussed first, followed by the network coding algorithm. Finally the results and conclusions are presented.

6.1 INTRODUCTION

Wireless networks are growing fast in recent years. Particularly, many wireless local area networks (WLANs) have been established based on the IEEE 802.11 protocol family. One important feature of this standard is that each adapter can be operating only on a half-duplex manner, which means that an adapter can either send or receive data at a given time but not both.

Because of this shared medium we have a great number of collisions which reduce greatly the throughput of the network. Thus many re-transmissions are required and a delay is then introduced.

In this chapter we introduce a mechanism to reduce the number of transmissions called network coding. Network coding takes advantage of the broadcast nature of the air by mixing data at intermediate nodes in order to increase the amount of possible data in a time interval.

Also to reduce the delay introduced by processing packets using network coding we proposed a multi-channel environment. The main idea is to divide a given bandwidth into 3 channels. Thus adapting ideas from the IEEE802.3 standard, in this case having 2 out of the 3 channels for receiving data and 1 used to send. The channel allocation problem is then addressed in a dynamic manner. Using fuzzy logic to adapt some intelligence in the intermediate nodes to select which is the best channel to transmit a packet a given time according to its utilization and the collisions occurring in the channel.

6.2 RELATED WORK

There has been some work related to the topics used in this work. [22] discusses a way of deploying sensor nodes in a distributed system using a fuzzy optimization algorithm (FOA). In the FOA method were taken into account the number of adjacent nodes in a given radio and the distance between them.

Meanwhile [23] uses a multiple channel environment to test the IEEE802.11n using packet aggregation which is adding redundancy data to a packet at the source node in order facilitate network coding and decoding at the destination node. However this work focuses more on the aggregation of packets, while uses the available channels in a static way with 25 multi-hop nodes topology.

In [53] we have the implementation of UWBMAC, a MAC protocol for wide-band networks. This is the protocol used in this work to add the channel selection functionality. UWBMAC uses packet aggregation to aggregate multiple upper-layer packets into one single packet, thus decreasing the delay and increasing the throughput.

[24] presents a static assignment of channels in a multi-hop multi-radio environment. The focus of the work is based on minimizing the size of collision domains using a mesh network relying on adjacent nodes to make the delivery of data.

Finally [40] presents COPE, a simple opportunistic network coding protocol applied to wireless networks, using probabilities as the base to decide when to apply network coding or not. On the other hand [42] shows another alternative of network coding, which is done at the physical layer, this mechanism is faster than the one presented in [40] but we will concentrate more on the network layer network coding since it gives us more freedom with multicast packets.

For more details on the related work please refer to the corresponding section (3.3 and 3.4) in chapter 3. Additional information on multiple channels can be found at [54–65]. Also more information on network coding can be found at [66–69].

6.3 DESIGN OF CHANET

In the previous discussion, we have demonstrated the importance of avoiding extra transmissions and allocating channels according to the environment. In this section, we elaborate on the design of CHANET. Fundamentally, CHANET is a sub-layer between the MAC (802.11) and the network layer (IP), allowing it to co-exist with the current protocol stack and integration without modifications to the actual protocols. We will now explain the two components of CHANET, 1) channel selection, and 3) network coding.

6.3.1 CHANNEL SELECTION

This subsection discusses how the use of Fuzzy Logic allowed the use of different channels in the network, taking always the best option.

The selection of channels will use a Fuzzy Logic code developed in C++. The measures that will be taken into account are the collisions of the network and the channels utilization.

Collisions $C \in \{none, low, medium, high, blocked\}$.

Utilization $\rho \in \{nothing, some, alot, overloaded\}$.

These sets will use normalize data in order to improve comparison of different channel utilization and collision percentages.

In figure 6-1 we can graphically appreciate the range of the collision sets. *None* is defined as the case when practically there are no collisions in the system. Initially this condition is not met and the idea is to try to get the system into that state. The *None* set is a z shape fuzzy set with maximum at 2% and minimum at 12%. Next we can see the *Low* collision fuzzy set, which is a triangular shaped set with left minimum at 10%, maximum at 22% and right maximum at 35%.

Following we have the *Medium* set, also triangular with left minimum, maximum and right minimum at 33%, 45%, 62% respectively. The other set is *High* which has triangular shape with thresholds at 60%, 75% and 90%. Finally is *Blocked* fuzzy set

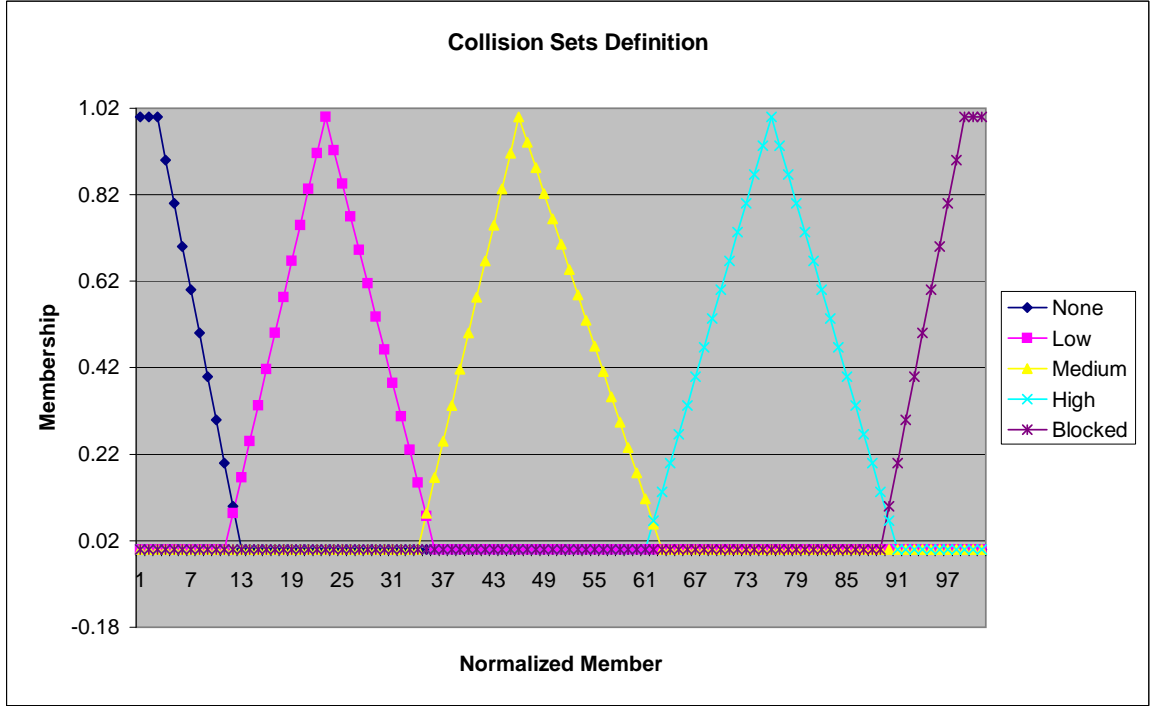


Figure 6-1: Collision sets definition

that represents the state where nothing can pass through the network due to the number of collisions occurring and is defined with as an s shaped set with minimum at 88% and maximum at 98%.

Now we will continue to present the four definitions of sets related to channel utilization.

In figure 6-2 there is a display of the sets of channel utilization. The first set we find is the set of *Nothing*. This set means that the channel is not been used by the network. It is shown as a z shape set with maximum at 2% and minimum at 22%. The two triangular shaped sets in the middle are the *Some* (when the channel is used but not much) and *Alot* (when the channel is almost overloaded since of the heavy load). They have thresholds of left minimum, maximum and right minimum at 18%, 33%, 44% and 40%, 60%, 75% respectively.

For the state where a channel is doing all the job and others are doing nothing we have the set *Overloaded*, meaning that an action has to be taken in order to

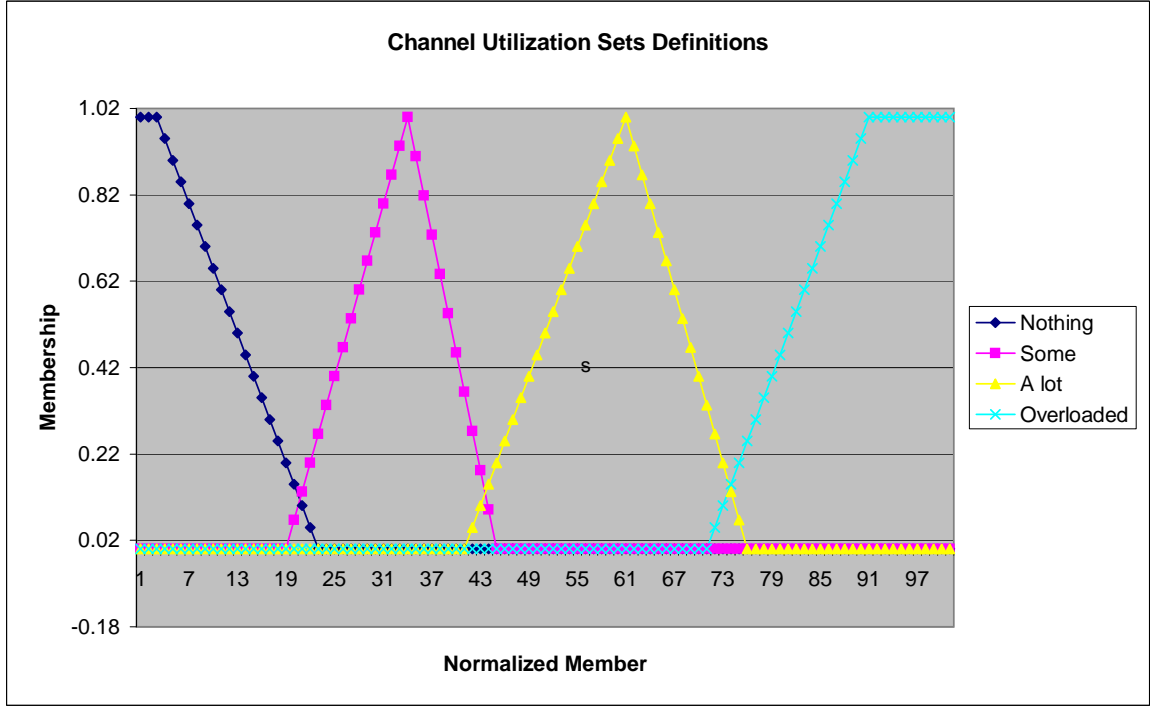


Figure 6-2: Utilization sets definition

distribute better the work load. This set is an s shape type with minimum at 70% and maximum at 90%.

These fuzzy sets previously presented were used in the fuzzy rules to make the decisions. The rules consisted of two inputs (collisions and utilization) and one output (utilization). The definitions of these rules are presented in algorithm 1.

In figure 6-3 can appreciate the steps of the decision making. At each channel we take into account its utilization percent along with the collisions in the network. These two values are then fuzzified, changed into a fuzzy set, to make the decision in the rules relation. Finally the resulting relation is then passed through the final block, the defuzzifier, to extract and find out which channel is best to use.

6.3.2 NETWORK CODING

The network coding part is presented in this section. To implement the network coding functionality a random linear coding algorithm was used. Random linear coding basically involves creating indecently linear equations where the terms are in this case the packets multiplied by a constant positive coefficient. These coefficients

Algorithm 1 Fuzzy Logic Algorithm

```

1: for Each Channel do
2:   overloaded AND blocked  $\rightarrow$  nothing
3:   overloaded AND (medium OR high)  $\rightarrow$  some
4:   overloaded AND low  $\rightarrow$  alot
5:   overloaded AND none  $\rightarrow$  overloaded
6:   alot AND blocked  $\rightarrow$  some
7:   alot AND (medium OR high)  $\rightarrow$  some
8:   alot AND low  $\rightarrow$  alot
9:   alot AND none  $\rightarrow$  a lot
10:  some AND blocked  $\rightarrow$  alot
11:  some AND (medium OR high)  $\rightarrow$  alot
12:  some AND low  $\rightarrow$  some
13:  some AND none  $\rightarrow$  some
14:  nothing AND blocked  $\rightarrow$  overloaded
15:  nothing AND (medium OR high)  $\rightarrow$  alot
16:  nothing AND low  $\rightarrow$  some
17:  nothing AND none  $\rightarrow$  nothing
18: end for

```

were randomly chosen from a field that contained only prime numbers to make the process of determine independence easier later. That is: $Z_p = \{p\}$ where p is a prime number. These coefficients are also generated by the intermediate nodes each time providing them linearly independent equations each time. When a packet is transmitted by an end node the coding vector is set to all zeroes to let the other nodes know that is a native packet to be encoded later by the intermediate nodes.

First we discuss the network coding send procedure shown in figure 6–5. It all starts with the best channel selection algorithm discussed previously. Afterwards the node checks if it can transmit. If transmission is not possible then it uses this time to prepare the encoded packet to be sent in the future.

To create this packet the first step is to determine if there are packets to be broadcasted. This is done by calculating the intersection of the pools of all the hosts. If there are packets that have to be broadcasted then these are mixed together by multiplying them by a random coefficient and adding them together. The coefficients used for that encoding are then copied to the coding vector of that packet. Since this

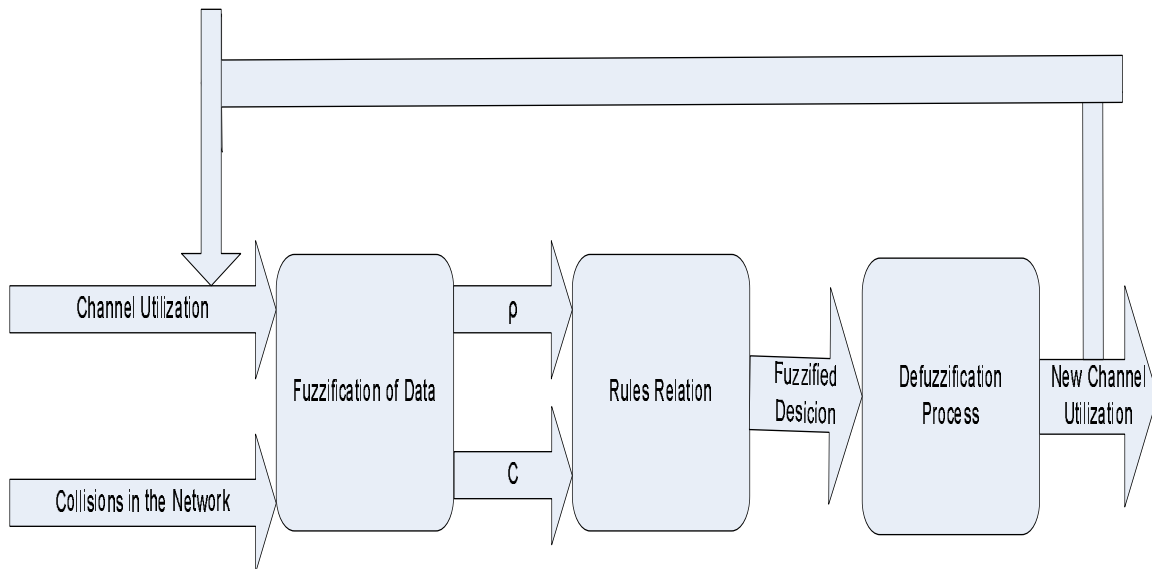


Figure 6–3: Graphical representation of the channel selection process

packet is a broadcast the destination address is changed to the broadcast address. This same process is repeated N times where N is the number of packets that are being mixed.

The alternative is that there are no packets to be broadcasted or multicasted if this is the case then the node passes to choose the first host that was pending to receive packets and mixes all the packets destined for that host. Afterwards the encoded packet is added to the output queue to be transmitted later.

Now if the node can send it passes then to check if there is a packet to send in the output queue. If so it sends the packet to the destination else if checks if the pools are empty to generate an output packet if they are not.

Now we discussed the algorithm for receiving a packet using network coding as seen in figure 6–6. The first step after receiving a packet is to determine whether or not is a native packet. A native packet is a packet that it has not been mixed with any other, that is its coding vector is full of zeroes. If this packet is native then the node passes to add it to the pool of heard packets. These heard packets are sort by destination address so that they may be used in encoding later if necessary.

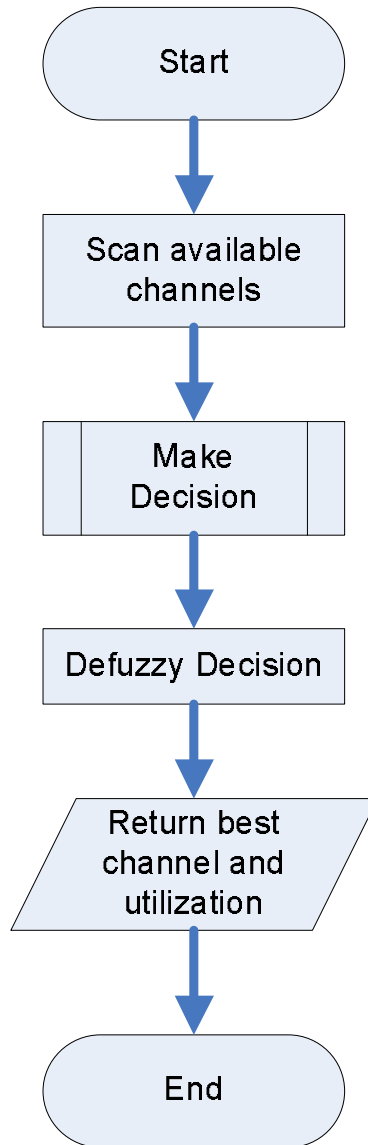


Figure 6–4: Channel selection flow chart

If the packet is not native then the node stores it in a temporary pool that will contains all packets related to the transmission if and only if the packet is linearly independent with the others already stored. Since the coefficients were chosen from prime numbers to determine independence of the packets is only necessary to check if the code vectors being use are independent, that is they can not be divided by each other. When the node receives N packets, where N is the length of the coding vectors been used for the transmission, it then passes to decode the N packets by

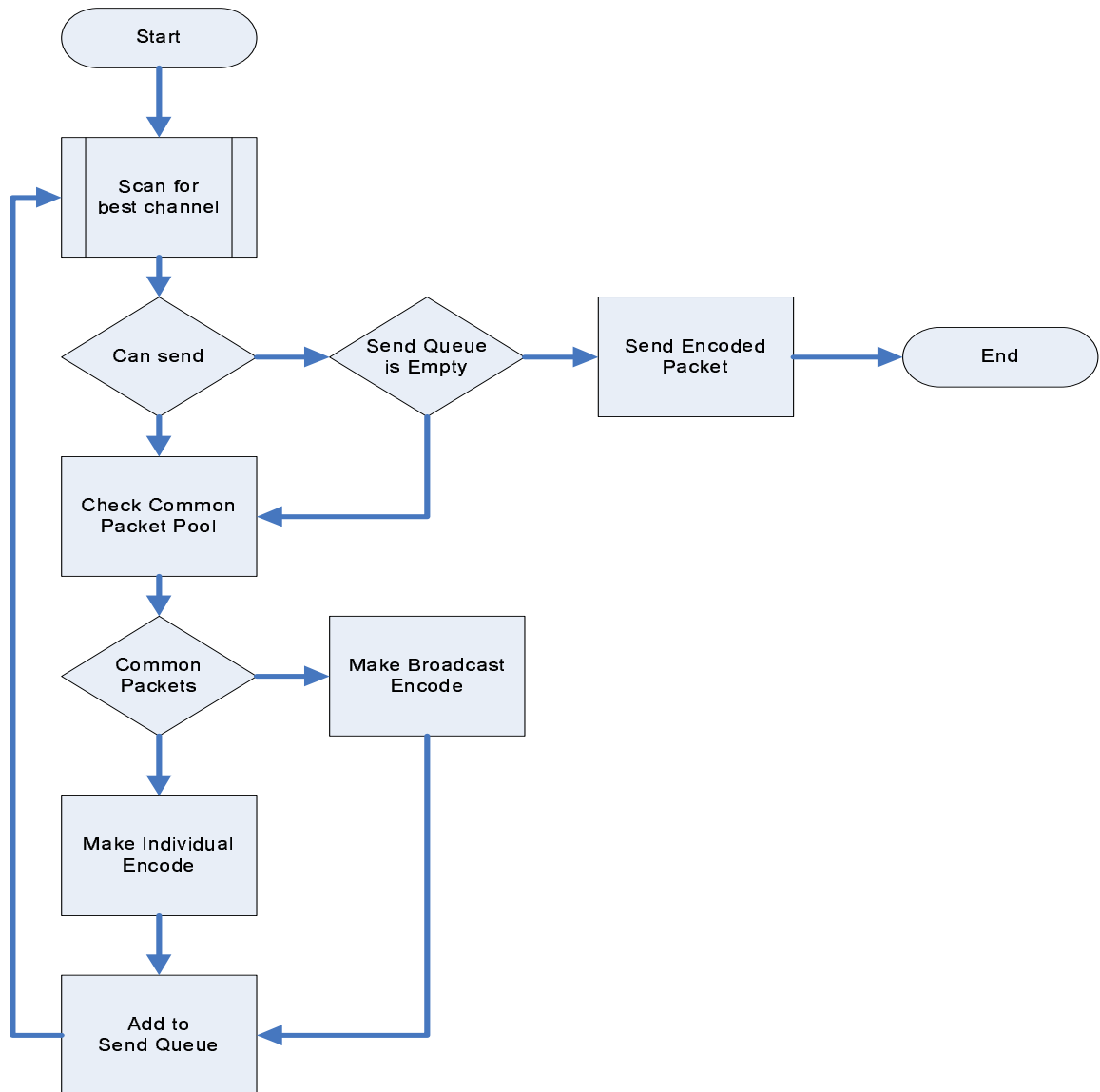


Figure 6-5: Network coding send flow chart

multiplying the inverse of the matrix formed by the coding vectors times the encoded packet themselves. Then it stores the N native packets in the corresponding pools.

Figure 6-7 shows an example of an encoding and decoding of two packets. The different colors represent the different channels. In this example the computer at the lower left of the mesh wants to send x and y to the other three computers. Later the three computers receive two packets each one. There is actually no gain in sending the mixture to its neighbor, since it still requires two transmissions. However

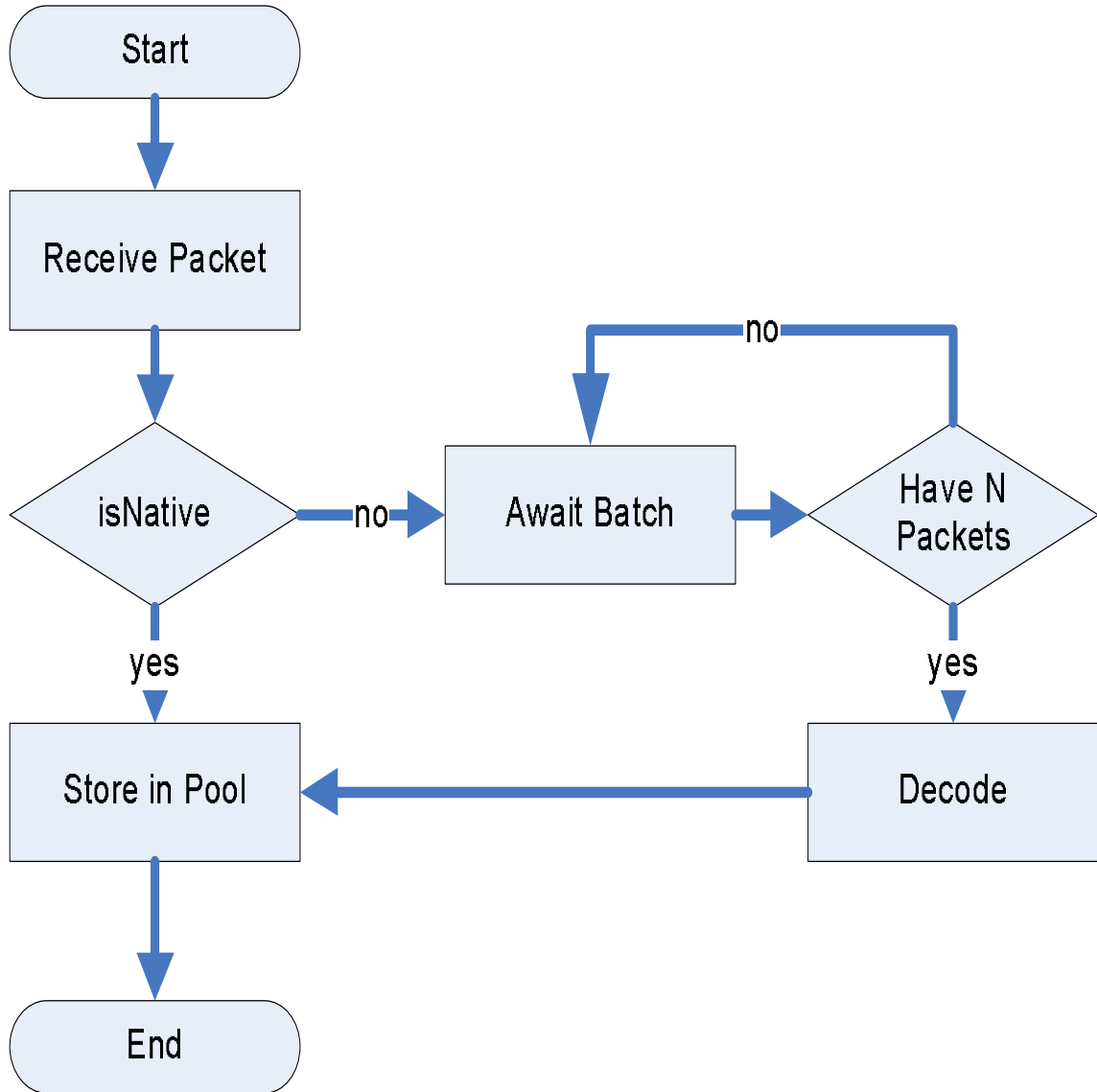


Figure 6–6: Network coding receive flow chart

the other two foreign computers will receive the two packets in two transmissions, instead of four.

6.4 EXPERIMENT AND DISCUSSION

In figure 6–8 we see the simulation environment, developed in NS2 [70] using C++ and TCL [71], to test CHANET. The C++ code used as base for CHANET was the UWBMAC developed by the authors in [53]. Table 6–1 shows that node 5 in the simulation was used as an intermediate node and all other nodes had a destination located at the opposite of them. All traffic was forced to passed by node

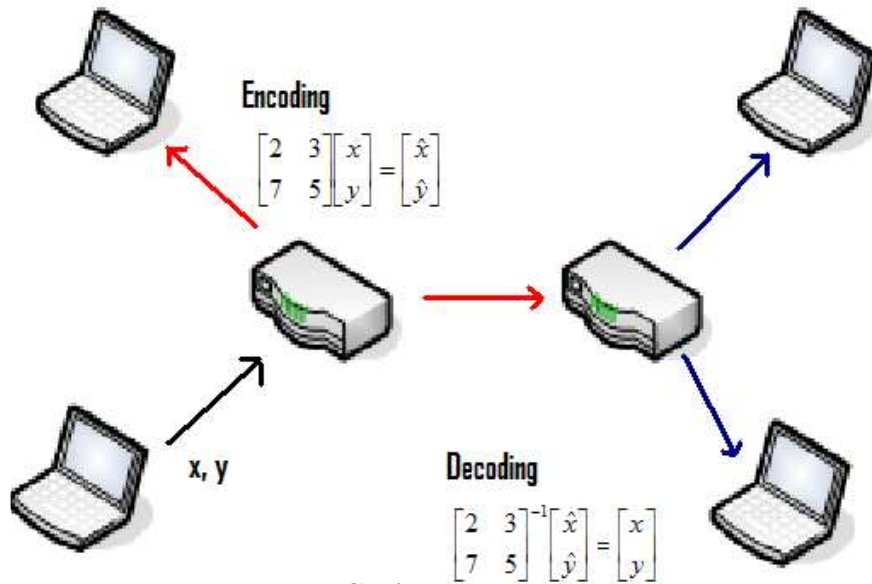


Figure 6-7: CHANET example scenario

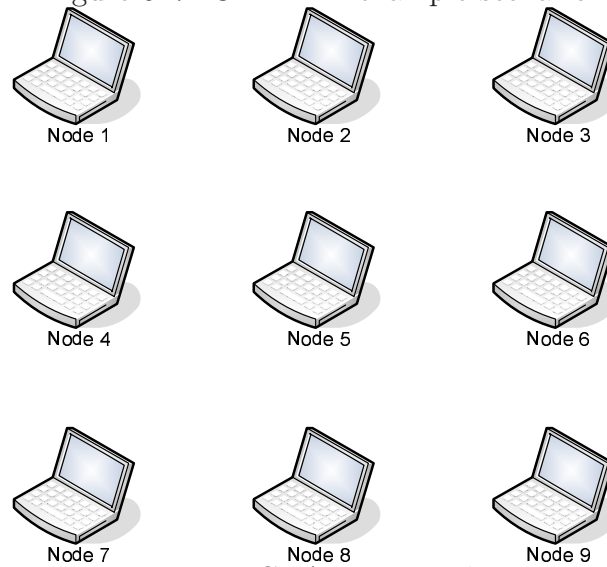


Figure 6-8: CHANET simulation

5 in order to reach its destination. Each simulation was run for an amount of 40 seconds at a data rate of 54MB per second. The results found after testing the simulations were very satisfactory. These results are presented in table 6-2.

As seen in the above table, the more channels are available the better the system performs. In the second row one can appreciate how the selection of the available channels for the run time successfully improved the system performance by 221% in terms of throughput and reduced the delay by 32.84%. These results were obtained because of the collision rate reduction, thus avoidance of many retransmissions. It

Table 6–1: Connection Flows

Source	Destination
1	7, 8, 9
2	7, 8, 9
3	7, 8, 9
4	3, 6, 9
6	1, 4, 7
7	1, 2, 3
8	1, 2, 3
9	1, 2, 3

Table 6–2: Performance Comparison

#Chan/Time(s)	Throughput (Mb/s)	Delay (s)
1 Chan/40 s	19.08	0.01623
3 Chan/40 s (fuzzy)	42.09	0.0109
3 Chan/40 s (CHANET)	81.12	0.00113

is important to notice that the advantage and increase in throughput is due to the fact that transmissions are multicasts, thus making it possible to achieve a capacity higher than 54Mbps, since one transmissions is meant to be for N hosts instead of having to send N transmissions in order to communicate with them.

Meanwhile in the third row we can appreciate what happens when we add the network coding functionality. Due to the reduction in transmissions and the mixed data that can pass at a given time through a channel the delay is reduced and also the collisions, making them practically zero. This caused an increase in throughput of almost twice (192.7%) the one obtained only using 3 channels. Since network coding was applied during the scan of channels then we can also obtain a smaller delay, that is 89.7% less. However this speed increase may not happen if network coding is applied at the moment of sending a packet, rather than when scanning for available channels.

6.5 CONCLUSION

Having into account the results obtained at the simulations one can say that it is encouraged to use the maximum number of channels available from the start in

order to reduce the percentage of collisions in the system if these channels do not overlap each other. For this reason we used 3 channels since the 802.11 standard has at most 3 non-overlapping channels.

The proposed system shows that fuzzy logic is indeed a powerful tool in the field of networking. It allows protocols to take dynamic decisions in an intelligent manner by simplifying complex scenarios into a logical statement and reducing the side effects of hidden terminals in the network without entering in the details of a complex mathematical model.

It is also shown that network coding increases the throughput by reducing the number of transmissions, thus reducing the number of collisions and increasing the speed of the network. Hence network coding contributes greatly in security since in order to find the content of a certain packet; more than one packet is needed.

As a contribution this work offers an alternative to effective dynamic channel selection without having into account network layer functions, thus simplifying the development of the protocol.

For future work, the proposed system will be extended to analyze collisions by channel instead of a whole network. This will improve even more the performance since it will only change the channels that are causing the problem and let the others that are not working without any changes.

The next step in the CHANET protocol would be now to implement it on a real testbed.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 CONCLUSIONS

In this thesis we studied the performance of wireless mesh networks with the current technology. As seen WMNs are an effective solution to provide many multimedia services because of their low cost implementation, scalability, and broadcast nature. However the current network architecture is not designed for such applications and although it can be used, it may not provide the service and management of it. Thus a service-oriented routing algorithm was proposed. Having SORA as a cross-layered design protocol it could use information of the network layer and the application layer to discriminate routes and calculate effectively the best option to deliver a service. SORA only needs to specify the IP address of the source to work while the destination address is selected by the mesh, where the best option is computed. SORA reduced the delay of a session reply since it got the service from the closest server, and managed to distribute the network load to keep it as local as possible.

It was also shown that WMNs are a promising technology, that allow expansion of range of the traditional wireless network, through the usage of intermediate nodes that act as relays. Further more we presented a way to make a WMN resemble even more a wired network by using multiple channels to obtain a full-duplex connection.

To accomplish this we took advantage of the fact that the 2.4GHz frequency band provides at most three non-overlapping channels. Hence we developed a Fuzzy Logic algorithm to dynamically select the best channel to transmit, meanwhile the

other two could be used for receiving packets, if the node happened to have more than one radio interface. This approach allowed the nodes to adapt to the environment even if new nodes were to be added to the WMN or interference became greater.

On the other hand, network coding turn out to be great for improving channel utilization. By mixing data network coding was able to transmit multiple packets through one channel at the same time. It was demonstrated in this thesis that the biggest achievement of network coding is gained by combining multiple techniques. In this work we applied network coding while the process of scanning for the best channel to transmit was performed, thus reducing the delay introduced in processing the packets.

7.2 FUTURE WORK

WMNs turned out to be a field where great development is needed. As future projects to follow this thesis we can mention the following:

- Implementation of CHANET in a real testbed: the fact that NS2 was used as platform for CHANET makes easier to perform the integration, since NS2 provides already a structural design of the network protocol stack. A challenge that may be found in this step is realizing that the protocol as it is consumes a lot of energy, hence modifications (e.g. see [72]) will have to be done in order to make it viable for a WMN to run it. It could happen also that the fuzzy sets created for this scenario is not the optimal in all cases, for which the use of a genetic algorithm could help to adapt these definitions according to each environment.
- Integration of SORA and CHANET into the same system: this will add some security to SORA, which right now is very vulnerable to intruders. By adding network coding then the intruder would need to intercept more than one packet and know the structure and decoding process to successfully find the real content of the packets.

- Adding QoS: to further extend the functionality of CHANET and SORA once they are integrated it is necessary to have in mind the quality of service. It is important that our protocols can guarantee certain quality on the network so that multimedia applications (e.g. VoIP, video streaming, IPTV, etc.) may use them as transport medium.
- Adding security: although network coding does provide a some sense of security, it is a technique for improving channel utilization and not security itself. This causes the algorithm to be weak to some hacking techniques. Hence for this reason we propose using an authentication scheme such as the one discussed in [73]. In this work the authors developed a key distribution scheme for wireless sensor networks. This same scheme could be use in a WMN and encode the data using one of the common keys. In this manner an attacker would have to know all the possible keys in order to decode correctly the data. These keys could be selected as prime numbers to speed up the key-to-encode selecting process. Further more this scheme could be used at the physical layer, allowing it to be computed with hardware, thus making it faster and more robust (i.e. harder to be decoded by intruders).
- Adding services to intermediate nodes: two make the delivery of the services even faster and more reliable, one can consider adding external hard drives to the intermediate nodes so that they can provide the services for themselves. Normally doing this would cause problems since routers should focus in communication only and have the least amount of processing to do. This is not the case for SORA, because by its nature SORA distributes the work load among the whole mesh.

APPENDICES

APPENDIX A

SORA SOURCE CODE FRAGMENTS

A.1 SORA PACKET STRUCTURE IN C

```
struct sora_packet{  
    struct in_addr client;  
    char flags;  
    unsigned int size;  
    char mess[MAXDATASIZE];  
};
```

A.2 SORA SERVICE SESSION STRUCTURE IN C

```
struct ServiceHash{  
    int clientfd;  
    struct in_addr clientip;  
    int service;  
};
```

A.3 SORA ROUTE TABLE STRUCTURE IN C

```
struct Route{  
    /*Type of service being offered*/  
    unsigned int service;  
    /*IP Address of the server with the service*/  
    struct in_addr server;  
    /*Interface where the next hop is*/  
    struct in_addr interface;
```

```

/*Number of hops to the server*/
unsigned int hop;
};

```

A.4 SORA ROUTER FORWARD REPLY PROCEDURE FRAGMENT IN C

```

while(1){
    unsigned int type = packet.flags;
    type &= 224;
    type >>= 5;
    returnCode = send(client, (struct sotp_request*)&packet,
                      sizeof(struct sotp_request), 0);

    if(type == MORE){
        int numbytes = 0;
        if((numbytes = recv(server, (struct sotp_request*)&packet,
                          sizeof(struct sotp_request), 0)) == -1)
        {
            printf("Error in router recv() \n");
            exit(-1);
        }
        returnCode = numbytes == -1 ? numbytes : returnCode;
    }
    else
    {
        break; /*Type is ANSWER... END TRANSMISSION*/
    }
}

```

A.5 SORA ROUTER FORWARD REQUEST SENDING PROCEDURE FRAGMENT IN C

```

for(i = 0; i < numberOfServers; ++i)
{
    if(servers[i].service == request){
        serverIndex = servers[serverIndex].service != request ||
            servers[i].hop < servers[serverIndex].hop ? i : serverIndex;
    }
}

```

A.6 SORA SERVER PROCEDURE FRAGMENT IN C

```

while(1) {
    sin_size=sizeof(struct sockaddr_in);
    if ((fd2 = accept(fd,(struct sockaddr *)&client, sin_size))== -1) {
        printf("error in accept()\n");
        exit(-1);
    }
    int numbytes = 0;
    if(fork() == 0){
        sleep(1); /*Force flow to parent so that
                    other incoming requests may be attended*/
        if ((numbytes=recv(fd2, (struct sotp_request*)&packet,
                           sizeof(struct sotp_request),0)) == -1){
            printf("Error en server recv() \n");
            exit(-1);
        }

        unsigned int type = packet.flags;

```

```

    type &= 224;
    type >>= 5;
    if(type == REQUEST)
    {
        unsigned int service = packet.flags;
        service &= 31;
        struct in_addr requester = packet.client;
        Byte* params = packet.mess;
        unsigned int length = packet.size;
        attendRequest(requester, service, params, length, fd2, client);
    }

    return; /*End child process*/
}

++inform_time;
if(inform_time == INFORMTIME){
    inform_time = 0;
    notifyServices(ipa, gate);
}
}

```

A.7 SORA AGENT FRAGMENT IN JAVA

```

while(true){
    Socket s1=s.accept(); // Wait and accept a connection
    int servicePort = getServicePort(s1);
    Socket localconn = new Socket("localhost", servicePort);
    DataInputStream is = new DataInputStream(localconn.getInputStream());
    DataOutputStream os = new DataOutputStream(s1.getOutputStream());
    byte[] out = new byte[10000];

```

```
is.read(out);  
os.write(new String(out));  
localconn.close();  
s1.close();  
}
```


APPENDIX B

CHANET SIMULATION

B.1 FUZZY.H

```
#include<iostream>

#include<stream.h>

#ifndef FUZZY_H
#define FUZZY_H

const int TRI_SHAPE = 0;
const int Z_SHAPE = 1;
const int S_SHAPE = 2;
const int D_MAX = 4;
const int D_MOM = 5;
const int D_COA = 6;
const int D_LMAX = 7;
const int OR = 8;
const int AND = 9;
const int NOT = 10;
const int UNKNOWN_TYPE = -1;

extern "C++"{
    class Fuzzy{
```

```
private:
    double min;
    double center;
    double max;
    double start;
    int length;
    int type;
    double* member;
    double* membership;

public:
    Fuzzy();
    Fuzzy(const Fuzzy& other);
    Fuzzy(double start, int length);
    ~Fuzzy();
    int getType();
    void setType(int type);
    double getMin();
    double getCenter();
    double getMax();
    int getLength();
    double* getMembers();
    double* getMemberships();
    double getMemberAt(int pos);
    double getMembershipAt(int pos);
    void setMin(double value);
    void setCenter(double value);
    void setMax(double value);
```

```

void setMemberAt(int pos, double value);
void setMembershipAt(int pos, double value);
double TriShape(double x);
double ZShape(double x);
double SShape(double x);
double ShapeValue(double x);
void generateMembership(int type);
double decode(int type);
Fuzzy& operator=(const Fuzzy& other);
Fuzzy operator~();
Fuzzy operator+(Fuzzy& other);
Fuzzy operator&(Fuzzy& other);
Fuzzy& operator[](int column);
friend ostream& operator<<(ostream& out, const Fuzzy& logic);
friend Fuzzy IfThen(Fuzzy set, double value, Fuzzy& thenClause);
friend Fuzzy IfThen(Fuzzy* sets, double* values, int numCond,
                    int conditions, Fuzzy& thenClause);
};
};
#endif

```

B.2 FUZZY.CC

```

#include <iostream>
#include <assert.h>
#include <stream.h>
#include "fuzzy.h"

```

```

Fuzzy::Fuzzy()

```

```

{
    this->min = 0;
    this->center = 0;
    this->max = 0;
    this->start = 0;
    this->length = 20;
    this->member = new double[20];
    this->membership = new double[20];
    for(int i = 0; i < length; ++i)
    {
        this->member[i] = 0;
        this->membership[i] = 0;
    }
}

Fuzzy::Fuzzy(const Fuzzy& other)
{
    this->start = other.start;
    this->length = other.length;
    this->member = NULL;
    this->membership = NULL;
    this->member = new double[other.length];
    this->membership = new double[other.length];
    for(int i = 0; i < this->length; ++i)
    {
        this->member[i] = other.member[i];
        this->membership[i] = other.membership[i];
    }
}

```

```

        this->min = other.min;

        this->center = other.center;

        this->max = other.max;

        this->type = other.type;
    }

Fuzzy::Fuzzy(double start, int length)
{
    this->min = 0;

    this->center = 0;

    this->max = 0;

    this->start = start;

    this->length = length;

    this->member = new double[length];

    this->membership = new double[length];

    for(int i = 0; i < length; ++i)
    {
        this->member[i] = start + i;

        this->membership[i] = 0;
    }
}

Fuzzy::~~Fuzzy()
{
    this->min = 0;

    this->center = 0;

    this->max = 0;

    delete[] this->member;

    delete[] this->membership;
}

```

```
        this->member = NULL;
        this->membership = NULL;
    }
    int Fuzzy::getType()
    {
        return this->type;
    }
    void Fuzzy::setType(int type)
    {
        this->type = type;
        this->generateMembership(this->type);
    }
    double Fuzzy::getMin()
    {
        return this->min;
    }
    double Fuzzy::getCenter()
    {
        return this->center;
    }
    double Fuzzy::getMax()
    {
        return this->max;
    }
    int Fuzzy::getLength()
    {
        return this->length;
```

```
}

double* Fuzzy::getMembers()
{
    return this->member;
}

double* Fuzzy::getMemberships()
{
    return this->membership;
}

double Fuzzy::getMemberAt(int pos)
{
    if(pos >=0 && pos < this->length)
    {
        return this->member[pos];
    }

    cerr << "Error: Invalid position in the array" << endl;
    return -1;
}

double Fuzzy::getMembershipAt(int pos)
{
    if(pos >=0 && pos < this->length)
    {
        return this->membership[pos];
    }

    cerr << "Error: Invalid position in the array" << endl;
    return -1;
}
```

```

void Fuzzy::setMin(double value)
{
    this->min = value;
}

void Fuzzy::setCenter(double value)
{
    this->center = value;
}

void Fuzzy::setMax(double value)
{
    this->max = value;
}

void Fuzzy::setMemberAt(int pos, double value)
{
    if(pos >=0 && pos < this->length)
    {
        this->member[pos] = value;
        return;
    }
    cerr << "Error: Invalid position in the array" << endl;
}

void Fuzzy::setMembershipAt(int pos, double value)
{
    if(pos >=0 && pos < this->length)
    {
        this->membership[pos] = value;
    }
}

```



```

    cerr << "Error: Invalid position in the array" << endl;
}

double Fuzzy::TriShape(double x)
{
    if(x <= this->min)
    {
        return 0;
    }
    else if(x > this->min && x <= this->center)
    {
        return (1/(this->center - this->min))*(x - this->min);
    }
    else if(x > center && x <= max)
    {
        return (1/(this->center - this->max))*(x - this->max);
    }
    else
    {
        return 0;
    }
}

double Fuzzy::ZShape(double x)
{
    if(x <= this->min)
    {
        return 1;
    }
}

```

```

else if(x > this->min && x <= this->center)
{
    return (1/(this->min - this->center))*(x - this->center);
}
else
{
    return 0;
}
}

double Fuzzy::SShape(double x)
{
    if(x <= this->center)
    {
        return 0;
    }
    else if(x > this->center && x <= this->max)
    {
        return (1/(this->max - this->center))*(x - this->center);
    }
    else
    {
        return 1;
    }
}

double Fuzzy::ShapeValue(double x)
{
    switch(this->type)

```

```

{
    case TRI_SHAPE:
        return TriShape(x);

    case Z_SHAPE:
        return ZShape(x);

    case S_SHAPE:
        return SShape(x);

    default:
        cerr << "Error: Unknown membership type" << endl;
}

return UNKNOWN_TYPE;
}

void Fuzzy::generateMembership(int type)
{
    this->type = type;
    switch(type)
    {
        case TRI_SHAPE:
            for(int i = 0; i < this->length; ++i)
            {
                this->membership[i] = TriShape(this->member[i]);
            }
            break;

        case Z_SHAPE:
            for(int i = 0; i < this->length; ++i)
            {
                this->membership[i] = ZShape(this->member[i]);
            }

```

```

        }
        break;
        case S_SHAPE:
            for(int i = 0; i < this->length; ++i)
            {
                this->membership[i] = SShape(this->member[i]);
            }
            break;
        default:
            cerr << "Error: Unknown membership type" << endl;
    }
}

double Fuzzy::decode(int type)
{
    double maxValue = 0;
    double sum = 0;
    double count = 0;
    double numerator = 0;
    double denominator = 0;
    double result = 0;

    for(int i = 0; i < this->length; ++i)
    {
        maxValue = this->membership[i] >= maxValue
            ? this->membership[i] : maxValue;
    }
    switch(type)

```

```

{
    case D_MAX:
        for(int i = 0; i < this->length; ++i)
        {
            if(this->membership[i] == maxValue)
            {
                return this->member[i];
            }
        }
        break;
    case D_LMAX:
        for(int i = 0; i < this->length; ++i)
        {
            if(this->membership[i] == maxValue)
            {
                result = this->member[i];
            }
        }
        return result;
        break;
    case D_MOM:
        for(int i = 0; i < this->length; ++i)
        {
            if(this->membership[i] == maxValue)
            {
                sum += this->member[i];
                ++count;
            }
        }

```

```

        }

    }

    return (sum / count);

break;

case D_COA:

    for(int i = 0; i < this->length; ++i)
    {
        numerator += (this->membership[i] * this->member[i]);
        denominator += this->membership[i];
    }

    result = denominator == 0 ? 0 : (numerator/denominator);

    return result;

break;

default:

    cerr << "Error: Unknown decoding method" << endl;

}

return result;

}

Fuzzy& Fuzzy::operator=(const Fuzzy& other)
{
    //if(this != &other)
    //{
        //if(this->length == other.length && this->start == other.start)
        //{
            delete[] this->member;

            delete[] this->membership;

            this->length = other.length;

```

```

        this->start = other.start;
        this->member = NULL;
        this->membership = NULL;
        this->member = new double[this->length];
        this->membership = new double[this->length];
        for(int i = 0; i < this->length; ++i)
        {
            this->member[i] = other.member[i];
            this->membership[i] = other.membership[i];
        }
        this->min = other.min;
        this->center = other.center;
        this->max = other.max;
        this->type = other.type;
    //}
//}
    return *this;
}

Fuzzy Fuzzy::operator~()
{
    Fuzzy tempNegation(*this);
    for(int i = 0; i < tempNegation.length; ++i)
    {
        tempNegation.membership[i] = 1 - tempNegation.membership[i];
    }
    return tempNegation;
}

```

```

Fuzzy Fuzzy::operator+(Fuzzy& other)
{
    Fuzzy tempUnion = Fuzzy(this->start, this->length);
    //if(this->length == other.length && this->start == other.start)
    //{
    for(int i = 0; i < this->length; ++i)
    {
        double value = 0;
        int otherType = other.type;
        value = this->membership[i] >= other.membership[i]
            ? this->membership[i] : other.membership[i];
        tempUnion.membership[i] = value;
    }
    return tempUnion;
    //}
}

Fuzzy Fuzzy::operator&(Fuzzy& other)
{
    Fuzzy tempIntersection = Fuzzy(this->start, this->length);
    //if(this->length == other.length && this->start == other.start)
    //{
    for(int i = 0; i < this->length; ++i)
    {
        double value = 0;
        int otherType = other.type;
        value = this->membership[i] <= other.membership[i]
            ? this->membership[i] : other.membership[i];
    }
}

```



```

        tempIntersection.membership[i] = value;
    }
    return tempIntersection;
    //{
}

Fuzzy& Fuzzy::operator[](int column)
{
    int length = sizeof(this)/sizeof(Fuzzy);
    assert(column < length);
    return this[column];
}

ostream& operator<<(ostream& out, const Fuzzy& logic)
{
    /*out << "Members:" << endl;
    for(int i = 0; i < logic.length; ++i)
    {
        if(i == logic.length-1)
        {
            out << logic.member[i] << " ";
        }
        else if(i == 0)
        {
            out << " " << logic.member[i] << " ";
        }
        else
        {
            out << logic.member[i] << " ";
        }
    }
    out << endl;
    */
}

```

```

    }
}
out << "\nMemberships:" << endl;
for(int i = 0; i < logic.length; ++i)
{
    if(i == logic.length-1)
    {
        out << logic.membership[i] << "";
    }
    else if(i == 0)
    {
        out << "" << logic.membership[i] << " ";
    }
    else
    {
        out << logic.membership[i] << " ";
    }
} */
out << "{";
for(int i = 0; i < logic.length; ++i)
{
    if(logic.membership[i] != 0)
    {
        out << logic.membership[i] << "/" << logic.member[i];
    }
    if(i != logic.length - 1 && logic.membership[i] != 0
        && logic.membership[i + 1] != 0)

```

```

        {
            out << "\t";
        }
    }
    out << "}";
    return out;
}

```

Fuzzy IfThen(Fuzzy set, double value, Fuzzy& thenClause)

```

{
    Fuzzy Conclusion(thenClause);
    double limit = 1;
    limit = set.ShapeValue(value);

    for(int i = 0; i < Conclusion.length; ++i)
    {
        Conclusion.membership[i] = (Conclusion.membership[i] < limit)
                                   ? Conclusion.membership[i] : limit;
    }

    Conclusion.type = thenClause.type;
    Conclusion.start = thenClause.start;
    Conclusion.length = thenClause.length;
    Conclusion.min = thenClause.min;
    Conclusion.center = thenClause.center;
    Conclusion.max = thenClause.max;
    return Conclusion;
}

```

```

Fuzzy IfThen(Fuzzy* sets, double* values, int numCond,
             int condition, Fuzzy& thenClause)
{
    Fuzzy Conclusion(thenClause);
    double limit = 1;
    switch(condition)
    {
        case AND:
            limit = 1;
            for(int i = 0; i < numCond; ++i)
            {
                limit = (sets[i].ShapeValue(values[i]) < limit)
                        ? sets[i].ShapeValue(values[i]) : limit;
            }
            break;
        case OR:
            limit = 0;
            for(int i = 0; i < numCond; ++i)
            {
                limit = (sets[i].ShapeValue(values[i]) > limit)
                        ? sets[i].ShapeValue(values[i]) : limit;
            }
            break;
    }
    for(int i = 0; i < Conclusion.length; ++i)
    {

```

```

        Conclusion.membership[i] = (Conclusion.membership[i] < limit)
                                ? Conclusion.membership[i] : limit;
    }

    Conclusion.type = thenClause.type;
    Conclusion.start = thenClause.start;
    Conclusion.length = thenClause.length;
    Conclusion.min = thenClause.min;
    Conclusion.center = thenClause.center;
    Conclusion.max = thenClause.max;
    return Conclusion;
}

```

B.3 CHANNEL UTILAZATION AGENT CODE IN C++

```

class UMAC_Agent : public Agent {
public:
    UMAC_Agent();
protected:
    int command(int argc, const char*const* argv);
private:
    int result;
    /* Fuzzy Logic Channel Selection */
    int newUtilization(int actualUtilization, int numOfRoutes,
        int numOfCollisions, int numOfTransmits);
};

static class UmacAgent : public TclClass {
public:

```

```

    UmacAgent() : TclClass("Agent/UmacAgent") {}

    TclObject* create(int, const char*const*) {
        return(new UMAC_Agent());
    }
} class_umac_agent;

UMAC_Agent::UMAC_Agent() : Agent(PT_UDP){
    result = 0;
    bind("res_", &result);
}

int UMAC_Agent::command(int argc, const char*const* argv){
    if(argc == 6) {
        if(strcmp(argv[1], "new-utilization") == 0) {
            int param1 = atoi(argv[2]);
            int param2 = atoi(argv[3]);
            int param3 = atoi(argv[4]);
            int param4 = atoi(argv[5]);
            newUtilization(param1, param2, param3, param4);
            return(TCL_OK);
        }
    }
    return(Agent::command(argc, argv));
}

int UMAC_Agent::newUtilization(int actualUtilization, int numOfRoutes,
                                int numOfCollisions, int numOfTransmits){

```

```

/* Percent of utilization for Fuzzy Sets */
double percentOfUtil = ((double)actualUtilization)/((double)numOfRoutes);
percentOfUtil *= 100;
double percentOfColi = ((double)numOfCollisions)/((double)numOfTransmits);
percentOfColi *= 100;

Fuzzy none(0, 100), low(0,100), medium(0,100), high(0,100), blocked(0,100);
Fuzzy nothing(0, 100), some(0,100), alot(0,100), overloaded(0,100);

/*Type definitions */
none.setType(Z_SHAPE);
low.setType(TRI_SHAPE);
medium.setType(TRI_SHAPE);
high.setType(TRI_SHAPE);
blocked.setType(S_SHAPE);

nothing.setType(Z_SHAPE);
some.setType(TRI_SHAPE);
alot.setType(TRI_SHAPE);
overloaded.setType(S_SHAPE);

/* Bounds definitions */
none.setMin(2);
none.setCenter(12);

low.setMin(10);
low.setCenter(22);

```

```
low.setMax(35);
```

```
medium.setMin(33);
```

```
medium.setCenter(45);
```

```
medium.setMax(62);
```

```
high.setMin(60);
```

```
high.setCenter(75);
```

```
high.setMax(90);
```

```
blocked.setCenter(88);
```

```
blocked.setMax(98);
```

```
nothing.setMin(2);
```

```
nothing.setCenter(22);
```

```
some.setMin(18);
```

```
some.setCenter(33);
```

```
some.setMax(44);
```

```
alot.setMin(40);
```

```
alot.setCenter(60);
```

```
alot.setMax(75);
```

```
overloaded.setCenter(70);
```

```
overloaded.setMax(90);
```



```

none.generateMembership(Z_SHAPE);
low.generateMembership(TRI_SHAPE);
medium.generateMembership(TRI_SHAPE);
high.generateMembership(TRI_SHAPE);
blocked.generateMembership(S_SHAPE);

nothing.generateMembership(Z_SHAPE);
some.generateMembership(TRI_SHAPE);
alot.generateMembership(TRI_SHAPE);
overloaded.generateMembership(S_SHAPE);

/*Desicion rules */
if(percentOfUtil > 50) //overloaded{
    Fuzzy firstRule = IfThen(blocked, percentOfColi, nothing);
    Fuzzy conditions[] = {high, medium};
    double values[] = {percentOfColi, percentOfColi};
    Fuzzy secondRule = IfThen(conditions, values,
        sizeof(conditions)/sizeof(Fuzzy), OR, some);
    Fuzzy thirdRule = IfThen(low, percentOfColi, alot);
    Fuzzy fourthRule = IfThen(none, percentOfColi, overloaded);
    /*Decoding the desicion using Center of Area method */
    Fuzzy fuzDecision = (firstRule + secondRule);
    fuzDecision = (fuzDecision + thirdRule);
    fuzDecision = (fuzDecision + fourthRule);
    /*Linear proportion to convert back to number of utilization */
    double newUtil = fuzDecision.decode(D_COA);
    newUtil *= ((double)numOfRoutes/100);

```

```

    newUtil = ceil(newUtil);
    result = (int)newUtil;
}

else if(percentOfUtil > 35) //a lot{
    Fuzzy firstRule = IfThen(blocked, percentOfColi, some);
    Fuzzy conditions[] = {high, medium};
    double values[] = {percentOfColi, percentOfColi};
    Fuzzy secondRule = IfThen(conditions, values,
        sizeof(conditions)/sizeof(Fuzzy), OR, some);
    Fuzzy thirdRule = IfThen(low, percentOfColi, alot);
    Fuzzy fourthRule = IfThen(none, percentOfColi, alot);
    /*Decoding the desicion using Center of Area method */
    Fuzzy fuzDecision = (firstRule + secondRule);
    fuzDecision = (fuzDecision + thirdRule);
    fuzDecision = (fuzDecision + fourthRule);
    /*Linear proportion to convert back to number of utilization */
    double newUtil = fuzDecision.decode(D_COA);
    newUtil *= ((double)numOfRoutes/100);
    newUtil = ceil(newUtil);
    result = (int)newUtil;
}

else if(percentOfUtil > 15){
    Fuzzy firstRule = IfThen(blocked, percentOfColi, alot);
    Fuzzy conditions[] = {high, medium};
    double values[] = {percentOfColi, percentOfColi};
    Fuzzy secondRule = IfThen(conditions, values,
        sizeof(conditions)/sizeof(Fuzzy), OR, alot);

```

```

    Fuzzy thirdRule = IfThen(low, percentOfColi, some);
    Fuzzy fourthRule = IfThen(none, percentOfColi, some);
    /*Decoding the desicion using Center of Area method */
    Fuzzy fuzDecision = (firstRule + secondRule);
    fuzDecision = (fuzDecision + thirdRule);
    fuzDecision = (fuzDecision + fourthRule);
    /*Linear proportion to convert back to number of utilization */
    double newUtil = fuzDecision.decode(D_COA);
    newUtil *= ((double)numOfRoutes/100);
    newUtil = ceil(newUtil);
    result = (int)newUtil;
}

else{
    Fuzzy firstRule = IfThen(blocked, percentOfColi, overloaded);
    Fuzzy conditions[] = {high, medium};
    double values[] = {percentOfColi, percentOfColi};
    Fuzzy secondRule = IfThen(conditions, values,
        sizeof(conditions)/sizeof(Fuzzy), OR, alot);
    Fuzzy thirdRule = IfThen(low, percentOfColi, some);
    Fuzzy fourthRule = IfThen(none, percentOfColi, nothing);
    /*Decoding the desicion using Center of Area method */
    Fuzzy fuzDecision = (firstRule + secondRule);
    fuzDecision = (fuzDecision + thirdRule);
    fuzDecision = (fuzDecision + fourthRule);
    /*Linear proportion to convert back to number of utilization */
    double newUtil = fuzDecision.decode(D_COA);
    newUtil *= ((double)numOfRoutes/100);

```

```
        newUtil = ceil(newUtil);  
        result = (int)newUtil;  
    }  
    return 0;  
}
```

REFERENCE LIST

- [1] I.F. Akyildiz and Xudong Wang. A survey on wireless mesh networks. *Communications Magazine, IEEE*, 43:S23–S30, 2005.
- [2] Oasis soa reference model. <http://www.oasis-open.org/>.
- [3] Robert David Callaway. *An Autonomic Service Delivery Platform for Service-Oriented Network Environments*. PhD thesis, North Carolina State University, 2008.
- [4] Xabier Larrucea, Gorka Benguria, and Stefan Schuster. Mdsoa for achieving interoperability. *IEEE ICCBSS 07*, 2007.
- [5] Stephane H. Maes. Service delivery platforms as it realization of oma service environment: Service oriented architectures for telecommunications. *IEEE WCNC 2007*, 2007.
- [6] Pasquale locola. When legacy meets soa: Achieving business agility by integrating new technology with existing software asset. *IEEE Systems Conference 2007*, 2007.
- [7] Harry M. Sneed. Integrating legacy software into a service oriented architecture. *CSMR 06*, 2006.
- [8] Soap online manual. <http://www.w3schools.com/>.
- [9] Kejie Lu, Yi Qian, and Hsiao-Hwa Chen. A secure and service-oriented network control framework for wimax networks. *IEEE Communications Magazine*, 45:124–130, 2007.
- [10] Gary Luckenbaugh, Scott Landriau, Jon Dehn, and Sid Rudolph. Service oriented architecture for the next generation air transportation system. *ICNS 07*, 2007.

- [11] Firat Kart, Gengxin Miao, L. E. Moser, and P. M. Melliar-Smith. A distributed e-healthcare system based on the service oriented architecture. *SCC 2007*, 2007.
- [12] Vincent Ricquebourg, David Menga, Bruno Marhic, Laurent Delahoche, David Durand, and Christophe Log. Service oriented architecture for context perception based on heterogeneous sensors network. *IECON 2006*, 2006.
- [13] Hanwei Chen, Jianwei Yin, Lu Jin, Ying Li, and Jinxiang Dong. Jtang synergy: A service oriented architecture for enterprise application integration. *CSCWD 2007*, 2007.
- [14] Peter Hrastnik and Werner Winiwarter. Coordination in service oriented architectures using transaction processing concepts. *DEXA 07*, 2007.
- [15] Ketil Lund, Anders Eggen, Dinko Hadzic, Trude Hafse, and Frank T. Johnsen. Using web services to realize service oriented architecture in military communication networks. *Communications Magazine*, 45, 2007.
- [16] P. Rajesh, S. Ranjiith, P. R. Soumya, V. Karthik, and S. Datthathreya. Network management system using web services and service oriented architecture. *NOMS 2006*, 2006.
- [17] Leobino Sampaio, Ivo Koga, Rafael Costa, Herbert Monteiro, Jose A. Suruagy Monteiro, Fausto Vetter, Guilherme Fernandes, and Murilo Vetter. Implementing and deploying network monitoring service oriented architectures. *LANOMS 2007*, 2007.
- [18] D. Cotroneo, C. Di Flora, and S. Russo. Improving dependability of service oriented architectures for pervasive computing. *Proceedings of the Eighth International Workshop on Object-Oriented Real-Time Dependable Systems, 2003. (WORDS 2003).*, 2003.
- [19] Zhenyu Liut, Ning Gut, and Genxing Yang. A reliability evaluation framework on service oriented architecture. *ICPCA 2007*, 2007.

- [20] An Min-Jeong, Lee Hong-Chul, and Jin Hye-Jin. Design of the material control system based on service oriented architecture. *ICCAS 07*, 2007.
- [21] Firat Kart, Zhongnan Shen, and Cagdas Evren Gerege. The midas system: A service oriented architecture for automated supply chain management. *SCC 06*, 2006.
- [22] Haining Shu and Qilian Liang. Fuzzy optimization for distributed sensor deployment. *Proc. IEEE WCNC 2005*, 3, 2005.
- [23] Omar Villavicencio, Kejie Lu, Hua Zhu, and Sastri Kota. Performance of IEEE 802.11n in multi-channel multi-radio wireless ad hoc network. *Proc. IEEE Milcom 2007*, 2007.
- [24] Arindam K. Das, Rajiv Vijayakumar, and Sumit Roy. Static channel assignment in multi-radio multi-channel 802.11 wireless mesh networks: Issues, metrics and algorithms. In *Proc. IEEE Globecom*, 2006.
- [25] Ashish Raniwala and Tzi cker Chiueh. Architecture and algorithm for an IEEE 802.11-based multi-channel wireless mesh network. In *in Proc. of IEEE INFOCOM'05*, volume 3, pages 2223–2234, 2005.
- [26] Arindam K. Das, Hamed M.K. Alazemi, Rajiv Vijayakumar, and Sumit Roy. Optimization models for fixed channel assignment in wireless mesh networks with multiple radios. In *Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005. 2005 Second Annual IEEE Communications Society Conference on*, pages 463–474, 2005.
- [27] Krishna N. Ramachandran, Elizabeth M. Beldin, Kevin C. Almeroth, and Milind M. Buddhikot. Interference-aware channel assignment in multi-radio wireless mesh networks. In *Proc of INFOCOM*, 2006.
- [28] Mark Felegyhazi, Mario Cagalj, and Jean-Pierre Hubaux. Multi-radio channel allocation in competitive wireless mesh networks. In *in Proc. of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS)*

Workshops, page 36, 2006.

- [29] Bong-Jun Ko, Vishal Misra, Jitendra Padhye, and Dan Rubensteinf. Distributed channel assignment in multi-radio 802.11 mesh networks. In *Columbia University Technical Report*, 2006.
- [30] Haitao Wu, Fan Yang, Kun Tan, Jie Chen, Qian Zhang, and Zhensheng Zhang. Distributed channel assignment in multi-radio multichannel multi-hop wireless networks. In *IEEE journal on Selected Areas in Communications*, volume 24, pages 1972–1983, 2006.
- [31] Ritesh Maheshwari, Himanshu Gupta, and Samir R. Das. Multichannel mac protocols for wireless networks. In *Proc SECON 2006, Reston Virginia*, 2006.
- [32] Jeonghoon Mo, Hoi-Sheung Wilson So, and Jean Walrand. Comparison of multi-channel mac protocols. In *Proc. of the 8th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'05), Montreal, Qc., Canada*, 2005.
- [33] Axel Davidian and Ioannis Oikonomidis. Network coding: An overview. *Seminar on Topics in Communications Engineering Master of Science in Communications Engineering Munich University of Technology*, 2005.
- [34] Luiz Filipe M. Vieira, Archan Misra, and Mario Gerla. Performance of network-coding in multi-rate wireless environments for multicast applications. *MILCOM 07*, 2007.
- [35] Tracey Ho Keesook Han, Ralf Koetter, Muriel Medard, and Fang Zhao. On network coding for security. *MILCOM 07*, 2007.
- [36] Haruko Kawahigashi and Yoshiaki Terashima. Security aspects of the linear network coding. *MILCOM 07*, 2007.
- [37] Majid Ghaderi, Don Towsley, and Jim Kurose. Network coding performance for reliable multicast. *MILCOM 07*, 2007.

- [38] Joon-Sang Park, Desmond S. Lun, Fabio Soldo, Mario Gerla, and Muriel Médard. Performance of network coding in ad hoc networks. *MILCOM 06*, 2006.
- [39] Christina Fragouli, Dina Katabi, Athina Markopoulou, Muriel Medard, and Hariharan Rahul. Wireless network coding: Opportunities & challenges. *MILCOM 07*, 2007.
- [40] Sachin Katti, Hariharan Rahul, Wenjun Hu, Dina Katabi, Muriel Medard, and Jon Crowcroft. Xors in the air: practical wireless network coding. *SIGCOMM 06*, 36, 2006.
- [41] Szymon Chachulski, Michael Jennings, Sachin Katti, and Dina Katabi. Trading structure for randomness in wireless opportunistic routing. *SIGCOMM 07*, 2007.
- [42] Shengli Zhang, Soung-Chang Liew, and Patrick P.Lam. Physical-layer network coding. *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking*, 2006.
- [43] Sachin Katti, Shyamnath Gollakota, and Dina Katabi. Embracing wireless interference: Analog network coding. *SIGCOMM 07*, 2007.
- [44] Kejie Lu, Yi Qian, Hsiao-Hwa Chen, and Fu Shengli. Wimax networks: From access to service platform. *IEEE Network*, 22:38–45, 2008.
- [45] J. Barata, L. Ribeiro, and Armando Colombo. Diagnosis using service oriented architectures (soa). *Industrial Informatics, 2007 5th IEEE International Conference*, 2007.
- [46] Suzette Stoutenburg, Leo Obrst, Deborah Nichols, Ken Samuel, and Paul Franklin. Applying semantic rules to achieve dynamic service oriented architectures. *Second International Conference on Rules and Rule Markup Languages for the Semantic Web*, 2006.

- [47] Sriram Anand, Srinivas Padmanabhuni, and Jai Ganesh. Perspectives on service oriented architecture. *IEEE International Conference on Services Computing 2005*, 2005.
- [48] Catharina Candolin. A security framework for service oriented architectures. *MILCOM 07*, 2007.
- [49] Ivar Jorstad, Schahram Dustdar, and Do Van Thanh. A service oriented architecture framework for collaborative services. *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, 2005.
- [50] Clemens Martin and Khalil A. Abuosba. Utilizing a service oriented architecture for information security evaluation and quantification. *BDIM 07*, 2007.
- [51] John F. Meyer. Service oriented architecture (soa) migration strategy for u.s. operational naval meteorology and oceanography (metoc). *OCEANS 2007*, 2007.
- [52] Openwrt. <http://www.openwrt.org>.
- [53] Kejie Lu, Dapeng Wu, and Yuguang Fang. A novel framework for medium access control in ultra-wideband ad hoc networks. *Dynamics of Continuous, Discrete and Impulsive Systems, Series B*, 12:427–443, 2005.
- [54] Mansoor Alicherry, Randeep Bhatia, and Li (Erran) Li. Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 58–72. ACM Press, 2005.
- [55] Padhye J. Draves R. and Zill B. Routing in multi-radio, multi-hop wireless mesh networks. In *in Proc. of ACM Mobicom'04*, 2004.
- [56] Kodialam M. and Nandagopal T. Characterizing the capacity region in multi-radio multi-channel wireless mesh networks. In *Proc. of ACM Mobicom'05*, pages 73–87, 2005.

- [57] Kodialam M. and Nandagopal T. On the capacity region of multi-radio multi-channel wireless mesh networks. In *1st IEEE workshop on wireless mesh networks (WiMESH)*, 2005.
- [58] Vaidya N. H. Kyasanur. Routing and link-layer protocols for multichannel multi-interface ad hoc wireless networks. In *ACM SIGMOBILE Mobile Computing and Communications Review*, volume 10, pages 31–43, 2006.
- [59] D. Lin, Teng-Sheng Moh, and M. Moh. A delay-bounded multi-channel routing protocol for wireless mesh networks using multiple token rings: Extended summary. In *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 845–847, 2006.
- [60] M.K. Marina and S.R. Das. A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks. In *Broadband Networks, 2005 2nd International Conference on*, pages 381–390 Vol.1, 2005.
- [61] S. Merlin. Resource allocation in multi-radio multichannel wireless ad hoc networks. In *internal draft, unpublished, available www.dei.unidp.it/merlo/papers/mc4.pdf*.
- [62] Das A.K. Vijayakumar R. Alazemi H.M.K. Ma H. Roy, S. and Alotaibi E. Capacity scaling with multiple radios and multiple channels in wireless mesh networks. In *in First IEEE Workshop on Wireless Mesh Networks (WiMesh'05)*, 2005.
- [63] J. So and N. Vaidya. Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver. In *Proc. ACM MobiHoc*, 2004.
- [64] Gupta H. Subramanian A.P. and Das S.R. Minimum-interference channel assignment in multi-radio wireless mesh networks. In *WINGS Lab technical Report, unpublished, available: <http://www.cs.sunysb.edu/hgupta/ps/channel.pdf>*, 2006.

- [65] G. Tang J., Xue and W. Zhang. Interference-aware topology control and QoS routing in multi-channel wireless mesh networks. In *Proc. of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, Urbana Champaign*, pages 68–77, 2005.
- [66] Sachin Katti, Dina Katabi, Hari Balakrishnan, and Muriel Medard. Symbol-level network coding for wireless mesh networks. *SIGCOMM 08*, 2008.
- [67] Sachin Katti, Dina Katabi, Wenjun Hu, Hariharan Rahul, and Muriel Medard. The importance of being opportunistic: Practical network coding for wireless environments.
- [68] Grace R. Woo, Pouya Kheradpour, Dawei Shen, and Dina Katabi. Beyond the bits: Cooperative packet recovery using physical layer information. *MobiCom 07*, 2007.
- [69] Raymond W. Yeung, Shuo-Yen Robert Li, Ning Cai, and Zhen Zhang. *Network Coding Theory*. Now Publishers, 2006.
- [70] Jae Chung and Mark Claypool. NS by example. <http://nile.wpi.edu/NS/>.
- [71] Tcl online manual. <http://www.tcl.tk/man/>.
- [72] Miguel A. Erazo and Yi Qian. Sea-mac: A simple energy aware mac protocol for wireless sensor networks for environmental monitoring applications. *Wireless Pervasive Computing, 2007. ISWPC '07. 2nd International Symposium on*, 2007.
- [73] Yi Qian, Kejie Lu, Bo Rong, Hector Lugo, and David Tipper. An optimal key management scheme for wireless sensor networks. *MILCOM 07*, 2007.

BIOGRAPHICAL SKETCH

Hector M Lugo-Cordero was born in July 15 of 1983 at the city of San Juan, Puerto Rico. Hector is the son of Hector M Lugo-Santiago and Diana Cordero-Arias. He has a younger brother names Luis A. Lugo-Cordero and a younger sister named Liz Y. Lugo-Cordero. Hector studied the elementary and middle school at the Colegio San Rafael at Quebradillas and his high school at the Colegio San Antonio at Isabela.

In June of 2006 he received the tittle Bachelor in Science in Computer Engineering with Magna Cum Laude (high honors) from the University of Puerto Rico Mayagüez Campus. He then continued his studies on August 2006 at the University of Puerto Rico Mayagüez Campus in the area of Computer Engineering with a specialty in Computer Networks under the supervision of Dr. Kejie Lu. On April 2006 Hector passed the Fundamentals of Engineering exam and then on October 2006 he passed the Principles and Practice of Engineering Exam.

His research interests include software development, computer networks, wired and wireless communications, network management and security, embedded systems, digital signal processing, and control systems.