UNIVERSIDADE FEDERAL FLUMINENSE
CENTRO TECNOLÓGICO
MESTRADO EM ENGENHARIA DE TELECOMUNICAÇÕES

RICARDO CAMPANHA CARRANO

IMPROVING THE SCALABILITY AND RELAIBILITY OF THE XO MESH NETWORK

NITERÓI
2008

RICARDO CAMPANHA CARRANO

IMPROVING THE SCALABILITY AND RELAIBILITY OF THE XO MESH NETWORK

Dissertação apresentada ao Curso de Mestrado em Engenharia de Telecomunicações da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Sistemas de Telecomunicações

Orientador: Prof^{o} Luiz Cláudio Schara Magalhães, Phd

NITERÓI

2008

RICARDO CAMPANHA CARRANO

IMPROVING THE SCALABILITY AND RELAIBILITY OF THE XO MESH NETWORK

Dissertação apresentada ao Curso de Mestrado em Engenharia de Telecomunicações da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Mestre. Área de Concentração: Sistemas de Telecomunicações

Aprovada em Dezembro de 2008

BANCA EXAMINADORA

_____

Prof. Dr. LUIZ CLAUDIO SCHARA MAGALHÃES

_____

Prof. Dr. MICHAEL ANTHONY STANTON

_____

Prof. Dr. ANTÔNIO JORGE GOMES ABELÉM

This work is dedicated to all people that work for improving the life of others.

May all children have access to good education and through knowledge achieve peace and happiness.

ACKNOWLEDGEMENTS / AGRADECIMENTOS

First, I would like to thank my advisor **Luiz Claudio Schara Magalhães** for his guidance and for all the exciting opportunities I have enjoyed as his student.

I also thank the teachers, colleagues and staff at the Masters Program of the Telecommunications Engineering Department at **Universidade Federal Fluminense**, especially Professors **Célio Vinicius Neves de Albuquerque** and **Débora Cristina Muchaluat Saade** and all my friends at **Midiacom**.

I have the deepest gratitude for the continuous support and inspiration from my boss at **One Laptop per Child**, **Michalis Bletsas.** Special thanks to him and to everyone at **OLPC**.

Many, many thanks are due to **Javier Cardona**, **Luis Carlos Cobo** and **Brian Cavagnolo** and everyone at **CozyBit;** and **Ashish Shukla, Rajesh Bhagwat**, **Shailendra Govardhan** and **Ronak Chokshi** at **Marvell** for their generosity in promptly answering my copious questions.

Last, but not least, I thank my precious wife **Erika**, for her love and support; my father **Jorge** and my mother **Wanda** for giving me the tools to make the best of my life.

This work was partially funded by **CAPES** (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), **RNP** (Rede Nacional de Ensino e Pesquisa) and by **One Laptop Per Child**.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AODV	Ad hoc On Demand Distance Vector
AP	Access Point
BSS	Basic Service Set
BSSID	Basic Service Set Identifier
BT	Blinding Table
CBR	Constant Bit Rate
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CTS	Clear-to-Send
CW	Contention Window
DA	Destination Address
dB	Decibel
dBm	Decibels in reference to 1 milliwatt
DCF	Distributed Coordination Function
DIFS	DCF Inter Frame Spacing
DO flag	Destination Only
DSDV	Destination-Sequenced Distance Vector routing
DSR	Dynamic Source Routing
ERP	Extended Rate PHY
ESSID	Extended Service Set Identifier
ETT	Expected Transmission Time
ETX	Expected Transmission Count
FWT	Forwarding Table
HWMP	Hybrid Wireless Mesh Protocol
IEEE	Institute of Electrical and Electronics Engineers
ISM	Industrial, Scientific and Medical
LAN	Local Area Network
MAC	Media Access Control

MANET	Mobile Ad hoc Network
MAP	Mesh Access Point
MDA	Mesh Deterministic Access
MP	Mesh Point
MPP	Mesh Portal Point
MPR	Multi-point Relay
NIC	Network Interface Card
NWB	Network Wide Broadcast
OFDM	Orthogonal Frequency-Division Multiplexing
OLPC	One Laptop Per Child
OLSR	Optimized Link State Routing Protocol
OSPF	Open Shortest Path First
PCF	Point Coordination Function
PDM	Path Discovery Mechanism
PER	Packet Error Rate
PERR	Path Error
PHY	Physical Layer
PLI	Path Length Index
PREP	Path Reply
PREQ	Path Request
RA	Receiver Address
RANN	Root Announcement
RA-OLSR	Radio-Aware Optimized Link State Routing Protocol
RF flag	Reply and Forward
RFC	Request for Comments
RNP	Rede Nacional de Ensino e Pesquisa
RTS	Request to Send
RTT	Round Trip Time
RUCA	Rede de Um Computador por Aluno
SA	Source Addess
SIFS	Short Inter Frame Space
SNR	Signal to Noise Ratio
SoC	System on a Chip

SSID	Service Set Identifier
STA	Station
TA	Transmitter Address
TTL	Time-to-Live
UCA	Um Computador por Aluno
USB	Universal Serial Bus
WDS	Wireless Distribution System
WLAN	Wireless Local Area Network
WMAN	Wireless Metropolitan Area Network
WMN	Wireless Mesh Network
WPAN	Wireless Personal Area Network
ZRP	Zone Routing Protocol

RESUMO

XO-Mesh é uma implementação de rede ad-hoc de múltiplos saltos criada pela OLPC (One Laptop per Child [49]) para seu laptop educacional de cem dólares – O XO. Este dispositivo foi projetado não apenas para o uso infantil, mas também para os rigores das regiões economicamente menos favorecidas e pode, em muitas partes do globo, ser o único meio de comunicação eletrônico disponível. Para que sua nuvem mesh possa crescer em termos de participantes e também na confiabilidade de seus serviços, tornando-se uma plataforma de comunicação útil e livre da necessidade de infra-estrutura pré-existente, alguns ajustes aos parâmetros gerais da rede sem fio e aos parâmetros específicos da rede em malha dos XO são necessários. Realizar testes que resultem nestas recomendação é o objetivo desta tese.

Este trabalho provê documentação inédita para alguns aspectos da implementação do XO-Mesh e documenta suas diferenças para o IEEE802.11s. Alguns dos mais importantes desafios à escalabilidade e à confiabilidade do Mecanismo de Descoberta de Caminhos do XO também são estudados e otimizações são propostas, baseadas em dados experimentais.

Palavras-chave: Redes em malha; protocolos de roteamento ad-hoc; IEEE802.11s; XO-Mesh; Redes em malha no nível dois; Redes ad hoc movies, MANET; XO e One Laptop per Child.

ABSTRACT

The XO-Mesh is a layer two ad hoc multihop wireless network, developed by the One Laptop Per Child Association (OLPC) for its hundred-dollar educational laptop – the XO. This device is designed not only for child use, but also for economically challenged regions and, in many parts of the world, it may be the only electronic communication device available. In order to form a mesh that will grow in number of participants and in reliability of services, providing its users with a useful infrastructure-free communication platform, some adjustments to the general wireless subsystem parameters and to specific mesh parameters of the XO-Mesh are necessary. Performing tests and coming out with a series of recommendations in that respect is the objective of this thesis.

We provide the first documentation to many of the XO-Mesh implementation and documents its differences from the IEEE802.11s. Some of the most important challenges posed to scalability and reliability of the XO Mesh Path Discovery Mechanism are also studied and optimizations are proposed based on experimental data.

Keywords: Mesh networks; Ad-hoc routing protocols; IEEE802.11s; XO-Mesh; Layer 2 Mesh Network; Mobile ad hoc Networks, MANET; XO and One Laptop per Child.

1. INTRODUCTION

1.1. OVERVIEW

For more than one hundred years now, the use of the electromagnetic spectrum for communications has gone through wave after wave of technological innovation, with the introduction of services and devices that completely changed the way people live, work and interact. Communication technologies helped shape the last century and wireless communication was central to this story.

But far from becoming static, wireless technologies are still evolving vigorously. Recently, mobile telephony swept the globe and achieved ubiquity in any industrialized region of the world. But a new upsurge is now facing us, and it promises to make the much anticipated paradigm of truly mobile computing a reality.

Data exchange through small untethered devices is evolving fast, supported not only by the cellular infrastructure of mobile telephony, but also by the advent and popularization of new standards for wireless communication that operate in unlicensed spectrum, freeing the user of the ties of government agencies and the charges of telecommunications companies.

These standards gave birth to a new generation of inexpensive radio interfaces that can be added to almost any device. Laptops and PDAs already benefit from customer grade wireless chips and even cell phones have begun to embed them. An immediate outcome of this is that the number of nodes connected to the Internet will keep growing fast for years to come.

Of the myriad of portable devices that are in the market today, most are “Wifi” capable. Wifi is a trade mark for an specific type of Wireless Local Area Network (WLAN), which is increasingly popular in great part because they are easy and inexpensive to deploy when compared to traditional wired networks and, on top of that, for the mere fact that they need no wiring.

The heart of this new wireless phenomenon is a set of standards created by the Institute of Electrical and Electronics Engineers (IEEE) during the last decade. Though

not the only alternative, the IEEE standards managed to achieve quick and large popularity worldwide.

The IEEE 802 family of standards is dedicated to the construction of Local Area Networks (LANs) and Metropolitan Area Networks (MANs). Distinguished members of this group are the IEEE 802.3 (Ethernet) and the now almost forgotten 802.5 (Token Ring) but most of the emerging standards in this family deals with networking over the wireless medium.

The 802.15, of which Bluetooth is a part, is designed to interconnect personal devices over a short range Wireless Personal Area Network (WPAN). For the creation of the wireless equivalent of a LAN (i.e. a Wireless Local Area Network or WLAN), the IEEE proposed the 802.11 standard; while the 802.16 (a.k.a WiMax) addresses the problem of city-wide networks or WMANs (Wireless Metropolitan Area Networks).

These three standards have in common the fact that they are strongly based on some sort of infrastructure. In a WPAN – a master device concentrates all traffic. For a WLAN, the access point plays a central role, by relaying all traffic between participating nodes. And, finally, WiMax is also infrastructure bound – its central node is a powerful and resourceful base station.

Though still easy to deploy when compared to their wired counterpart, these technologies are not viable in scenarios where no infrastructure at all is available, the classical example being a disaster area where a natural catastrophe or terrorist attack completely destroyed any infrastructure.

But there is a much more common scenario where infrastructure-free networks are needed – the developing and economically challenged regions where no investments exist to build or maintain a working infrastructure. A non-infrastructure or ad-hoc network may be the powerful digital inclusion tool needed to reduce poverty by means of extending access to information and educational contents.

An ad-hoc network is a self-forming, self-configuring network that dispenses any dedicated infrastructure, even an access point. In such a network a node will be able to communicate with any other node within range and also with nodes out of immediate

radio range. To implement the latter, an ad-hoc network relies on the nodes to relay traffic for the benefit of other nodes. This category of networks is being called MANETs for Mobile Ad-hoc Networks.

These multihop capabilities are one important component lacking in 802.11 and there are recent efforts to incorporate it in the three aforementioned IEEE wireless technologies. In this work, however, we are most interested in the implementation of a "mesh"[1] network with 802.11 devices, a goal being pursued by the IEEE 802.11 Task Group "s", namely IEEE 802.11s [44, 50, 51], and also by the One Laptop per Child Association [49].

1.2. MOTIVATION AND OBJECTIVES

The XO mesh network may be the only communication device available in many of the OLPC's deployments, as this organization is mostly focused in underserved regions of the globe. This means that being able to form a mesh is perceived as one very important feature offered by the XO.

Initial tests and reports from the field indicate that a mesh cloud formed by XOs can not support certain applications in a reliable manner if the number of nodes is greater than 3 or 4, depending on the demands of the application.

Although the proper redesign of some of these applications may be recommended, in order to make them less chatty, it is always possible to optimize the underlying mechanisms of the mesh, such as the path discovery mechanism (PDM) and the management traffic and help alleviate the traffic pressure over the XO-Mesh.

---

[1] The term "mesh" has been chosen by OLPC's to name to its wireless multihop network. Although, there is not a widely accepted precise definition for what would consist a mesh network, the term is usually associated to WMNs (Wireless Mesh Networks), while the collaborative network formed by XOs would better described by the acronym MANET (Mobile Ad-hoc Network). Both expressions (WMN and MANET) will be explained in Chapter 3.

This thesis is dedicated to the second approach. More specifically we will describe, investigate and propose optimizations to the software implemented in the XO wireless subsystem, especially the reactive path discovery protocol it employs.

The scalability issue will not be eliminated, there will be a limit to the size of the XO Mesh for a certain traffic demand, what we want is to stretch this limit. In doing so, we will be also improving the reliability of the XO-Mesh network as a congested network tends to be not reliable.

1.3. WORK ORGANIZATION

In the remaining of this introductory chapter we will describe the projects that made this work possible, the OLPC project itself and the RUCA project, which was one of the first initiatives to test the network capabilities of the XO (the OLPC's "one-hundred-dollars" educational laptop).

In the second Chapter we will cover the main aspects of IEEE 802.11, since this is the base standard on which all of this work stands on. We then proceed in Chapter 3 to the general analysis of Mobile Ad-hoc Networks (MANETs) including their taxonomy, routing protocols and metrics followed by a more detailed description of the emerging standard IEEE 802.11s in Chapter 4 and the particularities of its implementation by One Laptop per Child in Chapter 5.

Our contributions are described in Chapter 6, where we present a set of tests performed on an XO testbed which culminates in the conclusions and recommendations found in Chapter 7. Two appendixes are included with complementary information, and cover the XO's specification (Appendix A) and Python code used in the test tools (Appendix B).

1.4. THE RUCA PROJECT

In Brazil, as in other developing regions of the globe, there is a clear obstacle to the social and economic progress of a significant part of the population – deficient or unsatisfactory education. New communication technologies, like the Internet, and easy access to powerful computing devices are now commonalities in the industrialized

regions, and may either increase the gap between the rich and the poor (people or countries) or provide an opportunity to actually reduce the discrepancy.

Because of this dichotomy, much attention is being dedicated to the idea of digital inclusion. The motto is that either people leapfrog into this new digital-connected-computerized reality or they will be doomed to a life of lesser opportunities. Many initiatives worldwide are trying to address this threat by focusing mainly on the education of children. The UCA (Um Computador por Aluno) project is one of these initiatives. Being undertaken by the Brazilian Ministry of Education its final goal is to provide a laptop computer for every student in the public school system[2].

One chapter of the UCA project focused on the connectivity challenges of the initiative – the RUCA project (where R stands for "rede", network). The first RUCA project was administered by RNP (Rede Nacional de Ensino e Pesquisa), and encompassed six Universities in Brazil (UFF, USP, UNB, UFAM, UFPB, UFRGS).

The RUCA team was given the mission to determine the network capabilities of the laptop device under test[3] and to propose a connectivity model for the use of the laptop in the school and at home. The connectivity model can be summarized as follows:

(1) The laptop must work in the dense environment of a school, where hundreds of nodes will be spread over a building and dozens of them will be packed inside a relatively small indoor environment – the classroom. And all of this must work with inexpensive infra-structure (customer grade access points);

(2) That children will take their XO home and once distributed within a community they will be capable of forming a mesh network with its close neighbors. This may also extend Internet connectivity to this community since some of the laptops may

---

[2] It is actually much more than providing the laptop. The device is part of a bigger plan - a vehicle for a pedagogical process and for educational content

[3] When the RUCA project was carried out the only equipment under test was the OLPC's XO since it was the first one presented to the authorities and the only one with some of the required network capabilities, like the mesh network.

be accessing an antenna placed on the top of the school. Those connected nodes will act as Mesh Portal Points and further extend Internet connectivity to other nodes in the mesh cloud, as we will further explain (see Figure 1).

The tests were designed and coordinated by the Midiacom Laboratory at Universidade Federal Fluminense (UFF) and performed by all the Universities involved. It consisted of a series of isolated tests that would first determine if the basic necessary capabilities were present, checking for instance if the wireless network and the mesh were operational and if the devices performed satisfactory in terms of throughput or radio coverage and then try to emulate a set of scenarios, like the dense classroom scenario or a sparse community network.

The RUCA project was concluded by November 2007 and was the starting point of the present work. Its main results are summarized in Appendix A and will be used throughout this text. The present thesis describes a series of adjustments that will improve network performance and connectivity in the dense scenario and it can be considered a development of the RUCA project.

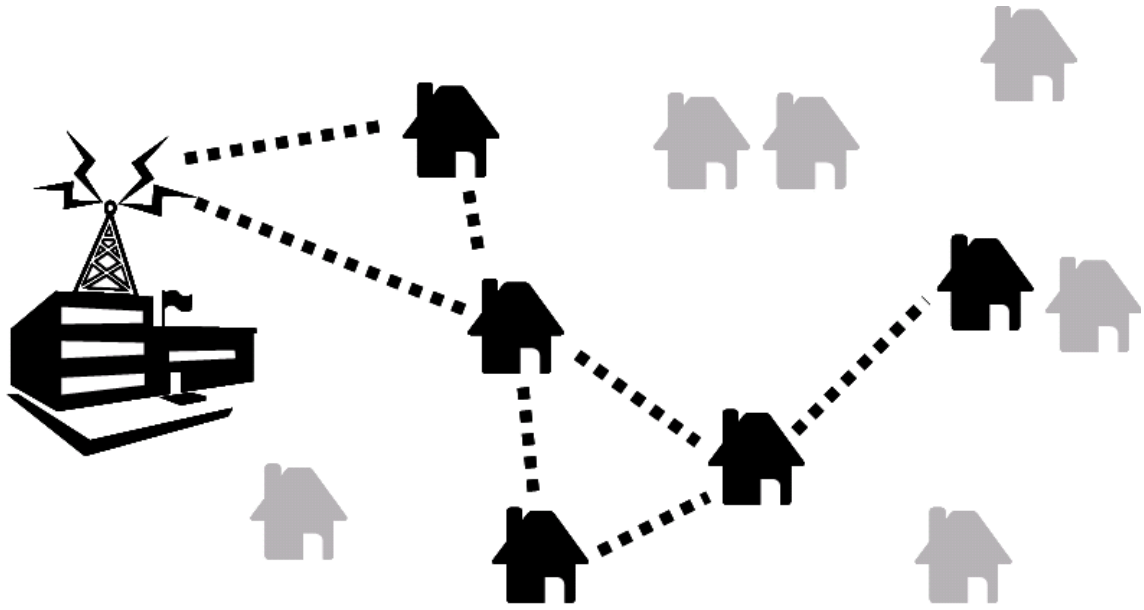

Figure 1 – Laptops are present in some of the houses in a give neighborhood and are connected to the school network directly or with the aid of the neighbors.

1.5. OLPC AND THE XO

One Laptop Per Child [49] is a non-profit organization based in the United States (Cambridge, MA), which conceived a low cost laptop - the XO, formerly known as the one hundred dollars laptop - especially designed for child use and highly adapted to operate in underserved regions of the world.

Figure 2 – OLPC's XO – the US$100 laptop.

The XO (Figure 2) was the first device to ever implement a version of the IEEE802.11s draft [50]. Other features that make it an attractive device for the challenged regions are its ruggedness (it can survive, for instance, a fall from 1.5 meters or accidental water spills over his keyboard), its augmented autonomy (an XO can

operate for more than 3.5 hours without the need of recharging the battery) and its dual, black-and-white and color, screen (that can be read under the sunlight).

XO laptops are being mass produced at Quanta, in Shanghai, China since November 2007 and are currently being deployed in countries like Uruguay, Peru and Mongolia. The dual layer innovative display is legible under the sun and the XO is also equipped with camera, microphone, SD slot, a capacitive touch pad and 3 USB slots. The complete specification of the XO can be found in Appendix A.

Of particular interest for this work is the XO's radio subsystem consisting of a Marvel 88W8388, which embeds an ARM9 processor, volatile and non-volatile memory and a Marvell 88W8015 radio transceiver.

The laptop has two *bunny ear* rotating antennae. During the initial tests of the RUCA project, prototypes were used, and there was no guarantee that two any devices would have equal reception or transmission capabilities, since many internal antennae designs were being tested, as we can see in Figure 3.

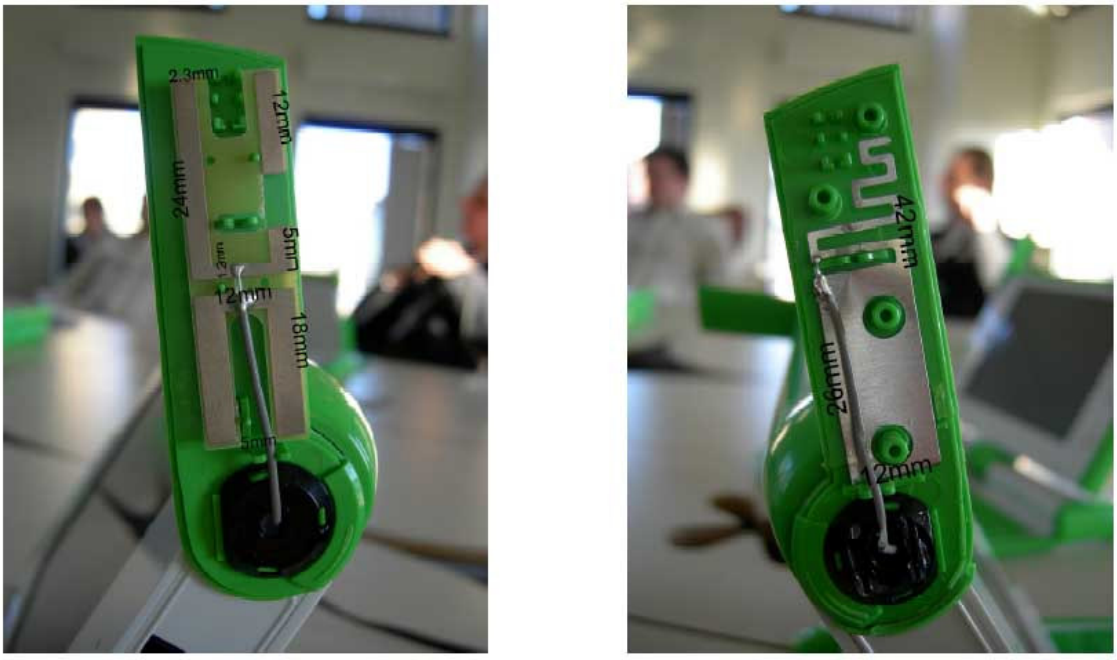

Figure 3 – two totally different designs for the antenna may produce differences in test results

The radio subsystem is connected to the main CPU (AMD Geode GX-500, for the prototypes and AMD Geode LX-700, for the mass produced units) via a Universal Serial Bus (USB) as depicted in Figure 4.

The USB controller chip bandwidth limits the throughput of a host application to about 14Mbps but the segregation of the radio subsystem has some advantages. The Marvell 8388 is a SoC (System on a Chip) capable of implementing not only the IEEE 802.11 MAC layer and IEEE802.11 "b" and "g" physical layers, but also of hosting OLPC's implementation of the IEEE802.11s [44, 50, 51] (all of these to be described in more detail in subsequent chapters).

Because of this partition an XO can forward frames, independent of the main CPU and consuming no more than 0.5 Watts. The host CPU will only be activated for local IP traffic. Figure 4 provides basic block schematics of the XO's radio subsystem.

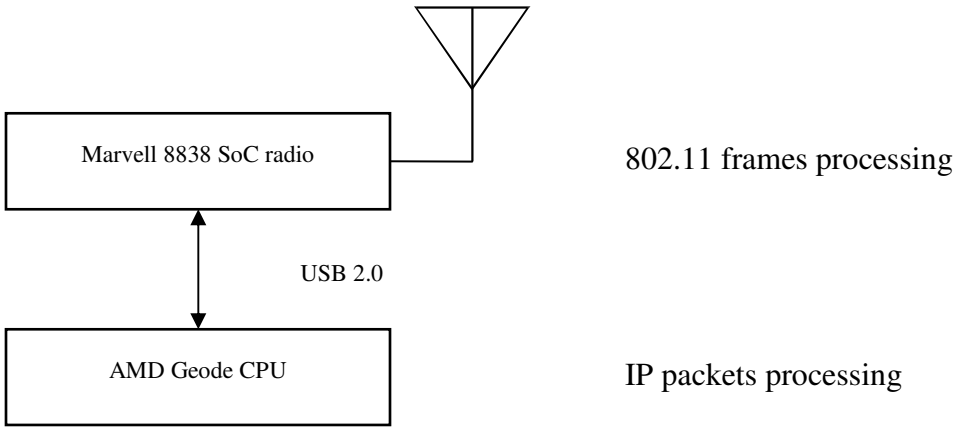

Figure 4 – the radio subsystem can handle 802.11 frame processing by itself and host OLPC's implementation of IEEE802.11s. The main CPU will handle IP packets.

The XO operating system is a customized version of the Fedora Core Linux. On top of it the user experience is governed by an especially designed graphic user interface named Sugar and a group of activities (applications in OLPC jargon) that were designed

with constructionist [4] goals in mind and optimized for efficient use of cpu cycles and low memory fingerprint.

The Linux kernel shipped with the XOs is also extremely customized in order to support many of the features introduced by OLPC, like a more aggressive suspend and resume for power saving, its security model (Bitfrost) which includes anti-theft techniques, or the new file system (olpcfs). An adapted version of the Libertas driver (Libertas is a codename for a Marvell family of chips) was also developed to operate in conjunction with a Libertas firmware, which is uploaded to the radio subsystem at every initialization.

[4] Constructionism is a learning theory inspired by Jean Piaget's constructivist theories. It holds that learning can happen spontaneously when the subject actively engages to solve a certain practical problem. A construcionist device or technique, like the XO and its software, will emphasize "learning to learn".

2. IEEE802.11 WIRELESS NETWORKS

2.1. OVERVIEW

Of all the wireless network technologies available today, none was so successful in extending local area networks to wireless nodes than Wifi, which is derived from the IEEE 802.11 standard. This standard describes the physical layer and the link layer for wireless communication in the 2.4GHz range, including the medium access mechanism, and since its release in 1997 it has been amended many times both to introduce new capabilities or to include different physical layers that could provide more bandwidth. Table 1 provides a list of the most popular amendments past and present and, as we can see, there are still many advancements being proposed.

Table 1 – Main 802.11 amendments

Standard/amendment	Description
802.11-1997	Original standard (from 1997) which described the MAC layer and the FHSS and DSSS modulation techniques (1 and 2Mbps)
802.11a	Approved in 1999 - introduces a new physical layer - OFDM. Incorporated to 802.11-2007.
802.11b	Approved in 1999 - introduces a new physical layer – HR/DSS. Incorporated to 802.11-2007.
802.11g	Approved in 2003 - introduces a new physical layer – ERP. Incorporated to 802.11-2007
802.11d	Approved in 2001 - introduces support compatibility with international regulations.
802.11e	Approved in 2005 - introduces quality of service (QoS)
802.11h	Approved in 2004 – adapts 802.11a to Europen Union regulations.
802.11i	Approved in 2004 – introduces new security mechanisms
802.11j	Approved in 2004 – adapts 802.11 to Japanese regulations
802.11-2007	Incorporates amendments “a”, “b”, “d”, “e”, “g”, “h”, “i” e “j” to IEEE802.11-1997.
802.11n (*draft*)	Task Group “n” (TGn) proposes techniques to achieve bands superior to 100Mbps (MIMO or Multiple Input, Multiple Output is possibly the most popular of these techniques)
802.11r (*draft*)	Task Group “r” (TGr) works on handoff mechanisms, particularly for fast moving devices (vehicles, for instance).
802.11s (*draft*)	Task Group “s” (TGs) is proposing a mesh network for 802.11 devices.

One of the main goals of the IEEE in creating the 802.11 standards was to provide a way of interconnecting wireless networks to Ethernet (802.3) networks. The wireless network is seen as an extension to the wired network and this implies the need for a series of mechanisms to provide interconnectivity. Recently, after the spread of wireless networks, the standard was augmented to improve bandwidth (IEEE802.11a, b and g, or

the recent draft n); make them more secure (802.11i); improve support for mobility (draft r); or introduce quality of service mechanisms (802.11e). Not to mention the mesh capabilities at the very core of this work.

2.2. THE PHYSICAL LAYER

802.11 networks use two portions of the spectrum that are unlicensed in most parts of the world. These are called ISM (*Industrial, Scientific and Medical*) bands and can be freely used by any device as long as the maximum transmitted power does not exceed a certain limit.

The first of the two is the S-ISM and extends from 2.4 to 2.483GHz (with minor variations around the world). This is the spectrum portion used by the physical layers proposed by both 802.11b and 802.11g, but Bluetooth and zigbee devices also operate in the S-ISM band.

The number of channels in this band varies between 11 and 14 around the world (11 in Brazil and in the United States, 13 in Europe, for instance). In any case, the central frequency for channel one is 2.412 GHz and each adjacent channels are separated by 5 MHz. Since 802.11 b/g spreads it signal over a 22MHz band, this means that adjacent channels interfere significantly and hence the common practice of allocating the three non-interfering channels, 1, 6 and 11 – separated by 25MHz – in deployments.

Because of its unlicensed nature and the popularity of the devices operating in this part of the spectrum, the ISM band is already crowded in most of the large urban areas around the world. A second band that is reserved to unlicensed use and employed in the IEEE802.11a physical layer lies around or after the 5GHz frequency and is not as uniformly allocated around the world as the 2.4 ISM band. But since “a” devices are less popular than “b” or “g” devices, the 5GHz band is still less subject to interference than the 2.4GHz spectrum.

Signals transmitted at the GHz portions of the spectrum (microwaves) do not bend around obstacles or penetrate walls so well (as FM radio signals, for instance) and given the legally allowed transmission powers, this limits a link between two nodes to some tens of meters, or a few hundred meters under near ideal conditions.

The original version of the IEEE 802.11 standard included two different physical layers. The first, Frequency-Hopping Spread Spectrum (FHSS) was the first spread spectrum technique to be widely employed and although still in use in some devices (like Bluetooth gadgets) in terms of IEEE802.11 it is now obsolete.

The second PHY was the Direct Sequence Spread-Spectrum, or DSSS. Both techniques provided codification rates of 1 or 2 Mbps, but in 1999 IEEE802.11b used DSSS to reach 5.5 Mbps and one variation of it - HR/DSSS (High Rate Direct Spread Spectrum) – made 11Mbps rates possible.

Curiously IEEE802.11a, which could reach data rates as high as 54Mbps was introduced in the same year (1999) and remained the fastest of the IEEE standards until 2003, when 802.11g was released. Both new physical layers (IEEE802.11a and IEEE802.11g) implemented schemes based on Orthogonal Frequency-Division Multiplexing (OFDM) as the modulation method. But although "a" devices used another portion of the spectrum – as we have seen, the 5GHz band – "g" devices shared the same spectrum as the previous "b" and original IEEE802.11 devices.

Table 2 - ERP (*Extended-Rate* PHY) modulation schemes

ERP subcategories	**Description**
ERP-DSSS e ERP-CCK	Two modulation techniques backward compatible to the original standard and also to the "b" amendment. Used to operate in at 1, 2, 5.5 and11 Mbps.
ERP-OFDM	Same modulation technique used in the amendment "a" but operating at the 2.4GHz band. Used to operate at 6, 9, 12, 18, 24, 36, 48 and 54Mbps.
ERP-PBCC	Optional method not implemented in most of the recently shipped devices. It provides the data rates of 22 and 33 Mbps.
DSSS-OFDM	Another optional method (also not available in most devices). It is a hybrid scheme that uses DSSS for transmitting the header and OFDM for the frame body. (allowing 802.11b stations to decode the header)

Backward compatibility may explain why IEEE802.11g chips (that hit the market in 2005) became more popular than IEEE802.11a chips, even though the latter were available many years before. Today IEEE802.11b/g or even IEEE802.11a/b/g devices are

the rule. And exactly because the "g" amendment is backward compatible with previous standards (but not with "a", of course) its physical layer is called ERP, or Extended-Rate PHY, which is an aggregation of many of the previous modulation techniques and supported data rates. Table 2 provides a summary of the modulation schemes that compose the ERP and its respective codification rates.

Figure 5 lower part displays all the physical layers (PHYs) currently available in the IEEE802.11 standard.

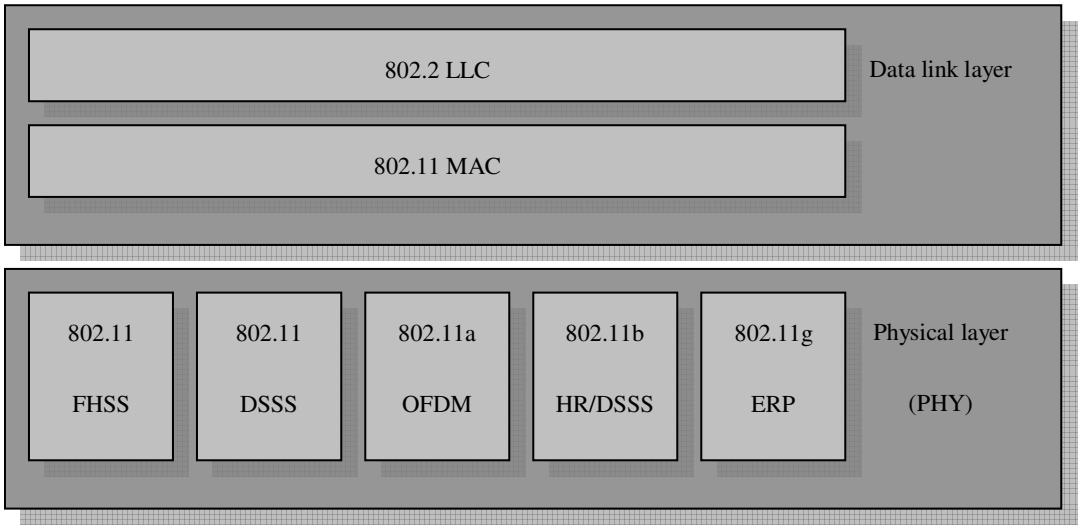

Figure 5 – Physical and MAC layers in the IEEE 802.11 standard

2.3. MEDIUM ACCESS CONTROL

Despite the common goals, the medium access control mechanism described in 802.11 differs from that of 802.3 (Ethernet) because of the specific characteristics of the wireless medium, that presents challenges and constraints unknown to the wired networks.

The mandatory medium access mechanism for IEEE802.11 stations – the Distributed Coordination Function (DCF) is based on CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance) which contrasts with the collision detection mechanism – CSMA-CD (Carrier Sense Multiple Access with Collision Detection) – implemented in Ethernet LANs.

A collision *avoidance* method is preferred since single interfaced radio systems cannot listen to the medium while transmitting and are thus unable to detect a collision. Also, the costs of collisions in wireless networks are higher than in a wired network, where transmission rates are usually higher. And, finally, the loss of a frame due to a corrupted transmission is more commonplace in wireless transmissions than in wired.

Inferring medium availability in order to avoid collisions is the main goal of the IEEE802.11 MAC, but this cannot be achieved solely by listening to the medium. The mere fact that the medium is free at a given moment does not mean that it is safe to transmit, i.e. that no collision will result. One example that illustrates this is the scenario known as hidden node.

Figure 6 illustrates the hidden node problem. Station "A" wants to transmit to "C" but it is completely unaware of a transmission going on from "B" to "C", since "B" is too far away for "A" to detect its transmission. If "A" senses the medium free and decides to transmit, this will cause a collision that will render both frames useless (A's and B's frames).

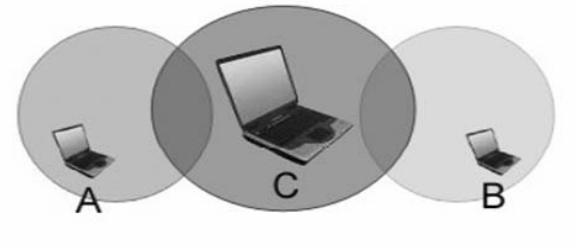

Figure 6 – the hidden node problem

To address the hidden node problem, IEEE802.11 implements the so called RTS/CTS scheme. When a node wants to transmit, it will first try to reserve the medium by broadcasting a Request to Send (RTS) frame. The RTS advertises a duration field – an estimation of the time necessary for the subsequent transmission to succeed, including the time taken for the recipient to confirm its reception by means of an ACK frame.

Upon the receipt of an RTS, the intended destination responds with a Clear to Send (CTS) frame, which also includes the estimated time that the medium will be busy. This way, even nodes that could not listen to the original RTS (potential hidden nodes) will receive the medium reservation information.

The use of RTS/CTS is optional and clearly only pays off for frames bigger than the RTS frame itself (20 bytes), otherwise the cost of losing the frame will not justify the additional overhead of the medium reservation. Hence, 802.11 stations will have a configurable RTS threshold that will set the minimum frame size to trigger the RTS/CTS scheme.

Figure 7 shows an example of a transmission with RTS/CTS. Neighbor stations will update their NAVs (Network Allocation Vector) with the times advertised in the RTS and CTS frames and will refrain from transmitting anything during this period. This mechanism that prevents a node to transmit even if the medium seems to be free is usually referred to as *Virtual Carrier Sensing*. The standard describes different silence intervals between the frames (Interframe Spaces), as DIFS (DCF Interframe Space) and SIFS (Short Interframe Space). This different interval will be used according to the status of the transmission and type of frame to be transmitted [2].

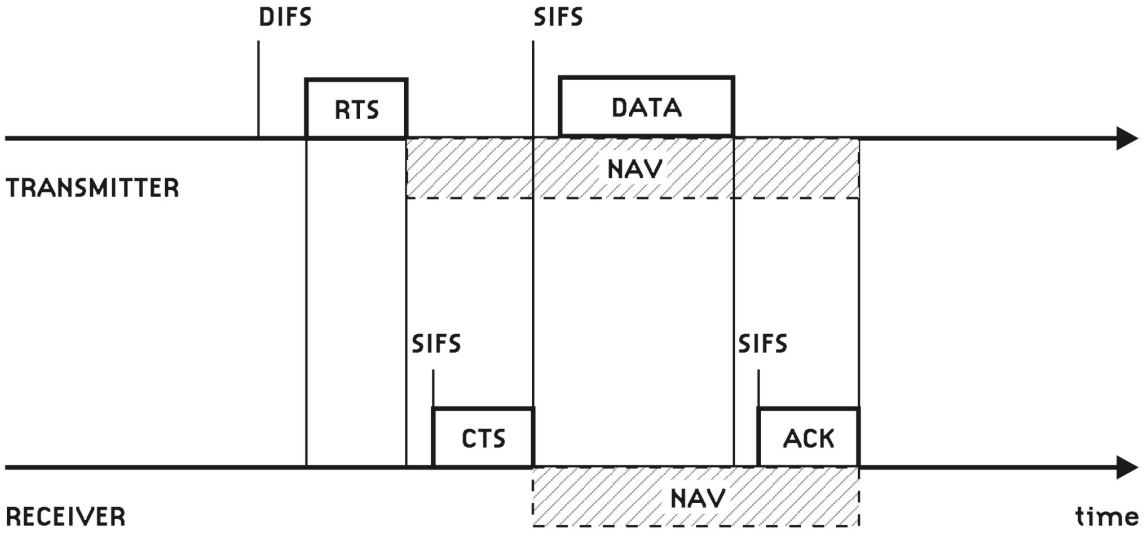

Figure 7 – RTS/CTS mechanism timeline

RTS/CTS frames need to be transmitted at lower rates in order to be successfully decoded by all the 802.11 nodes in the vicinity – either because they are distant or simply because they do not implement faster modulation techniques. This means that even being small these control frames will take a relatively long time to be transmitted[5].

Even with the RTS/CTS mechanism and even with Carrier Sensing collisions may happen. To understand that let us see what happens when a station wants to transmit. According to the DCF protocol, the station will check the medium and proceed according to the following protocol:

(1) If the medium is free, the station will wait a predefined time, the DIFS, and check the medium again. If it is still free the station will transmit.

(2) If the medium is sensed busy, the station will start a *Backoff* procedure, choosing a random (integer) number of time slots drawn from a uniform distribution over the interval [0, CW]. This random number will be the station's *Backoff Timer*. It is not difficult to see that the probability of collision is higher when the medium becomes idle just after a busy period. That is why every station contending for the medium will wait for a time equal to DIFS and then start the backoff period.

During the backoff period the station will check the medium at the start of each time slot. If the medium is sensed free the station will decrement its Backoff Timer. If, on the other hand, the medium is busy, the station will freeze the Backoff Timer until the medium is sensed free. When the Backoff Timer reaches zero, the station is allowed to transmit. If the frame transmitted is destined to a unicast address, the station will wait for an ack. If the ack is never received the transmission is considered failed and the frame will have to be retransmitted. In this case, the CW value will be doubled until it reaches a limit value, the CWmax.

Contention Window (CW) values are configured in each station to vary within a minimum (CWmin) and a maximum (CWmax) value. Every time a frame is successfully

---

[5] Furthermore, during the RUCA project tests indicated that the RTS/CTS frame seems particularly unfeasible for multihop wireless networks as can be seen in [1].

transmitted, CW is reset to its minimum value (CWmin) and every time a transmission fails, CW is doubled unless it is already equal to CWmax. The practical result is that backoff intervals respond to frame loss treating them as collisions – a bigger number of time slots to drawn from will reduce the probability of two stations to choose the same slot.

Finally, every time a transmission fails the station will increment the *Retry Counter*. If the Retry counter reaches a certain limit, the frame will be discarded. Actually there are two Retry Counters, one for small frames and other for larger frames.

Typical values for the parameters discussed in this section for 802.11g stations are summarized at Table 3. Inter frame space times (SIFS and DIFS) are mandatory. The other values will be set at vendor's discretion.

Table 3 – Typical values for CW and Retry limits and standardized values for IFSs in 802.11g

Parameter	**Value**
SIFS time	10μs
DIFS time	28μs
CWmin	31
CWmax	1023
Short retry limit	4
Long retry limit	7

2.4. 802.11 FRAME FORMAT

Figure 8 shows the format of an IEEE802.11 frame. One of its noteworthy aspects is the presence of four MAC addresses. As mentioned, wireless networks were originally projected as extensions to its Ethernet wired counterparts. In contrast with the Ethernet frame, which brings only two MAC addresses, the 802.11 frame will possibly be relayed by intermediary nodes – an access point, for instance – before it reaches its final destination, and thus the need for more addresses.

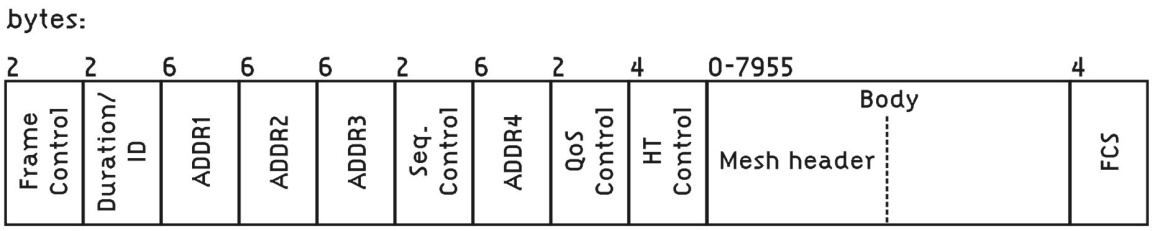

Figure 8 – IEEE802.11 frame format

The four MAC addresses are numbered instead of named because their meaning is not always the same. To illustrate this we use an example, depicted in Figure 9, where shows two stations communicate through an access point and station A sends a frame destined to B.

If we examine the addresses in each frame we will observe that:

For frame 1 (from A to AP):

- ADDR1 will be the AP's MAC address;
- ADDR2 will be A's MAC address;
- ADDR3 will be B's MAC address.

And for frame 2 (from AP to B):

- ADDR1 will be the B's MAC address;
- ADDR2 will be the AP MAC address;
- ADDR3 will be A's MAC address.

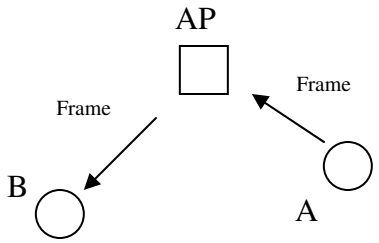

Figure 9 – A sends a frame to B in infra mode

In this example, ADDR1 can be thought as a recipient address for both the frames 1 and 2. Likewise, ADDR2 would be the transmitter address for frames 1 and 2. But ADDR3 would render different meanings. For frame 1, it indicates the destination address of the frame (B) while for frame 2 it informs the source address of the frame (A).

A MAC address in an 802.11 frame may have the following roles:

- Destination address (the final recipient of a frame);
- Source address (the address of the node that originated the frame);
- Recipient address (the station that processes the frame);
- Transmitter address (the station that sent the frame out);
- Basic Service Set ID (BSSID). Since many wireless networks may coexist, the BSSID provides identification for the set of nodes that participate in the same network.

Conventionally, an address with zero in the first bit will be a unicast address, while if it starts with one; the address will refer to a multicast (group) address. Moreover, an address consisting of 1's only will be the broadcast address.

The Frame control field contains many subfields that indicate, for instance, the protocol version (currently set to zero in all frames), the use of fragmentation at layer two, whether the frame has been retransmitted, among others. The frame type and subtype is also part of the Frame Control field and indicates if the frame is a *Management*

frame, that implements network operations (like Beacons or Association Requests); a *Data frame*, the very reason of the wireless network; or a *Control frame* that helps providing reliability to the data frame transmissions (ACK, RTS, CTS).

The duration field brings the estimated time that nodes should add to their Network Allocation Vectors in order to implement the Virtual Carrier Sensing, as explained previously. Sequence control is used to allow reassembling of fragments, if fragmentation is used, and also to help detect duplicated frames. A checksum trails every frame and provides integrity checking based on Cyclic Redundancy Check (CRC).

The frame body in Figure 8 is already updated to its new size introduced by 802.11n (it used to be up to 2,312 bytes). As we will see later, the new mesh header will be prepended to the frame body exactly like the subheaders QoS, for Quality of Service and HT for High Throughput networks (802.11n).

2.5. TYPES OF 802.11 NETWORKS

The IEEE 802.11 standard describes two distinct types or networks, or modes, depending on whether there is or there is not the participation of a specialized node called an Access Point. The first, and by far most common, is called infrastructured network, in reference to the presence of an access point which will mediate all communication between the nodes which are *associated* with it (a process to be explained soon). Normally, the access point is also connected to a wired network to which it will extend access to the wireless nodes.

Figure 10 shows a wireless local network (WLAN) where many access points (AP) are interconnected via a wired distribution system (DS). A group of nodes (stations – STA) connected to a same access point defines a BSS (Basic Service Set), while the union of all interconnected access points, bridged due to the presence of the distribution system, is called an ESS (Extended Service Set).

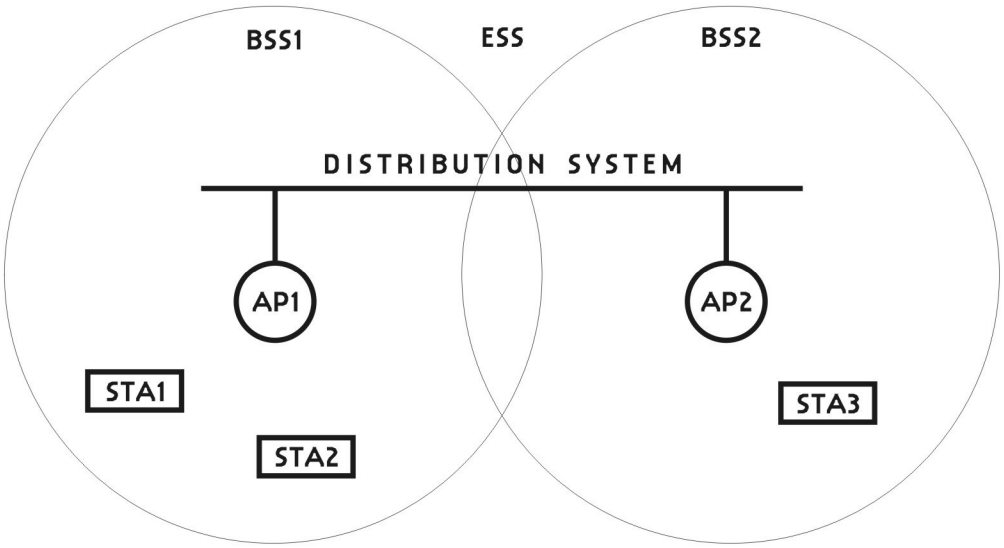

Figure 10 – An Extended Service Set formed by two Basic Service Sets and a Distribution System

In order to join an infrastructured network a wireless node first has to find it. This process is called *scanning* and can be achieved *passively*, by listening to the medium for special announcement frames that an access point frequently broadcasts, called *beacons*; or *actively* in which case the node will send out *Probe Request* frames and wait for *Probe Responses* from available access points.

The actual selection of a BSS to join can be automatic, for instance based on the signal strength, or user selected. It is also possible to search for a specific AP to attach, based on a BSSID. Either way, the internal process of selecting an access point is called *joining*, and it should happen before the *association* process itself.

The association starts with the exchange of authentication frames in a process that actually does not provide any real authentication of the nodes involved (the station or the access point). True authentication, if it is to be employed, will take place after the association process. The association process will finish with the station sending an *Association Request*, to be responded by the AP with an *Association Response*.

Once associated, the station will be able to exchange frames with the access point, with other nodes connected to the same BSS, and to other nodes outside the BSS if the access point happens to be connected to a wired network or if it is part of an Extended Service Set (ESS).

The second class of IEEE802.11 (IBSS – Independent Basic Service Set) networks consists entirely of stations (no access points) that connect to each other in a peer-to-peer on demand fashion, in what is called *ad-hoc mode* (Figure 11). In ad-hoc mode there is no provision for multihop paths and nodes are only capable of communicating to peers to which an ad-hoc connection has been established.

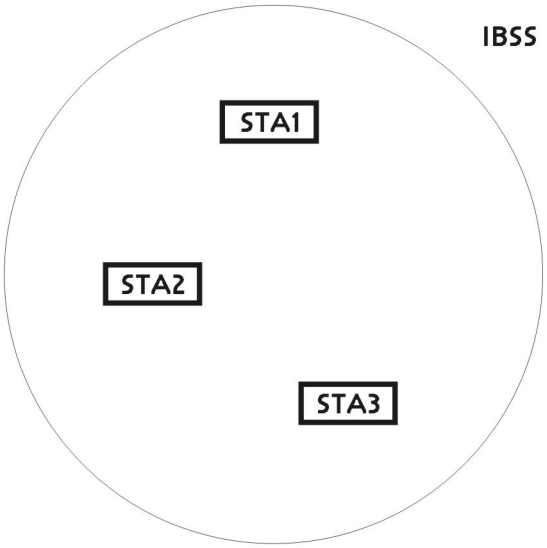

Figure 11 – An Independent Basic Service Set

Based on the non-infrastructure approach of the ad-hoc mode, a diversity of mechanisms has been proposed to add routing capabilities to the nodes and provide the means for multihop communication. The first implementations employed traditional network layer routing protocols, as depicted in Figure 12 – a Mesh Network.

After a decade of research in ad-hoc networks, the amendment "s" is the first to propose a multihop mechanism to be implemented at the link layer. The next chapter will briefly describe some of the most important implementations of multihop ad-hoc networks in layer three. The IEEE's proposal of a layer two implementation will be studied in more detail in Chapter 4.

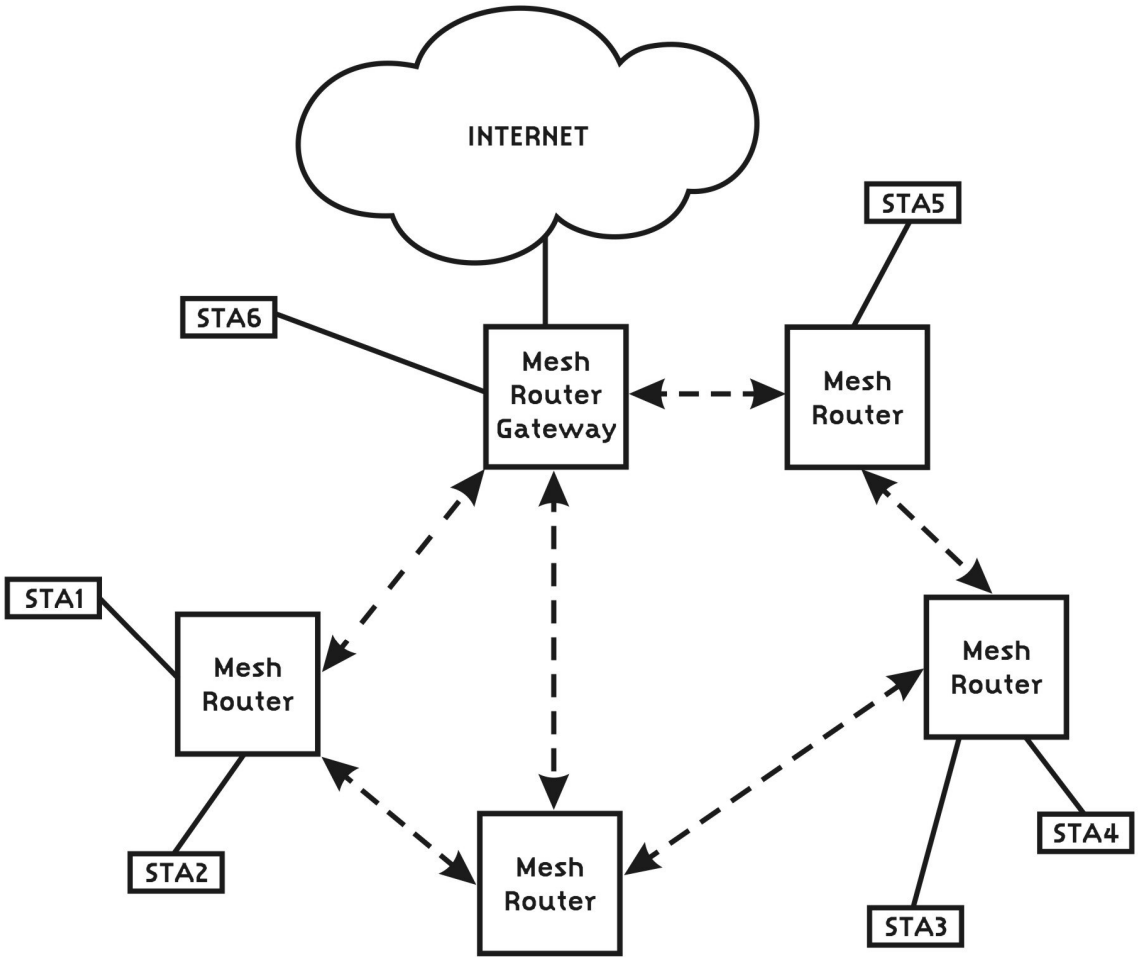

Figure 12 – A Mesh Network implemented at Layer 3. Mesh Routers are typically wireless routers with customized software that will implement not only the Access Point capabilities - to provide connectivity to the Stations (STAs) - but also mechanisms to create a self-forming, self-healing "mesh" topology (dashed arrows) with similarly configured devices, with the final goal (usually) of forwarding STA's traffic to the Internet.

3. MANETS (MOBILE AD-HOC NETWORKS)

3.1. INTRODUCTION

Multihop wireless networks fall into many categories. Probably the most widespread of these categories is the Wireless Mesh Network (WMN). A WMN (Figure 13) is basically a collection of fixed nodes, most of the times consisting of regular access points running adapted software. Its main goal is to provide an inexpensive and easily deployable wireless backhaul that will connect distant LANs or WLANs.

Operational WMNs can be found around the world and are mostly based on traditional network layer routing. Some examples are the FunkFeuer Net in Austria [3], MIT's RoofNet [33,34], VMesh, in Greece [40], UCSB's MeshNet [37,39] CUWin in Urbana [38], Microsoft Mesh [35.36] among others [41]. In Brazil, the Universidade Federal Fluminense has deployed the ReMesh Network in the city of Niterói [4].

A WMN is not necessarily an ad-hoc network, in the sense that it can benefit from previous planning of the position of the nodes (the case of UFF's Remesh). Nonetheless, nothing prevents it from growing organically like the FunkFeuer Network, in Austria, or the Meraki Public Network in San Francisco [5].

Sensor networks are another increasingly important class of multihop wireless network. Their main goal is the consolidation of information collected from distributed nodes – the sensors – spread over a given area. The sensor might be mobile, like the collars attached to coyotes in UCSC's CARNIVORE Project [6] or to zebras in Princeton's ZebraNet [7], projects designed to study wildlife; or fixed, like the ones installed in tree tops to collect environmental data, as temperature or the incidence of light, just to cite a few among many applications of sensor networks.

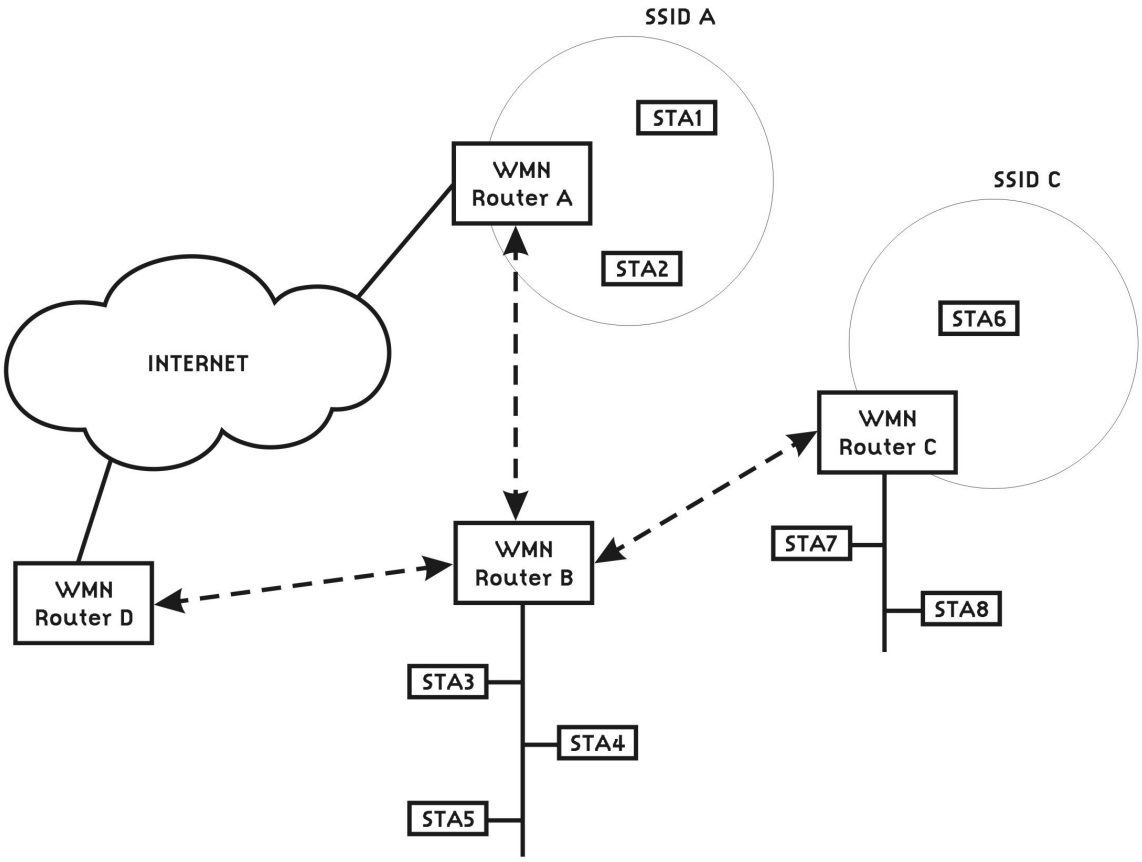

Figure 13 – In a WMN the routers provide a wireless backhaul that interconnects wired and wireless stations. WMN routers can serve wired Stations (B and C), wireless Stations (A and C), both (C), or serve no Station directly (D). The latter will contribute by forwarding traffic coming from other WMN routers. Some WMN routers can be connected to a external network (A and D) like the Internet while others may be only "internal" (C and B).

Independent of the application or the scale, one important characteristic of a sensor network is that the information travels to central nodes, where the researcher or a monitoring process analyze the data and infer its conclusions.

A third category of multihop wireless networks, and the one that we are most interested in, is the Mobile Ad-hoc Mobile Network, or MANET. These networks are designed to provide connectivity to mobile computing devices without the aid of an infrastructure. In this text, when we refer to a mesh, or to a mesh cloud, we are actually referencing MANETs.

Differently from a WMN, a MANET is a self-configuring network where there are no fixed routers. In a MANET, routers are free to move (it is an ad-hoc network[6]) and the topology of the network can change dramatically and quickly. Traffic routing functions will be carried on by some or all of the participating nodes. Moreover, differently from a sensor network, there may be no clear concentration of traffic to a given node. Though some concentration may occur if one node offers an attractive service to the mesh cloud – like gateway provisions to the Internet – any two nodes might want to communicate.

Be it ad-hoc or carefully planned, or somewhere in between these two paradigms, wireless multihop networks share a common challenge: the development of routing protocols able to cope with the specific challenges posed by wireless networks: node/router mobility, fast changing characteristics of the radio environment and medium access contention. After some decades of research in routing algorithms and routing metrics, there is a natural tendency for these new routing protocols to be based, in varying degrees, on preexisting routing protocols.

The traditional approach has been the implementation of routing protocols at the network level, which brings the obvious advantage of being link-layer independent. After all, internetworking has been the realm and main goal of the routing protocols. But when it comes to wireless networks, the alternative of implementing multihop capabilities in layer two is increasingly common.

---

[6] Ad hoc networks are wireless multihop networks where there are no fixed or dedicated routers (every node carry out the function of forwarding frames). In that sense, a WMN would not be truly ad-hoc, while a MANET would be. This is why the expression “wireless multihop network” has been used throughout this text, since it applies to WMNs, MANETs and Sensor Networks.

3.2. ROUTING PROTOCOLS

One of the most common ways of classifying routing protocols[7] for multihop wireless networks is based on the way a route discovery is triggered. As we will see in the following subsections, there are two opposed approaches, called *proactive* and *reactive*, and some attempts to combine both, in *hybrid* mechanisms.

3.3. PROACTIVE PROTOCOLS

The collection of route information can be carried out in a scheduled manner, independently of the needs of the nodes – in the so called pro-active approach. In a proactive protocol, routers are constantly exchanging information. Based on this information, forwarding tables are immediately calculated so that, whenever a node has traffic to send out, the routing information will be readily available.

The obvious drawback of this approach is the overhead traffic posed to the network. Routing information is to be exchanged whether or not this is necessary. Because wireless networks are dynamic in nature (even if the nodes are not mobile, but even more if they are), routing tables may age rapidly and there is the need of constant update messages, which means the reduction of the already relatively scarce bandwidth available for user applications.

Examples of proactive protocols specific to wireless networks are the Optimized Link State Routing (OLSR) [8] and the Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) [9]. The OLSR protocol is a popular routing protocol with one open source implementation [10] and at least two commercial implementations [11, 12] and is described in an Experimental RFC [8]. It is, as the name implies, a *Link State* protocol, a classic category in which the OSPF (Open Shortest Path First) [13] widely employed in the wired domain, is probably the best known example.

---

[7] We will use the term “route” and its derivative expressions (“router”, “routing protocols”, etc) when we refer to them irrespective of the layer in which they are implemented. Once we begin to study layer 2 protocols, we will favor the terms path and it derivatives (“path discovery participant”, “path discovery protocols”, etc)

In Link State protocols each node will flood the network with messages informing the status of its links to immediate neighbors. Since bandwidth in a wireless network is a relatively scarce resource, flooding the network constantly may seriously degrade the overall network throughput. One of the optimizations employed by OLSR to reduce the negative impact of the control messages is the Multipoint Relay (MPR) [14] mechanism.

In MPR a node will send link state messages only to a subset of all its immediate neighbors. This subset will be such that all nodes that are two hops away will be reached when the MPRs relay the message (see Figure 14).

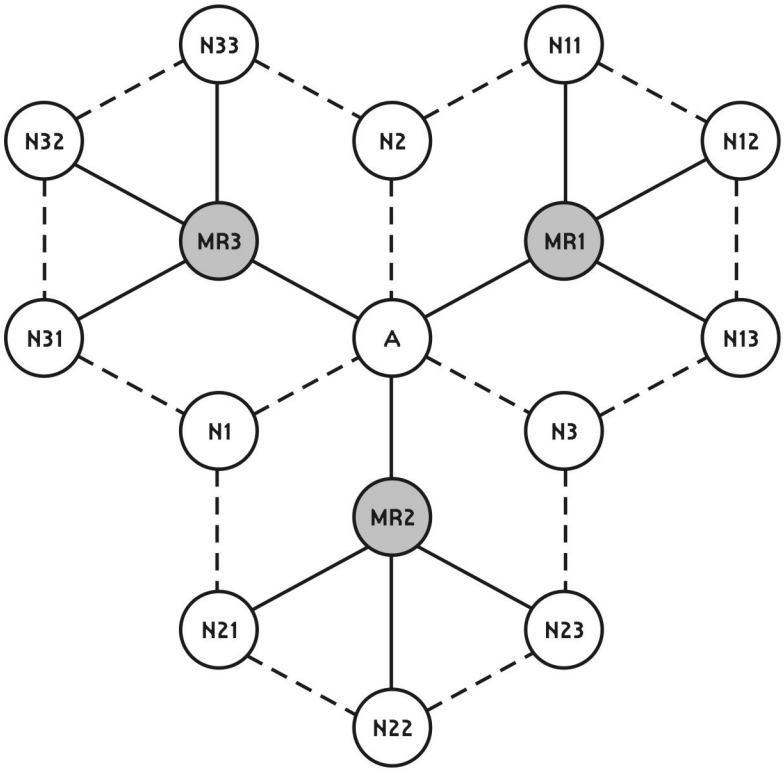

Figure 14 – Through the Multipoint Relays MR1, MR2 and MR3, A can reach all nodes two hops distant.

Published in 1994, the DSDV was one of the first attempts to create a routing protocol capable of handling the constraints of the wireless medium. However, as opposed to OLSR, DSR is not a Link State protocol and relies on the other classic routing protocol approach, the *Distance Vector* algorithm which, in the wired realm, is associated with the RIP [15] protocol.

Distance Vector protocols are known to be less computationally expensive than the Link State candidates, but they pose the risk of creating route loops. The traditional ways to address routing loops in Distance Vectors protocols, namely the Poison-Reverse and Split-Horizon [15] techniques cannot be implemented in the wireless medium due to its inherent broadcast nature.

DSDV is mostly an implementation of the Bellman-Ford algorithm adapted to address the routing loops issue by the use of *Sequence Numbers*. An analytical prove that the use Sequence Numbers eliminates the routing loops can be found in [17]. This mechanism has been implemented in many other ad-hoc protocols, including the reactive protocols that we will see soon.

3.4. REACTIVE PROTOCOLS

Another category of routing protocols are the reactive protocols, where route discovery mechanisms are activated only when necessary. Nodes will have to wait until the information is gathered, what means a somehow delayed beginning of the transmission. Here, again, routes can rapidly and constantly become unavailable what means that route maintenance mechanism may be activated many times during a transmission. Examples of reactive protocols are the Dynamic Source Routing (DSR) and the Ad-hoc On-Demand Distance Vector [16].

Source Routing is the base of the DSR protocol and consists of a technique where every frame includes a sequential list of hops – the route – on which it should relay in order to reach its destination. This route was discovered previously by a mechanism that is common, with minor variations, to all reactive protocols, the *Route Request/Route Reply* cycle, which will be studied in detailed when we describe the IEEE802.11s path discovery mechanism later in Chapter 4. In short, if S wants to find a route to D, it will flood the network with a Route Request frame, which will eventually reach D, which, in turn, will respond with a Route Reply. Once again the network is flooded with control messages and, this time, in a less predictable way than the proactive control messages do it.

Although its proponents advocate that Source Routing poses less traffic overhead than the Distance Vector or Link States paradigms, we should note that, though there is partial elimination of the control traffic (there is still the Route Request/Reply messages), an overhead is posed to each data packet, since they will be carrying the source routing information.

The Ad-hoc On Demand Distance Vector (AODV) is a popular distance vector reactive protocol which, like OLSR is also described in an experimental RFC [17] and has more than one implementation from which the Uppsala version [18] is possibly the most popular. Its route discovery mechanism is also based on Route Request/Response Messages and like many other protocols; it avoids route loops through the use of the sequence numbers technique introduced by DSDV. Contrary to DSR, AODV is a next hop protocol and thus does not piggyback routing information on data frames.

Although reactive, an AODV node may send periodic Hello messages to its neighbors. This way, if a node does not receive any traffic from a neighbor after a certain time it may consider that the link to this neighbor is no longer active. Another usual way of detecting broken links, which is usually employed on all protocols that use Path Request/Response frames, is the transmission of Path Error frames. Since AODV is the main inspiration for the HWMP protocol, which we will study in detail on Chapter 4, we will save more details for later.

3.5. HYBRID PROTOCOLS

In trying to gather the advantages of both proactive and reactive approaches, hybrid proposals have emerged. The HWMP [19], part of the IEEE 802.11s draft, is an example of a hybrid protocol, another example is the Zone Routing Protocol [20].

The Zone Routing Protocol (ZRP) tries to address the limitations of both the proactive and reactive protocols, namely the high cost of the control messages and the high path acquisition time. The idea is that each every node would be the center of a *Zone* that would have a certain diameter, counted in hops. See Figure 15.

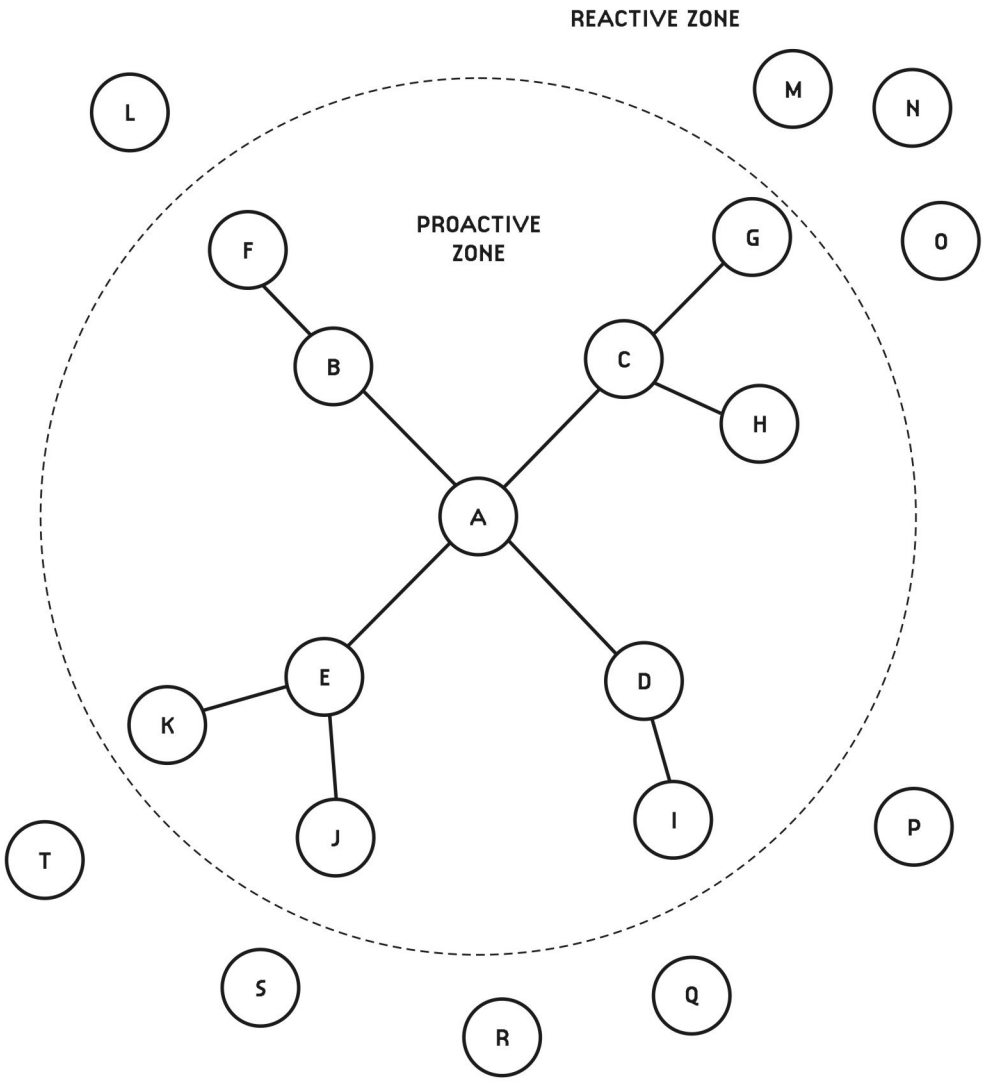

Figure 15 – In ZRP, A uses proactive mechanisms to discover paths for nodes within its 2-hop-radius proactive zone (nodes B to K) and reactive (on-demand) method to reach the other nodes (L to T).

If node B is within the Zone of node A – i.e. if it is less than the configured number of hops (the diameter) away, A will exchange proactive control messages with B. Reactive path discovery mechanisms will be applied only to find nodes outside the zone (for example, C in Figure 26). In this case, A will send unicast Path Requests to the nodes on the border of its zone and those would carry on the task of finding a path to C.

The assumption that the traffic generated by a node is mostly directed to the neighbors nearby (inside the zone) may not always hold correct, but there is merit to the hybrid approach, since it introduces a certain hierarchy that may be necessary in a large

network. An analogy with intra-domain, like OSPF, and inter-domain, like BGP, protocols may come to mind.

The way HWMP is hybrid is completely different than what is proposed by ZRP. In HWMP, proactiveness is employed to advertise some special nodes, as we will see in Chapter 4.

3.6. METRICS

Another characteristic by which routing protocols can be classified is the metric used in the routing decision. In a network in which nodes move quickly, the links will break and form continuously and the routing protocol must be able to converge to the new topology in a short interval. In such an environment, hop count seems to be a natural choice, particularly if we assume that traffic seems to flow to and from gateways connecting the mesh network to the wired Internet. But it is important to observe that 802.11 networks are multirate networks. It is common for a node to support more than ten codification rates and higher rates mean shorter range. For that reason, hop count might not always be the best option. In today's multihop wireless networks, the medium is the scarcest resource and it makes sense to privilege the higher rates, for they consume less airtime, even if this will result in longer paths.

One of the earliest proposed metrics – the Expected Transmission Count (ETX) [21] – computes the expected number of times a packet would have to be transmitted to successfully reach a neighbor. An evolution to ETX is the ETT metric [22], where the number of tries is replaced by the expected time a node would need to successfully forward a frame. This way, the metric would account for the different rates at which nodes can transmit in a wireless network.

Recent WMNs implementations may take advantage of some more advanced techniques as the simultaneous use of orthogonal radio channels. This brings new demands in terms of metrics – they will have to account for intra-flow interference (when two nodes transmitting packets from the same flow interfere with each other) and inter-flow interference (when it happens among concurrent flows). A protocol which deals with inter-flow interference is Weighted Cumulative ETT (WCETT) [22], while the

Metric of Interference and Channel-switching (MIC) [23] and iAWARE [24] are designed to deal with both inter and intra flow interferences.

Quick and unpredictable variation of the link quality is another phenomenon to take into consideration when determining an effective metric for wireless networks. Some metrics take the standard deviation in addition to link quality average values. Examples of instability-aware metrics are the modified ETX (mETX) and Effective Number of Transmissions (ENT) [25].

Another increasingly popular approach is the use of power-aware metrics, which accounts for the battery power available for a given node in a mesh. Power aware metrics are more keen to MANETs where the routing nodes are mobile (and thus battery powered) than to WMNs, where routing nodes may in general be AC powered.

As we will see in the next Chapter, the IEEE 802.11 Task Group “s” decided on an Airtime Link Metric, which main goal is to save the medium, by taking into account not only the datarates that a given link can support, but also the probability of success on the transmission of frames.

4. IEEE802.11S

4.1. INTRODUCTION

In September 2003, IEEE started a study group to investigate adding wireless mesh networks as an amendment for its IEEE 802.11 standard. One year later, the study group became the Task Group “s" (TGs), which issued its first draft later in March 2006. By the time of writing, IEEE 802.11s is still a draft (currently in version 2.00) [44, 50, 51], therefore some degree of change should be expected before IEEE 802.11s becomes a standard. In fact, many improvements have been made in the current draft, considering previous versions of the document, and it is important to keep in mind that this is still a work in progress. Nevertheless, commercial implementations of this draft are already available in some wireless devices, like the XO.

The recent emergence of handheld communication devices, constrained in many ways (power, processing, memory), demands a solution that may be easily embedded in network interface cards (NIC) and in systems-on-chip (SoC). A MAC layer solution, being lightweight in contrast to a full implementation of ad hoc routing, fits that purpose.

In order to support multihop forwarding at the MAC layer, the current draft introduces changes in the MAC frame format, and an optional medium access method as well as many other optimizations to improve performance and security of wireless mesh networks. In this section, we focus on path selection mechanisms since this is the item most related to our research, but we will also discuss the new frame format introduced by the amendment. Additional features are briefly discussed at the end of this Chapter.

Originally, two path selection mechanisms were proposed in the draft. RA-OLSR (Radio-Aware Optimized Link State Routing) [26], which is a proactive controlled-flooding protocol based on OLSR but adapted to work at layer two instead of three, and a hybrid traffic-aware protocol, named HWMP (Hybrid Wireless Mesh Protocol), based on AODV, which is actually the mandatory protocol and the only one remaining on the current proposal (version 2.00). RA-OLSR was removed in favor of an extensible path

selection framework that enables alternative implementations of path selection protocols and metrics within the mesh framework.

Before going into the path selection mechanisms though, we must briefly discuss the mesh creation mechanisms and describe the architecture proposed by the emerging standard.

4.2. MULTIHOP-MAC MESH NETWORK ARCHITECTURE

According to the IEEE 802.11s draft, nodes in a mesh network fall into one of the four categories as illustrated in Figure 16:

♦ Client or Station (STA) is a node that requests services but does not forward frames, nor participates in path discovery mechanisms;

♦ Mesh Point (MP) is a node that participates in the formation and operation of the mesh cloud;

♦ Mesh Access Point (MAP) is an MP which has an attached access point (AP) to provide services for clients (STA); and

♦ Mesh Portal Point (MPP) is an MP with the additional functionality of acting as a bridge or gateway between the mesh cloud and external networks.

Figure 16 illustrates a possible ad hoc topology for this architecture. The doted lines represent the mesh network itself (mesh cloud) in which other non-802.11s nodes may participate indirectly (solid lines) connecting to mesh nodes extended with access point functionalities (MAPs).

In this topology example, there is only one MPP (PORTAL/MP), but nothing prevents a mesh network from having many. In that case each node must dynamically choose one of them for sending traffic outside the mesh network bounds.

Figure 16 must be understood as a snapshot for a dynamic topology, where nodes may move in unpredictable and diverse ways, and links are formed or disrupted not only because of mobility, but also due to the changing conditions of the wireless medium. In that sense, the role of MPP is opportunistic and the network should provide the means (protocols and mechanisms) for announcing throughout the mesh cloud the set of nodes

that are able to work as MPPs. These announcement mechanisms will be described in the following sections.

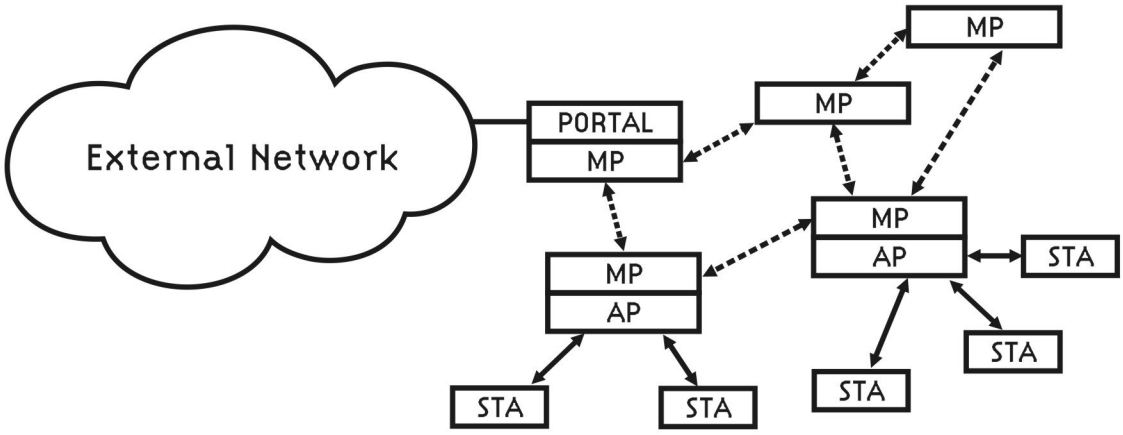

Figure 16 – The 802.11s network architecture. MPs are the Mesh Points and STAs represent the Stations. Two MAPs (Mesh Access Points) are displayed as the aggregation of a Mesh Point (MP) and an Access Point (AP), while the MPP is Mesh Point (MP) augmented with Gateway (Portal) capabilities.

4.3. MESH CREATION

In infrastructured wireless networks, a Service Set Identifier (SSID) is used to distinguish the set of access points, which maintains a certain functional correlation and belong to the same local area network.

In a mesh network the same need for an identity exists, but instead of overloading the definition and function of the SSID, the draft proposes a Mesh identifier or Mesh ID. Similarly to 802.11, beacon frames are used to announce a Mesh ID, which should not be confused with the standard SSID employed by regular infrastructured wireless networks. To avoid misleading a non-mesh station when trying to associate to a mesh network, Mesh Points (MPs) broadcast beacons with the SSID set to a wildcard value.

The Mesh ID is one of the three elements that characterize a mesh network. The other two are a path selection protocol and a path selection metric. Together these three elements define a *Profile*. A Mesh Point may support different profiles, but all nodes in a mesh cloud, at a given moment, must share the same profile.

The 802.11s mandatory profile defines HWMP as the path discovery mechanism and the Airtime Link metric as the path selection metric, as it will be described in the following sections. The draft does not prevent other protocols or metrics from being used in a mesh cloud and even defines frameworks for those alternative mechanisms, but it advises that a mesh network shall not use more than one profile at the same time. This recommendation may be interpreted as an attempt to avoid complexity of profile renegotiation that may be too expensive for a simple device to handle. If a mesh cloud is formed with non-mandatory elements (protocol and metric), it is not obliged to fall back in order to accommodate a new mesh member that only supports the mandatory profile.

A mesh network is formed as MPs find neighbors that share the same profile. The neighbor discovery mechanism is similar to what is currently proposed by the 802.11 standard – active or passive scanning. In order to achieve this, regular (802.11) beacon frames and probe response frames are extended to include mesh related fields. As it will be discussed in the following sections, the draft does not only introduce new frames but also extends pre-existent ones.

To conclude our analysis on the mesh creation procedures we should comment on the establishment of the *Peer links* – the edges of a mesh graph. A Mesh Point will create and maintain peer links to its neighbors that share its active profile (an MP may keep many profiles, but only one is active at a given moment). Once a neighbor candidate is found, through active or passive scanning, an MP uses the Mesh Peer Link Management protocol [44] to open a mesh peer link.

A mesh peer link is univocally identified by the MAC addresses of both participants and a pair of link identifiers, generated by each of the MPs in order to minimize reuse in short time intervals.

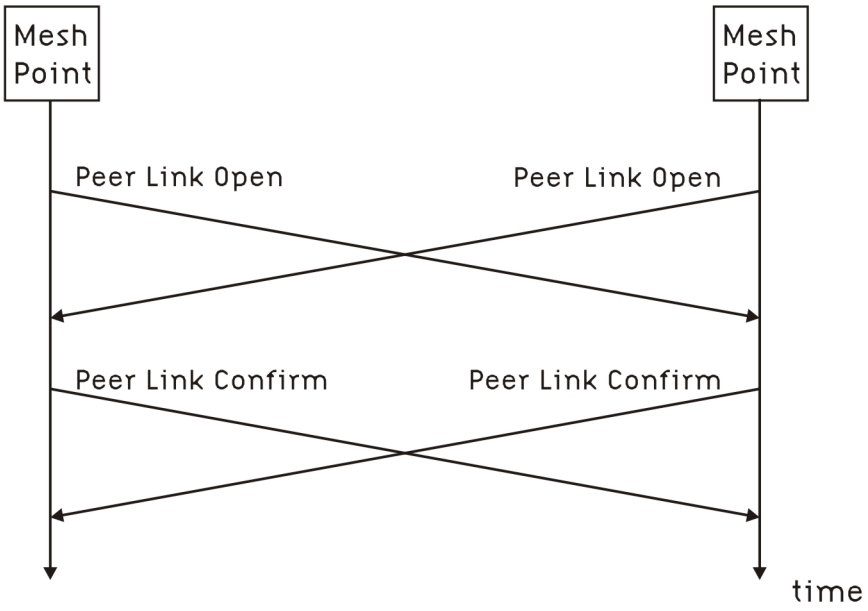

Figure 17 – The establishment of a link in 802.11s

To establish a peer link, both MPs exchange *Peer Link Open* and *Peer Link Confirm* frames as depicted in Figure 17. Whenever an MP wants to close a peer link it should send a *Peer Link Close* frame to the peer MP.

4.4. PATH SELECTION MECHANISMS

IEEE 802.11s proposes a mandatory path selection protocol: a hybrid (proactive/reactive) protocol named HWMP - Hybrid Wireless Mesh Protocol. Although the standard assures compatibility between devices of different vendors by dictating mandatory mechanisms (HWMP and the Airtime Link Metric), it also includes an extensible framework that may be used to support specific application needs.

In order to exchange these configuration parameters, a *Mesh Configuration* element is transported by beacon frames, Peer Link Open frames and Peer Link Confirm frames. The Mesh Configuration element contains, among other sub-fields, an *Active Path Selection Protocol Identifier* and an *Active Path Selection Metric Identifier*.

4.5. HWMP AND AIRTIME LINK METRIC

As a hybrid protocol, HWMP aims at merging advantages of both proactive and reactive approaches. It is inspired by the Ad Hoc On Demand Distance Vector (AODV) protocol [45] and on its extension AODV-ST [46].

HWMP can be configured to operate in two modes: on-demand reactive mode and tree-based proactive mode. On-demand mode is appropriate to establish a path between MPs on a peer-to-peer basis, while in proactive mode; a tree-based topology is calculated once an MP announces itself as a root MP. The tree-based approach can improve path selection efficiency when there is a tendency to forward significant portions of network traffic to some specific nodes, for instance to a Mesh Portal Point (MPP).

What makes the protocol truly hybrid is the fact that both modes may be used concurrently. The main advantage of this approach is that, in certain circumstances, although readily available, the tree-based path may not be optimal and an on-demand path discovery may be employed to determine a more appropriate path.

One example of such a circumstance is the case where two non-root nodes are able to exchange data through a low cost path (even directly by a single mesh link), but instead they are forced to send their frames to a distant root node up and down the tree.

In 802.11s the mandatory metric is the Airtime Link metric. This metric accounts for the amount of time consumed to transmit a test frame and its value takes into account the bit rate at which the frame can be transmitted, the overhead posed by the PHY implementation in use and also the probability of retransmission, which relates to the error rate in a link. The draft does not specify how to calculate the frame loss probability, leaving this choice to the implementation. Nodes transmitting at low data rates may use all the bandwidth in a network with their long transmissions the same way a high error rate link can occupy the medium for a long time. The Airtime Link metric is designed to avoid both. According to the standard, the Airtime Link metric is calculated as:

$$c_a = \left[O + \frac{B_t}{r} \right] \frac{1}{1 - e_f}$$

(**Equation 1**)

Where O is a constant overhead latency that varies according to the PHY layer implementation, B_t is the test frame size (1024 bytes), r is the data rate in Mb/s at which the MP would transmit a test frame and e_f is the measured test frame error rate.

During path discovery, each node in the path contributes to the metric calculation by using management frames for exchanging routing information. Independently of the operating mode (proactive or reactive), HWMP functions are carried on by management frames with the following set of information elements:

♦ Path Request (PREQ) elements are broadcast by a source Mesh Point that wants to discover a path to a destination Mesh Point;

♦ Path Reply (PREP) elements are sent from the destination Mesh Point back to the source Mesh Point, in response to a PREQ;

♦ Path Error (PERR) elements are used to notify that a path is not available anymore; and

♦ Root Announcement (RANN) elements are flooded into the network in one of the proactive operation modes (there are two proactive modes in HWMP as it will be described later).

The above-listed frames are employed in all of the three mechanisms HWMP provides. The mechanisms are summarized in Figure 18. The first one, which is reactive, is called on-demand path selection. The other two are proactive and are named PREQ and RANN mechanisms.

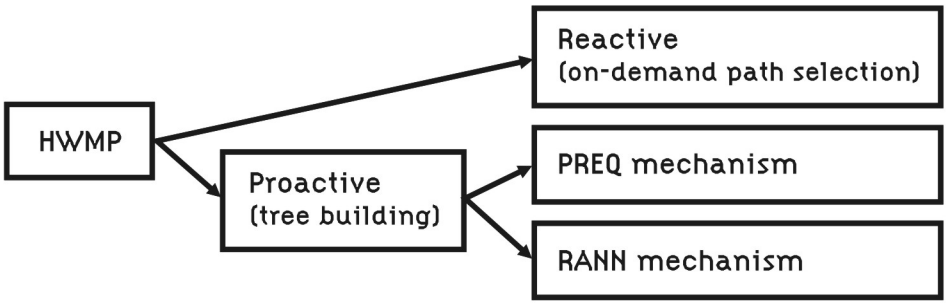

Figure 18 – Path selection mechanisms in 802.11s

Figure 19 displays an example for the on-demand path discovery mechanism. The Source Mesh Point (S-MP) needs to find a path to the Destination Mesh Point (D-MP) and in order to do so S-MP needs the cooperation of Intermediate Mesh Points (I-MPs).

The mechanism works as follows. First, S-MP broadcasts a PREQ frame. Whenever an I-MP node receives a PREQ, it checks to see if it already knows a path to D-MP. If this is the case, this I-MP node issues a PREP frame back to S-MP. S-MP can prevent intermediate nodes from answering PREQs by setting a DO (Destination Only) flag in the PREQ frame. In that case, only D-MP is allowed to respond with a PREP frame. Therefore, when receiving a PREQ with DO set to “1", any I-MP node may broadcast the PREQ frame again and the process repeats until the request eventually gets to D-MP. Only if the DO flag is not set, an IMP node (that knows a path to D-MP) may answer PREQ with a PREP frame. Solid-line arrows in Figure 19 represent PREQs while dotted-line arrows represent PREPs.

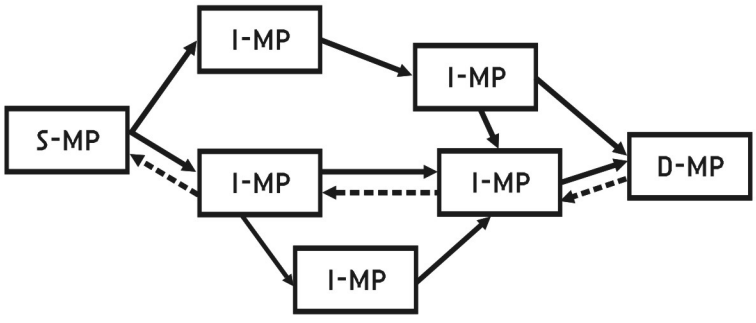

Figure 19 – On demand path discovery where S-MP wants to find a path to D-MP

Another flag, RF (Reply and Forward) can also be used to control the behavior of intermediate nodes. If RF is set to 1, and DO is set to 0, an intermediate node may respond with a PREP frame but it must also broadcast the PREQ frame. Likewise, if both DO and RF flags are set to zero, an intermediate node responds but it does not broadcast the request farther. Hence, the RF flag can limit the quantity of PREPs received by S-MP.

Whenever an I-MP node receives a PREQ, it learns a path back to S-MP. This path is the reverse path and it may be used later (in case this I-MP node is in the selected path) to forward RREP frames to S-MP. Response frames can be unicasted using this reverse path.

Both PREQ and PREP frames carry a metric field and each I-MP node must increment this metric field accordingly. That is how the destination node (D-MP) is able to choose a reverse unicast path among many possibilities (in a dense mesh) and this is also how the source node (S-MP) chooses the forward path at the end of the cycle.

Regarding the denseness of a mesh cloud, we should note that, in a wireless medium, coverage and high data rate are conflicting objectives, and increasing one will decrease the other. Broadcast and multicast frames are usually transmitted at low rates in order to reach most nodes, since distant nodes will then have a greater probability of receiving them. On the other hand, those frames will take a longer time propagating through the cloud, which may be problematic in a dense environment.

Besides the on-demand path discovery mechanism, HWMP provides two different mechanisms for proactively building a forwarding table, as previously stated. The first is based on the PREQ frames and called “Proactive PREQ mechanism" and the second is based on the RANN frames, therefore named “Proactive RANN mechanism".

In the proactive PREQ mechanism, when configured to work as a root MP, a node broadcasts a PREQ frame with DO and RF flags set to 1. This PREQ is sent periodically and every receiving MP updates the PREQ frame (decreasing the hop count and updating the path metric) and broadcasts the PREQ again, which eventually reaches all nodes in the mesh cloud.

Whether or not a node answers with a PREP frame upon receipt of a proactive PREQ depends in part on the setting of another flag, the “Proactive PREP". If the root MP sets it on, all receiving nodes shall send a proactive PREP back to it. A node may send a PREP frame back if it has data to send to the root node and if it wants to establish a bidirectional link, even if the Proactive PREP is not set.

The proactive PREQ mechanism is clearly chatty, particularly in its proactive PREP version. An alternative method is presented by the proactive RANN mechanism. Here, instead of sending PREQs out, a root node can flood the mesh with Root Announcement frames. Nodes willing to form a path to the root MP answer with a PREQ frame. This PREQ is sent in unicast mode to the root MP, through the node by which the RANN frame was received, and is processed by intermediate nodes with the same rules applied to PREQ broadcasts in the reactive mode.

The root node answers each of the received PREQs with a respective PREP, thus forming a forward path from each MP to the root MP. At the end, the RANN mechanism introduces one additional step and may be advantageous if compared to the PREQ mechanism only if a small subset of MPs wants to establish paths with the root node.

Finally, it is worth to comment about the role of PERR frames in the mechanisms previously described. Whenever a frame cannot be forwarded by an intermediate node, this fact should be informed to the previous nodes in the path, until it reaches the original sender that will then start a new path discovery cycle. Thus, PERR frames are used for announcing a broken link to all traffic sources that have an active path over this broken link.

4.6. INTERNETWORKING WITH 802.11S

The multihop capabilities of an 802.11s mesh network would be not very useful without the ability to connect the mesh cloud to other networks such as the wired Internet, as illustrated in Figure 20, which shows two examples of internetworking with mesh networks. As previously mentioned, the IEEE 802.11s draft names gateway nodes MPPs (Mesh Portal Points).

Figure 20(a) illustrates the use of MPPs to interconnect mesh clouds to other LAN networks, when they act like bridges and all nodes belong to the same subnet. Figure 20(b) depicts another scenario where MPPs act as gateways to different layer-three subnets. In a MANET where all nodes are potentially routers they are also potentially gateways to an infrastructured network.

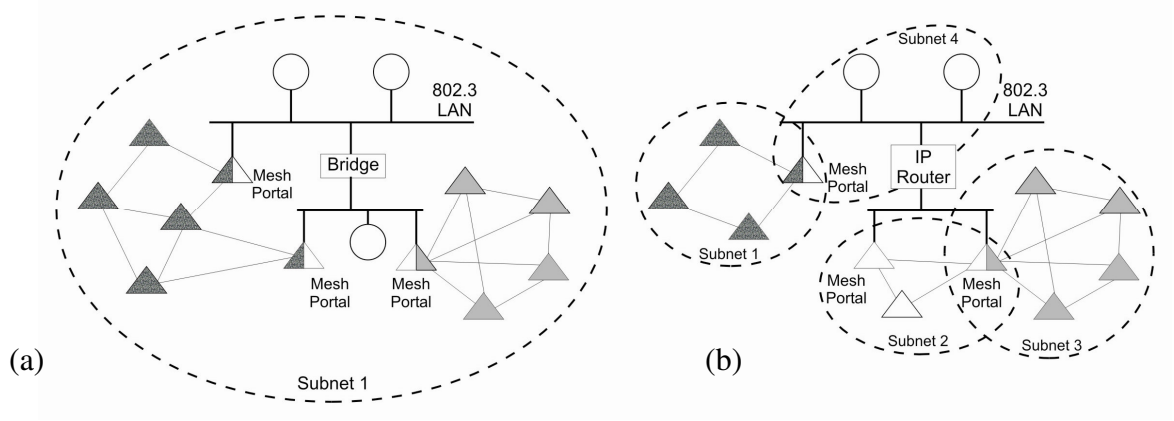

Figure 20 – IEEE802.11s internetworking scenarios. Triangles are MPs, circles are non-MPs (a) Layer 2 bridging – from the perspective of the network (IP) layer all devices are in the same subnet (b) Layer 3 internetworking – a more "traditional" approach, segments of the network are interconnected by an IP Router.

An MPP basic characteristic is the fact that it is a mesh node (MP) that is also connected to another network, and this capability has to be announced for other MPs to benefit from its connectivity. Thus, once configured as an MPP, a node spreads the news sending a Portal Announcement (PANN) frame.

An MP that receives a PANN frame records the MPP MAC address and the associated path metric and then broadcasts the PANN frame again. Each mesh point in the cloud keeps a list of available MPPs and is able to choose among them when it needs to send traffic outside the mesh network limits.

A Mesh Portal Point may also interconnect mesh networks running different path selection protocols. It is also easy to design the interconnection of many wired 802.3 networks and mesh clouds in a big layer-two bridged network using protocols like 802.1D [42].

4.7. FRAME STRUCTURE AND SYNTAX

In order to allow multihop functions at the MAC layer, the IEEE 802.11s emerging standard extends the original 802.11 frame format and syntax. The new frame format supports four or six MAC addresses and new frame subtypes are introduced as it will be described in the present section.

The first two octets of an 802.11 frame contain the Frame Control field and the third and fourth bits of this field identify the frame type, as shown in Table 4.

Besides those two bits, there are also four more bits devoted to a frame subtype. A beacon, for instance, is a management frame (0x0) of the beacon subtype (0x8), while an acknowledgement is a control frame (0x1) of subtype 0xD.

Table 4 – Frame types in IEEE802.11

00 = management frames	01 = control frames
10 = data frames	11 = reserved

Since 802.11s is an amendment to 802.11, the frames it introduces must fall into the four pre-existing categories. Initially the reserved (0x3) type was considered for mesh traffic. Instead, it was decided to extend the data and management frames in the following ways:

♦ data exchanged between MPs are transported by Mesh Data frames, defined as data frames (type 0x2), where a mesh header is included in the frame body; and

♦ mesh-specific management frames, such as PREP or PREQ, belong to type 0x0 (management) and subtype 0xF, which was formerly reserved. This new subtype was named Multihop Action frames.

Another characteristic of the new frames is the use of the FromDS and ToDS flags. In 802.11, those bits marked frames as being originated from or destined to a distribution system, which is the infrastructure that interconnects access points.

Figure 21 depicts a wireless distribution system that connects two access points (AP1 and AP2) and allows two stations (STA1 and STA2) to exchange frames without the intervention of layer-three protocols. In other words, the distribution system provides bridging for the extended service set.

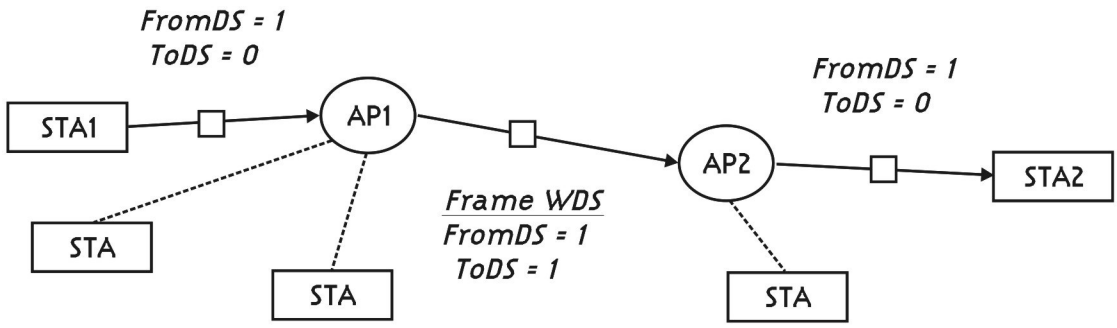

Figure 21 – A frame travels through a Wireless Distribution System

In a wireless distribution system, or WDS, the backhaul connecting the access points is, as the name implies, wireless. A WDS frame is used to exchange frames between them and has both FromDS and ToDS frames activated. Its original role is to allow transmissions between stations connected to two different access points in the same wireless local area network. IEEE 802.11s also sets FromDS and ToDS flags in frames transmitted inside a mesh cloud.

The IEEE 802.11 standard defines frames where FromDS and ToDS flags are set to 1 as “data frames using the four-address format". This definition will be changed to “A data frame using the four-address MAC header format, including but not limited to mesh data frames" when the “s" amendment is approved. The fact that WDS implementations are vendor specific may potentially rise up issues of compatibility with the emerging standard.

As we described the use of the DS flags, we should notice that both are set to zero in ad hoc 802.11 frames. In an ad hoc network, peer-to-peer transfers can happen opportunistically in a way that should not be confused with that proposed by a mesh network, where frame forwarding, i.e. multihop forwarding, capabilities are present.

In Figure 8 we showed the general structure of an IEEE 802.11 frame extended by a Mesh Header (included in the frame body). The Mesh Header is represented in Figure 22 and contains four fields.

Currently, only the first two bits of the Mesh Flags field are defined. They inform the length of the last field in the header - the Mesh Address Extension field - and vary between zero and three, meaning the number of MAC addresses carried in the Mesh Address Extension field.

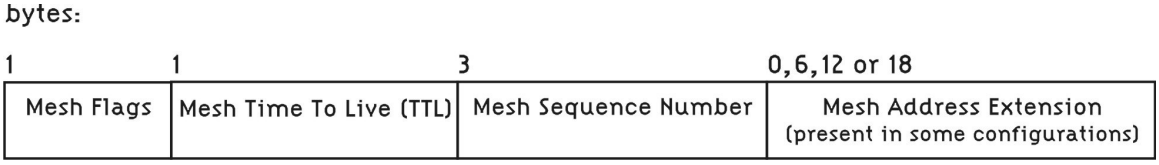

Figure 22 – Mesh Header introduced by 802.11s

The Mesh TTL (Time To Live) field is decremented by each transmitting node, limiting the number of hops a frame is allowed to take in the mesh cloud and avoiding indefinite retransmissions in the case of a forwarding loop.

The three-octets-long Mesh Sequence Number identifies each frame and allows duplicate detection, preventing unnecessary retransmissions inside the mesh cloud. Finally, the aforementioned Mesh Address Extension field carries extra MAC addresses, since the mesh network might need up to six addresses as it will be discussed in the next section.

4.8. STAS CONNECTIVITY AND FRAME ADDRESSING

According to IEEE 802.11s, non-mesh nodes (STAs) can participate in the mesh network through a Mesh Point with Access Point capabilities - MAPs in Figure 16. STAs communicating through the mesh cloud are proxied by their supporting MAPs and this scenario constitutes one example where the novel six-address frame format is employed.

We start our discussion on frame addressing by the more general four-address frame format, which may be used for both data or management frames. The four MAC addresses in this case are:

♦ SA (source address) is the MAC address of the frame source - the node that generated the frame;

♦ DA (destination address) is the MAC address of the node that is the final destination of the frame;

♦ TA (transmitter address) is the MAC address of the node that transmitted the frame. It can be the same as the source address, or the address of any MP that forwards the frame on behalf of the source (any intermediate node); and

♦ RA (receiver address) is the MAC address of the node that receives the frame. It is the address of the next-hop node and, on the last hop to the destination, it becomes the same as DA.

In short, SA and DA are associated to the endpoints of a mesh path, while TA and RA are the endpoints of each single link. Four-address frames are originally supported by IEEE 802.11 for transmissions using a WDS (Wireless Distribution Systems) but, as mentioned before, they are not enough to implement all features described in the emerging standard.

As we exemplified above, if two non-mesh STAs are communicating through the mesh, two additional addresses will be necessary - the Mesh Source Address (Mesh SA) and the Mesh Destination Address (Mesh DA). In order to understand them, DA and SA entities are defined in a more general way:

♦ Mesh SA - In a six-address frame, the SA (source address) is the source communication endpoint, that is, the node outside the mesh cloud that originates the frame. Then, the Mesh SA is the node that introduces the frame in the mesh cloud (on behalf of the SA); and

♦ Mesh DA - Likewise, the DA is defined as the final destination of the frame, while the Mesh DA must be understood as the address of the last node of the mesh cloud that handles the frame.

In Figure 23 we depict a scenario where STA1 wants to communicate to STA2, which is associated to another MAP in the mesh cloud. During this transmission, if we

analyze a frame while it is being forwarded from node MP1 to node MP2, the six-address scheme will be in use (addresses are shown in the figure).

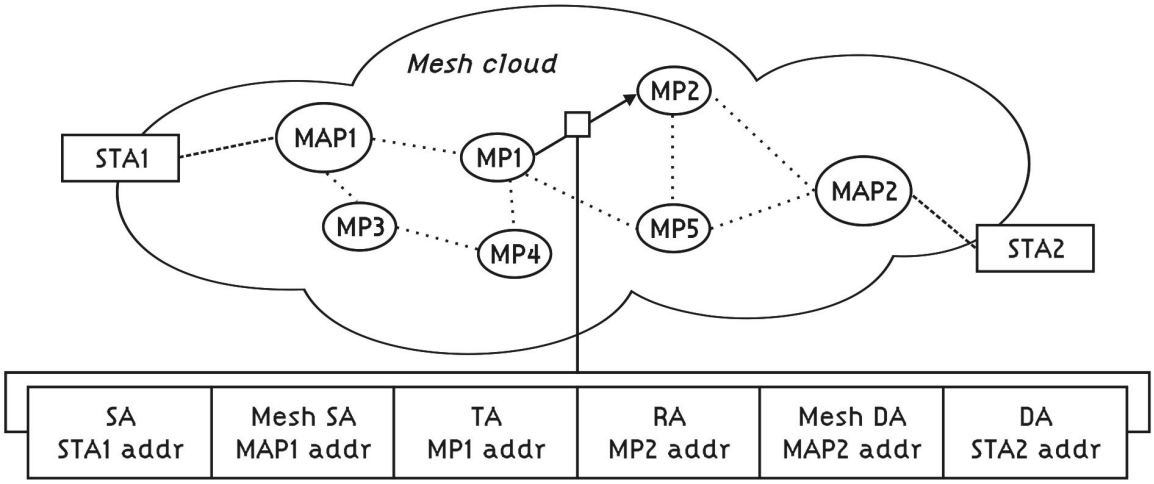

Figure 23 – The six MAC addresses in a frame sent from STA1 and destined to STA2

Another case where the six address format is to be used comes from the HWMP tree-based mode, where two nodes can communicate through a root MP. In this scenario the complete path includes two sub-paths – one from the source MP to the root MP and another from the latter to the destination MP. Finally, mesh points can also communicate with the “outside world" through MPPs. In all those cases, more then four addresses are necessary.

In Figure 24 we present a scenario where MAP2 is substituted by an MPP node. In this case five different addresses are necessary. The six-address frame format is employed again and both the Mesh DA and DA are set to the MPP MAC address. It is responsibility of the MPP to act as a gateway and forward the traffic to an STA outside the mesh cloud, possibly using layer-three traditional routing.

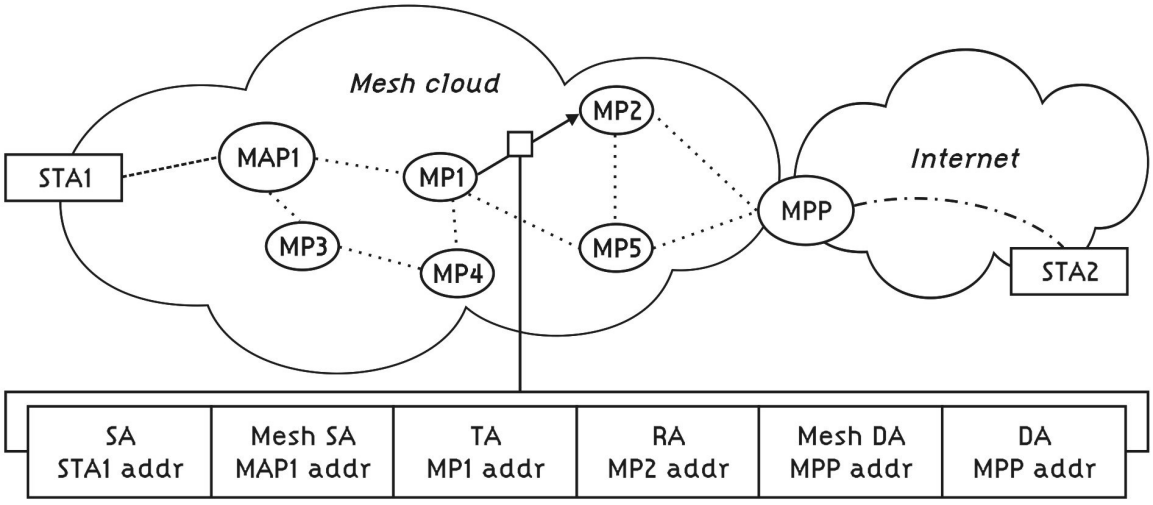

Figure 24 – The six MAC addresses in a frame sent from STA1 and destined to STA2, via an MPP.

4.9. ADDITIONAL FEATURES

The IEEE 802.11s standard covers much more ground than we could address in a book chapter. We have tried to cover the most important points that touch the operation of a mesh network, but there are still some interesting aspects [47].

IEEE 802.11s introduces a medium access method called Mesh Deterministic Access (MDA), which helps to reduce contention through the use of a new coordination function. This mechanism is optional and may be implemented by a subset of the MPs present in a mesh cloud. As a consequence, MDA-enabled Mesh Points must be able to interoperate with non-MDA MPs, potentially hurting the efficacy of the scheme.

The core idea of MDA is the introduction of periods of time, called MDAOPs (MDA Opportunities), during which MDA-capable nodes have the opportunity to access the medium with less contention (there may still be contention due to the presence of non-MDA nodes). MDA is implemented through five new action frames: MDA Setup Request, MDA Setup Reply, MDAOP Advertisement Request, MDAOP Advertisements, MDAOP Set Teardown [44].

Congestion Control is only briefly addressed in the standard proposal. A congestion control mechanism must be selected for the whole network and will be also advertised in the Mesh Configuration element, along with the path selection protocol and metric. The draft describes the format of the Congestion Control Notification frame to be

sent by an MP to its peer MP (or MPs) in order to indicate its congestion status. However, details on how congestion is detected or what triggers the announcement of congestion are considered beyond the scope of the future standard.

Power savings, on the other hand, received more attention in the draft. The main idea is that some capable nodes, named Power Save Supporting MPs, will buffer frames to other nodes, called Power Saving MPs, and transmit them only at negotiated times. It is a service similar to the one access points may provide to its associated nodes in IEEE 802.11 networks.

In terms of security, IEEE 802.11s describes mechanisms to provide both authentication and privacy. Security is based on Mesh Security Association (MSA) services that provide link security between two MPs and may operate even if there is no central authenticator, i.e. it also supports distributed authentication.

Once configured to enable security, an MP shall establish only secure peer links and renegotiate pre-existing unsecure links. The establishment of a secure peer link involves the exchange of extra frames (a four-way handshake) that will start immediately after the initial exchange of Peer Link Open and Peer Link Confirm frames.

The IEEE 802.1X [43] standard is in the MSA core, but pre-shared keys (PSK) may also be employed, which seems viable only for centrally administered mesh networks.

5. THE OLPC IMPLEMENTATION

5.1. DIFFERENCES TO IEEE802.11S

OLPC's XO was the first device to implement a mesh network based on the IEEE802.11s, but this implementation has its own particular characteristics and diverges from the current state of the IEEE802.11s draft at some points. In this section we are interested in explaining exactly what these differences are and in presenting the mesh mechanisms in more detail.

5.1.1. Path asymmetry

In IEEE802.11s' HWMP a path between S and D is assumed to be also a good path between D and S. As we described in Chapter 4, whenever S wants to find a path to D, it broadcasts a PREQs frame that will eventually be responded with PREPs. For the IEEE proposal the reverse path (learned by every intermediary node when it rebroadcasts a PREQ) and the forward path (learned by the intermediary nodes that forward the PREP back to the source) are the same. When D has frames to send back to S it will just use that reverse path without the need to start a new discovery cycle to S.

Differently from the IEEE proposal, in OLPC's implementation a new path discovery mechanism will be started to find a path between D and S. In one hand, radio links are known to be asymmetrical – the fact that a node A successfully decodes a frame from B, does not imply that the opposite is correct. We must recognize that A and B will suffer different interference and contention levels by the mere fact that they are in different positions – a phenomenon perfectly negligible in wired networks, but typical of a radio transmission.

On the other hand, though accounting for path asymmetry may result in more robust forwarding, it is also true that an additional path discovery cycle will increase route acquisition time. This approach may be considered advantageous if (1) path acquisition time is not critical or (2) most paths will be asymmetrical.

5.1.2. Metrics

The calculation of link cost is one item where the IEEE proposal and the OLPC implementation are very different. The first introduces an Airtime link metric that reflects the time necessary for the frame to be successfully transmitted, and this time calculation takes the probability of error into account. For OLPC the cost for a given link will be solely derived from the data rate at each of the PREQs that make from the source to the destination, as we will see shortly. There is no account for the probability of error other than the successful reception of the PREQ frame at a given data rate.

To determine a data rate for a link (and hence its cost) an XO will send a cluster of PREQs consisting of four frames sent at many different transmission rates. The data rates are 54Mbps, 36Mbps, 11Mbps and 1Mbps. The default associated cost for each of these rates is presented in Table 5. A lower rate will have a higher (hence worse) cost. One of the contributions of this thesis is the proposal of adjusted values these costs as we will see in the next Chapter.

Table 5 – Costs associated to each of the PREQs.

Data rate (Mbps)	Associated cost
54	13
36	28
11	42
1	64

5.1.3. The path discovery mechanism in detail

After the initial cluster of PREQs is broadcast, intermediary nodes will begin to rebroadcast them and flood the entire mesh cloud – a process called Network Wide Broadcast, or NWB – in an attempt to reach the destination. In the current implementation, the first PREQ received by an intermediary node will be immediately rebroadcast, but a delay time will be respected before additional PREQs are forwarded. During this delay time, the node may receive multiple PREQs, not only with different metrics, but also from different transmitters, but only the PREQ with the best metric will be rebroadcast after this time expires. This mechanism will avoid unnecessary

consumption of airtime. After all, reactive protocols can be extremely bursty and rebroadcasting every single PREQ a node receives would make things even worse.

Since broadcast frames are not acknowledged, they lack any retransmission mechanism and this fact alone advocates for some level of redundancy as necessary or many path discovery cycles would not succeed. On the other hand, another means for improving broadcast/multicast reliability is reducing the transmission data rate for multicast/broadcast frames. But since lower transmission rates mean longer transmission times, there is a clear tension between coverage/reliability on one side and spectrum efficiency on the other.

After the PREQ delay period is finished (by default this time is 10ms, but it is adjustable as we will see in the next chapter), a new timer will be fired, for the intermediary node may still receive additional PREQs. In this case, the node will retransmit only the best PREQ received during this second period.

But an intermediary node will not simply rebroadcast a single PREQ, it must actually make a new PREQ cluster based on the first PREQ received and also for every best PREQ received over the delay time period. This means that an intermediary node will rebroadcast at least a complete cluster (4 frames) and maybe more, depending on the configured delay time and on the mesh denseness. It is an inherent characteristic of a dense cloud that an intermediary node will receive many PREQs coming from different paths, with different metrics and hop counts. It is not impossible that PREQs with lower costs (better metrics) are received after PREQs with higher metrics. Thus, in a dense mesh, a short delay time will pose a higher control overhead on the cloud.

It is also worth noting that there is no DO or RF flags present in the XO path discovery frames. In fact, there is an implicit DO flag, since an intermediary node will not respond to PREQs. In short, a description of the path discovery mechanism implemented in the XO follows:

(0) – S wants to discover a path to D.

(1) – S will check its forwarding table for a valid path to D. A valid path will be one not expired. Route expiration time defaults to 10 seconds, currently.

(2) – If (1) fails, S will broadcast a cluster of PREQs, consisting of four frames sent back to back at the data rates of 54, 36, 11 and 1Mbps. Each of these frames will have an associated cost (default values can be seen in Table 5).

(3) – All intermediary nodes will rebroadcast the first PREQ received in a new cluster. If, for example, the intermediary node successfully decodes the 54Mbps PREQ from S, it will immediately broadcast a new cluster with values 13+13, 13+28, 13+42 and 13+64 for PREQs at 54, 36, 11 and 1Mbps, respectively.

(4) – An intermediary node will wait a configurable time – the rreq_delay – before relaying another PREQ cluster if a PREQ with a better associated cost happens to be received later. Note that it is perfectly possible for a node to listen to the 11Mbps PREQ from S (with cost 42) and only then receive another PREQ, relayed by another intermediary node, in a two hops path, with a lower cost of 26. Figure 25 illustrates this scenario.

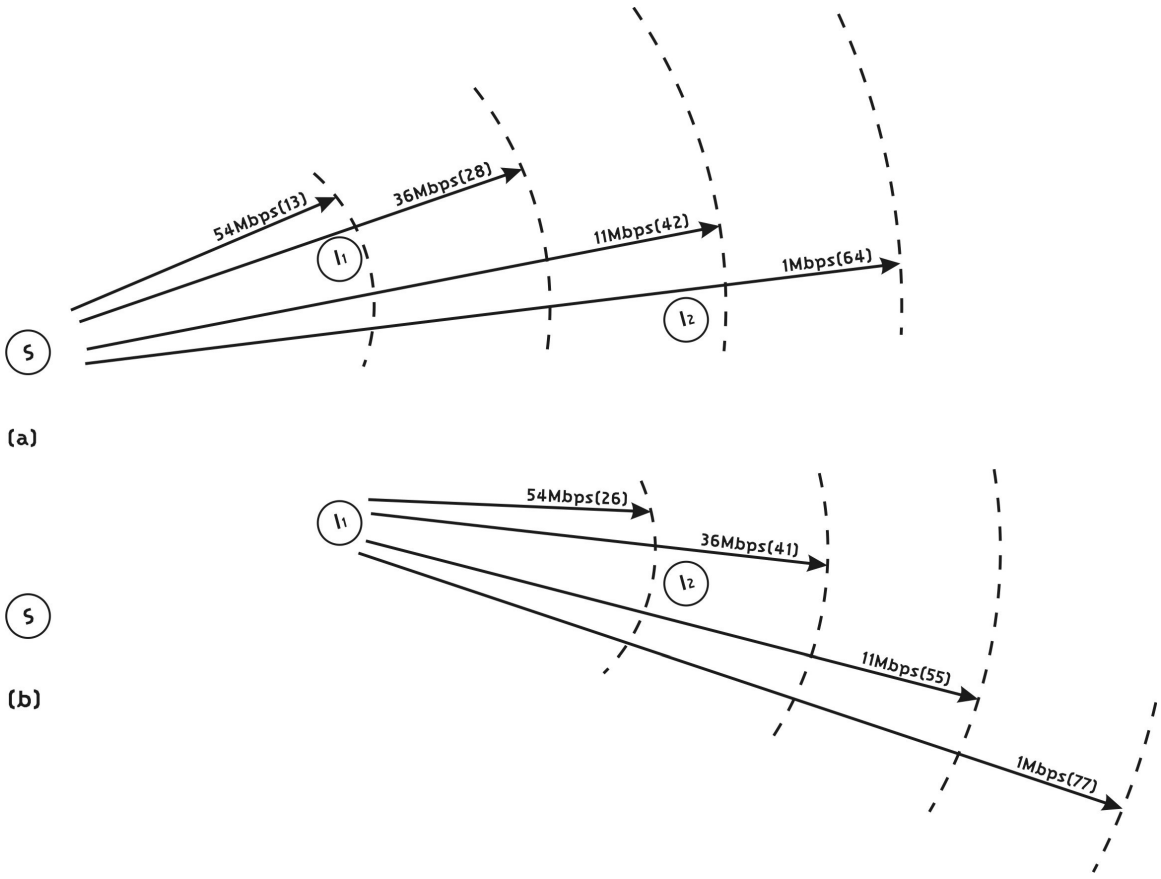

Figure 25 – I_2 receives PREQs directly from S and also with one additional hop, from I_1

All but the first PREQ received (3) will be stored for rreq_delay before being retransmitted. And the intermediary node will only retransmit the best PREQ received during this period. The rreq_delay currently defaults to 10ms.

(5) – The PREQ will eventually hit the destination node D which will respond with a PREP. At this point the procedure is the same of what is described in the 802.11s proposal, besides the fact that if D needs to send traffic back to A, a new path discovery will start. Note that this is only necessary if the application that triggered the path discovery mechanism at A is bidirectional, which is the case for all TCP applications, but not necessarily the case for UDP-based applications.

5.2. NON IMPLEMENTED FEATURES

Some features described in the 802.11s draft are not implemented in OLPC's mesh and this is mainly for two reasons. First, when the first prototypes of the XOs started being tested, the 802.11s draft was still in version 0.01 and many points of the proposal were still being discussed.

The second reason is simplicity. The XO is projected to consume low power – about 5% of what a regular laptop would do – and this led to the use of the Marvell 8388 SoC, primarily designed for cell phone use and thus limited in memory and processing power. For this reason every aspect of the draft that is not fundamental was not be implemented. A little gain in efficiency on the path discovery mechanism or in the mesh operation could mean a lot of extra complexity and computer power requirements and, in this case, the simpler approach was preferred.

5.2.1. Link establishment

For OLPC, there is no idea of a link as being established. There are no Peer Link Open, Peer Link Confirm or Peer Link Close elements and no periodic messages to keep or check if a link is still active (as the Hello messages in AODV). An active neighbor will be a one hop distant neighbor for whom there is an active path in the forwarding table, and nothing else.

5.2.2. Security

Currently no security mechanism is implemented at the link layer. Security is left to be implemented at higher layers (examples: network IPsec, transport, application), though it is true that none of these mechanisms would prevent authentication issues or spoofing, for instance.

5.2.3. Medium access protocol

An XO will contend for the medium using the Distributed Coordination Function (DCF) as described in the 802.11 standard. The MDA is not implemented. The only optimization in that concern was the introduction of a Dynamic Contention Window mechanism as a result to the tests performed during the research phase of this thesis. This will be described in the next Chapter, along with other proposals that derived from this work.

5.2.4. Congestion Control

There is no congestion control mechanism currently. Congestion detection is currently under discussion but only to support local decisions – as increasing the Contention Window, for instance.

5.2.5. Power Savings mode

Though OLPC is implementing an aggressive suspend and resume mechanism that is able to keep an XO running for 20 hours in *Ebook* mode, there is no mechanism in the mesh that will make a node buffer frames for other nodes that are suspended.

5.3. IMPLEMENTATION DETAILS

A standard does not cover every detail necessary for its implementation. Many choices are left to the discretion of vendors and implementers. This is not different for the IEEE802.11s. While the last section described differences between the proposed standard and the OLPC implementation, the current section describes implementation decisions.

5.3.1. Forwarding table

Every XO is capable of keeping up to 64 paths in its forwarding table. An entry on the forwarding table consists of the following fields.

Entry	Meaning
DA	Destination MAC Address: the node this path entry is for.
RA	Receiver MAC address: the next hop for this path.
Valid	If the route is valid or not
Metric	The total cost for the path
Dir	Direction (1 for forward path, 0 for reverse path)
Rate	Data rate used to transmit to the next hop (RA)
SSN	Source Sequence Number (time at the RA for reverse routes. Default is 0)
DSN	Destination Sequence Number (time at the DA for direct routes. Default is 0)
Hopcount	Hop count (currently unused, default is 0)
TTL	Time to live, as usual in hops (Only used in reverse entries)
Expiration	Entry expiration (in ticks, where a tick is 1024us, or approximately 1ms. default is 0 which means never expires)
Sleepmode	RA's sleep mode (currently unused, default is 0)
SNR	SNR in the link to RA (currently unused, default is 0)
Precursor	Predecessor's MAC in direct routes

DA or Destination address is the target for the path, while *RA* will be the next hop in this path. In single hop paths, i.e. paths to direct neighbors they will be the same.

Metric will be the sum of all the costs associated to the PREQs that were successfully transmitted, hop-by-hop, from the current node to the destination (DA). Possible values for a mesh with no more than 5 hops can be seen in Table 13.

Routes are marked *invalid* when MPs detect that the next hop is unreachable. Invalid routes are deleted only when FWT table is full and the XO wants to insert a new route. Note that valid routes can be expired or not expired. A time stamp is associated with every frame added to the forwarding table. This time stamp is provided by the Wireless Module and it is started every time the XO is booted. A route will be considered valid if:

```
Current Time <= Entry Timestamp + Route Expiration Time
```

(Equation 2)

The Route Expiration Time is configurable and currently defaults to 10 seconds.

Every path has a direction (*Dir*). A direct (forward) path will be marked with one, reverse paths will be marked with zero in the *Dir* field.

The *Rate* at which this node will transmit to the next hop – the *RA* – is derived by using a mesh rate adaptation algorithm based on a Packet Error Rate (PER) algorithm. Details of the implementation are property of Marvell Technology Group. The basic idea is to lower the data rate every time a number of successive transmissions fail (no ack is received) and, likewise, raise the rate, if possible, when a given number of consecutive transmissions succeed.

The Source Sequence Number (*SSN*) and the Destination Sequence Number (*DSN*) are used to avoid path loops as explained in [16]. *Hop Count* is the number of hops to the destination (DA) while *TTL* keeps the number of hops to the Source in reverse routes.

The field *Sleepmode* is currently unused, since there is currently no support to mesh power saving mode. The *SNR* value is also not used by the firmware, and is derived when the path is learned and only updated every when the path expires.

Finally, the *Precursor* is the predecessor in a direct path, thus it is the one to which the node will send an Path Error frame (PERR) in case it fails to forward data frames to the next hop (*RA*).

5.3.2. Blinding table

This is a table that, in its default *direct* mode, lists nodes that will not be seen by the XO, i.e. frames coming from that node will be dropped. Conversely, the blinding table can also be put in the *inverted* mode, meaning that the nodes listed in the table will be the only ones seen (frames not dropped). This is mostly a testing feature and has been used thoroughly during this work.

5.3.3. Avoiding Duplicates

In a mesh network, it is common for a node to receive more than one copy of a given frame. To avoid the waste of resources associated to processing or forwarding

duplicates, every node will keep a cache of all the frames recently broadcasted. Entries in the recently broadcasted table expire after a variable timeout between 5 and 320 seconds. The entries in this table are indexed by Source address and a Mesh Sequence Number.

Currently the table can hold 64 entries and the criterion for discarding frames is to pick at random between all the entries on the table excluded the most recent entry. The reason for the variable timeout (5 to 320 seconds) is that there is one single timer for the entire table. It is a watchdog set to expire in 5 seconds. Each new entry added to the table resets the watchdog. On expiration of the watchdog timer, the entire table is cleared. On each insertion, if the time since last table flush is greater than the upper timeout limit, the table is cleared.

5.3.4. Configurable parameters

Most of the parameters that can be tweaked in the mesh are implemented as *ioctls* that can be accessed by the user-space command "iwpriv". An ioctl is a system call that allows a program running on a Unix/Linux system to send commands to a device driver, i.e. to the part of the operating system's kernel that actually controls the device. In terms of the wireless subsystem, there are many ioctls available and most of them were used at some point during this project. Table 6 brings a list of the ioctls currently implemented.

Table 6 – ioctls currently present in the wireless subsystem of the XO

Forward Table management	
fwt_add	Add a path to the forward table
fwt_del	Remove an entry from the forward table
fwt_list	List an entry from the forward table
fwt_lookup	Search for the best path to a given destination in the forward table
fwt_list_route	Same as fwt_list
fwt_list_neigh	List a neighbor from the forward table (a subset of the whole table)
fwt_reset	Removes all entries from the forward table
fwt_cleanup	Remove invalid or expired entries from the forward table
fwt_time	Returns an internal time used to check the expiration of the paths
Blinding Table Management	
bt_add	Add an entry to the blinding table
bt_get_invert/bt_set_invert	Change the mode (0 = direct; 1 = inverted)

bt_del	Remove an entry from the blinding table
bt_list	List an entry in the blinding table
bt_reset	Remove all entries from the blinding table
Mesh Behavior	
mesh_get_ttl/mesh_set_ttl	Get/set maximum number of hops (ttl) a frame can traverse
mesh_get_bcastr/mesh_set_bcastr	Get/set the rate used to transmit multicast and broadcast frames
get_rreq_delay/set_rreq_delay	Get/set the time a node will wait before re-broadcasting a PREQ
get_route_exp/set_route_exp	Get/set the path expiration time
get_link_costs/set_link_costs	Get/set the costs associated to each PREQ (54, 36, 11 and 1Mbps)
Management traffic control	
setprspretrylt	Sets the number of retries when sendinf Probe Reponses
bcn_control	Sets the frequency of beacons
Miscelaneous	
ledbhv/ledgpio	Controls the behavior of the two wireless activity leds
getregioncode/setregioncode	Get/Set the Region Code (Spectrum regulations)

The core of the present work related to the setting some of the parameters in the Mesh Behavior category in Table 6. A detailed analysis of the impact on changing each of these parameters is presented in the next chapter.

6. ANALYSES AND PROPOSED OPTIMIZATIONS

6.1. INTRODUCTION

This chapter will describe our proposed contributions to the OLPC path discovery mechanism implementation and also to other more general aspects of the wireless network employed by the XO. The contributions are the result of research started in early 2007, during the RUCA project, and later as the author worked as an OLPC contractor, and are presented in the form of analyses.

Many of these analyses resulted in changes in the OLPC implementation, some of them served as confirmation that a given mechanism works, and others are currently being studied and may or may not be implemented.

The first group of analyzes is dedicated to the problem of airtime consumption as this is critical to the mesh scalability, verbose protocols and control mechanisms will render a wireless network useless. A dense cloud will be a congested cloud after a given number of nodes are activated and this first group of analyzes aims at increasing this number by pointing out opportunities to reduce the management traffic.

In the sequence, we investigate the Path Discovery Mechanism (PDM) implemented by the XO. Aspects like the costs associated to each PREQ in the PREQ cluster are studied along with the forming of unexpected and suboptimal multihop paths. We also analyze the Path Expiration time parameter and its impact on the PDM performance and reliability.

The third and last group of analyzes is also dedicated to the problem of congestion, but instead of focusing on traffic (as the first group of analyzes) it studies the automated mechanisms of the wireless interface, such as the contention window size and rate adaptation logic.

This set of experiments will result in our recommendations at the conclusion of this work as consolidated in Chapter 7.

6.2. METHODOLOGY AND TESTBED

The results presented in this thesis are based in real world tests, not in simulations. The test bed for the experiments consisted of ten XOs placed in a low radio noise environment, where the noise floor was less than -96dbm.

The XOs were concentrated in a dense setting – the ten occupied a flat surface with approximately 2 meters by 1 meter. The noise floor was measured with the Wispy [29] spectrum analyzer, used to monitor the spectrum conditions during all the experiments.

The wireless traffic was captured by a Cacetech's AirPcap [27] that recorded the captured frames in the tcpdump format with the Wireshark tool [28]. Spectrum conditions were monitored with the Wispy [29] spectrum analyzer.

Python scripts were written in order to process the capture files. Two of them are described in the following sections: The airtime tool and the PDM-analysis tool (with code available in Appendix B).

Additional details for each test will be added to the specific section where necessary. In short,

6.3. ANALYSIS 1 – AIRTIME CONSUMPTION

Wireless medium is a scarce resource. Current data rates commercially available are not only some orders of magnitude slower than those of wired networks but also, to make things worse, the inherently shared medium, particularly on the unlicensed portions of the spectrum, may exhaust airtime availability easily.

For any wireless network to scale it is fundamental to make the best use of the spectrum. Even considering that the XO's primary target are regions where contention or interference from other sources are not as strong as what we would find in the urban developed regions of the world, its mesh network will have to make good use of the airtime in order to scale to many tens of nodes.

In this section we will describe the tests, analysis and adjustments that were performed in order to limit the impact of a specific category of traffic, namely the

management frames that convey link layer presence and capabilities information: Beacons and Probe Requests/Responses.

As the airtime analysis is obviously a very important one in the study of wireless networks, an airtime tool was developed (under the license terms of the GPL license) and in is described at the end of this section. The code is available in Appendix B.

6.3.1. Beacon Backoff

The first analysis for this item was related to the aggregated traffic generated by a group of XOs beaconing. Since the default frequency in which an XO broadcasts beacons is 10Hz, it is easy to see that this traffic could exhaust airtime in a dense mesh if no counter measure was introduced.

XO's beacons typical size is 114 bytes. Since they are transmitted at 1 Mbps, this means that each one of them consumes 1.104 ms or airtime. 30 XOs would, thus, suffice to take about one third of the available airtime.

To avoid this, XOs implement a backoff mechanism that reduces the number of beacons one node will transmit in response to the number of beacons it can hear from other nodes. Figure 26 shows that the backoff mechanism effectively avoids a *beacon storm*. In this experiment the number of beacons sent by one XO was 8.8 per second, while with ten XOs this number raised to only 11.8.

The fact that only 8.8 beacons per second were captured, when we would expect to see 10 beacons per second, can be explained by the fact that the traffic was captured by a monitoring station, that can fail to capture all the traffic and also because some of the beacons might have collided. Finally the frequency may fluctuate because of the backoff mechanism itself – nodes will send beacons according to what they hear – or because of the buffer conditions on the node.

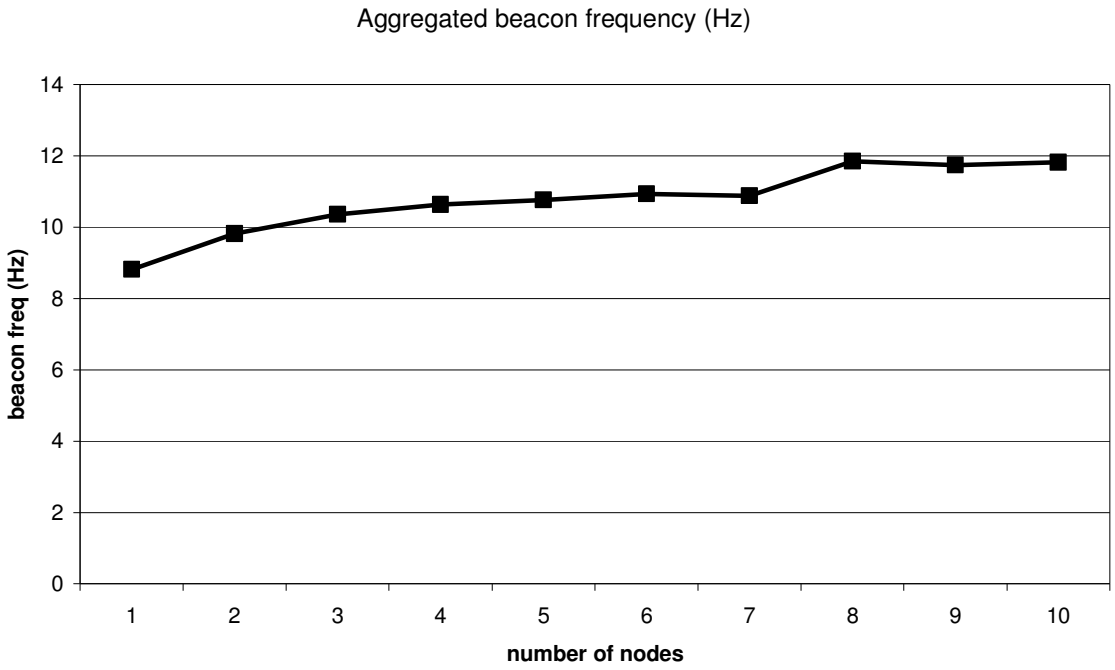

Figure 26 – The aggregated frequency of beacons for a mesh cloud starting with one and growing to 10 nodes.

It was observed that in an environment with more than 50 active XOs, the aggregated beacon traffic was inferior to 20Hz, and although the backoff mechanism was proved effective and might suffice to avoid a spectrum meltdown, another control was added that can reduce or even disable beacon traffic. This control is implemented in the form of a private ioctl and can be accessed via the iwpriv command:

```
iwpriv msh0 bcn_control 0|1 <interval>
```

The default interval is 100ms (10Hz) and values from 20ms (50Hz) to 1 second (1Hz) are accepted. With this new feature the test described above was repeated. All ten nodes were configured to the new frequency of beacons and the aggregated traffic varied from 55Hz (1 node) to 57Hz (10 nodes).

In fact there is no reason to raise the beacon frequency and with the current default value of 10Hz (which can be lowered or even disabled) 10 XOs would consume

1% of the airtime. Beacons are currently only used to maintain the ad-hoc mode compatibility.

We conclude that we can reduce the frequency of beacons to one second and free about 0.9% of airtime and we also conclude that there is no risk of beacons clogging a mesh cloud of any feasible size (a mesh cloud with hundreds of nodes is not foreseen at present).

6.3.2. Probe Requests/Probe Responses

XOs periodically send out Probe Requests and respond to each others` Probe Requests with Probe Responses. Contrary to the Beacons, there is no backoff mechanism and Probe traffic can harm network performance.

What makes Probe traffic problematic is not the frequency probe requests are transmitted – about 0.1 Hz – but the burstiness of the traffic it generates. This can be particularly serious in dense mesh clouds where each single Probe Request will trigger *N* nearly simultaneous Probe Responses.

In a ten node network, one Probe Request and its nine Probe Responses will generate a traffic pattern like the one depicted in Figure 27. As we can see in this real and typical traffic capture, there is a high contention window that will last for about 12 milliseconds. During this significantly long time, data frames will have little chance of being successfully transmitted.

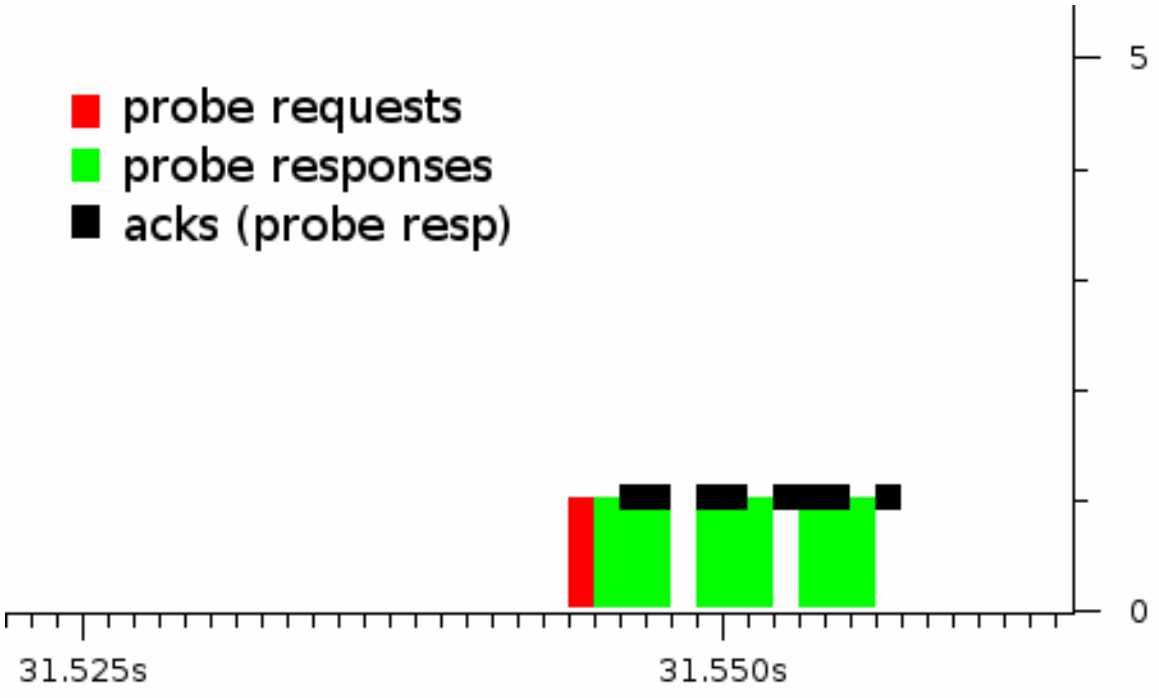

Figure 27 – a burst of probe responses (and respective acks) triggered by a single probe request in a ten nodes mesh cloud captured by a monitoring tool.

Differently from Probe Requests, which are broadcast frames, Probe Responses are unicast transmissions that will be repeated in case the destination (the node that originated the Probe Request) does not acknowledge the reception. Because of the nearly simultaneous transmissions of Probe Responses, they will face a high level of collisions and subsequent retransmissions. It was obvious, from the very beginning of this investigation, that a mechanism to reduce Probe traffic should be implemented. Figure 28 shows the same probe response burst depicted in Figure 27, this time marking the retried transmission.

To address the *probe burst* another iwpriv control was added that allowed setting the number of retries a node should use specifically for the probe responses:

```
iwpriv msh0 setprepretrylt <0|15>
```

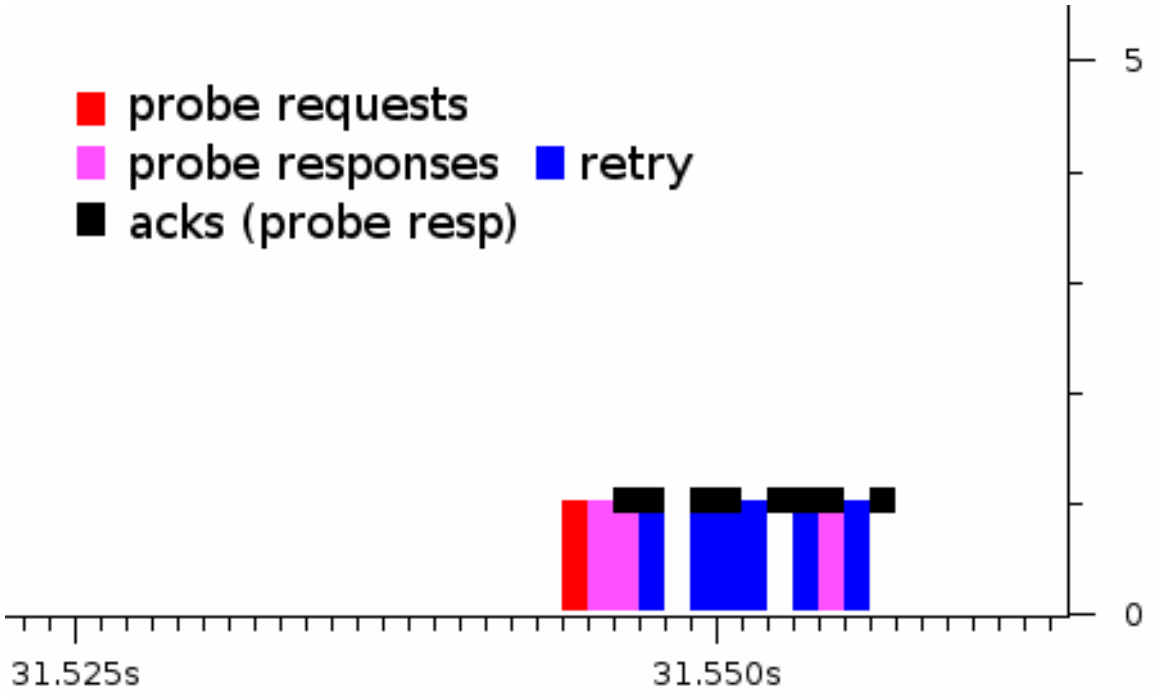

Figure 28 – Same as Figure 27 but marking retries (in blue). Six out of the nine probe responses captured were marked as retransmissions.

The control over the probe responses helped reduce the management traffic, but the most interesting outcome of this experiment was the perception that a mechanism to mitigate the harmful effects of bursts was necessary.

The number of retransmissions that could be observed throughout the experiment accounts to 76% of all probe responses as shown in Figure 29, and can be expected to happen to all traffic that present a similar pattern, i.e. occurs in bursts, and a counter measure to improve this – an adaptive contention window – will be described in Analysis #3.

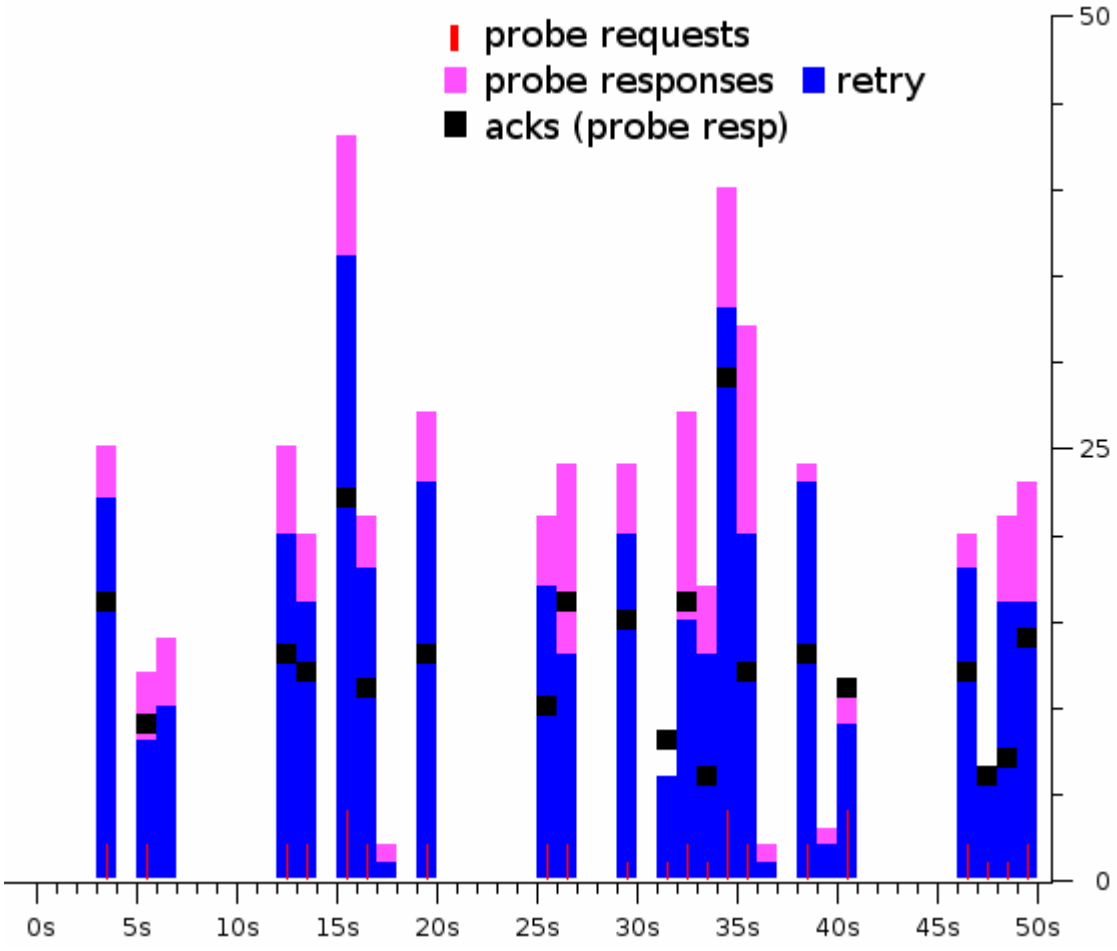

Figure 29 – The number of retries in probe responses transmissions can reach 76% even in an idle network. For this graph, the consolidation interval is 1 second.

6.3.3. The airtime analysis tool

In wired networks the total available bandwidth or the bandwidth required by a given application are very important measures of network capacity or demands. The number of packets exchanged in a given interval is also an important number that can help project or select routers and switches.

Wireless networks, nonetheless, are multirate networks, where a scarce resource is shared not only between nodes, but between distinct networks. This resource is airtime. Open source tools, like *Wireshark*, give the investigator a powerful insight on the frames transferred between the stations, but do not help in airtime analysis.

It is not enough to find out, for instance, that a single node will broadcast 1,140 bytes per second in beacons. It is more meaningful, for the purposes of projecting a wireless network, to discover that they will consume 0.9% of the available airtime.

The time necessary to transmit a given frame is function not only of the transmission rate; it is also affected by the modulation technique in use. An XO will transmit frames in two different modulation mechanisms (CCK and OFDM) according to Table 7.

Table 7 – Modulation technique for each data rate used by the XO.

Modulation technique	**Data rates**
CCK	1, 2, 5.5 and 11Mbps
OFDM	6, 9, 12, 18, 24, 36, 48 and 54Mbps

Given:

- Frame size in bytes: *S*
- Datarate in Mbps: *R*
- Airtime in microseconds (us): *T*

For frames transmitted in CCK:

```
T(µs) = 192 + (S*8)/R
```

Equation 3

Notes:

- The 192 µs term is the sum of the preamble duration (144µs) and the PLPC header duration (48µs).
- Supposing long preambles are being used in all transmissions.

For frames transmitted at more than 11Mbps (OFDM):

```
T = 26 + (S*8)/R
```

Equation 4

Note:

- The 26µs term results from the sum of the signal extension time (6µs) and the preamble (20µs)

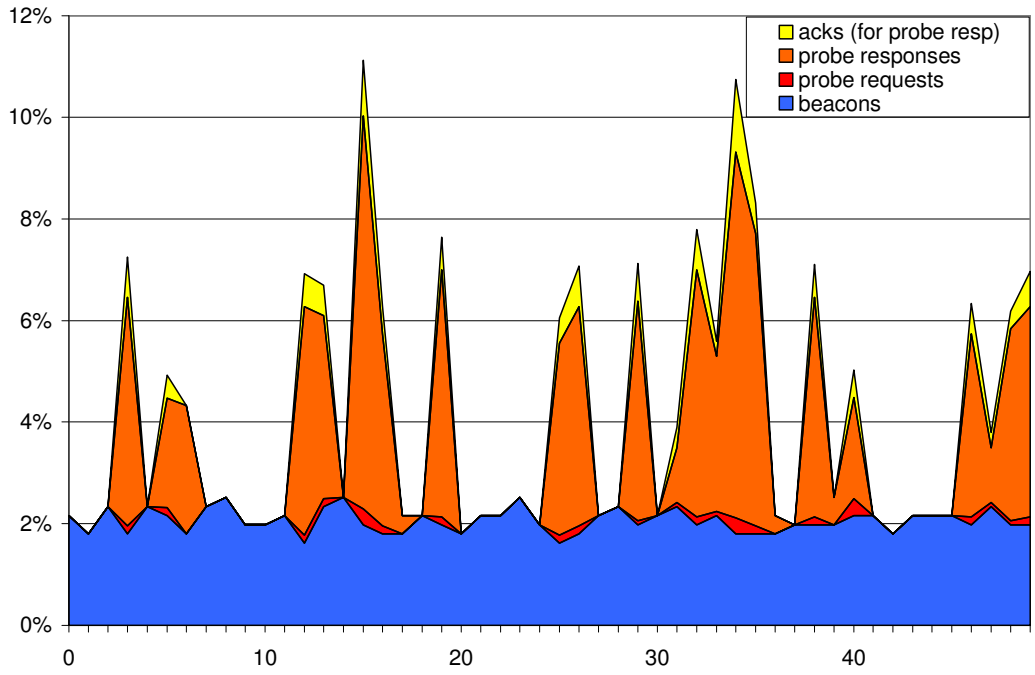

Figure 30 – The airtime consumption for each category of management traffic on a 20 node mesh cloud.

The above math was incorporated in a single python tool that frontends the *tshark* tool (part of the Wireshark traffic analysis tool) and given an input file and a traffic filter, outputs the airtime consumed (percentage) in a configurable consolidation interval.

The python code for the airtime tool, as well as instructions on how to use it, is given in Appendix B. The graph in Figure 30 was drawn on a comma-separated values (CSV) output generated by the *Airtime* tool.

6.4. ANALYSIS 2 – PATH DISCOVERY MECHANISM

The path discovery mechanism was the single area to which most research time was devoted. In this section we present three analyses of the efficiency of the mechanism and a tool created to help in present and future analysis.

6.4.1. Route (path) expiration time

After being discovered, a path will be added to the forwarding table along with a timestamp provided by the radio subsystem. This timestamp is started at zero at boot time and incremented at about every millisecond. At any given time a path will be considered expired if the following condition is met:

```
Current Time > Entry Timestamp + Expiration Time
```

Where *Current Time* is the current timestamp provided by the radio subsystem and can be displayed with:

```
iwpriv msh0 fwt_time
```

The *Expiration time* is a configurable parameter that defaults to 10 seconds, and can be displayed or set with:

```
iwpriv msh0 get_route_exp
```

```
iwpriv msh0 set_route_exp <N(seconds)>
```

Expired entries will be kept in the table and used in case the path discovery mechanism fails to find a new path. The forwarding table is currently capable of storing sixty four entries.

Path expiration is important to maintain the forwarding table up to date. In a highly mobile network a long expiration time will freeze the topology and increase the probability of path breakage. On the other hand, a short expiration time will produce a constant flow of control messages within the mesh cloud particularly because in the XO's implementation paths will expire even if they are in use.

A mesh cloud formed with XOs carried by their users – children – is a real world MANET and some real world use cases can be assumed to better optimize the route

expiration time. First, we may assume that during most of the time a child uses an XO he or she will be static or moving at a very slow pace. When inside a classroom or gathered in a community meeting place, or even when sitting inside their houses and chatting with the neighbor next door, nodes will be mostly stationary.

Even if mobile, it seems reasonable to assume that the nodes will move at a rate inferior to 0.5 meters/second, which means that the distance between any two particular XOs would vary no more than 1 meter per second, in the worst case scenario. It is important to recognize, though, that the physical distance may not be the only factor influencing the quality of a link: crossing a door and interposing any obstacle between the XOs may have more influence than the actual changes in distance. These influences are, on the other hand, too random to be computed, so it seems a reasonable approach to concentrate on the scenarios that not only can be predicted, but are also more typical, particularly because the route expiration time is a static (non-adaptive) value.

There are other factors that influence the quality of a link, even if XOs are stationary:

(1) A child may turn off its laptop and hence stop acting as an intermediary node

(2) A child may join the mesh and possibly introduce new optimal paths

(3) Spectrum conditions may change due, for example, to a varying interference source

All of the above would justify the discovery of new paths but, as previously stated, (1) the XO PDM implementation is the simplest possible, mainly due to memory limitations on the Marvell 8388 SoC and (2) there is no mechanism to announce such changes. Actually, a path will expire after the configured time, irrespective of its current status.

The fact that at every route expiration time a path will be rediscovered brings more tension to the conflicts between shorter times (high overhead) and longer times (high probability of path breakage). Fortunately there is a mechanism to deal with broken paths – the PERR notification.

Compared to the periodic rediscovery, the PERR notification represents the additional delay of having the PERR travel all the way from the node before the broken link to the node at the beginning of a path. If this path is not that long, for example, one or two links, this extra delay will be in the order of 30 or 60 milliseconds.

Expiration is probably not the best approach to deal with broken routes. It brings a proactive element to a purely reactive mechanism, meaning that it will pose a maybe unnecessary overhead, particularly when it recalculates the same path currently in use. On the other hand, if a node is transmitting on a given point of time (on the scale of seconds), it is highly probable that it will be transmitting in the next point in time, this, of course, depending on the average duration of a transmission. In an XO, where collaboration software seems to dominate, longer sessions (and connections) are to be expected.

For a scenario where mobility is not too high, it seems reasonable to tolerate suboptimal paths in exchange of airtime. To deal with broken paths there is the Path Error mechanism. One can argue that keeping the route expiration time low would reduce the probability of breaking the route but then we need to acknowledge that there is no significant difference in cost between rediscovering a route because it was notified as broken by a PERR message and rediscovering it due to expiration. Figure 31 shows that in a mesh formed with 10 XOs and set to the default route expiration time (10 seconds), the PDM traffic will take 15% of the airtime (at every second).

The tests summarized in Figure 31 and Figure 32 measured the effect of increasing the route expiration time in a static dense mesh cloud. The results show, not surprisingly, that the airtime consumption will be reduced by increasing the route expiration time.

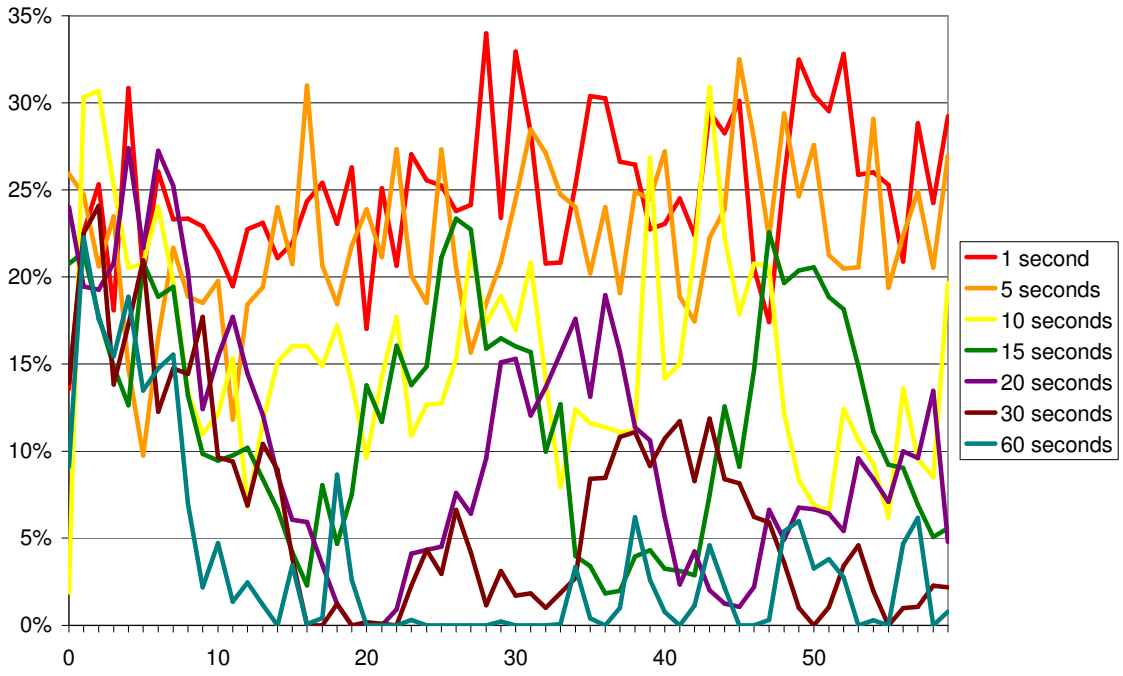

Figure 31 – Airtime consumed as the route expiration time changes (consolidation interval is 1 second)

Figure 33 shows that path breakage on the other hand displays a different correlation with the route expiration time. It is tempting to conclude that the Path Discovery Mechanism itself interferes with the data traffic and causes paths to be marked as broken – that would account for the first part of the curve (5, 10 seconds) – while longer expiration times (like 25 or 30 seconds) cause the expected increase in path breakage. On the other hand, the overall results present high deviation and any clear conclusion would lack statistical support.

This high deviation in this test is in itself an interesting result. They indicate a complex correlation between the PDM and the data traffic, possibly with feedback mechanisms (PDM traffic interferes with data traffic and causes paths to break, triggering even more PDM traffic). The bursty nature of both the PDM traffic and the data traffic used for this tests (multicast pings) aggravates the unpredictability of these results, since it all comes down to the millisecond scale when bursts may or may not coincide. To obtain statistically meaningful results with real tests (as opposed to simulations) an unfeasible number of repetitions would be necessary.

What can safely be conclude is that harmful interactions between data traffic and path discovery mechanism traffic, and even self-interference in path discovery traffic (as shown in the next analysis) do exist and suggest that decreasing the control traffic is an important goal, particularly if the network is not too mobile.

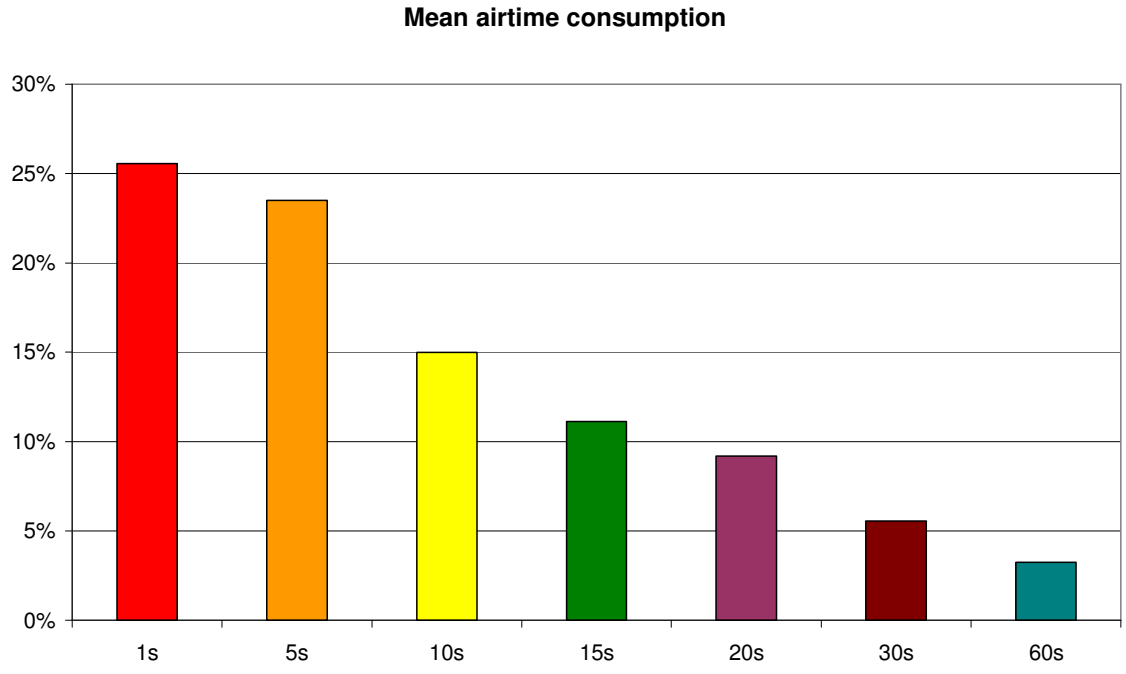

Figure 32 – Mean consumed airtime over the experiment (see Figure 31)

In short, it is safe to assume that nodes will be static or any two nodes will not vary their distance in a rate greater than one meter per second. This will account for most of the usage scenarios. Other factors that may introduce the need of path recalculations are too random to predict.

In a nearly static mesh, like one formed inside a classroom[8] or when a group of children meet to collaborate with their XOs elsewhere, it is highly beneficial to increment this time from its actual default time of 10 seconds to 20 seconds.

Even in the moderately mobile mesh cloud that we would get by the children moving around, the distances traveled during 10 seconds would hardly be superior to 10

[8] Though the suggested way to interconnect XOs in a classroom is still the use of Access Points.

meters. If two XOs move from 40 to 50 meters of distance, for example, there is a good chance that they will not be able to exchange frames at 54Mbps, but this does not mean the disruption of the path only the lowering of the data rate of the link. It is also possible that moving 10 meters will break a marginal link. In this case, a Path Error notification will be issued and a new path will be discovered if an intermediary node is available. Moreover, if an intermediary node was available in the first place, the path discovery mechanism would probably select a multihop path as we will see soon when we discuss the costs of the links.

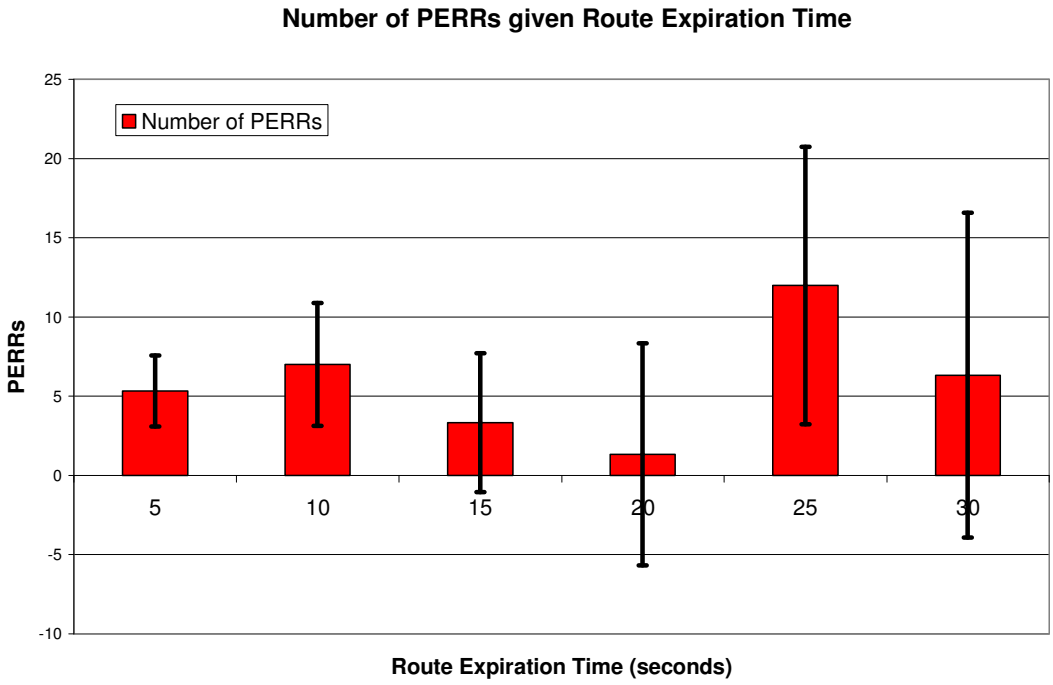

Figure 33 – Path Error frames as a function of the route expiration time.

For all of the above we believe that 20 seconds would be a better value for the route expiration time as a static value.

In a possible adaptive approach, a node would gain insight on the mobility of the mesh cloud by observing the breakage of paths. This could be achieved by counting the number of PERRs over a given time period. But we believe that a highly mobile scenario does not represent a typical use case, and the added complexity would not pay off.

A far more interesting result that came from the path expiration time analysis though was that reducing the amount of control traffic actually seems to improve the

sanity of the path discovery mechanism in dense scenarios. This observation deserved another analysis that we present in the following section.

6.4.2. Path Discovery Mechanism Sanity

This analysis is dedicated to the following investigation: Does the Path Discovery Mechanism (PDM) implemented in the XO behave as expected in terms of the path length?

During tests with dense mesh clouds, there have been observations that the implemented PDM was wrongly creating unnecessarily long paths. If all nodes are close to each other – for example, within a few meters – one would not expect to see multihop paths forming.

The present analysis was designed to determine:

(1) If the multihop paths were actually forming and with which frequency

(2) What could be the causes

(3) How (and if) they can be eradicated

In trying to answer the questions above, other ones came up. For instance: how data traffic and PDM traffic influence each other. Will the burstiness of PDM's traffic significantly degrade data throughput? And will data traffic prevent path formation or induce unnecessarily long paths?

To determine if multihop paths form even when all nodes are within the transmission range of each other a test bed with ten XOs was prepared in which the nodes were placed in a very dense configuration (less than 2 square-meters area).

In such a situation one would expect the paths to be mostly, if not all, single-hop paths, because a successful link at the top datarate of 54Mbps between any two nodes is theoretically possible. A path with two or more hops would have to be explained and, the longer the path, the more suboptimal the results would be.

The tests were performed in a quiet environment. The workload used for this particular test was multicast pings (addressed to 224.0.0.1, default size, 1Hz). The test was performed three times (1 minute for each test). Multicast pings represent an easy way

to test and stress the PDM, since, in order to respond to an echo request sent to the multicast address, all the nodes must find a path to the requesting node.

Table 8 shows the results for a single run of the test for node J (nodes in the test are named A to J). In this case the Echo Reply frames captured by a monitoring station were used and the data shows clearly that multihop paths were selected by the PDM. In one case, the path for node *D* has reached five hops and only to three nodes the paths behaved as expected (C, F, I) all the time, i.e, for these nodes single paths were selected all the time.

Table 8 – The forming of multihop paths between node "J" and other nodes in the mesh.

Destination node	**1 hop**	**2 hops**	**3 hops**	**4 hops**	**5 hops**	**PLI**
A	90.0%	10.0%	0.0%	0.0%	0.0%	1.05
B	82.1%	17.9%	0.0%	0.0%	0.0%	1.09
C	100.0%	0.0%	0.0%	0.0%	0.0%	1.00
D	59.0%	24.0%	4.0%	6.0%	7.0%	1.33
E	87.1%	8.6%	4.3%	0.0%	0.0%	1.08
F	100.0%	0.0%	0.0%	0.0%	0.0%	1.00
G	89.6%	10.4%	0.0%	0.0%	0.0%	1.05
H	86.5%	13.5%	0.0%	0.0%	0.0%	1.07
I	100.0%	0.0%	0.0%	0.0%	0.0%	1.00

During the one minute experiment, every path would be rediscovered at least six times, since the route expiration time was set at its default value of ten seconds. In order to make the data easier to read a *Path Length Index* (PLI) for each node was calculated. This index indicates how far from the expected value (that would equal 1.00) paths formed at each node for every other node were during the experiment. This index was calculated as follows:

$$PLI = \sum_{1}^{n} i \bullet P_i$$

Where P_i is the percentage of the traffic captured over a path formed by *i* hops. In the performed experiments, n equals five, since the *Mesh TTL* (the maximum number of hops that a frame can take before being discarded) defaults to five.

Table 9 consolidates the results for the three runs of the multicast ping test described above. For each node, PLIs were calculated for every other node and mean values, as well as the standard deviation.

Table 9 – PLI for each node

Node	Mean PLI	stdev
A	1.07	0.08
C	1.05	0.07
C	1.04	0.06
D	1.07	0.08
E	1.06	0.07
F	1.05	0.07
G	1.05	0.07
H	1.04	0.05
I	1.06	0.07
J	1.06	0.08

This experiment proved without doubt that multihop paths are actually forming. In addition to the index used in Table 8 andTable 9 it is interesting to note that exactly 9.99% of all data traffic (pings in this case) exchanged during the tests were transmitted over a multihop path.

The multihop paths in a dense environment introduce unnecessary overhead in a scenario where contention for the medium is already high. It is important, then, to determine if the situation can be aggravated by an increase in the number of nodes. Since only ten XOs were available for these tests, one way to increase the PDM traffic was to reduce the Route Expiration Time.

In the next experiment, we vary the Route Expiration Time and recalculate the Path Length Index and the percentage of traffic forwarded via multihop paths. Figure 34 shows that the percentage of frames transmitted over a multihop path drops from 9.99 to

6.24 percent if we increase the route expiration time from 10 seconds (the default) to 30 seconds. And if we fix it in 5s, this percentage grows to more than 16%. Likewise, Figure 35 shows that increasing the route expiration time will cause a decrease in the path length.

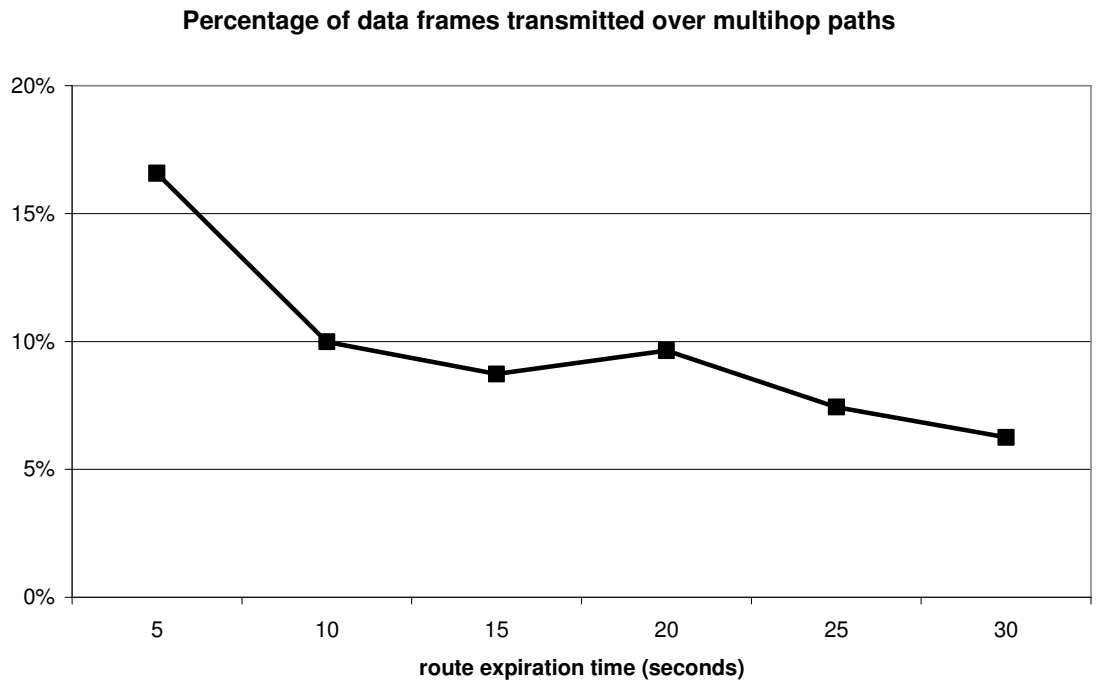

Figure 34 – Percentage of traffic delivered through multihop paths given the route expiration time

Based on these results we can infer that increasing the number of XOs in a dense mesh cloud will increase the PDM traffic and cause longer paths as a result, indicating that the PDM traffic interferes with itself. And since PDM traffic self-interferes it seems safe to assume that any traffic will potentially interfere with the path discovery traffic and, as a result, increase the average path length.

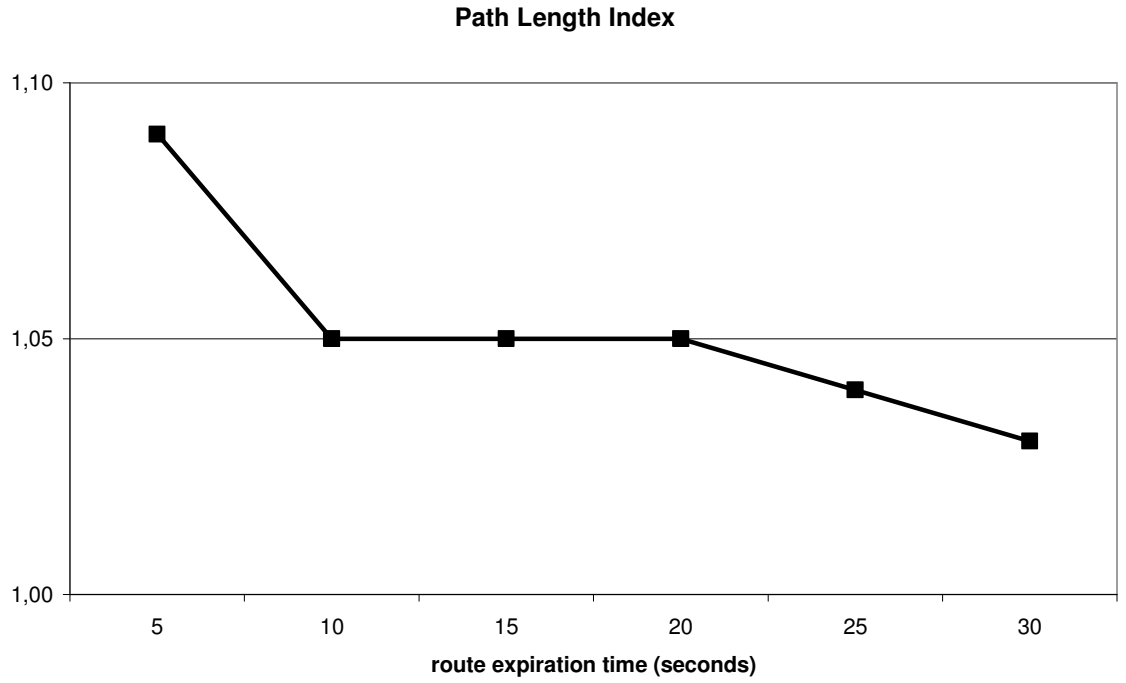

Figure 35 – Path length index given the route expiration time

The natural question at this point would be why more traffic will cause longer paths. A first attempt to answer this question would be to associate it with higher contention which would produce more collisions. But the loss of PREQs cannot account for the forming of multihop paths alone. It is not difficult to see that if a PREQ is completely lost, i.e. not decoded by any participating node it will not contribute for the forming of multihop paths.

A common misunderstanding about dense wireless environments is that every node will capture the same traffic simply because they are very close and share a common transmission area. If this was the case, either a frame would be received by all the nodes or a collision would render a frame useless to all of them. But this is not a correct assumption.

- Obstructions and reflections and the resulting multipath effect change the probability that a frame is successfully decoded by a node. Even nodes some centimeters apart will experience different signal levels.

- The internal condition of a node (from the signal levels to the protocol stack) may cause a frame that was successfully decoded by a nearby neighbor to be lost.
- Slight differences in the antenna gain and in the radio sensitivity between two neighbors (to name only two items) are normal and may also explain some differences.
- And, more interestingly, even if none of the above held true, and we had this perfect scenario (no obstructions or reflections, no differences in the devices or in the internal states), two nodes would still not always successfully decode the same subset of the transmitted frames because of the capture effect [52].

In order to prove that not all nodes will decode the same set of frames, it is enough to compare traffic captures from distinct nodes and check for differences. In the following experiment, 4 XOs (B,D,G,I) captured traffic while 2 other XOs (A,J) exchanged UDP datagrams. Each node transmitted 100 datagrams to a multicast address. So, ideally, each of the four monitoring stations would capture 400 UDP frames (100 from A, 100 from J, the 100 from J retransmitted by A, and the 100 from A retransmitted by J). The results are shown in Table 10.

Table 10 – Number of captured frames per node

NODE	fromA	from J	retransmitted by J	retransmitted by A
A	-	97	-	-
B	98	100	98	97
D	98	100	98	97
G	98	98	97	97
I	98	100	97	97
J	98	-	-	-

The fact that none of the monitoring stations captured all the 400 transmitted frames is not actually important for this experiment. It is the fact that nodes G and I

captured less frames than nodes B and D though they were placed on the same table and, moreover, the fact that they captured a different subset of the transmitted datagrams that is relevant. Both G and I captured 97 frames retransmitted by J, but only 96 frames in common – a slight difference that nevertheless proves our point.

In other words, we are not interested in proving that some frames will be lost (and they will), but that frames may be lost to *some* of the nodes in our dense environment. This *difference in perspective* will explain, at least in part, the multihop paths. In order to prove that this difference will happen irrespective of traffic class and will also affect the path discovery mechanism, another experiment was performed.

This time nodes A and J exchanged traffic while E and F captured the frames. Table 11 shows the difference of the traffic captured by E and F, paying special attention to PDM traffic.

Table 11 – Comparison on the number of captured frames

Monitoring station	Total frames	from A	to A	PREQs from A	PREQs from A (54Mbps)	from J	to J	PREQs from J	PREQs from J (54Mbps)
E	13,618	1678	469	145	29	1790	430	145	36
F	13,496	1687	466	150	32	1681	428	140	36

Table 11 shows that it is possible that a PREQ is successfully decoded by a group of nodes, but not by other nodes like, for instance, the intended destination of the PREQ and Table 12 proves that this happens irrespective of the modulation technique.

Table 12 – Captured PREQs by transmission rate

Monitoring station	PREQs	PREPs	54	36	11	1
E	7334	326	4525	1741	1947	2992
F	7298	328	4484	1735	1923	2978

Note that although F decoded 0.9% less frames than E, that does not mean that it will perform worse in respect of every frame. Node F actually captured a little more of the PREPs – 328 against 326 captured by E.

Also interesting is the fact that the frames transmitted at the more robust data rate of 1Mbps (like the PREQs in Table 12) or 2Mbps (like the UDP datagrams in Table 10) are not free of this effect either. We have no reason to believe that the differences in the set of decoded frames are dependent on data rate but neither have we stated that they are not. This is just irrelevant for the current analysis.

A next step would be to associate the fact that not all nodes will decode the same frames with the forming of multihop paths. We may start by proposing a scenario:

Node "A" broadcasts a path request at 54Mbps to find a path to node "J". Node "J" does not decode this frame, but some other nodes around, say node "C", do. Some milliseconds before that, node "C" will rebroadcast the PREQ from "A", and this time, "J" may listen and respond with a PREP. The result is a two-hop path, where a single hop would be preferred.

Note that it doesn't make any difference if B receives or not any of the other PREQs sent from A (at the datarates of 36, 11 or 1Mbps). And this is where the other contributing factor for the multihop paths enters – the costs associated to each PREQ. Particularly the fact that two PREQs transmitted at 54Mbps (second line of Table 13) will beat one 36Mbps (third line of Table 13).

Table 13 – Lowest cost paths with current metrics

Total cost of the path	**Data rates of the PREQs**
13	54
26	54,54
28	36
39	54,54,54
41	54,36
46	11
52	54,54,54,54
54	54,54,36
56	36,36
59	54,11
64	1
65	54,54,54,54,54

We can conclude that the fact that not all nodes decode the same subset of frames plus the way the path costs are currently calculated is one cause for the multihop paths observed. This brings us to the question of whether these multihop paths can be eliminated by adjusting the costs associated to each PREQ.

Before we try to answer this, we should acknowledge that even if we used hop count as a metric the fact that nodes do not decode the same subset of the transmitted frames is alone a cause for multihop paths. In fact, this is the very reason that we need multihop paths. Because of that the answer to the above question is no, changing metrics will never completely eliminate the issue as we will prove shortly. If fixing is not possible, how could they be reduced?

Let's give a more detailed look at the scenario we just proposed in which “A” wants to discover a path to “J”. In that example, a two multihop path formed because:

- The target "J" did not decode the 54Mbps PREQ (with cost 13) from "A"
- Node "A" won't retransmit a 54Mbps PREQ, unless the whole PDM cycle fails. Instead it will send a cluster of PREQs back-to-back (consisting of three more PREQs, at 36, 11 and 1Mbps).
- Node "C", an intermediary node, decoded this original PREQ from "A" and retransmitted it in a new cluster (the 54Mbps PREQ on this new cluster will have cost 26).
- Though "J" may have decoded the 36Mbps PREQ from "A", its associated cost is 28 so, when it receives the PREQ from "C" with the cost 26, the two-hop path (via "C") will be selected.

If the cost associated to the 36Mbps PREQ sent from "A" and received by "J" was smaller than 26, that particular multihop path wouldn't form. So, it is possible to adjust metrics to avoid at least this case. In the next analysis, new costs will be proposed to each of the PREQs in the cluster.

On the other hand, in order to prove that changing the costs does not suffice to eliminate the multihop paths in a dense mesh, we performed another experiment in which

we switched our metrics to hop count, by setting all costs in all PREQs to 1 (any value would cause the same effect as long as it was configured in all PREQs).

As the Table 14 shows, switching to a hop count (by making all the costs associated to datarates the same) reduces the average Path Length Index: it dropped from 1.05 to 1.02 in comparison to Table 9, and the percentage of frames transmitted over a multihop path reduced from 9.99 to 4.77%. This confirms that even a hop count metric will not completely eliminate multihop paths and it is also an indicator that the way the metrics are calculated is in part causing the multihop paths.

Table 14 – Path Length Index with Hop Count metric

Node	Mean PLI	Standard deviation
A	1.03	0.07
B	1.02	0.04
C	1.03	0.05
D	1.03	0.05
E	1.01	0.03
F	1.04	0.06
G	1.02	0.05
H	1.01	0.02
I	1.03	0.05
J	1.02	0.04

We conclude that it is true that multihop paths form where you wouldn't expect them – in dense mesh clouds. We have reason to believe that the more nodes you add to the mesh the longer the paths will get. We identified at least one cause for this, and this cause is composed of two contributing factors: (1) the way that link costs are calculated and (2) the fact that not all nodes will decode the same frames (even when they are quite close to each other).

We also found out that this phenomenon cannot be completely eliminated, but we identified two ways to reduce its occurrence. The first is related to the costs in the PREQs. In the next section we will propose new static values for the metrics. Another possible approach would be to detect the dense mesh scenario and adapt to it, by turning,

for instance to a hopcount metrics. An adaptive approach to a dense mesh will be studied in Analysis 4.

The second approach that has an impact on the path discovery mechanism is the reduction of the control traffic caused, for instance, by the increase of the route expiration time, as suggested in previous analysis, or by the reduction of the PREQ cluster.

6.4.3. Metric costs

This section is dedicated to a proposal of new values for the costs associated to each frame in the PREQ cluster. The proposal tries to be the *simplest* possible, thus:

- It does not change the current algorithm
- It does not change the current cluster of PREQs (54,36,11,1Mbps)

In short, the idea is to propose a starting point based on the current mechanism implemented in the XO – static costs, depending solely on the PREQ rate. So, the challenge would be to find optimal costs to associate to each PREQ in the cluster. Current values can be found in Table 5. For the current analysis the following notation will be used:

- C_{54} = The cost associated to a 54Mbps PREQ
- Cs = $\{C_{54}, C_{36}, C_{11}, C_1\}$

Even in a simple approach like that, it is necessary to decide what we are optimizing the costs for. If we were solely worried with the airtime consumption, it would suffice to use the time necessary for transmitting each PREQ and use this as metric. That would mean for example that $C_1 \sim 50 * C_{54}$. But this relation actually depends on the frame size you want to optimize for, since for 100 bytes frames $C_1 = 24 * C_{54}$ while for 1,500 bytes frames $C_1 = 49 * C_{54}$.

If, on the other hand, we would like to take the number of hops into consideration, and use the airtime only to select between paths of the same hop length, we could choose

values where $C_{54} >> (C_1 - C_{54})$, for example Cs == {1013, 1028, 1042, 1064}. In this case, hop count would clearly prevail over the data rates on PREQs.

Clearly none of the approaches are good enough. Keeping the medium free is important, but saving queue space and reducing frame loss due to excessive relays is also needed. The current values on the other hand tend to form multihop paths as shown in Table 15(a). The relative costs between the 54Mbps and the 36Mbps PREQs also do not seem ideal, since there is no clear reason why two 54Mbps links should be preferred over a single 36Mbps link. Since the latter consumes less airtime and also no resource from intermediary nodes.

This proposal suggests optimizing for latency. Not that keeping RTT low is more important in itself, but because latency is one thing that will take into account:

- The transmission time and hence airtime
- The processing time for the frame when it is relayed by an intermediary node.

Given:

- S: Frame size in bytes
- R: Data rate in Mbps
- T: Air time in microseconds (μs)
- I: Time necessary for an intermediary node to relay the frame

And calculating air time consumption with Equation 3 and Equation 4 gives us Equation 5

```
Cs = {(T + 26 + (S*8)/54), (T + 26 + (S*8)/36),
(T + 192 + (S*8)/11), (T + 192 + (S*8))}
```

Equation 5

A simple experiment was conducted to determine the value of T. The idea was to compare the time necessary to transmit a frame over a single hop path to the time necessary to transmit the same frame over a multihop path (2 hops and 3 hops).

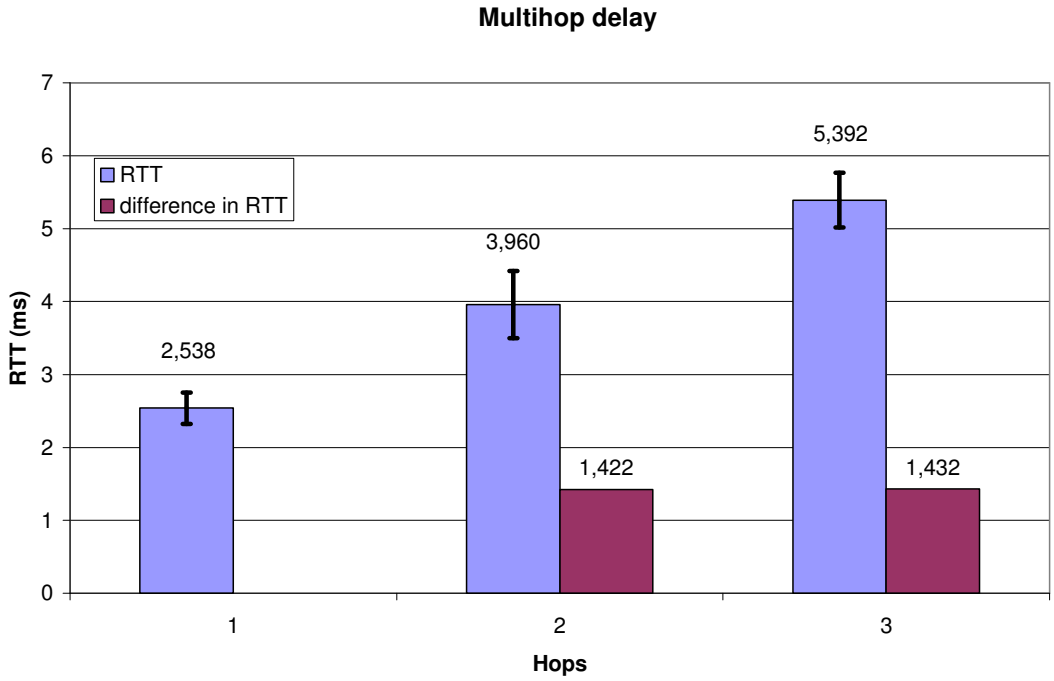

Figure 36 – Additional delay posed by mutihop forwarding on ping datagrams

Figure 36 suggests that the additional delay posed by multihop paths is of the order of 1.427ms and this value will be used from now on.

The size of the frame (S) is a more arbitrary choice. Mean traffic size is very difficult to determine since traffic patterns are totally dependent of the applications selected by the user, and the XO is a general use communication platform, where all kinds of traffic are to be expected. One option would be to choose frame sizes for collaboration software currently running on the XO. For instance, the chat activity is supported by multicast frames of about 460bytes, which is also a good midterm between the control traffic frames (about 100 bytes) and the data frames of applications that transmit large chunks of data, like a file transfer or web surfing (1,500 bytes).

It is always reasonable to consider that the 1,500 byte frames will typically dominate the traffic and adjust the metrics to this frame size. We will show results for both sizes.

For T = 1,427 μs and S=460 bytes we have:

Cs = {1521, 1555, 1954, 5299}

For T = 1,427 μs and S=1,500 bytes we have:

Cs = {1675, 1786, 2710, 13619}

Table 15 shows the resulting costs optimizations with frame sizes of 460 and 1,500 bytes compared to the currently implemented values (a). The proposed metrics (both) are clearly less prone to the formation of multihop frames, because it adds a cost component based on the time needed for the intermediary node to handle the frame.

The differences between both proposals (b) and (c) are not big. For larger packets, the transmission rates are more important to the calculations and a 1Mbps link will be used with less probability. Other than that both proposals are more similar to a hop count metrics than the current implementation.

Table 15 – Current values and proposed values for A=1

(a) Current values		**(b) T=1427; S=460; A=1**		**(c) S=1427; S=1500; A=1**	
total cost	**data rates for links**	**total cost**	**data rates for links**	**total cost**	**datar ates for links**
13	54	1521	54	1675	54
26	54,54	1555	36	1786	36
28	36	1954	11	2710	11
39	54,54,54	3042	54,54	3350	54,54
41	54,36	3076	54,36	3461	54,36
46	11	3110	36,36	3572	36,36
52	54,54,54,54	3475	54,11	4385	54,11
54	54,54,36	3509	36,11	4496	36,11
56	36,36	3908	11,11	5025	54,54,54
59	54,11	4563	54,54,54	5136	54,54,36
64	1	4597	54,54,36	5247	54,36,36
65	54,54,54,54,54	4631	54,36,36	5358	36,36,36
67	54,54,54,36	4665	36,36,36	5420	11,11
69	54,36,36	4996	54,54,11	6060	54,54,11
72	54,54,11	5030	54,36,11	6171	54,36,11

Until now, since we are basing the metrics on latency measurements, we have been implying that the airtime and relaying-time are equally important, but we could introduce an adjustment term *A*. So that:

```
Cs = {(A*T + 26 + (S*8)/54), (A*T + 26 + (S*8)/36), (A*T + 192 + (S*8)/11), (A*T + 192 + (S*8))}
```

If we think that the processing time is only half as important as the airtime, *A* would be 1/2. Therefore *A* expresses the relative importance between the processing time and the airtime. The *A* factor could also be used as a means to account for other metrics, in case the costs were calculated dynamically instead of statically as it is currently. One such metric could be the current condition of the battery or the average length of the transmission queue. In this scenario, A=1 would be a subset of all possibilities and only in that case we could consider our metrics based on latency.

Table 16 – Comparison of resulting costs for A=0.5

(a) **Current values**		**(b)** **T=1427; S=460; A=0.5**		**(c)** **T=1427; S=1500; A=0.5**	
total cost	**data rates for links**	**total cost**	**data rates for links**	**total cost**	**data rates for links**
13	54	808	54	962	54
26	54,54	842	36	1073	36
28	36	1241	11	1924	54,54
39	54,54,54	1616	54,54	1997	11
41	54,36	1650	54,36	2035	54,36
46	11	1684	36,36	2146	36,36
52	54,54,54,54	2049	54,11	2886	54,54,54
54	54,54,36	2083	36,11	2959	54,11
56	36,36	2424	54,54,54	2997	54,54,36
59	54,11	2458	54,54,36	3070	36,11
64	1	2482	11,11	3108	54,36,36
65	54,54,54,54,54	2492	54,36,36	3219	36,36,36
67	54,54,54,36	2526	36,36,36	3848	54,54,54,54
69	54,36,36	2857	54,54,11	3921	54,54,11
72	54,54,11	2891	54,36,11	3959	54,54,54,36

Table 16 shows that with A=0.5 we have a result more similar to the current metrics in that multihop paths are more likely to be formed, as we would expect, since *A* is associated to the cost of adding hops.This proposal has some advantages:

- It is simple to implement and does not require any change in current software (firmware and driver) other than setting the values for C_{54} ... C_1
- It allows the counterbalance of two important aspects: medium efficiency and node resource.

Choosing the proper value for *A* is probably the most important decision on this scheme. Also choosing a value for frame size (S) has some impact also. As we have seen, 460 bytes is a midterm between the small management frames (about 100 bytes) and the big data frames (1500 bytes, typically). It is also the typical size of a chat frame. More importantly, choosing proper static values is just a starting point for the proposal of an adaptive metrics – as we will present in our conclusions.

6.4.4. The PDM Analysis Tool

The same way we identified that the airtime tool would be very important to measure progress in the goal of saving airtime, a similar need was identified in terms of the Path Discovery Mechanism traffic. In order to allow improvements on the XO implementation to speed up, another python tool was created.

The PDM-Analysis tool provides a report based on the observed (captured by a monitoring station) PDM traffic and outputs a report like the following:

```
------------------------------------------------------------------------
ROUTE DISCOVERY REPORT (for ../route_exp/captures/10s-mcastping-1.pcap)
------------------------------------------------------------------------
Captured route discovery frames:
 14152 [RREQs 12936 (91.41%) RREPs 1141 (8.06%) RERRs 75 (0.53%)]
Route discovery clusters: 852
Average # of frames captured per cluster:
 16.52 (stdev = 11.57) [RREQs 15.18, RREPs 1.34, RERRs 0.09]
Replied requisitions (with any rrep captured): 509 (59.00%)
Completed requisitions (with final rrep captured): 158 (18.00%)

RREQs:
```

```
By Hop Count:
 1: 2165 (16.74%) 2: 4449 (34.39%) 3: 3354 (25.93%) 4: 2280 (17.63%) 5: 688 (5.32%)
By Datarate:
 54: 2374 (18.35%) 36: 2470 (19.09%) 11: 3991 (30.85%) 1: 4101 (31.70%) Unknown: 0

RREPs:
By Hop Count: 1: 93 (8.15%) 2: 362 (31.73%) 3: 478 (41.89%) 4: 180 (15.78%) 5: 28 (2.45%)
Estimated % of multihop paths: 34.26%

First RREP elapsed time (ms): 62.27 (stdev = 35.94) MIN: 14.87 MAX: 183.31
Route acquisition time (ms): 80.74 (stdev = 50.73) MIN 14.87 MAX 263.99
Route discovery lifespan (ms): 94.71 (stdev = 57.99) MAX 263.99
RREQ distribution for 1 hop(s) (ms): mean = 11.02 (stdev = 24.53) MIN: 0.00 MAX: 189.35
RREQ distribution for 2 hop(s) (ms): mean = 34.07 (stdev = 38.90) MIN: 1.69 MAX: 232.48
RREQ distribution for 3 hop(s) (ms): mean = 52.19 (stdev = 40.12) MIN: 4.94 MAX: 227.01
RREQ distribution for 4 hop(s) (ms): mean = 64.19 (stdev = 39.30) MIN: 10.75 MAX: 194.59
RREQ distribution for 5 hop(s) (ms): mean = 82.54 (stdev = 42.35) MIN: 18.09 MAX: 189.81

PDM Airtime (ms): 5778.6543
PDM Activity Interval (s): 68.53
PDM Airtime %: 8.43

Participants: 10

00:17:c4:05:2c:ae: FWD 1518           ORIGIN 65 => 1309              TARGET 65 => 1199
00:17:c4:05:23:3a: FWD 1475           ORIGIN 61 => 1217              TARGET 83 => 1325
00:17:c4:0c:e8:eb: FWD 1307           ORIGIN 145 => 2205             TARGET 82 => 1463
00:17:c4:02:2f:59: FWD 1325           ORIGIN 127 => 1823             TARGET 91 => 1711
00:17:c4:05:27:98: FWD 1691           ORIGIN 149 => 1968             TARGET 78 => 1351
00:17:c4:05:23:02: FWD 1080           ORIGIN 43 => 874               TARGET 109 => 1615
00:17:c4:03:57:1b: FWD 1357           ORIGIN 60 => 1006              TARGET 81 => 1361
00:17:c4:05:2a:79: FWD 1525           ORIGIN 68 => 1217              TARGET 98 => 1701
00:17:c4:03:56:e0: FWD 1477           ORIGIN 67 => 1264              TARGET 83 => 1121
00:17:c4:05:27:a3: FWD 1397           ORIGIN 67 => 1194              TARGET 82 => 1230
```

We briefly describe each section of this report:

The header informs the name of the file used to generate the report

```
---------------------------------------------------------------------
ROUTE DISCOVERY REPORT (for ../route_exp/captures/10s-mcastping-1.pcap)
---------------------------------------------------------------------
```

GENERAL INFORMATION SECTION: The total number of PDM frames captured [by type]:

```
Captured route discovery frames:
 14152 [RREQs 12936 (91.41%) RREPs 1141 (8.06%) RERRs 75 (0.53%)]
```

The number of times the PDM was started and the average composition of a cluster:

```
Route discovery clusters: 852
Average # of frames captured per cluster: 16.52 (stdev = 11.57) [RREQs 15.18, RREPs 1.34, RERRs 0.09]
```

For how many of the clusters at least one RREP was captured, and for how many a final rrep (i.e. one with the transmission address of the originator of the cluster):

```
Replied requisitions (with any rrep captured): 509 (59.00%)
Completed requisitions (with final rrep captured): 158 (18.00%)
```

Note: The above, of course, does not mean that only 18% of the path discovery attempts were complete, but it can be used to compare the results between two different experiments (an increase in this number is an indicator of higher efficacy)

ROUTE REQUESTS SECTION:

```
RREQs:
```

A RREQ cluster will be propagated through the mesh cloud in many hops (in this case, an interesting topology was forced with Blinding Table):

```
By Hop Count: 1: 2165 (16.74%) 2: 4449 (34.39%) 3: 3354 (25.93%) 4: 2280 (17.63%) 5: 688 (5.32%)
By Datarate: 54: 2374 (18.35%) 36: 2470 (19.09%) 11: 3991 (30.85%) 1: 4101 (31.70%) Unknown: 0 (0.00%)

Note: The position of the monitoring station should be take into account. If placed far away from the observed traffic lower rates will dominate.
```

ROUTE RESPONSES SECTION:

```
RREPs:
```

This gives a good insight on the typical path length (in this case, an interesting topology was forced with Blinding Table):

```
By Hop Count: 1: 93 (8.15%) 2: 362 (31.73%) 3: 478 (41.89%) 4: 180 (15.78%) 5: 28 (2.45%)
Estimated % of multihop paths: 34.26%
```

TIME LINE SECTION:

Time necessary for the first RREP to each cluster to be transmitted.

```
First RREP elapsed time (ms): 62.27 (stdev = 35.94) MIN: 14.87 MAX: 183.31
```

Time necessary for the last RREP to each cluster to be transmitted.

```
Route acquisition time (ms): 80.74 (stdev = 50.73) MIN 14.87 MAX 263.99
```

PDM traffic keeps being transmitted even after the path was discovered (because it floods the mesh)

```
Route discovery lifespan (ms): 94.71 (stdev = 57.99) MAX 263.99
```

The RREQs by hop count in a time line.

```
RREQ distribution for 1 hop(s) (ms): mean = 11.02 (stdev = 24.53) MIN: 0.00 MAX: 189.35
RREQ distribution for 2 hop(s) (ms): mean = 34.07 (stdev = 38.90) MIN: 1.69 MAX: 232.48
RREQ distribution for 3 hop(s) (ms): mean = 52.19 (stdev = 40.12) MIN: 4.94 MAX: 227.01
RREQ distribution for 4 hop(s) (ms): mean = 64.19 (stdev = 39.30) MIN: 10.75 MAX: 194.59
RREQ distribution for 5 hop(s) (ms): mean = 82.54 (stdev = 42.35) MIN: 18.09 MAX: 189.81
```

PDM AIRTIME STATISTICS:

Total airtime consumed by all PDM frame in the input file

```
PDM Airtime (ms): 5778.6543
```

Interval between the first and the last PDM frames in the input file

```
PDM Activity Interval (s): 68.53
```

The percentage of the airtime consumed by the PDM traffic (in the interval it was detected)

```
PDM Airtime %: 8.43
```

PARTICIPANTS SECTION:

```
Participants: 10
```

The number of PDM frames forwarded (FWD) by each node. The number of PREQ clusters originated by each node (ORIGIN) and how many frames were triggered by the original RREQs. The number of PREQ clusters destined (targeted at) the node and how many frames were triggered by the original RREQs.

```
00:17:c4:05:2c:ae: FWD 1518          ORIGIN 65 => 1309            TARGET 65 => 1199
00:17:c4:05:23:3a: FWD 1475          ORIGIN 61 => 1217            TARGET 83 => 1325
00:17:c4:0c:e8:eb: FWD 1307          ORIGIN 145 => 2205           TARGET 82 => 1463
00:17:c4:02:2f:59: FWD 1325          ORIGIN 127 => 1823           TARGET 91 => 1711
00:17:c4:05:27:98: FWD 1691          ORIGIN 149 => 1968           TARGET 78 => 1351
```

```
00:17:c4:05:23:02: FWD 1080          ORIGIN 43 => 874              TARGET 109 => 1615
00:17:c4:03:57:1b: FWD 1357          ORIGIN 60 => 1006             TARGET 81 => 1361
00:17:c4:05:2a:79: FWD 1525          ORIGIN 68 => 1217             TARGET 98 => 1701
00:17:c4:03:56:e0: FWD 1477          ORIGIN 67 => 1264             TARGET 83 => 1121
00:17:c4:05:27:a3: FWD 1397          ORIGIN 67 => 1194             TARGET 82 => 1230
```

6.5. ANALYSIS 3 – WIRELESS CONGESTION DETECTION AND ADAPTATION

6.5.1. Contention Window Size

In a mesh cloud formed by XOs, each node will contend for the medium according to the rules of the Distributed Coordination Function (DCF) described by 802.11. The DCF, as described in Chapter 2, dictates that when a node that wants to transmit senses the medium to be busy it must wait a random time before attempting to transmit. This random time is calculated according to an exponential backoff mechanism.

A big contention window will reduce the probability of two nodes transmitting at the same time and consequently colliding. On the other hand it will increase latency and reduce the efficiency in the use of the spectrum, since there is a higher probability of wasting time slots.

Our tests showed a high rate of retransmissions of unicast traffic when many nodes try to transmit simultaneously. In a mesh network there are many examples of traffic that triggers simultaneous retransmissions from all nodes and these may fall into two categories:

(1) Simultaneous retransmissions. A node broadcasts a frame that should reach all other nodes in the mesh cloud (a Network-Wide Broadcast). One example of such a frame is the Path Request (PREQ) sent to start a Path Discovery. In general all broadcast frames will trigger this behavior.

(2) Simultaneous responses. A node sends a broadcast or even a multicast frame that should be responded by a large number of nodes. An example is the Probe Request

frame, that will be responded (Probe Response) at about the same time by all listening stations.

Simultaneous transmissions may pose scalability issues even to a network that is not congested, because of the high probability of collision directly related to *N* (the number of nodes). Traffic burstiness may easily reduce the reliability of protocols, particularly of those that rely on multicast/broadcast transmissions (and hence, are more susceptible to causing bursts).

A contending station that has frames not related to the burst to send – for instance, part of an ongoing TCP transmission – will have a reduced chance to do it during this period, since it will have to compete with *N* stations. If this station happens to sort a big contention window size it will probably not transmit its frame and report this to TCP, which will react by decreasing the throughput. In short, because of the burst, caused for instance by the mere transmission of a probe request, a TCP connection may interpret a rather idle spectrum as congested.

Increasing the retry limit, which is typically four for small frames and seven for large frames, could improve the chances of transmitting a unicast frame at the cost of throughput, since backlogged traffic would have to wait longer on buffer and also because this would increase latency in general and recovery times for broken paths. Moreover, this would have no impact on the probability of collision and no effect on multicast/broadcast transmissions which are not retransmitted.

The probability that two nodes choose the same time slot to transmit, and collide, is a function of the number of contending nodes and also of the contention window size. To prove the correlation between the observed high rates of retransmissions and the contention window parameters, a comparative test was performed.

A baseline was established in a test where ten nodes simultaneously sent icmp echo requests to a multicast address, triggering simultaneous responses from all other nodes. In this case, the contention window parameters were kept at the default values – CWmin = 7 and CWmax = 31. These values were changed to CWmin=31 and CWmax=1024 and the tests were repeated. The results are shown in Figure 37.

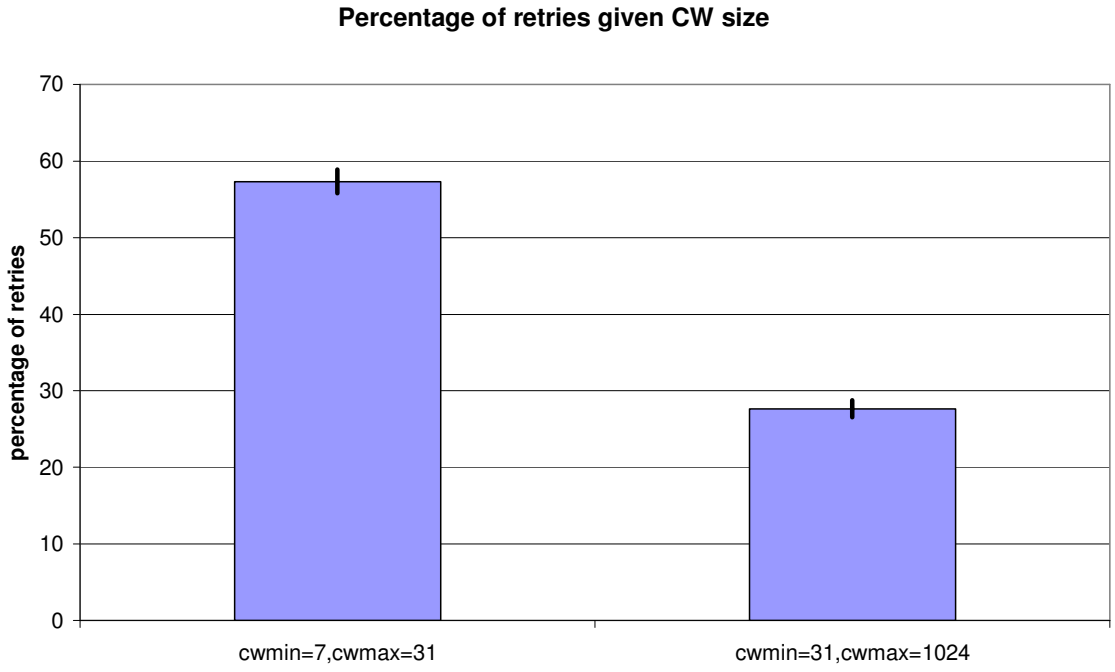

Figure 37 – The percentage of retries in icmp echo responses (with error bars) triggered by ten nodes pinging the multicast address (224.0.0.1).

Based on these results Marvell implemented an adaptive contention window that tries to adjust CWmin and CWmax to the number of nodes in the mesh which are detected by listening to the number of Beacons and the number of Path Responses from other neighbors. Unfortunately, the details of this implementation were not disclosed. Nevertheless the implementation was tested and presented good results, reducing the number of retries to 26% on the same multicast ping test.

Although the mechanism reduced the number of retransmissions in a mesh cloud it falls short at least in two aspects:

(1) It does not account for stations in infra-structure mode (XOs or not). Non-XO stations will be a source of contending traffic as XOs associated to access points (that will not start Path Discoveries)

(2) It is based on counting frames that can be configured to be less or more frequent. Beacons frequency can be changed or they can even be disabled, as shown in Analysis 1, and also the Route Expiration Time can be

changed, as explained in Analysis 2, resulting in a decrease of Path Response traffic.

6.5.2. Rate adaptation logic

Another important adaptive behavior of a wireless station relates to the transmission rate it uses to send frames to a given neighbor. Most devices will try to use the highest rate possible, i.e. the one by which it can successfully deliver the frame to its destination. The higher the data rate the more susceptible to interference it is and, because of that, distance and high transmission rate are opposing goals.

Due to limited memory resources the XO implements a simple Frame Error Rate algorithm that lowers the transmission rate if a given number of consecutive frames are lost and, likewise, raises that rate if a given number of consecutive frames are successfully sent.

The problem with that scheme lies exactly on the idea of reacting to a failed transmission. For 802.11 every unicast frame must be acknowledge and if a transmitting node does not receive the correspondent ack, it will consider the frame lost and retransmit it. But this method draws no distinction between a frame that was lost due to poor link quality and a frame that was lost due to a congested scenario, where both the frame and the ack have a higher probability of colliding.

When a node reacts to a congested scenario by lowering the transmission range it is actually making things even worse. The following experiment was designed to demonstrate that this is exactly the case for the XO rate adaptation logic.

In this case, three nodes are used. The first, “A” pings the second “B” while a contending traffic generated by “C” is switched on and off. This contending traffic is an UDP flow that will vary its throughput throughout the experiment. We repeat the experiment with the rate adaptation logic off and the transmission rates fixed at 54Mbps.

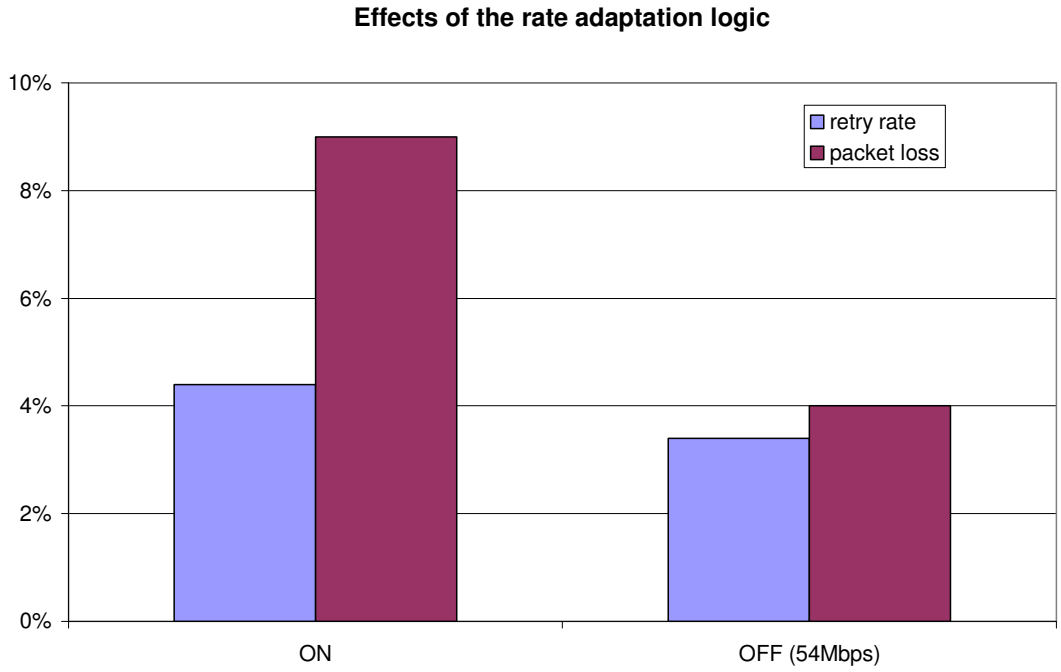

Figure 38 – Packet loss and retry rate for the rate adaptation logic enabled (ON) and disabled (OFF – with the rate fixed at 54Mbps)

As expected results with the rate adaptation logic off were clearly superior, with the rate of retries dropping from 9% to 4% and the delivery rate improving 1% (Figure 38). Figure 39 shows how the rate adaptation logic can increase airtime consumption by decreasing transmission rates. In this case, the contending traffic on node "C" was activated after 8 seconds and, at about 19 seconds, collisions led the rate adaptation mechanism to decrease rates, which caused the harmful feedback (collision causes the lowering of rates that causes more collisions) peaking to more than 2% of airtime consumption – twenty times the normal.

Similar results are shown in Figure 40 where the increase in retransmissions result in peaks of latency that can reach up to 400ms, against less than 3ms for a transmission that is successful at the first attempt.

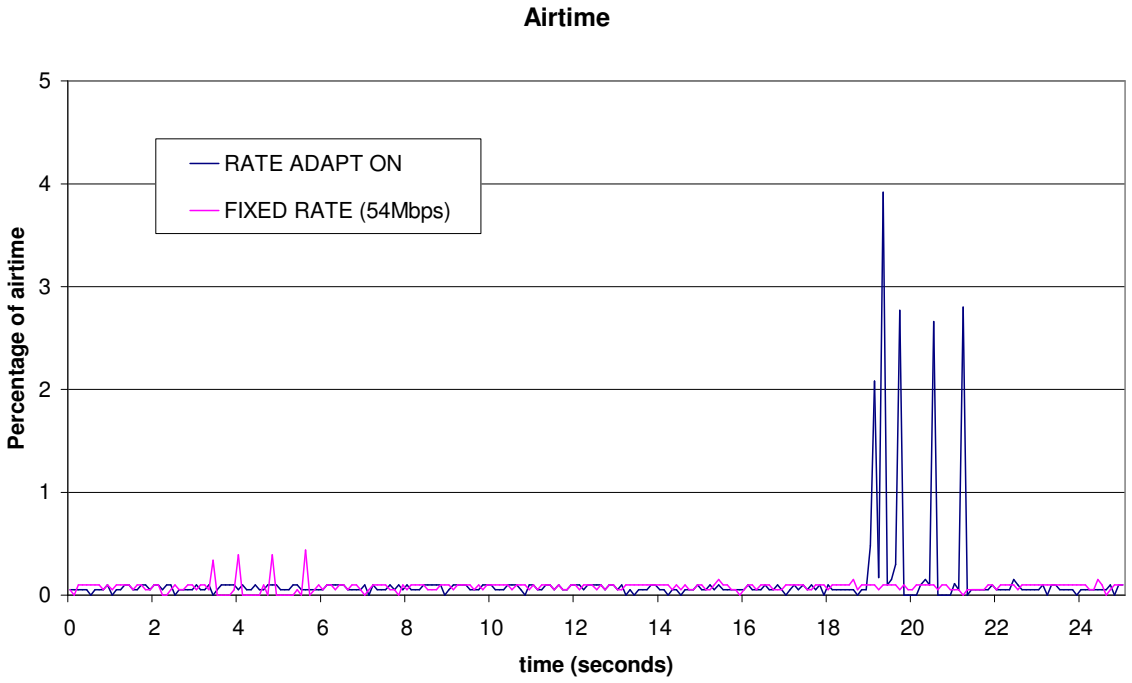

Figure 39 – Peaks in airtime consumption of the icmp traffic due to the rate adaptation logic.

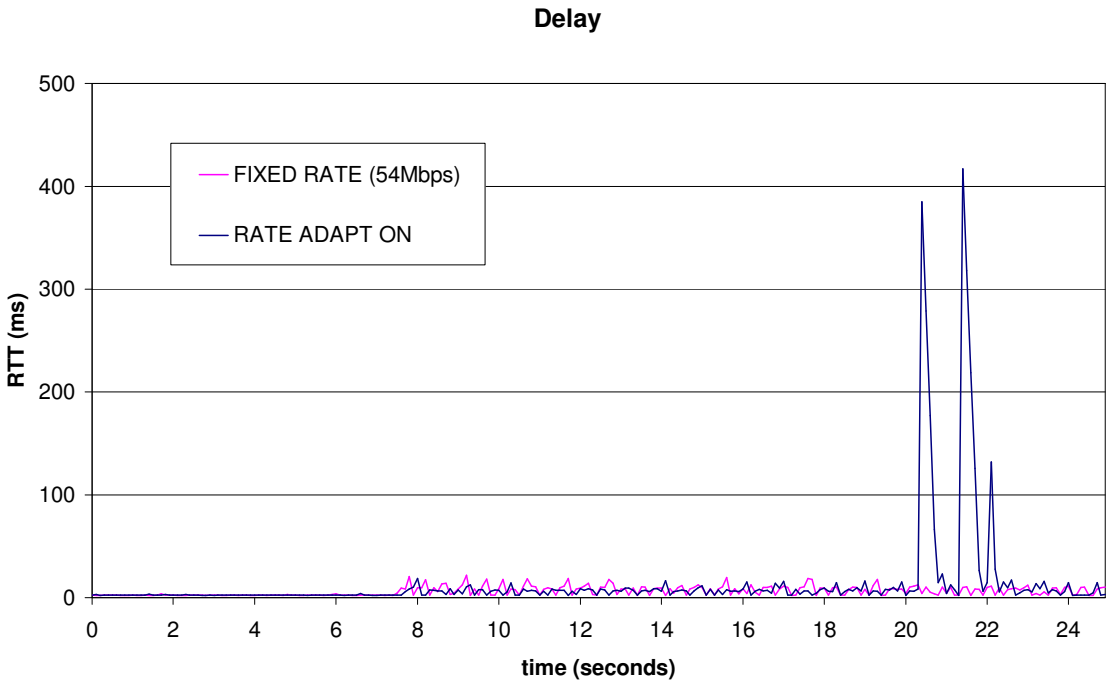

Figure 40 – Peaks in RTT for the icmp traffic due to the adaptation logic.

Figure 41 shows the effects on airtime consumed by the ping traffic, as the CBR is switched on and off with different intervals between datagrams (100ms, 50ms and 10ms) and, once again, shows that whenever the contending source reaches a certain throughput, the collisions will cause the rate adaptation to react. The consolidation interval for this graph is one second, and this explains why the peaks are lower than the ones presented in Figure 39 where the consolidation interval was 100 milliseconds.

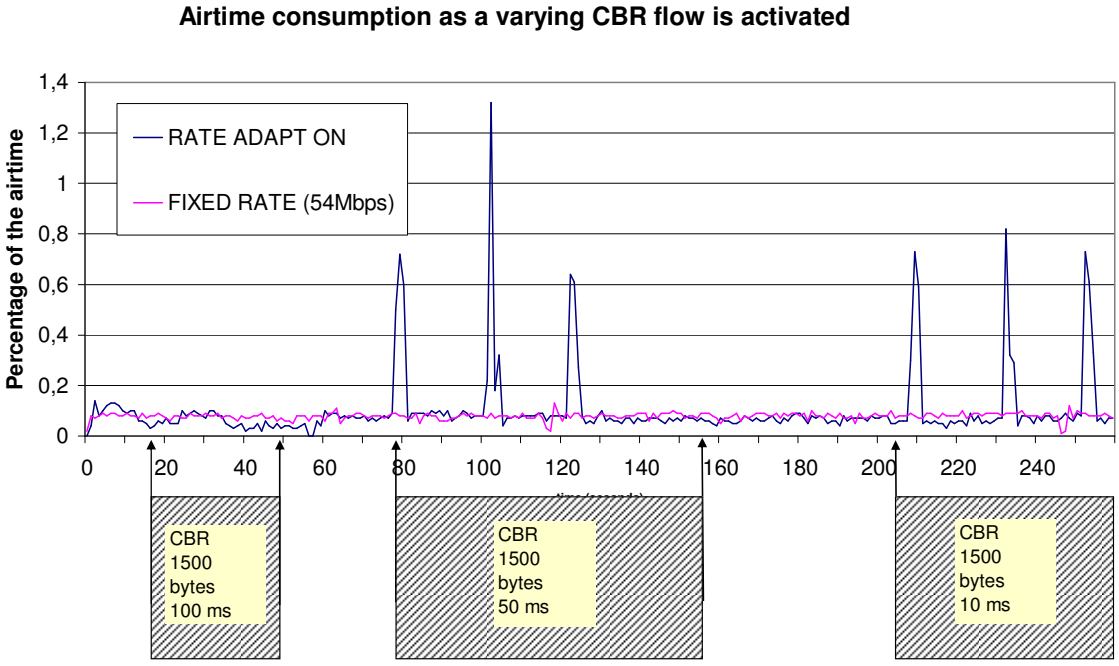

Figure 41 – As the CBR is switched on and off the rate adaptation logic reacts wrongly, potentially congesting the network.

6.6. ANALYSES CONSOLIDATION

Before we move to the next Chapter and write the recommendations that resulted from the analyzes above, we summarize them, with their motivation and main conclusions in Table 17.

Table 17 – The analyzes performed and what can be concluded from them

	Analyses	Investigates	Concludes
1 - Airtime consumption	Beacon Backoff	If a increase in the number of nodes in the XO mesh would result in beacons taking away all the available airtime	The backoff mechanism works and there is no risk of rendering a XO-Mesh useless because of the beacon traffic. However, the frequency of beacons can be reduced, saving airtime
	Probe Request / Probe Responses	If Probe Request and Responses can generate excessive traffic and harm scalability and reliability	Because of the bursty nature of the probe responses, there is indeed negative impacts of this category of traffic, that should be reduced (by decreasing the number of retries for the probe responses, or eliminated)
2 – Path Discovery Mechanism	Route Expiration Time	If the route expiration time parameter is set in a sound way, finding the balance that will avoid, in one extreme, the freezing of the network paths into suboptimal states or, in the other extreme, producing excessive control traffic.	The current 10 second time is small and produces significant control traffic. Increasing this value may free invaluable airtime without limiting node mobility or accommodating for the changing of spectrum conditions.
	Path Discovery Mechanism Sanity	If the PDM is indeed resulting in unnecessary lengthy paths and, in this case, its causes and possible counter-measures.	There is indeed the forming of multihop paths where single hop paths would be expected. Though these long paths can not be completely eliminated (due to the fact that not every node will receive the same set of frames) they can be reduced, by reducing the PDM traffic overhead and by changing the costs associated with each PREQ in the PREQ cluster.
	Metric costs	If the costs associated to each PREQ in the PREQ cluster are well selected.	Even if a non-adaptive (static) value are to be chosen, the relation between the costs associated with the 54Mbps PREQ and the 36Mbps PREQ are introducing unnecessary long paths (as seen in the previous analysis). We suggest that new costs should be tried, and propose a logic to derive such costs.
3 – Wireless Congestion detection and adaptation	Contention Window size	If configured CW sizes are optimal	Setting such small values for the minimum (7) and maximum (31) size of the CW parameters are producing a lot of collisions and should be avoided in favor of standardized values (31 and 1023, respectively).
	Rate adaptation logic	If the rate adaptation algorithm implemented is satisfactory	The Packet Error Rate algorithm currently in place, does not draw a distinction between poor link quality and congestion. When faced with the latter, the mechanism reduces the transmission rate, what makes the congestion even worse. More sophisticated algorithms should be implemented if possible.

7. CONCLUSION

Based on the tests and analysis we just described, we can propose some recommendations that, we believe, will improve scalability and reliability of XO's mesh clouds. These recommendations are grouped into five items, the last of which being a suggested theme for future work.

7.1. REDUCE OVERHEAD WHERE POSSIBLE

Since airtime is so scarce we propose reducing beacon frequency to its minimum frequency (1Hz). It will be necessary to adjust the mechanism of detection of active neighbors to this change (it expected the nodes to beacon at 10Hz). One option is to take the Beacon interval field in the beacons into consideration. In a ten node mesh, reducing the beacon frequency will save an average airtime of more than 1% (for a consolidation interval of one second).

Probe Requests/Responses may be disabled. Mechanisms that rely on them should be redesigned to work passively, i.e. by listening to beacons, instead of actively sending probes. The only mechanism known to use Probes are the 802.11 ad-hoc mode. This will remove the probe response bursts that can take up to 10% of the airtime (in one second consolidation interval) or clog the network for periods as long as 20ms.

The default non-adaptive value for the Route Expiration Time should be raised to 20 seconds, reducing the PDM (Path Discovery Mechanism) traffic overhead to about half. This will save another 6% of the average airtime (consolidated at one second) if nodes in the mesh cloud constantly exchange data in a peer-to-peer fashion.

If, on the other hand, multicast/broadcast transmissions are dominant, the PDM traffic will be not so severe, but measures must be considered in order to reach all nodes, or all participating nodes in the case of multicast, in a more efficient way, i.e. consuming less airtime.

It may also be possible to change the PREQ cluster composition from four (54, 36, 11 and 1) to three frames (54, 22 and 1) without harming PDM reliability and saving

0.242 ms in each PREQ cluster that would reduce PDM airtime in about 20%, but tests of this were not yet performed.

In general, a dense mesh cloud is not a substitute for infrastructure mode if the nodes are concentrated in a classroom, for instance. Reducing the PDM traffic by interconnecting the nodes via infrastructure may permit more nodes to share the spectrum, particularly if they are to interconnect more than twenty nodes. Applications should also be designed to make the best use of the spectrum, by reducing payload size or the frequency of transmissions every time this is possible.

7.2. IMPROVE NETWORK WIDE BROADCAST SPECTRUM EFFICIENCY

Mechanisms to reduce the traffic associated with sending frames to all nodes in the mesh are also necessary. Many approaches are already proposed in the literature, and a good summary can be found in [30] which classifies the protocols in Simple Flooding (which is the one in use currently); Probability Based Methods, in which nodes will rebroadcast the frame based on a probability that this retransmission is necessary to reach other nodes; Area Based Methods, in which nodes rely or estimate positional information; and Neighbor Knowledge Methods, in which a node will proactively determine to which other nodes it should retransmit the broadcast frames in order to achieve all the nodes in the mesh, in a process similar to the Multirelay Point mechanism.

Though we have not studied these methods, we believe we advanced enough to prove that they are necessary, and we suggest this as a future research topic. One possibility would be to randomize the PREQ_delay time may have a positive impact in avoiding the bursts and transmissions of duplicates and beacons can be used to convey neighbor information if a Neighbor Knowledge Method is to be applied.

7.3. ADJUST METRICS

Current cost values need to be changed, particularly the relation between the costs associated with 54Mbps and 36Mbps. We proposed a new mechanism to recalculate the

costs, which considers not only airtime, but also resources consumed (buffer space and battery) for retransmitting the frames.

We believe that the introduction of the weighting term "A" makes it possible to consider other important metrics, such as the current available battery in a node. For instance, we could increase *A* if the battery has reached a certain low level or even disabling participating in path discovery if this level is too critical. Likewise, AC-powered XOs could decrease *A*.

OLPC is now manufacturing standalone modules that may be distributed over an area in order to improve connectivity in sparse scenarios. The standalone module, or active antenna, consists of an autonomous XO radio subsystem that can be attached to a USB port (in a XO or in another computer) or simply powered with a 5V DC power adaptor. In these modules, a lower value of *A* could be selected in order to make it preferred over other XOs with equivalent links and, this way, saving battery and buffer spaces on these XOs. The difference in *A* would be small enough to resolve ties between equivalent paths and not to force traffic through the active antennas, if this would increase airtime consumption.

7.4. CHANGE DEFAULT VALUES FOR PARAMETERS

As a result of the tests performed, we are proposing adjustments in some of the default values for wireless/mesh parameters as shown in Table 18. The basic idea around this new values is to save airtime (Route expiration time), increase tolerance to bursts (CWin) and improve path discovery efficiency (Metric Costs).

Table 18 – New suggested default values

Parameter	**Current default**	**Suggested default value**
Route expiration time	10s	20s
CWsize (CWmin, CWmax)	(7,31)	(15,255)
Metric costs (C_{54}, C_{36}, C_{11}, C_1)	(13,28,46,64)	XO: (963, 1073, 1997, 12906) Active antenna: (962, 1072, 1996, 12905)

Further tests should be performed in the future, to optimize the parameters in Table 19. Increasing the RREQ_delay may reduce PDM traffic overhead while increasing path acquisition time. Similarly, increasing the transmission rate used for broadcast and multicast transmissions will save airtime at the cost of harming coverage.

Table 19 – Other parameters to adjust

Parameter	Current default
RReq_delay	10ms
Multicast/Broadcast rate	2Mbps

All of these parameters may also be dynamically set, to adapt to conditions like congestion, denseness, mobility and battery status. This is the topic of the next and final item.

7.5. FUTURE WORK: DETECT AND ADAPT

In Analysis 3 we demonstrated how important is to detect and react correctly to congestion and to distinguish congestion from poor link quality. Unfortunately, the IEEE802.11 standard introduces no mechanism to make this distinction.

Adapting is a key to the success of a MANET. During the course of our studies, it became clear that a sparse and a dense mesh should be treated differently if we want to achieve both scale and reliability. Setting static values and expect then to work in both scenarios will simply fail.

One simple example is the broadcast/multicast rate parameter that configures the transmission rate for both multicast and broadcast frames. The default value of 2Mbps is clearly too low for a network where the nodes are sufficiently close to each other (a dense mesh). If a 54Mbps transmission is possible, every rate slower than that will be a waste of precious airtime. On the other hand, in a sparse mesh, where the nodes are distributed over a larger area, tens or hundreds of meters apart from each other, such high rates will clearly break connectivity.

The same analysis could be applied to the Path Discovery Mechanism. Broadcasting a cluster of four Path Request frames is a very wasteful approach for a dense mesh, but one necessary if we want to improve coverage. In wireless networks

there is a constant tension between transmission rate and transmission range and, hence, between coverage and spectrum efficiency.

All things considered, detecting congestion may not suffice; we should try to avoid it first. For this to be possible, detecting denseness of a mesh cloud may be extremely important. If nothing is done to prevent it, a dense network will be congested by its own control traffic.

Figure 42 consolidates all adaptive measures that we propose as a result of this study. The three boxes at the top are grouped because they are all related to an increase in contention. The first of the three "IDLE DENSENESS" refers to a situation where the node detects the presence of many other nodes, by counting the beacons received over a certain period. In such a scenario, the node may take prophylactic measures to avoid congestion.

If a given number of these detected nodes are active, and this would be determined by listening to path responses from these nodes, we would achieve the "ACTIVE DENSENESS" status, and the node would increase its contention window even more and also increase multicast and broadcast rates, in order to save airtime.

One final stage in this group would be to detect congestion. Congestion may happen because of other sources of traffic not related to the mesh, coming, for example, from stations in infrastructure node that happen to share the same channel.

There is no current way to detect congestion in the XO. As we stated previously, the Frame Error Count mechanism currently implemented may harm network scalability exactly because there is no distinction between poor link quality and congestion. We propose that congestion can be detected by counting the number of times the backoff timer is stopped since this event is clearly correlated to contention.

If the backoff time is stopped frequently this is a clear indication of denseness/congestion, while if this happens infrequently and a frame is lost anyway, there is a high probability that the problem happened due to poor link quality, since we are dealing with short distances (in long links one node may face a congested spectrum while

the other does not). The use of the backoff counter to infer congestion is present in other proposals such as Idle Sense [31] and in the YARA rate adaptation mechanism [32].

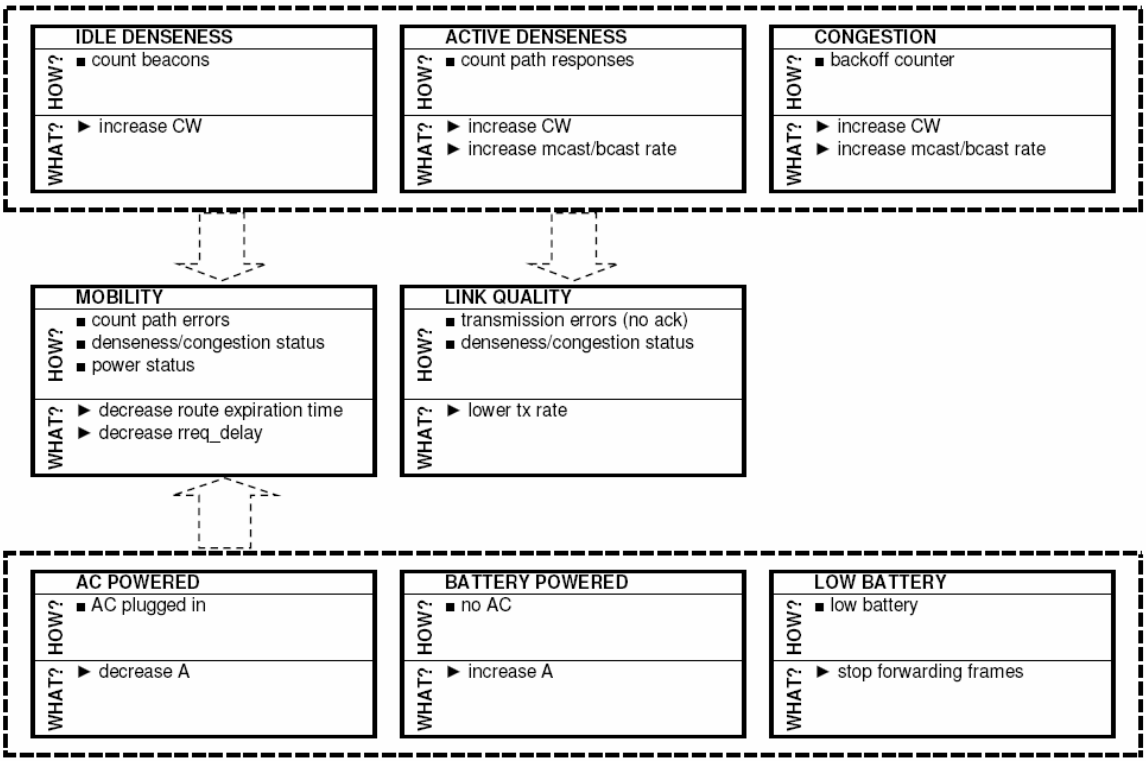

Figure 42 – Adaptation mechanisms

In the center of Figure 42, we depict two other adaptive factors – Mobility and Link Quality. A node would detect mobility by counting PERR frames, i.e. by observing the disruption of paths, but it would also take into account the denseness or congestion of the network. In a congested network paths will also break, so this extra input helps the node to distinguish between mobility and congestion.

Another source of mobility information can be derived from the power status of each node in a path. This could be achieved by the introduction of an extra flag in the Path Reponse frames, that would be 1 if all nodes in the path are battery powered and 0 if not. To convey this information, every node would check this flag and before forwarding the PREP it would:

(1) If the flag is zero, keep it zero.

(2) If the flag is one, keep it as one if the node is AC powered, otherwise change it to zero.

The originator of the path request would store this information on the forwarding route. A route marked as “1” would be one where all nodes are AC powered and hence, a non mobile path. The route expiration time for non mobile routes would have longer expiration times.

As described in Chapter 5, the RREQ_delay is a wait time observed by each node before retransmitting RREQ frames (except the first, which is immediately retransmitted). Reducing the RREQ delay is a means of reducing the route acquisition time, particularly because we are suggesting an increase of this default value from 10 to 20ms in order to reduce the overall PDM traffic.

Denseness/Congestion information would also input the other adaptive factor in the center of the figure – Link Quality. This would permit the rate adaptation mechanism to react properly to loss of frames. If the network is congested, we need to avoid lowering the data rate. If no congestion is detected it is safer to assume that we are dealing with poor link quality.

The last group of adaptive factors is related to the power available. If a node is powered through an AC line, it is in better condition to contribute to the forwarding of frames and it should lower its *A* factor, making it more probable that paths will form through it. Likewise, *A* should be increased if the node is battery powered and, ultimately the node should stop forwarding PREQs (and consequently stop forwarding other nodes data frames) if the battery reaches a low threshold.

7.6. FINAL REMARKS

A scalable and reliable reactive mesh cloud is a challenge. In order to achieve these goals it will be necessary to achieve more with less traffic possible. That means:

(1) Trimming out any unnecessary traffic;

(2) Implementing efficient mechanisms to broadcast information to all nodes;

(3) Implementing metrics that take into account not only airtime, but also processing time and battery conditions in the forwarding nodes;

(4) Finding the most appropriate default values for wireless/mesh parameters;

(5) Adapting to congestion, denseness, mobility, link quality and battery status.

We proposed changes in all of these five topics. Some of the proposals are already effective while others are being considered by OLPC. All of these proposals were designed to be simple to implement, in respect to the memory and power constraints of the radio subsystem. Also, all necessary controls are already in place, with the exception of the changing in the PREP format to include the "AC" field and the backoff counter – and both seem simple enough to implement.

It is also important to stress that most of the experiments we performed were carried out in dense mesh scenarios and there is still much to test and analyze in sparse scenarios. We also concluded that infrastructure mode (use of access points), where possible, still represents the best method to interconnect more than a few tens of nodes and, moreover, we believe that, even with the proposed improvements to scalability, a dense node with hundreds of XOs will be still not feasible.

The *XO mesh* represents a new paradigm, where communication and collaboration is possible without any infrastructure and in an energy efficient way. We believe the green laptop will revolutionize education and provide a powerful and flexible communication platform and, through both, improve the quality of life and future prospects of millions of children.

APPENDIX A – XO SPECS

Physical dimensions

- Approximate dimensions: 242mm × 228mm × 32mm (see drawing to the right for detailed dimensions)
- Approximate weight:
    - XO laptop with LiFePO4 battery: 1.45KG (~3.20lbs);
    - XO laptop with NiMH battery: 1.58KG (~3.48lbs);
- Configuration: Convertible laptop with pivoting, reversible display; dirt-and-moisture-resistant system enclosure; no fan.

Core electronics

- CPU: x86-compatible processor with 64KB each L1 I and D cache; at least 128KB L2 cache;
    - AMD Geode LX-700@0.8W
- CPU clock speed: 433 Mhz;
- ISA compatibility: Support for both the MMX and 3DNow! x86 instruction-set extensions;
    - Athlon instruction set (including MMX and 3DNow! Enhanced) with additional Geode-specific instructions
- Companion chips: PCI and memory interface integrated with CPU;
    - North Bridge: PCI and Memory Interface integrated with Geode CPU
    - South Bridge: AMD CS5536
- Graphics controller: Integrated with CPU; unified memory architecture;
- Embedded controller: ENE KB3700 or ENE KB3700B;
- DRAM memory: 256 MiB dynamic RAM; data rate: dual-DDR333-166Mhz;
- BIOS: 1024KiB SPI-interface flash ROM;
- Open Firmware used to load the operating system;
- Mass storage: 1024 MiB SLC NAND flash, high-speed flash controller;
- Drives: No rotating media.

- CAFE ASIC (camera- and flash-enabler chip provides high-performance camera, NAND FLASH and SD interfaces); Marvell 88ALP01: CAFE Specification

Display

- Liquid-crystal display: 7.5” dual-mode TFT display;
- Viewing area: 152.4 mm × 114.3 mm;
- Two modes: (1) grayscale (B&W) reflective mode (for outdoor use—sunlight-readable); and (2) color backlight Mode (for indoor use);
    - reflective mode: high-resolution (200 DPI), 1200(H) × 900(V) grayscale pixels, power consumption 0.1–0.2Watts;
    - backlight mode: built in sub-pixel sampling of the high-resolution display results in approximately 984(H) × 738(V) color pixels, power consumption 0.2–1.0Watts;
- The display-controller chip (DCON) with memory that enables the display to remain live with the processor suspended. The DCON also formats data for the display.
- This Liquid-crystal display is the basis of our extremely low power architecture. The XO is usable while the CPU and much of the motherboard is regularly turned off (and on) so quickly that it's imperceptible to the user. Huge power savings are harvested in this way (e.g. by turning stuff on the motherboard off when it's not being used (if even for a few seconds), while keeping the display on).

Integrated peripherals

- Keyboard: 80+ keys, 1.0mm stroke; sealed rubber-membrane key-switch assembly;
- Keyboard Layouts includes: English, Arabic, Thai, West African (Nigeria), Portuguese, Spanish, Amharic, French, Urdu, Cyrillic, Turkish (not final), Nepali, Mongolian, Kazakh, Devanagari, Uzbek, Pashto, Dari, Pulaar (Fula), Italian
- Gamepad: Two sets of four-direction cursor-control keys;
- Touchpad: Dual capacitance/resistive touchpad; supports written-input mode;
- ALPS Electric Dual capacitance/resistive touchpad;
- Audio: AC’97 compatible audio subsystem; Internal stereo speakers and amplifier; internal monophonic microphone; jacks for external headphones or microphone;
    - Analog Devices AD1888 and Analog Devices SSM2302 for audio amplification

- Camera: integrated color video camera; 640 x 480 resolution at 30 FPS; independent (and undefeatable by software) display of microphone and camera recording status; the camera and device driver support disabling AGC and automatic color balancing, to enable its use as a photometric sensor for educational applications;
    - Omnivision OV7670
- Wireless Networking: Integrated 802.11b/g (2.4GHz) interface; 802.11s (Mesh) networking supported; dual adjustable, rotating antennas support diversity reception; capable of mesh operation when CPU is powered down;
    - Marvell Libertas wireless chipset, 88W8388 controller and 88W8015 radio
- Status indicators: Power, battery, and WiFi (2), visible with lid open or closed; Microphone In-Use, and Camera In-Use, visible when lid is open.

External connectors

- DC power: 6mm (1.65mm center pin) connector; 11 to 18 V input usable, –32 to +40V input tolerated; power draw limited to 18 W; - see power connector dimensions at Battery and power.
- Headphone output: standard 3.5mm 3-pin switched stereo audio jack;
- Microphone input: standard 3.5mm 2-pin switched mono microphone jack; selectable 2V DC bias; selectable sensor-input mode (DC or AC coupled);
- USB: Three Type-A USB 2.0 connectors; Up to 1A power supplied (total);
- Flash Expansion: SD Card slot.

Battery

- Pack type: 2 or 4 cells LiFePO4; or 5 cells NiMH, approx. 6V series configuration (subject to change);
- Capacity: 16.5 Watt-hours (NiMH), 22 Watt-hours (LiFeP);
- Fully-enclosed “hard” case; user removable;
- Electronics integrated with the pack provide:
    - Identification;
    - Battery charge and capacity monitoring chip (Maxim DS2756);
    - Thermal and over-current sensors along with cutoff switch to protect battery;
- Minimum 2,000 charge/discharge cycles (to 50% capacity of new).

- Power Management will be critical

BIOS/loader

- Open Firmware (including hardware initialization and fast resume).

Environmental specifications

- Temperature: UL certification planned to 45C in Q32007, pending 50C certification in mid-2008;
- Humidity: UL certification planned to IP42 (perhaps higher) when closed, the unit should seal well enough that children walking to and from school need not fear rainstorms and dust;
- Maximum altitude: –15m to 3048m (14.7 to 10.1 PSIA) (operating), –15m to 12192m (14.7 to 4.4 PSIA) (non-operating);
- Shock 125g, 2ms, half-sine (operating) 200g, 2ms, half-sine (non-operating);
- Random vibration: 0.75g zero-to-peak, 10Hz to 500Hz, 0.25 oct/min sweep rate (operating); 1.5g zero-to-peak, 10Hz to 500Hz, 0.5 oct/min sweep rate (nonoperating);
- 2mm plastic walls (1.3mm is typical for most systems).

APPENDIX B – PYTHON TOOLS

B.1 – THE AIRTIME TOOL

```
#!/usr/bin/python
#
# Copyright (c) 2007 OLPC
# Author: Ricardo Carrano <carrano at laptop.org>
# Version 0.1.2
#
# this program is free software; you can redistribute it and/or modify
# it under the terms of the gnu general public license as published by
# the free software foundation; either version 2 of the license, or
# (at your option) any later version.
#
# this program is distributed in the hope that it will be useful,
# but without any warranty; without even the implied warranty of
# merchantability or fitness for a particular purpose.  see the
# gnu general public license for more details.
#
# you should have received a copy of the gnu general public license
# along with this program; if not, write to the free software
# foundation, inc., 51 franklin st, fifth floor, boston, ma  02110-1301  usa


import sys
import commands
import math
from optparse import OptionParser

arguments = OptionParser()
arguments.add_option("-f","--pcap-file",
                     dest="pcapfile",
                     help="Capture dump")
arguments.add_option("-t","--text-file",
                     dest="textfile",
                     help="Capture already converted/filtered")
arguments.add_option("-i","--interval",
                     dest="interval",
                     help="Consolidation interval in seconds")
arguments.add_option("-w","--filter",
```

```
                    dest="filter",
                    help="Wireshark filter")
arguments.add_option("-o","--output-format",
                    dest="output",
                    help="Output Format [csv, lines] ")
arguments.add_option("--no-fcs",
                    action="store_false",
                    dest="crc",
                    default=True,
                    help="don't check if frames have bad crc")
(options, args) = arguments.parse_args()

if not (options.pcapfile or options.textfile) :
   print "input file is mandatory"
   sys.exit(0)
filter_exp = ''
filter = ''
if options.crc == True:
   filter += 'wlan.fcs_good == 1'
   if options.filter:
      filter += ' and '+options.filter
else:
   filter += options.filter
if options.crc or options.filter:
   filter_exp = '-R "'+filter+'"'
if options.pcapfile:
   pcapfile = options.pcapfile
   inputfile = pcapfile
if options.textfile:
   textfile = options.textfile
   inputfile = textfile
else:
   textfile = pcapfile+'.tmp3'
   filter_cmd='tshark -r %s %s -T fields -e frame.time_relative -e radiotap.datarate -e
frame.len > %s' % (pcapfile, filter_exp, textfile)
   s, o = commands.getstatusoutput(filter_cmd)
if options.interval:
   interval = float(options.interval)
else:
   interval = 1
timeslot = 0
lastslot = 0
airtime = [0]
```

```
fd = open(textfile, 'r')

cck_datarates = ('2', '4', '11', '22')
ofdm_datarates = ('12', '18', '24', '36', '48', '72', '96', '108')

for line in fd:
   time, rate, size = line.split('\t')
   size = size.strip('\n')
   if rate in cck_datarates:
      airsize = 192 + float(size) * 16 / float (rate)
   elif rate in ofdm_datarates:
      airsize = 26 + float(size) * 16 / float (rate)
   else:
      airsize = 0
   timeslot = int(math.floor(float(time) / interval))
   if timeslot > lastslot:
      for slot in range(lastslot, timeslot):
         airtime.append(0)
   airtime[timeslot] += airsize / (interval * 1000000)
   lastslot = timeslot

if options.output == "csv":
   for i in airtime:
      print str(i)+',',
else:
   for i in range(0, len(airtime)):
      print "[%s - %s): %.2f%%" % (i*interval, (i+1)*interval, airtime[i] * 100)
```

B.2 – THE PDM ANALYSIS TOOL

```
#!/usr/bin/python
#
# Copyright (c) 2007 OLPC
# Author: Ricardo Carrano <carrano at laptop.org>
# Version 0.2.0
#
# this program is free software; you can redistribute it and/or modify
# it under the terms of the gnu general public license as published by
# the free software foundation; either version 2 of the license, or
# (at your option) any later version.
```

```
#
# this program is distributed in the hope that it will be useful,
# but without any warranty; without even the implied warranty of
# merchantability or fitness for a particular purpose.  see the
# gnu general public license for more details.
#
# you should have received a copy of the gnu general public license
# along with this program; if not, write to the free software
# foundation, inc., 51 franklin st, fifth floor, boston, ma  02110-1301  usa


import sys
import commands
from stats import mean,stdev
from optparse import OptionParser

arguments = OptionParser()
arguments.add_option("-f","--pcap-file",
                     dest="pcapfile",
                     help="Capture dump")
arguments.add_option("-t","--text-file",
                     dest="textfile",
                     help="Capture already converted/filtered")
arguments.add_option("--no-fcs",
                     action="store_false",
                     dest="crc",
                     default=True,
                     help="don't check if frames have bad crc")
(options, args) = arguments.parse_args()

if not (options.pcapfile or options.textfile) :
   print "input file is mandatory"
   sys.exit(0)
if options.pcapfile:
   pcapfile = options.pcapfile
   inputfile = pcapfile
if options.textfile:
   textfile = options.textfile
   inputfile = textfile
else:
   textfile = pcapfile+'.tmp'
   filter_exp = "wlan.fcs_good == 1 && wlan_mgt.fixed.action_type == 1"
   if not options.crc: filter_exp = "wlan_mgt.fixed.action_type == 1"
```

```
   filter_cmd='tshark -r %s -R "%s" -T fields -e frame.time_relative -e wlan.sa -e
wlan.da -e wlan_mgt.fixed.sa -e wlan_mgt.fixed.da -e wlan_mgt.fixed.mesh_action -e
wlan_mgt.fixed.metric -e wlan_mgt.fixed.hopcount -e wlan_mgt.fixed.ttl -e wlan.seq -e
wlan_mgt.fixed.ssn -e wlan_mgt.fixed.dsn -e radiotap.datarate > %s' % (pcapfile,
filter_exp, textfile)
   s, o = commands.getstatusoutput(filter_cmd)


fd = open(textfile, 'r')


maxttl = 5
datarates = ('108', '72', '22', '2', '0')


groups={}
rreqs={}
rreps={}
rerrs={}
participants={}
originators={}
targets={}
spread_times={}
rreqs['count'] = 0
rreps['count'] = 0
rerrs['count'] = 0
rreqs['datarate']={}
rreqs['hops']={}
rreps['hops']={}
for datarate in datarates:
   rreqs['datarate'][datarate]=0
for ttl in range(1, maxttl+1):
   rreqs['hops'][ttl]=0
   rreps['hops'][ttl]=0
   spread_times[ttl] = []


# Airtime costs
# PREQ = 86 bytes
# PREP = 80 bytes
preq_airtime = {}
preq_airtime['108'] = 0.0387
preq_airtime['72'] = 0.0457
preq_airtime['22'] = 0.2545
preq_airtime['2'] = 0.8800
preq_airtime['0'] = 0.0
prep_airtime = 0.8320
airtime = 0
starttime = 0
```

```
endtime = 0


multihop_percent = 0


def stdev2(serie):
   try:
      sd = stdev(serie)
      return sd
   except TypeError:
      return 0


def formattime(timetoformat):
   if timetoformat:
      return (mean(timetoformat)*1000, stdev2(timetoformat)*1000, min(timetoformat)*1000,
max(timetoformat)*1000)
   else:
      return "no availabe data"


#
# PARSING THE INPUT FILE and POPULATING DICTS
#
for line in fd:
   time, transmitter, receiver, origin, target, action, metric, hops, meshttl, msn, ssn,
dsn, datarate = line.split('\t')
   datarate = datarate.strip('\n')
   for xo in transmitter, receiver, origin, target:
     for array in participants, originators, targets:
        if not array.has_key(xo) and xo != 'ff:ff:ff:ff:ff:ff' and xo != '':
           array[xo] = {}
           array[xo]['counter'] = 0
           array[xo]['sd'] = 0 # source or destination
   if not starttime:
      starttime = time
   endtime = time
   participants[transmitter]['counter'] += 1
   if action != '0x0002':
      originators[origin]['counter'] += 1
      targets[target]['counter'] += 1
      if action == '0x0000':
         rreqs['count'] += 1
         rreqs['datarate'][datarate] += 1
         rreqs['hops'][int(maxttl)-int(meshttl)] += 1
      group=origin+'-'+target+'-'+ssn
      if groups.has_key(group):
```

```
         groups[group]['count'] += 1
         groups[group]['lasttime'] = time
      else:
         groups[group]={}
         groups[group]['count'] = 1
         targets[target]['sd'] += 1
         originators[origin]['sd'] += 1
      if action == '0x0000' and transmitter == origin and datarate == '108': # The
original 54Mbps RREQ
            groups[group]['starttime'] = time
      if action == '0x0000' and groups[group].has_key('starttime'):
            spread_times[maxttl-int(meshttl)].append(float(time)-
float(groups[group]['starttime']))
      if action == '0x0001':
         groups[group]['replied'] = True
         rreps['count'] += 1
         rreps['hops'][int(hops)] += 1
         if receiver == origin: # The last RREP
            groups[group]['endtime'] = time
         if transmitter == target: # The first RREP
            groups[group]['resptime'] = time
   else:
      rerrs['count'] += 1



#
# ACCOUNTING (processing the DICTS)
#
disc_frames = int(rreqs['count']) + int(rreps['count']) + int(rerrs['count'])
discovery_times = []
end_times = []
last_times = []
time_spread = {}
counts = []
replied_counter = 0
for group in groups.keys():
   if groups[group].has_key('starttime') and groups[group].has_key('resptime'):
      discovery_time = float(groups[group]['resptime']) -
float(groups[group]['starttime'])
      discovery_times.append(discovery_time)
   if groups[group].has_key('starttime') and groups[group].has_key('endtime'):
      end_time = float(groups[group]['endtime']) - float(groups[group]['starttime'])
      end_times.append(end_time)
   if groups[group].has_key('starttime') and groups[group].has_key('lasttime'):
```

```
      last_time = float(groups[group]['lasttime']) - float(groups[group]['starttime'])
      last_times.append(last_time)
   if groups[group].has_key('replied'):
      replied_counter += 1
   counts.append(groups[group]['count'])



#
# REPORT GENERATION
#
print "-----------------------------"+"-"*len(inputfile)
print "ROUTE DISCOVERY REPORT (for %s)" % (inputfile)
print "-----------------------------"+"-"*len(inputfile)
print "Captured route discovery frames: %s [RREQs %s (%.2f%%) RREPs %s (%.2f%%) RERRs %s
(%.2f%%)]" % (disc_frames, rreqs['count'], float(rreqs['count'])*100/float(disc_frames),
rreps['count'], float(rreps['count'])*100/float(disc_frames), rerrs['count'],
float(rerrs['count'])*100/float(disc_frames))
print "Route discovery clusters: %s" % (len(groups))
print "Average # of frames captured per cluster: %.2f (stdev = %.2f) [RREQs %.2f, RREPs
%.2f, RERRs %.2f]" % (mean(counts), stdev2(counts),
float(rreqs['count'])/float(len(groups)), float(rreps['count'])/float(len(groups)),
float(rerrs['count'])/float(len(groups)))
print "Replied requisitions (with any rrep captured): %s (%.2f%%)" %
(replied_counter,replied_counter*100/len(groups))
print "Completed requisitions (with final rrep captured): %s (%.2f%%)" %
(len(end_times),float(len(end_times)*100/len(groups)))
print "\nRREQs:"
print "By Hop Count:",
for ttl in range(1, maxttl+1):
   print "%s: %s (%.2f%%)" % (ttl, rreqs['hops'][ttl],
float(rreqs['hops'][ttl]*100)/float(rreqs['count'])),
print "\nBy Datarate:",
for datarate in datarates:
   txrate = int(datarate)/2
   if datarate == '0': txrate = 'Unknown'
   print "%s: %s (%.2f%%)" % (txrate, rreqs['datarate'][datarate],
float(rreqs['datarate'][datarate])*100/float(rreqs['count'])),
   airtime += int(rreqs['datarate'][datarate]) * preq_airtime[datarate]
airtime += int(rreps['count']) * prep_airtime
print "\n\nRREPs:"
print "By Hop Count:",
for ttl in range(1, maxttl+1):
   print "%s: %s (%.2f%%)" % (ttl, rreps['hops'][ttl],
float(rreps['hops'][ttl]*100)/float(rreps['count'])),
   if ttl > 1:
      multihop_percent += float(rreps['hops'][ttl]*100)/float(rreps['count'])/int(ttl)
print "\nEstimated %% of multihop paths: %.2f%%\n" % (multihop_percent)
```

```
print "First RREP elapsed time (ms): %.2f (stdev = %.2f) MIN: %.2f MAX: %.2f" %
(formattime(discovery_times))

print "Route acquisition time (ms): %.2f (stdev = %.2f) MIN %.2f MAX %.2f" %
(formattime(end_times))

if last_times: print "Route discovery lifespan (ms): %.2f (stdev = %.2f) MAX %.2f" %
(mean(last_times)*1000, stdev2(last_times)*1000, max(last_times)*1000)

else: print "Route discovery lifespan (ms): no available data"

for ttl in range(1, maxttl+1):

   if spread_times[ttl]:

      print "RREQ distribution for %s hop(s) (ms): mean = %.2f (stdev = %.2f) MIN: %.2f
MAX: %.2f" % (ttl, mean(spread_times[ttl])*1000, stdev2(spread_times[ttl])*1000,
min(spread_times[ttl])*1000, max(spread_times[ttl])*1000)

print "\nPDM Airtime (ms):", airtime

print "PDM Activity Interval (s): %.2f" % (float(endtime)-float(starttime))

print "PDM Airtime %%: %.2f" % (float(airtime)/(10*(float(endtime)-float(starttime))))

print "\nParticipants: %s\n" % (len(participants))

for xo in participants.keys():

   print "%s: FWD %s\t\tORIGIN %s => %s\t\tTARGET %s => %s" % (xo,
participants[xo]['counter'], originators[xo]['sd'], originators[xo]['counter'],
targets[xo]['sd'], targets[xo]['counter'])
```

REFERENCES

[1] Leonardo Hideki, Raphael Martins, Arthur Guerrante, Ricardo Carrano, Luiz Magalhães. Evaluating the impact of RTS-CTS in OLPC's XOs' Mesh Networks. In: XXV Simpósio Brasileiro de Telecomunicações (SBrT'07), 2007, Recife. Anais do XXV Simpósio Brasileiro de Telecomunicações (SBrT'07), 2007

[2] Walke, B., Mangold, S., Berlemann, L. (2006). IEEE 802 Wireless Systems – Protocols, Multi-hop Mesh/Relaying, Performance and Spectrum Coexistence, John Wiley & Sons.

[3] FunkFeuer Free Net – http://www.funkfeuer.at/

[4] ReMesh - Grupo de Pesquisa em Redes Mesh – http://mesh.ic.uff.br/

[5] Meraki Public Network in San Francisco – http://sf.meraki.com/map

[6] Movement of Top Predators: Combining Sensor Technology and Biology - http://www.steps.ucsc.edu/collabs.html

[7] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In ASPLOS, San Jose, CA, October 2002.

[8] Clausen, T. and Jacquet, P. (2003). Optimized link state routing protocol (OLSR), IETF Network Working Group RFC 3626.

[9] C. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In Proc. of the ACM SIGCOMM, October 1994.

[10] OLSRd – an adhoc wireless mesh routing daemon – http://www.olsr.org/

[11] Luceor OLSR – http://www.luceor.com/article.php3?id_article=48

[12] 6WINDGate software - http://www.6wind.com/6WINDGate-software/Control_Plane_Modules

[13] Moy, J. (April 1998). OSPF Version 2. Internet Engineering Task Force.

[14] A. Qayyum, L. Viennot, A. Laouiti, Multipoint relaying: An efficient technique for flooding in mobile wireless networks, RR-3898, INRIA, March 2000.

[15] C. Hedrick. Routing Information Protocol. RFC 1058. June 1988.

[16] C. E. Perkins, "Ad-hoc on-demand distance vector routing," in MILCOM '97 panel on Ad Hoc Networks, Nov. 1997

[17] RFC 3561 C. Perkins, E. Belding-Royer, S.Das. July 2003

[18] Ad-hoc On-demand Distance Vector Routing – Uppsala University – http://core.it.uu.se/core/index.php/AODV-UU

[19] Bahr, M. (2006). Proposed routing for ieee 802.11s wlan mesh networks, in WICON '06: Proceedings of the 2nd annual international workshop on Wireless internet (ACM, New York, NY, USA), ISBN 1-59593-510-X, p. 5.

[20] Z. Haas. A new routing protocol for the reconfigurable wireless networks. In Proc. of the IEEE Int. Conf. on Universal Personal Communications., Oct 1997.

[21] Couto, D. S. J. D., Aguayo, D., Bicket, J. and Morris, R. (2003). A high-throughput path metric for multi-hop wireless routing, in ACM International Conference on Mobile Computing and Networking (MobiCom), pp. 134-146.

[22] Draves, R., Padhye, J. and Zill, B. (2004). Routing in multi-radio, multi-hop wireless mesh networks, in ACM International Conference on Mobile Computing and Networking (MobiCom), pp. 114-128.

[23] Yang, Y., Wang, J. and Kravets, R. (2005). Designing routing metrics for mesh networks, in IEEE Workshop on Wireless Mesh Networks (WiMesh).

[24] Subramanian, A. P., Buddhikot, M. M. and Miller, S. C. (2006). Interference aware routing in multi-radio wireless mesh networks, in IEEE Workshop on Wireless Mesh Networks (WiMesh), pp. 55-63/

[25] Koksal, C. E. and Balakrishnan, H. (2006). Quality-aware routing metrics for time-varying wireless mesh networks, IEEE Journal on Selected Areas in Communications 24, 11, pp. 1984-1994.

[26] Zhang, Y., Luo, J. and Hu, H. (2007). Wireless Mesh Networking Architectures, Proto-cols and Standards, chap. 12, Wireless Networks and Mobile Communications Series (Auerbach Publications, Taylor & Francis Group), pp. 391-423.

[27] CACE Technologies - AirPcap Wireless Capture Adapter for Windows - http://www.cacetech.com/products/airpcap_family.htm

[28] Wireshark tool - http://www.wireshark.org/

[29] MetaGeek's Wispy Spectrum Analyzer http://www.metageek.net/Products/Wi-Spy

[30] T. Camp and B. Williams. Comparison of broadcasting techniques for mobile ad hoc networks. In Proceedings of The Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC 2002

[31] Idle sense: an optimal access method for high throughput and fairness in rate diverse wireless LANs ACM SIGCOMM Computer Communication Review Volume 35 , Issue 4 (October 2005). Pages: 121 – 132.

[32] - Adaptação Automática de Taxa em Redes 802.11 Densas - Kleber Vieira Cardoso1, José Ferreira de Rezende - 26° Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos Rio de Janeiro – RJ – Brasil (2008)

[33] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and Evaluation of an Unplanned 802.11b Mesh Network", ACM MobiCom, agosto de 2005.

[34] D. Couto, D. Aguayo, J. Bicket and R. Morris, “A High-Throughput Path Metric for Multi-Hop Wireless Routing”, ACM MobiCom, San Diego, CA, setembro de 2003.

[35] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-radio, Multi-hop Wireless Mesh Networks", ACM MobiCom, Philadelphia, PA, 2004.

[36] R. Draves, J. Padhye, and B. Zill, "Comparison of Routing Metrics for Static Multi-Hop Wireless Networks", ACM SIGCOMM, Portland, 2004.

[37] C. Ho, K. Ramachandran, K. C. Almeroth and E. M. Belding-Royer, "A Scalable Framework for Wireless Network Monitoring", 2nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH), Philadelphia, PA, setembro de 2004.

[38] M. Lad, S. Bhatti, S. Hailes and P. Kirstein, "Enabling Coalition-Based Community Networking", The London Communications Symposium (LCS), setembro de 2005.

[39] K. Ramachandran, E. M. Belding-Royer and K. C. Almeroth, "DAMON: A Distributed Architecture for Monitoring Multi-hop Mobile Networks", IEEE First International Conference on Sensor and Ad hoc Communications and Networks (SECON), Santa Clara, CA, 2004.

[40] N. Tsarmpopoulos, I. Kalavros and S. Lalis,"A Low Cost and Simple-to-Deploy Peer-to-Peer Wireless Network based on Open Source Linux Routers", IEEE First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities (TRIDENTCOM), pp. 92-97, 2005.

[41] S. Weber, V. Cahill, S. Clarke and M. Haahr, "Wireless Ad Hoc Network for Dublin: A Large-Scale Ad Hoc Network Test-Bed", ERCIM News, vol. 54, 2003.

[42] IEEE (1998). IEEE 802.1d. media access control (MAC) bridges, Standard.

[43] IEEE (2001). IEEE p802.1x.port-based network access control, Standard.

[44] IEEE (2008). IEEE p802.11s;/d2.00, draft amendment to standard IEEE 802.11: ESS mesh networking, Standard.

[45] Perkins, C. E. and Royer, E. B. (1999). Ad hoc on-demand distance vector routing, in IEEE Workshop on Mobile Computing Systems and Applications, pp. 90-100.

[46] Ramachandran, K. N., Buddhikot, M. M., Chandranmenon, G., Miller, S., Belding-Royer, E. M. and Almeroth, K. C. (2005). On the design and implementation of infrastructure mesh networks, in IEEE Workshop on Wireless Mesh Networks (WiMesh).

[47] Camp, J. and Knightly, E. (2008). The IEEE 802.11s extended service set mesh networking standard, IEEE Communications Magazine (to appear).

[48] IEEE Std 802.11-2007. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications

[49] OLPC – One Laptop per Child – http://www.laptop.org

[50] Albuquerque, C; Saade, D.; Tarouco, L.; Magalhães, L.; Gomes, A.; Carrano, R. . Multihop MAC: Desvendando o Padrão IEEE 802.11s. Livro de Minicursos SBRC. Rio de Janeiro, 2008.

[51] Carrano, R. ; Saade, D, ; Campista, M. ; Moraes, I.; Albuquerque, C.; Magalhães, L; Rubinstein, M.; Costa, L; Duarte, O. Multihop MAC: IEEE 802.11s Wireless Mesh Networks. In: Dharma Agrawal. (Org.). Encyclopedia on Ad Hoc and Ubiquitous Computing.

[52] L.G.Roberts, “ALOHA packet system with and without slots and capture”, Comput. Commun. Rev., vol. 5, pp. 28-42, Apr. 1975.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo