Departamento de Ciências e Tecnologias de Informação

Mesh Networks for Handheld Mobile Devices

Carlos José Pereira da Silva Meralto

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Engenharia de Telecomunicações e Informática

Orientador(a):
Prof. Rui Neto Marinheiro, Professor Auxiliar,
ISCTE-Instituto Universitário de Lisboa

Coorientador(a):
Prof. José Moura, Professor Auxiliar,
ISCTE-Instituto Universitário de Lisboa

Outubro, 2015

Abstract

Mesh communications emerge today as a very popular networking solution. Mesh networks have a decentralized and multihop design. These characteristics arouse interest in research for relevant novel features, such as cooperation among nodes, distribution of tasks, scalability, communication with limited infrastructure support, and the support of mobile devices as mesh nodes.

In addition to the inexistence of a solution that implements mesh networks with mobile devices at the data link layer (Layer 2), there is also a need to reconsider existing metrics with new information to tackle the intrinsic characteristics of mobile devices, e.g., the limited energy resources of their battery.

To tackle this problem, this thesis presents a detailed study about projects, routing protocols and metrics developed in the area of mesh networks. In addition, two data link layer solutions, Open802.11s and B.A.T.M.A.N-advanced, have been adapted and deployed in a real mesh network testbed with off the shelf routers devices installed with a customized operating system. From this testbed, Open802.11s has proved to offer better performance than B.A.T.M.A.N-advanced. Following this, a breakthrough in this work has been the integration of the 802.11s on an Android mobile device and its subsequent incorporation in the mesh network. This allowed the study of eventual limitations imposed by the mobile device on the operation of the mesh network, namely performance and energy scarcity. With this, another major novelty has followed, by designing, implementing and evaluating several energy related metrics regarding the battery status of mobile devices. This has enabled the participation of mobile devices in mesh routing paths in an efficient way.

Our main objective was to implement a mesh network with mobile devices. This has been achieved and validated through the evaluation of diverse testing scenarios performed in a real mesh testbed. The obtained results also show that the operation of a mesh with mobile devices can be enhanced, including the lifetime of mobile devices, when an energy-aware metric is used.

Keywords: Mesh Networks; Mesh networks with mobile devices; Routing protocols for mesh networks; Metrics for mesh networks; Energy-aware routing metrics

Resumo

As redes *mesh* surgem hoje em dia como uma solução de rede em crescimento e expansão. Neste tipo de redes o comportamento entre os nós é descentralizado e numa topologia de multihop. Estas características despertam interesse na pesquisa e desenvolvimento de novas funcionalidades tais como: cooperação entre nós, distribuição de tarefas, escalabilidade da rede e comunicações mesmo em casos de uma infraestrutura limitada e o suporte de dispositivos móveis como nós de uma rede mesh.

Associado à inexistência de um projecto que implemente redes mesh em dispositivos móveis na camada de ligação de dados (*Layer* 2), surge a necessidade de repensar as métricas já existentes com novas informações que façam face às novas características dos dispositivos móveis, neste caso, os recursos limitados de bateria.

Por forma a resolver este problema, este trabalho apresenta um estudo detalhado sobre os projetos, protocolos de *routing* e métricas desenvolvidas na área das redes mesh. Além disso, duas soluções que utilizam a camada de ligação de dados, Open802.11s e BATMAN-advanced, estes foram adaptadao e implementados num testbed real utilizando routers com um sistema operacional costumizado instalado. Deste testbed, concluiu-se que o Open802.11s obtem um melhor desempenho que o BATMAN-advanced. Assim, um dos avanços deste trabalho foi a integração do Open802.11s num dispositivo móvel Android e sua posterior incorporação na rede mesh. Isto permitiu o estudo de eventuais limitações impostas pelo dispositivo móvel ao funcionar numa rede mesh, ou seja, desempenho e a escassez de energia. Com isso, foi concebida outra novidade, através da concepção, avaliação e implementação de várias métricas relacionadas com a energia e que têm por base o estado da bateria do dispositivo. Isto permitiu que os dispositivos móveis participem na rede mesh e a sua gestão de bateria seja feita de forma eficiente.

O principal objectivo era a implementação de uma rede mesh com dispositivos móveis. Este foi alcançado e validado através de diversos cenários de teste reais. Os resultados obtidos demonstram também que o funcionamento de uma rede mesh com dispositivos móveis pode ser melhorada, incluindo o tempo de vida dos dispositivos móveis, quando uma métrica que considera a energia é utilizada.

Palavras chave: Redes *Mesh*; Redes mesh com dispositivos móveis; Protocolos de *Rotuting* em redes *mesh*; Métricas para redes *mesh; Métricas de energia*

Agradecimentos

A conclusão desta tese só foi possível com a ajuda de um grupo de pessoas que conseguiram cada uma à sua maneira e com o seu ponto de vista motivar-me em cada um dos 365 dias decorridos desde o início deste projecto. Nesse sentido, estas palavras são o reconhecimento para todos aqueles que contribuíram para a conclusão do meu mestrado.

Em primeiro lugar quero agradecer aos meus professores orientadores, Professor José Moura e Professor Rui Neto Marinheiro, pela constante orientação, disponibilidade, sugestões, criticas e motivação para atingir todos objectivos propostos no inicio.

À minha mãe que para além de me proporcionar a possibilidade de realizar os meus estudos superiores, me acompanhou ao longo destes cinco anos incutindo-me sempre um espírito de responsabilidade e motivando-me dia a após dia para que a conclusão desta tese fosse possível.

À minha irmã, ao João Coelho e à Suzanne Tavares pela motivação, aconselhamentos e revisão na escrita deste documento em inglês.

À minha namorada Filipa Vitorino, que ao longo do último ano acompanhou esta fase da minha vida com uma enorme paciência, ajudando-me em tudo aquilo que estava ao seu alcance.

A todos os meus colegas de curso, em especial ao Pedro Almeida e à Sara Santos, por todo o acompanhamento durante o percurso académico, tendo sido uma ajuda essencial ao longo dos últimos cinco anos. Por fim, aos meus amigos mais próximos (David Santos, João Melo, Mário Carvalho, Marino Soares e Rui Gonçalves), que souberam sempre ouvir as minhas preocupações e acima de tudo com a boa disposição que lhes reconheço estar presentes em todos os momentos da realização desta tese.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AODV	Ad hoc On-Demand Distance Vector
ARP	Address Resolution Protocol
B.A.T.M.A.N	Better Approach To Mobile Ad-hoc Networks
CMMBCR	Conditional max-min battery capacity routing
DSR	Dynamic Source Routing
DSVD	Destination Sequenced Distance Vector
EARM	Energy aware routing metric for IEE802.11s
ECC	Elliptic Curve Cryptography
ENT	Effective number of transmission
ETE	Expected Transmission Energy route metric
ETT	Expected transmission time
ETX	Expected transmission count
HWMP	Hybrid Wireless Mesh Protocol
iAWARE	Interference Aware Routing Metric
IEEE	Institution of Electrical Engineers
MAC	Medium Access Control
MANET	Mobile ad hoc networks
MCS	Modulation and Coding Scheme
MBCR	Minimum Battery Cost Routing
mETX	Modified expected transmission count
MMBCR	Min-Max Battery Cost Routing
MPR	Multi Point Relay
MREP	Maximal residual energy path routing
MTPR	Minimum Total Power Routing
OLSR	Optimized Link State Routing Protocol
OLSRd	Optimized Link State Routing Protocol daemon
OMG	Originator Message
OS	Operative System
OSI	Open Systems Interconnection
SINR	Signal to Interference and Noise Ratio
SNR	Signal to Noise Ratio
SPAN	Smart Phone Ad-Hoc Networks
SQ	Sequence Number
TC	Transmission Control

TTL	Time To Live
VPN	Virtual Private Network
WCETT	Weighted cumulative ETT
WMN	Wireless Mesh Network

1. Introduction

Mesh networks is today a technology with a very strong expansion. From everywhere arise useful usage scenarios for these type of networks such as riots, disasters and emergencies. In these scenarios, the lack of direct connectivity among mobile nodes and the network infrastructure occurs very often. In addition, the fact that, as far as we know, there is not an explicit standard for mesh networks incorporating handheld mobile devices that takes into account their specific limitations. This urges for novel research that studies the deployment of mobile devices in mesh networks.

It can be predictable that in such scenarios the existence of a technology that allows sending a text message or making a voice call through a mobile device becomes very useful and translates into a direct gain for the users.

In short, the main motivation behind this study is to improve some relevant network characteristics such as reliability, coverage, interoperability, performance and capacity of mobile mesh networks without penalizing the battery autonomy of terminals. The final goal is to deploy a real mesh network that enables the mobile devices to communicate with each other without the support of any infrastructure.

Wireless mesh networks (WMN) have emerged to address some limitations of traditional wireless networks. WMNs often consist of mesh clients and mesh routers. Normally, the mesh routers are static and power-enabled. The mesh routers also form a wireless backbone for the WMNs because these routers are directly connected directly to the wired network. Mesh clients are mobile and try to access the network directly via mesh routers or forming a multi-hop mesh topology with other mesh nodes. Therefore, based on these characteristics, the implementation of such networks satisfies the requirements of mobile devices. [1]

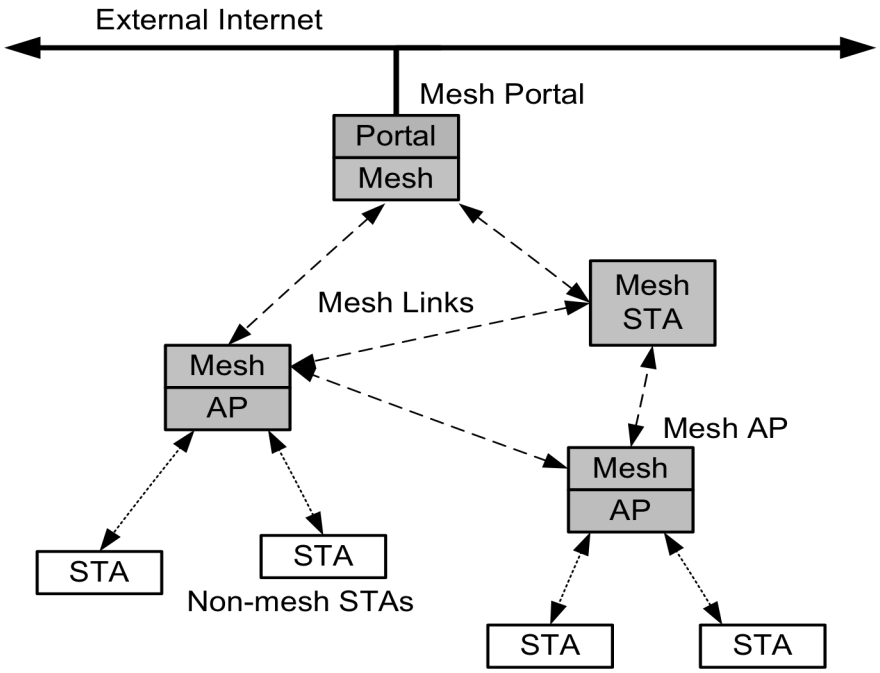

Figure 1.1 - Mesh network architecture

The research of mesh networks on mobile devices is a relatively new area and brings questions such as: the analysis of the best routing algorithms, routing metrics, optimizations in terms of battery autonomy, delay and the implementation of new functionalities not covered by the standard [2] .

The current contribution explores the concept of using mobile phones, which run the Android operating system, to create a mesh network. This type of implementation has different difficulties for implementing a traditional mesh network. Therefore, there are issues such as battery consumption, hardware limitations and the need to have root access to the mobile device.

Our work was initially based on a detailed study about the state of the art on mesh networks, more specifically in the topics of path selection and routing metrics. Throughout this research, it was possible to identify previous projects concerning mesh networks and mobile devices, as well as perceiving that any of these projects deployed mesh networks at Layer 2. This particular architectural aspect is very appealing because it ensures lower reaction delays to network failures than other upper layers implementations. We have also studied relevant routing protocols and mesh networks metrics. In this way, we have identified an open research issue. In fact, there was no available contribution in the literature, as far as we know, which used energy-aware metrics to control both the path selection and routing protocol within a mesh network formed by mobile devices.

The outcome of current work aims to answer some still open research questions in the area of mesh networks with mobile devices, such as:

- Are the routing protocols and metrics defined in the current available standards suitable for the real implementation of a mesh network with mobile devices, or other proposals are necessary?
- Is it possible to implement a reliable and full-operational mesh network with mobile devices without compromising the delay and battery autonomy?

Some recent proposed solutions enable the interconnection of devices in a mesh network configuration at Layer 2. These proposals explore new interconnection opportunities on mobile devices that were not possible until now, and launch new research challenges to be successfully addressed. Therefore, the main objective of the current work is the implementation of a mesh network using handheld mobile devices. Related to this main goal there are two other objectives, as follows:

- Optimize the battery power management of mobile devices within a mesh.
- Limiting the delay in the mesh configuration, diminishing the setup and the recovery time after a node failure.

The current dissertation used the research method designated by the Design Science Research Methodology. This research method was fundamental to structure and plan all the necessary work to develop and evaluate a mesh network with mobile devices.

Figure 1.2 visualizes the methodology process we have used during the current work.

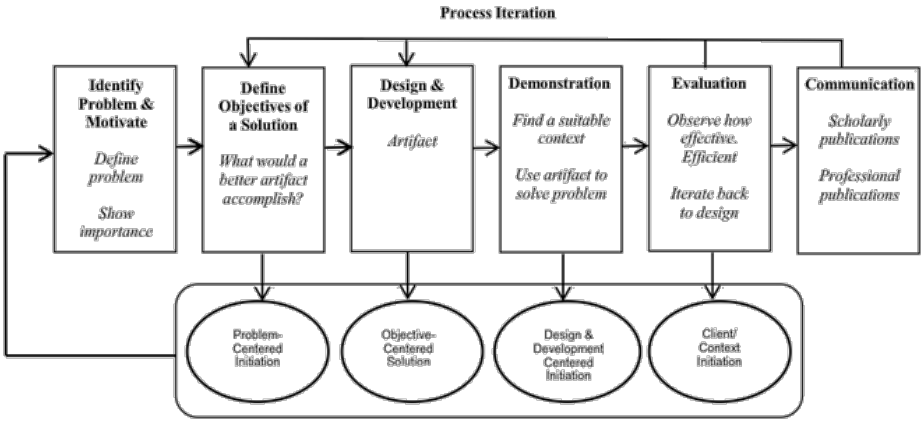

Figure 1.2 – Design science research methodology process model

Based on the methodology shown in Figure 1.2 it is possible to detail further the main tasks that within each phase of that methodology we have performed, such as following described:

- Problem Identification: it is performed a problem analysis regarding to mesh networks implementations on mobile devices. That analysis was based on existing solutions and the problems that that cannot be solved.

- Define Objectives for a solution: this stage defined the work requirements, it also defined the evaluation criteria, which is based on a test scenario to evaluate the performance and viability of the mesh network.

- Design and Development: the process of design and development is to implement a mesh network with mobile devices. On each iteration will be evaluated the feasibility of the implemented features. As the implementation is based on the existing standard, it is necessary to make an incremental implementation of various specifications.

- Demonstration: the demonstration phase is based on functional and non-functional tests on the mesh network. On this phase, results will be generated using essentially a testbed.

- Evaluation: the mesh network proposal is subjected to the analysis and evaluation, based on the results collected on the demonstration phase. On this phase is verified if existing problems with the development should be corrected in the next iteration of development process.

- Communication: finally, the communication concerns in the writing of the dissertation and its final presentation and discussion. In additional, several scientific papers should disseminate the diverse results obtained from this work.

The major contributions of the current work, in the area of mesh networks, focus on enhancing the delay and the battery autonomy of mobile terminals without compromising the global mesh performance. In this context, we intend to study and compare the different solutions, including the standard one, for implementing a real mesh network on mobile devices. Additionally, we aim to study a new energy metric as an important criterion to discover the more convenient paths across the mesh infrastructure.

This current work has contributed to the research in the area of mesh networks with two publications: “Mesh Networks for Handheld Mobile devices”, a paper accepted and presented during the Conftele conference [3]; and a pending submited chapter [4] “Wireless Mesh Sensor Networks with mobile devices: a comprehensive review” for the Handbook of Research on Advanced Wireless Sensor Network Applications, Protocols, and Architectures.

The current document has the following structure:

- Chapter 2 presents a review of the state of the art. It is divided in 5 sections. The first section is about projects that use mesh networks with mobile devices. Next, we discuss various solutions for protocol routing and path selection in mesh networks to understand if these solutions are compatible with the operation of mobile devices. The third section studies routing metrics for wireless mesh networks. Given the fact that a mobile device has to take into account the energy consumption, the next section introduces energy-aware routing metrics and the viability to work on mesh networks. The last topic discusses some relevant results obtained from two testbeds.

- In Chapter 3 is described the solution and implementation for implementing a mesh network with mobile devices. It is also shown the necessary steps to reach the final formula for the two proposed metrics. This chapter is divided into 4 sections: mesh network protocol selection, mesh networks in mobile devices, new energy-aware metric for IEEE802.11s and integration of new metrics in mobile device.

- Chapter 4 presents four testbeds scenarios. The first test was made with Open802.11s to verify if the network topologies and test procedures/configurations were appropriated. In a second test, two protocols were compared (Open802.11s and B.A.T.M.A.N-advanced) to decide which one is the more suitable for a WMN with mobile devices. Afterwards, it was build another test to verify if there is any kind of limitation of the mobile device when operating in a mesh network. Finally, the last real test was performed to study the new energy-aware metrics.

- Chapter 5 concludes the current contribution and points outs some relevant future work in the related area of the current work.

2. Literature review

Mesh Networks have recently become an area of great interest in the field of research and have contributed to major advances in the area of wireless networks.

This chapter analysis existing projects that use mesh networks with mobile devices in order to understand which of the layers of the OSI model are the more appropriate for implementing this type of network (Section 2.1).

In addition, existing routing protocols for mesh networks were analyzed to identify which ones can be more suitable to be deployed in mesh networks with mobile devices (Section 2.2). At a later stage, several metrics for mesh networks were analyzed to understand their characteristics and if they can be used to control how the routing paths are selected within the mesh infrastructure (Section 2.3). Because of the previous study, we have concluded that there are no metrics considering the battery status and, regarding that, it urges intensifying the research in the area of energy-aware routing metrics to increase the battery autonomy of mobile devices without penalizing too much the mesh performance (Section 2.4). Finally, a survey of the existing testbeds with mesh networks was made (Section 2.5).

2.1. Mesh networks implemented with mobile devices

Mesh Networks on mobile devices have been the focus of various research projects and feature a wide variety of proposals and solutions for each specific case study. It is then necessary to analyze and compare current relevant solutions, finding out the most interesting features to be implemented, especially in mesh scenarios with handheld mobile devices.

The implementation of a mesh network in mobile devices can be done at different layers of the OSI model. The standard mesh solution [2] is implemented at Layer 2 (data link layer). There are also implementations at Layer 2.5 (with additional software between Layers 2 and 3), Layer 3 (Network) and Layer 7 (Application).

a. Open Garden

The Open Garden [5] is an existing implementation of mesh networks and mainly focuses on establishing a connection among devices in situations where there is no coverage or network capacity. It allows routing and multihop discovery so that users can share their own Wifi/3G/4G connection, which operates as a model of "crowd-source bandwidth". This solution uses the application layer to implement the mesh network.

The main advantage of Open Garden is to offer a mesh network without requiring a root access on the mobile device. This project generated the FireChat application [6] which has often been referred to in the media [7] due to its extensive use in recent protests.

However, Open Garden contains bugs in relation to VPN operation in the Android operating system [8] [9] [10] and has crashed on versions higher than 4.4.

b. Serval Project

The Serval Project [11] is a mobile ad-hoc network that allows mobile phones to communicate when cellular coverage fails and is designed to operate in diverse scenarios such as rural, remote and disaster. The main objective is to achieve a basic communication service to everyone, with a special focus in rural areas. This project uses both the network and application layers to implement a mesh solution.

The Serval Project provides end-to-end privacy communication on voice calls and text messages using strong 256-bit Elliptic Curve Cryptography (ECC). This is one of the main advantages of the project, as it is a highly valued feature in communication systems. The ECC mechanism provides authentication and digital signature, data encryption and decryption, and identity management for the mesh network.

The roadmap of the project demonstrates that this is one of the most solid mesh networks for mobile devices. The team also developed specific hardware that can communicate with devices [12], achieving a broader coverage for the network.

c. SPAN Framework

The Smart Phone Ad-Hoc Networks (SPAN) [13] is an open source implementation of a generalized Mobile Ad-Hoc Network (MANET) framework. The main objective of the project is to provide a framework that is able to support any routing protocol that

works on a mesh network. This framework is injected into the existing Android network stack between OSI layers 2 and 3.

The SPAN project began as an internal investigation based on Wireless Tether for Root User, further developed by the MITRE Corporation and made available as open-source code on a GPLv3 license in December, 2011. SPAN is also based on the Serval project, with the main difference being the ability to use an arbitrary routing protocol. This is the opposite of what occurs in the Serval Project, which only uses the Better Approach To Mobile Ad-hoc Networks (B.A.T.M.A.N) routing protocol.

The possibility of multihop (shared with other projects) allows telephones with cellular access to perform a gateway role in the mesh network, providing Internet access to other devices. The most notable advantages offered by SPAN are the transparent functioning of the superior layers in relation to the network layer and the option to choose an arbitrary routing algorithm.

d. Commotion

Commotion [14] is an open-source project that uses wireless devices to create mesh networks. The project was developed by the Open Technology Institute in 2011 and launched in March 2013.

The main objective of Commotion is to provide an easy way to implement a mesh network for a wide audience through a specific software package. Commotion implements the mesh network at Layer 3.

The project is based on various open-source projects. Some of these projects are the Optimized Link State Routing Protocol daemon (OLSRd), which is used as the routing protocol for mesh network, OpenWRT, which is the operating system installed on routers and the Serval Project [11] that allows the use of security mechanisms for communication.

The project's main focus is to create a mesh network based on access points that can be extended to Android mobile devices. However, the main disadvantage is that it relies on a “Commotion infrastructure” for the terminals to communicate between each other.

e. Summary

Table 2.1 - **Comparison between mesh network projects with mobile devices** presents the more important characteristics that allow us to compare the mesh network projects with mobile devices discussed previously.

Table 2.1 - Comparison between mesh network projects with mobile devices

	Implementation Layer	Multihop	Standardized	Routing Protocol
Open Garden	Application (7)	Yes	No	Proprietary
Serval Project	Network and Application (3,7)	Yes	No	BATMAN
SPAN	Layer 2.5	Yes	Yes	Adaptable
Commotion	Network (3)	Yes	No	OLSRd

The currently study concludes that there is no project that uses Layer 2 to implement the mesh network. So, none of the projects mentioned above follows the recommendations and specifications of IEEE 802.11s [1]. A very important reason for choosing Layer 2 to implement the mesh network on mobile devices is to have more information available from the physical and MAC layers, which can be used by the routing algorithm. Additionally, operating at Layer 2 has strong benefits, by reducing the time delay to react to unexpected problems and to converge to a new operational configuration, as compared with implementations at higher layers.

In terms of routing protocols, each project chooses a different protocol. Nevertheless, the SPAN allows the usage of more than one routing protocol without any significant changes in the network configuration

2.2. Path selection and routing protocols

The routing protocols for mesh networks can be classified into three main categories: reactive (on-demand), proactive and hybrid. For the on-demand routing protocols, the path between a sender node and a receiver node is created only after the former transmitted a packet to the latter node. Conversely, in proactive protocols, each node maintains path information to all other nodes on its routing table before any transmission. Finally, the hybrid solution can support both categories described above. In the following text, is discussed several routing protocols for WMNs.

a. AODV

Ad-hoc On-Demand Distance Vector (AODV) [15] is a reactive-routing protocol for WMN. As its name indicates the algorithm is based on a distance vector protocol.

The path discovery is done by sending a message route request (RREQ) to the neighbors with the destination address and the sequence number. The use of a destination sequence number is a mechanism to avoid loops, similar to other algorithms based on distance vector. The nodes that receive the message increment the sequence number and update their routing table.

When the destination node receives the request message, it responds with a route reply (RREP) message back to the requesting node. When an intermediate node does not know the route that intermediate node sends a route error (RERR) message to the source node and the path discovery process is repeated.

b. HWMP

The Hybrid Wireless Mesh Protocol (HWMP) is defined in IEEE 802.11s [2] as the default routing protocol at the MAC layer. The HWMP protocol is based on Ad-hoc On Demand Distance Vector (AODV) and can be categorized as a hybrid solution because it supports the following modes: reactive (on-demand mode) and proactive.

On-Demand Mode

In On-Demand mode, a Path Request (PREQ) message is broadcasted by a Mesh STA (source) with information about the MAC address of destination Mesh STA. All PREQ messages contain a unique sequence number that is used to know the freshness of the PREQ on receiver. When an intermediate Mesh STA receives a PREQ, it creates or updates the path to the source only if the sequence number is higher than the previous know one or it has the same value but with a better metric. If the intermediary Mesh STA does not have the path to the destination, then the received PREQ message is forwarded to the next node until the destination Mesh STA is reached. When this destination node receives the PREQ, it sends a unicast Path Reply (PREP) message to the source Mesh STA.

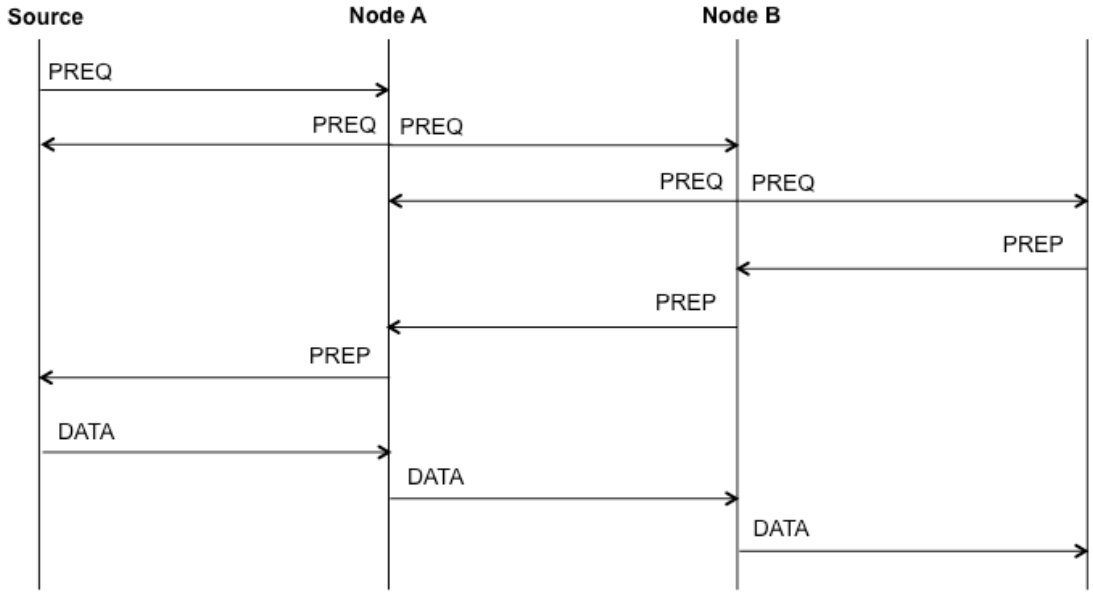

Figure 2.1 - HWMP on-demand route discovery

Proactive Mode

In the proactive mode, one of the nodes is selected as the root node. This node periodically sends PREQ messages. These PREQ messages are sent in broadcast to the neighbors. Each node after receiving a PREQ message sends a PREP back to the root. Alternatively, any node can send to the root a Route Announcement (RANN) message that enables the former node to build on-demand a path to the root node. Thus, the root gets a routing table fulfilled with all the possible paths traversing the mesh, forming a tree-based network topology.

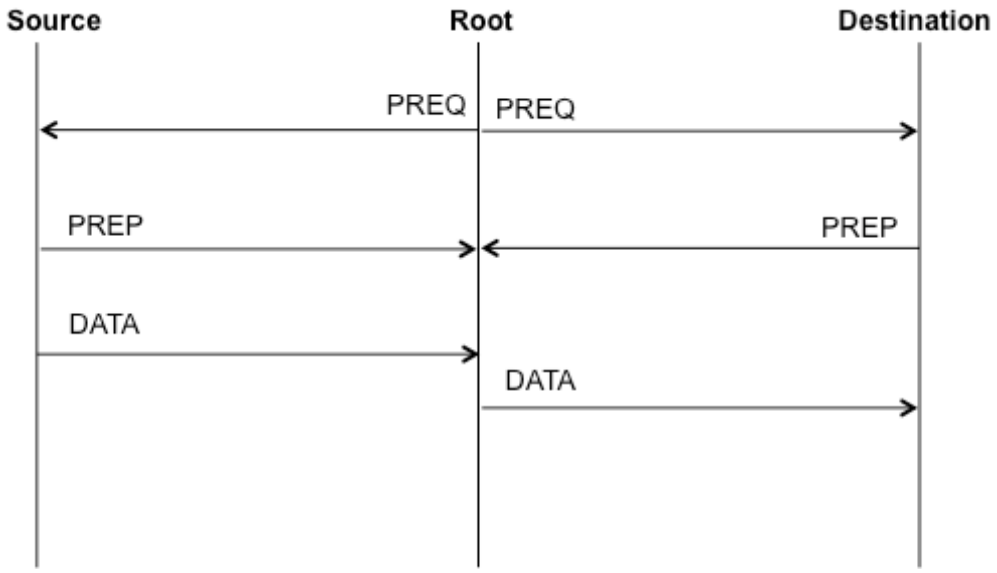

Figure 2.2 – HWMP proactive mode route discover

c. B.A.T.M.A.N

The B.A.T.M.A.N. [16] is a proactive routing protocol for WMN. The operating principle is to spread the knowledge about the best destination paths among all nodes. Each node receives and maintains only the information about the next hop towards remainder destination. Another important feature is the flooding mechanism based on events that prevents the occurrence of contradictory messages about the network topology (usually the reason for the existence of loops) and limits the number of messages in the network.

Each node *n* broadcasts an originator message (OMG) to inform the neighboring nodes of its existence. OMG messages are small packets that contain information about the original address, the address of the node that transmits the packet, Time to Live (TTL) and sequence number (SQ). Neighbors resend the message to its neighbors to inform the existence of the node *n*. Each node can make a re-broadcast of an OMG only once, and only if it is received by the node that is currently the best next hop to the creator of the OMG.

Currently the algorithm exists in two different versions: Batmand (Batman daemon) and Batman-advanced (Barman-adv). The difference between the two versions is how they are implemented, although the main operation is the same. The Batmand is implemented at layer 3 of the OSI model while Batman-adv works at layer 2.

The main differences are related to the implementation in distinct layers. The Batman-adv only needs MAC addresses to work while Batmand requires IP addresses. Batman-adv emulates an Ethernet bridge, causing all nodes appear to be connected by a direct link; so the above layer protocols are unaware of the multihop network topology.

d. DSDV

Destination Sequenced Distance Vector (DSDV) [17] is a proactive routing protocol for WMN based on Bellman-Ford algorithm.

In this protocol all the nodes keep in their routing table the next hop record and the number of hops required to reach the possible destinations from this node, where each entry is associated with a sequence number.

The updates are transmitted periodically or immediately when significant topology changes are detected. The update can be done in two ways: in “full dump” the node

transmits the complete routing table or “incremental update”, where the node sends only the new entries.

When a node receives an update message (full dump or incremental) it compares the information to that saved in the routing table and considers the entry with the most recent sequence number as the next to be used in the current routing decision. If the sequence number is equal then it considered the entry with the best routing metric.

Due to its architecture and operation, DSDV is suitable for networks with few devices, as they require regular incremental updates to their routing tables. This feature cause certain limitations related to battery consumption, which is a major drawback in networks with limited resources in terms of energy.

e. OLSR

The Optimized Link Sate Routing (OLSR) [18] is a proactive routing protocol for WMN based on classic link state routing.

The basic operation of OLSR is the same as the classic link-state routing, although with some additional features. One of the differences from the classic version is the use of MultiPoint Relays (MPR) to forward the packets through the network. This feature reduces the number of transmissions required to create the network topology and the number of control messages to keep the network operating in a correct state. This protocol has two types of control messages: neighborhood and topology messages, called respectively Hello messages and Topology Control (TC) messages. The Hello message has the function of neighbor discovery and TC messages are used for topology dissemination.

The OLSR protocol is especially useful for large and dense networks because the MRP’s technique works well in these scenarios.

f. BABEL

BABEL [19] is another proactive routing protocol for mesh networks that follows the distance-vector routing principles. It is based on DSDV [17] and the main difference to the classical distance-vector protocols is that is loop-free. This problem is avoided by using feasibility conditions and sequence numbers.

For selecting the more suitable route the BABEL protocol uses historical information about link quality to prevent a node changing several times the routes to the destinations, limiting the network instability.

In order to discover paths, each node periodically sends hello messages to its neighbors with a sequence number. Furthermore, "I heard you" messages (IHU) are also sent as replies to neighbors that previously sent hello messages. So based on these two messages is calculated the cost of each link connection.

The BABEL protocol has limitations for stable and large networks because their operation is based on constant updates of routing tables, which causes a high amount of traffic. This operation may cause a huge energy consumption for devices and for this reason this protocol is not adequate for networks with mobile devices.

g. DSR

Dynamic Source Routing (DSR) [20] is a reactive protocol conceptualized for mesh multi-hop wireless and uses IP source routing. The main characteristic of DSR is that each node stores the complete path to destination instead of the next hop (unlike AODV).

As such, all the DSR messages contain the list of all necessary nodes to reach the destination. This protocol allows the network to be self-organized and self-configured without an infrastructure.

h. Summary

Table 2.2 summarize the main characteristics of the discussed mesh routing protocols.

Table 2.2 - Comparison between mesh routing protocols

	Type	Routing Metric	Energy Efficient	Mobility	Load Balancing	Reliability
HWMP	Hybrid	Airtime	No	Yes	No	Yes
BATMAN	Proactive	Transmission Quality (TQ)	No	Yes	Yes	Yes
AODV	Reactive	Hop, ETX, ETT or iAWARE	No	Yes	No	Yes
OLSR	Proactive	Hop, ETX or ETT	No	Yes	No	Yes
DSDV	Proactive	Hop	No	Yes	No	Yes
BABEL	Proactive	Hop, ETX	No	Yes	No	Yes
DSR	Reactive	Hop	No	Yes	Yes	Yes

Regarding the protocol types, the HWMP is the only that can operate in either proactive or reactive mode. The B.A.T.M.A.N, OLSR, DSDV and BABEL are proactive routing protocols. The reactive algorithms are AODV and DSR.

All analyzed protocols implement mobility and reliability features. The load balancing feature is only supported by DSR and B.A.T.M.A.N.

It can be seen that none of the analyzed protocols consider the energy consumption. This is one of the main critical issues and concerns in in networks with limited resources in terms of energy and routing protocols for mesh should be aware of it. We will discuss this novel aspect in section 2.4.

2.3. Metrics for mesh networks

The study of routing protocols has been a very active area of research in mesh networks. In this sphere, the type of metric used is important in controlling the network operation. The routing metrics have diverse goals such as to optimize performance, maximize battery lifetime, increase throughput and minimize latency.

a. Hop Count

In Ad hoc networks, the hop count routing metric is very common due to its very simple implementation.

The hop count is the simplest metric routing because it is evaluated as the number of necessary hops to reach the destination. For this reason, the hop count is by default used by many major routing protocols such as AODV, DSR, DSDV and OLSR.

The main advantages of this metric are not only its deployment simplicity but also its stability, especially for networks with mobility, where it reduces the number of changes applied to the network when compared with other metrics.

On the other hand, the fact that all links are managed in the same way is a disadvantage, as parameters such as link capacity, diversity mechanisms, link load and interference are not taken into account, which may result in path choices being made with high delays and low throughput. Consequently, the use of hop count may result in degraded network performance.

b. Airtime Metric

The airtime [21] is the default routing metric for mesh networks specified in the IEEE 802.11s standard [2]. Airtime analyzes information related to the utilization of the channel during a transmission, for example, overhead, transmission time and frame error rate. It represents the amount of channel resource consumed by transmitting a frame over a particular link.

The Airtime metric is given by following expression:

$$C_a = \left[O_{ca} + O_p + \frac{B_t}{r}\right] x \ \frac{1}{1 - e_{pt}} \qquad (1)$$

where O_{ca} , O_p and B_t are constants that representing, respectively, channel access overhead, protocol overhead and the number of bits in the test frame. The term r is the data rate in Mbps and e_{pt} is the frame error rate. The way of measuring e_{pt} is not specified in 802.11s [2].

Figure 2.3 illustrates the two-way Airtime calculation process .

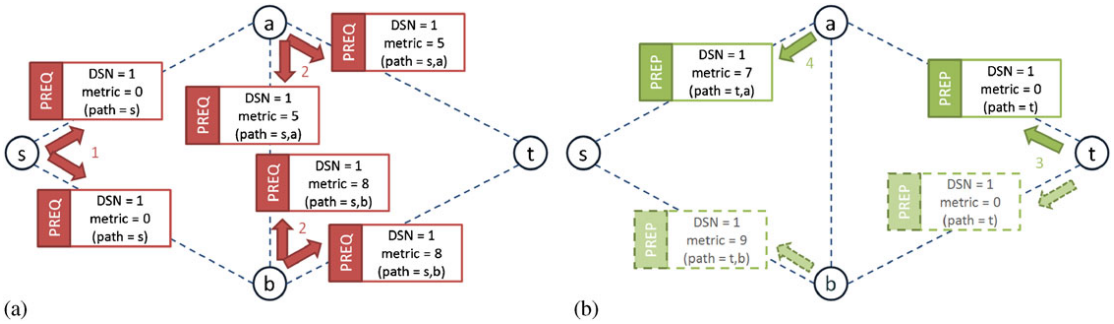

Figure 2.3 – PREP and PREQ exchanged messages to calculate de Airtime Link metric cost [22].

In the first step node *s* broadcasts a PREQ message, next *a* and *b* re-broadcasts the PREQ with the metric value updated towards *s*. The next phase (response), *t* sends a PREP in response to PREQs. Finally, *a* or/and *b* updating the metric and forward the PREP towards *t*.

The airtime does not define any load balancing mechanism that can lead to network paths through congested links. Ignorance of the existence of intra-flow interference can also have a negative impact on the network performance. For these reasons, analyzing the trend of airtime metric, one can not predict the current state and quality of a network link. [23]

c. Transmission Quality (TQ)

The TQ [24] is the routing metric used by B.A.T.M.A.N. and represents the quality of the path for a specific originated message. TQ as the name suggests, represents only the capability of sending data and not receiving. The local transmission quality value can not be directly calculated, because the source node does not have sufficient information. For this reason Receive Quality (RQ) and Echo Quality (EQ) need to be calculated.

The local Transmission Quality is given by in (2).

$$TQ = \frac{EQ}{RQ} \qquad (2)$$

The RQ value is calculated as the fraction of the received OGMs from a node and the number of expected ones. The EQ value is computed as the fraction of locally generated OGMs and the number of times that it receives the rebroadcasted OGM back from a neighbor.

The Path TQ represents the total transmission quality for an entire path. When a node generates a OGM, the TQ value is set to 255 and after is broadcasted. The value is updated by neighbors with the local TQ value and rebroadcasted, until reach the destination node. Therefore, the total TQ is expressed in expression (3).

$$PathTQ = PathTQ \times LocalTQ \quad \textbf{(3)}$$

To assure that only the best bidirectional links are chosen, the TQ metric assigns a penalty to links that communicates in one direction but not via the other. Therefore if the RQ value is poor the path TQ will be affected significantly. Thus, its expression is given in (4).

$$PathTQ = PathTQ \times (1 - (1 - RQ)^3) \quad \textbf{(4)}$$

Another penalty is relative to the number of hops. This penalty is assigned in relation to a fixed value (TQ_HOP_PENALTY) every time that a OGM is rebroadcasted. The Hop penalty is applied according to equation (5).

$$PathTQ = PathTQ \times (1 - TQ\_HOP\_PENALTY) \quad \textbf{(5)}$$

These penalties are very important to save bandwidth, reduce delay and avoid the reduction of throughput caused by self-inference.

d. Expected Transmission Count (ETX)

The ETX [25], as its name indicates, is defined by the number of transmissions required to transmit a packet successfully through a wireless link. The total number of the transmissions associated to a network path is the sum of transmission through all the individual links of that path.

To calculate the metric ETX associated to an individual link, each node periodically sends a fixed number of *N* probe frames in broadcast over a fixed time period *T*. Upon receiving the messages, the nodes calculate the forward delivery ratio d_{fwd} (probability that the packet was successfully received at the source node) by the expression (6).

$$d_{fwd} = \frac{R_f}{N} \quad (6)$$

Where R_f is the number of received frames from the previous sent N frames in the forward direction. When the nodes receive a probe frame, they count the probe frames received via piggyback in message d_{fwd} in order to compute the reverse delivery ration d_{rvs} (probability that acknowledgment of the packet was successfully received at the source node).The ETX metric can be evaluated as shown in (7)

$$ETX = \frac{1}{d_{fwd} \times d_{rvs}} \quad (7)$$

The main advantages of ETX are following explained. It considers packet loss ratios in its computation. It is an isotonic function that guarantees an efficient computation of paths with a low computing cost. It ensures a loop-free final result. However, the ETX does not consider interference or the characteristic that links may have different transmission rates. The ETX is not suitable for networks with high mobility because it takes some time to compute the necessary values (d_{fwd} and d_{rvs}).

e. Modified Expected Transmission Count (mETX) and Effective Number of Transmission (ENT)

In order to overcome some disadvantages of ETX, mETX and ENT have been both developed with certain changes on the link variance of the transmission to make the ETX more robust and to consider the routing quality. [26]

Modified ETX is evaluated in (8).

$$mETX = exp\left(\mu + \frac{1}{2}\sigma^2\right) \quad (8)$$

Where μ represents mean packet loss ratio and σ^2 is the variance of packet loss ratio. The main difference between mETX and ETX is that mETX considers probe losses. Based on the corrupted bit position in the probe message, the mETX metric computes bit error probability.

ENT is expressed as indicated in (9).

$$ENT = exp\left(\mu + \frac{1}{2}\delta\sigma^2\right) \quad (9)$$

Where δ is the strictness of the loss rate requirement. ENT is very similar to mETX, with the difference of the parameter δ, that allows an additional degree of freedom.

Both, mETX and ENT are ETX improvements, although they still fail to address intra-flow (nodes on the same path or flow on the same channel competing between each other for channel bandwidth) and inter-flow (caused by other flows that are operating on the same channel) interference problems in network.

f. Expected Transmission Time (ETT)

The ETT [27] is an improvement over the ETX routing metric described above. In order to overcome the main disadvantage of this metric a parameter was added that considers the link transmission rates.

Thus, ETT can be expressed as show in (10).

$$ETT = ETX \times \frac{s}{b_t} \quad (10)$$

Where *s* is the packet size and b_t is the transmission rate of each individual link. Therefore, with these additional parameters is possible to take into account the average time of a link transfer regardless of whether or not the transmission is successful.

However, the ETT still has the disadvantage of not fully analyzing the intra-flow and inter-flow interference in the network. This drawback makes it possible for this routing metric to choose a route that uses only one channel instead of other alternative routes with higher channel diversity.

g. Weighted Cumulative ETT (WCETT)

The WCETT [27] is an improvement over the ETT, also considering intra-flow interference.

WCETT is expressed as illustrated in (11).

$$WCETT = (1-\beta) \times \sum_{i=1}^{n} ETT_i + \beta \times \max_{1 \le j \le k} X_j \quad \textbf{(11)}$$

Where *n* represents the total number of nodes, *k* specifies the total number of different channels used in path, β is a tunable parameter with values between 0 and 1, X_j is the sum of transmission times required for all hops on channel j and is defined as shown in (12).

$$X_j = \sum_{i \text{ uses channell } j} ETT_i \quad 1 \le j \le k \quad \textbf{(12)}$$

From (11), the first parameter is the sum of each ETT link and favors the shorter and high quality paths. The second parameter is the sum of ETT for all links of a particular channel, choosing that which has a higher value. The result is, therefore, a higher value for paths with a large number of links operating on the same channel, which hence favors channels with high diversity and low intra-flow interference.

However, there are still several problems with this metric. In fact, the WCETT metric only considers the number of links that operate on the same channel and its ETTs, but it does not take into account the location of these links. WCETT is not isotonic (not ensure that the order of the weights of two paths are preserved if they are appended or prefixed by a common third path), what makes it difficult to use in efficient algorithms to find minimal weight paths, such as Bellman-Ford or Dijkstra's algorithm.

h. Interference Aware Routing Metric (iAWARE)

The iAWARE [28] appears as the first routing metric that takes into account intra-flow and inter-flow interference. The iAWARE uses ETT and values of Signal to Noise Ratio (SNR) and Signal to Interference and Noise Ratio (SINR) in order to obtain information about interference with its neighbors.

The iAWARE metric is expressed by the equation shown in (13).

$$iAWARE = (1-\alpha) \times \sum_{i=1}^{n} iAWARE_i + \alpha \times \max_{1 \le j \le k} X_j \quad \textbf{(13)}$$

where *Xj* is the inter-flow interference in the network, *k* is the total number of channels, *n* is the total number of links, α represents the trade-off parameter between intra-flow and inter-flow interferences in path.

The term $iAWARE_i$ is detailed in (14).

$$iAWARE_i = \frac{ETT_i}{IR_i} \quad \textbf{(14)}$$

where IR_i is the interference ratio of the link *I* and is given by the equation shown in (15).

$$IR_i = \frac{SINR_i}{SNR_i} \quad \textbf{(15)}$$

The term and X_j is expressed in (16).

$$X_j = \sum_{i\ uses\ channell\ j} ETT_i \quad 1 \leq j \leq k \quad \textbf{(16)}$$

The iAWARE metric is not isotonic and cannot be used in a link state routing protocol. The interference is only calculated on the receiver side and this parameter is ignored on the sender side. This routing metric also captures the effects of variation in link loss-ratio as well as the differences in transmission rate.

The iAWARE uses the value of the ETT and IR to evaluate the metric. If the IR value is higher than the value of ETT, the term $iAWARE_i$ will have a lower value and consequently the metric will choose a path with a lower ETT but higher interference. So, the main disadvantage is that it gives a greater weight to the ETT compared to the interference link.

i. Summary

Table 2.3 - Comparison between mesh metrics

	Quality-aware	Loss ratio	Data rate	Packet size	Intra-flow	Inter-flow	Isotonic	OSI Layer
Hop count	No	No	No	No	No	No	Yes	Network, Link
Airtime	Yes	Yes	No	Yes	No	Yes	Yes	Link
ETX	Yes	No	No	No	No	No	Yes	Network
mETX	Yes	No	Yes	Yes	No	No	Yes	Network
ENT	Yes	No	Yes	Yes	No	No	Yes	Network, Link
ETT	Yes	Yes	Yes	Yes	No	No	Yes	Network
WCETT	Yes	Yes	Yes	Yes	Yes	No	No	Network
iAWARE	Yes	Yes	Yes	Yes	Yes	Yes	No	Link

The metric types listed in Table 2.3 can be divided into two categories: Topology Based and Active Probing Based. In Topology Based category is inserted the Hop Count metric that is based exclusively on the network topology to make decisions. All the others are Active Probing Based and perform active measurements and use probe packets to estimate the metric values.

The main objective of the metrics in Table 3 are to minimize the delay and to maximize the probability of data delivery. The only metric that allows the detection of intra-flow and inter-flow interference is iAWARE, however this metric not have an isotonic behavior.

After analyzng the existing metrics TQ (B.A.T.M.A.N.) and Airtime (standard IEEE802.11s), it turns out that none of the used metrics take into account the power consumption, making it a major drawback to use in sensor networks. In the next section is presented proposals for metrics that already take into account the energy consumption, but so far have not been sufficiently studied in mesh networks with mobile devices.

2.4. Energy-aware routing metrics

Energy consumption can be an important constraint in a network with mobile devices and therefore it is crucial that the routing protocol should use a metric that has this aspect into account. Energy-aware routing metrics can be managed at a single layer or using a cross-layered design. In that sense the parameters used in the metric can be obtained from some layers such as: Physical Layer (transmission power control), Data Link Layer (MAC protocol operation) and Network Layer (routing algorithm).

The main objectives of such metrics are: minimization of overall energy consumption to maximize the time until a node turn off due to lack of battery.

a. Minimum Total Power Routing (MTPR)

The MTPR [29] was one of the first proposed metrics that consider the power consumption. The main objective of this metric is selecting the path with the minimum total transmission power. Therefore the metric calculates the total energy consumed over the route and makes a decision based on that value.

If we assume a route $r_d = n_0, n_1, \dots, n_d$ where n_0 is the source node and n_0 is the destination node, the MTPR metric is expressed by the following equation:

$$P(r_d) = \sum_{i=0}^{d-1} T\,(n_i, n_{i+1}) \qquad \textbf{(17)}$$

$T(n_i, n_j)$ represents the energy consumed for a transmission over the hop (n_i, n_j).

A disadvantage of this metric is that it does not take into account the battery of the nodes. It is expected that nodes that need to forward a lot of traffic have a shorter battery lifetime and do not contribute to a reliable network operation.

b. Minimum Battery Cost Routing (MBCR)

In order to account the battery capacity for all nodes over the network, MBCR [30] is based in the sum of the remaining battery capacity for the all intermediate nodes. Assuming that all nodes have the same battery capacity in full mode, the battery cost function for node n_i is given by the equation shown in (18).

$$f_i(E_i) = \frac{1}{E_i} \quad \textbf{(18)}$$

where E_i is the residual battery capacity.

The total battery cost R_j among the complete path *p*, with *D* nodes is expressed by the equation (19).

$$R_j = \sum_{n_i \in p} f_i(E_i) \quad \textbf{(19)}$$

Finally, to discover the route that minimizes the total cost among the nodes, we select a route *p'* with the minimum battery cost, as shown in (20).

$$p' = min\{R_j \mid j \in P\} \quad \textbf{(20)}$$

Where *P* is the full set of possible paths.

If all nodes have the same battery cost then MBCR selects the sorter hop path. The main disadvantage of this metric is the fact that uses only the total path value of battery cost. In cases when the route selected have a large discrepancy between nodes, for example nodes with full-charged battery and others with a low level of battery energy, this can be a strong drawback because some nodes may be turned off.

In [31] the authors propose an improvement to MBCR, resulting in a battery capacity defined as a cost function considering the number of hops. This change is expressed in equation (18).

$$R_j = \frac{1}{H} \sum_{n_i \in p} f_i(E_i) \quad \textbf{(21)}$$

Where *H* is the number of hops for the path.

This main aim of this solution is reduce the total transmission power consumption and balance the energy consumption among all nodes. Although, it does not yet consider the individual cost of nodes.

c. Min-Max Battery Cost Routing (MMBCR)

MMCBR [30] is an improvement to the original MCBR to manage the nodes more fairly. MMCBR avoid nodes with very low remaining battery and ones with large battery capacity are favored to be elected to establish routes.

The objective of this metric is for each route, select the battery cost with the maximum value among all nodes. Hereupon we can change the Equation (18) to consider that as shown in (22).

$$R_j = max_{i \in p} f_i(E_i) \quad \textbf{(22)}$$

Then, the route is selected accordingly to (23).

$$p' = min\{R_j \mid j \in P\} \quad \textbf{(23)}$$

With MMCBR there is no guarantee of minimum total transmission power in all cases. This creates a tradeoff between the priority to an individual node and the overall system energy optimization.

d. Conditional Max-Min Battery Capacity Routing (CMMBCR)

CMMBCR [32] is a combination of two previously described protocols into on a single hybrid metric, MTPR and MMBCR. In the above metrics was considered a cost function but in this metric is used the battery capacity. A threshold parameter γ is used to define the minimum remaining percentage of battery capacity for each node that ranges between 0 and 100.

For each node j CMMBCR finds the minimum capacity R_j for all nodes in the route. Finally, the R_j and γ value is compared. If $R_j \geq \gamma$ then MTPR is applied, else the route is selected using MMBCR.

With this hybrid metric the objectives of maximize the lifetime of each node and use the battery fairly are both contemplated.

e. Maximal Residual Energy Path Routing (MREP)

MREP [33] is a metric proposed by Chang and Tassiulas based on the remaining battery capacity and the necessary transmission energy. The cost of a node is inversely proportional to the remaining battery capacity. The main objective of this

metric is to balance the energy consumption in all paths, through the choice of the path with maximum residual energy.

Considering E_j the residual energy at node j and $e_{i,j}$ the energy cost to send a packet from node i to node j, the authors proposed two metrics.

The reciprocal of the residual energy $d_{i,j}$ for a node is expressed by equation (24).

$$d_{i,j} = \frac{1}{E_j - e_{i,j}} \quad (24)$$

MREP using the lexicographical ordering, and compare two paths by the examination of the highest cost node on each path. The path with highest cost node has the lowest cost in lexicographical ordering..

Other proposals for a simpler implementation were presented by Jae-Hwan Chang and Tassiulas. The first modification stores and uses the largest element of the path for comparison and is given by (25).

$$D_i^{(n+1)} = \min\left[\min\left\{\max\left(d_{i,j}, D_j^{(n)}\right)\right\}, D_i^{(n)}\right] \quad (25)$$

where $D_i^{(n)}$ is the distance of the path from node i to the destination at step n. Other solution uses the Bellman-Ford equation as shown in (26).

$$D_i^{(n+1)} = \min\left[\min\left(d_{i,j} + D_j^{(n)}\right), D_i^{(n)}\right] \quad (26)$$

The fourth proposal is based on the number of packets that can be delivered with the remaining energy of the nodes. This can be expressed as shown in (27).

$$d_{i,j} = \frac{e_{i,j}}{E_j} \quad (27)$$

In [34] another expression to calculate the metric is presented as it is visualized in (28).

$$d_{i,j} = \frac{e_{i,j}^{x_1} E_i^{x_3}}{R_i^{x_2}} \quad (28)$$

where R_i is the residual energy at node i and x_1, x_2 and $x_3$represent weighting non-negative values .

A disadvantage of this metric is the difficulty of determine the remaing energy on each node. This metric aims to maintain the fairness of the entire network and always trying to use the paths where there is a maximum value of residual energy, however does not take into account the power consumption.

f. Expected Transmission Energy Route Metric (ETE)

ETE [35] is a metric for WMSNs with the aim of improving the Airtime Link metric to take into account the energy consumption. This metric tries to ensure that all nodes have an identical energy consumption and simultaneously keeping the average power consumption at low values. To make this possible ETE introduce a threshold mechanism. When the energy of a node is below the threshold, this one does not join the metric calculation and does not forward packets.

The ETE metric is expressed by the equation (29).

$$c'_a = \left[O_{ca} + O_p + \frac{B_t}{r} + \sum_{i=1}^{n_j} \frac{E_{init}}{100 E_i} \right] \frac{1}{1 - e_{pt}} + \frac{\sum_{i=1}^{n_j} \frac{E_{ic}}{E_{init}}}{n_j} \qquad (29)$$

where E_i is the remaining energy of node i after the transmission, E_{ic} is the energy consumption of node i and n_j is the number of nodes along the selected path. O_{ca}, O_p, B_t, r and e_{pt} are constants defined for Airtime Link Metric and described above.

The tests carried out by simulation show that the ETE has better performance in prolonging the lifetime of the network and maximize the throughput when compared with Hop Count and Airtime Metric.

g. Energy Aware Routing Metric for IEE802.11s (EARM)

In order to address the lack of metrics for standard 802.11s mesh networks, has been proposed a new metric to replace the Airtime Link [36].

This proposal takes into account the residual energy of a node by calculating the total energy consumed by a node based on its state. The metric defines five states, IDLE: the node is idle; CCA_BUSY: at this stage the node is busy; TX: the node is transmitting packets; RX: the node is receiving packets and SWITCHING: the node is switching from a channel to another one.

The residual energy $\mathrm{R}(\mathrm{n_i})$ for each node $\mathrm{n_i}$ is given by (30).

$$R(n_i) = E_{current}(n_i) - E_{con}(n_i) \quad (30)$$

where $\mathrm{E_{current}}(\mathrm{n_i})$ is the current energy of the node $\mathrm{n_i}$ and $\mathrm{E_{con}}(\mathrm{n_i})$ is the energy consumed by node $\mathrm{n_i}$. To determine the consumed energy is used the equation (31).

$$E_{con}(n_i) = Current(n_i) \times Voltage(n_i) \times Duration \quad (31)$$

where $\mathrm{Current}(\mathrm{n_i})$ is the current in Ampere and depends on the current state of the node (IDLE, CCA_BUSY, TX, TX and SWITCHING), the $\mathrm{Voltage}(\mathrm{n_i})$ is the electrical potential in Volts and the $\mathrm{Duration}$ is the time interval since the last energy update in seconds.

For each node $\mathrm{n_i}$ is calculated an energy cost $\mathrm{C}(\mathrm{n_i})$ expressed by the equation shown in (32).

$$C(n_i) = \frac{E_{init}(n_i)}{R(n_i)} \quad (32)$$

So, the total cost for a specific route p from source node $\mathrm{n_i}$ to destination node $\mathrm{n_d}$, is evaluated in (33).

$$E_p = \sum_{n_i \in p,\, n_i \neq n_d} C(n_i) \quad (33)$$

Then, the selected route l satisfies the condition shown in (34):

$$E_l = min\{E_p : p \in V\} \quad (34)$$

where V is the set of all possible routes.

The authors tested the metric by simulation and compared their behavior with the Airtime Link metric at the level of the throughput and battery lifetime. The proposed metric showed up better results for the battery lifetime, however it was slower to react to network changes.

h. Summary

Table 2.4 summarize the main characteristics of the discussed energy-aware routing metrics.

Table 2.4 - Comparison between energy-aware metrics

	Path Battery	**Node Battery**	**Energy Transmission**	**802.11s Metric Characteristics**
MTPR	No	No	Yes	No
MBCR	Yes	No	No	No
MMBCR	Yes	Yes	No	No
CMMBCR	Yes	Yes	Yes	No
MREP	Yes	Yes	Yes	No
ETE	Yes	Yes	No	Yes
EARM802.11s	Yes	Yes	No	Yes

As discussed above it has been found that there are some metrics (see Table 4) that take into account the energy consumption, but there are some limitations still present and in the specific case of mesh networks there are only two proposals which have only been tested by simulation.

As described in each of the energy-aware metrics discussed above there is always a tradeoff to manage. This tradeoff is between the energy consumption and delay/throughput.

An important factor when choosing a particular metric it is the compatibility with the routing protocol. The information needed to calculate the metric value depends on the knowledge that the protocol has about the network. The topology, dimension and specific characteristics of the network must always be considered in the routing metric choice.

2.5. Testbeds

There have been developed several testbeds with analysis and comparisons in mesh networks. In many of these tests is analyzed and compared the performance of different routing protocols.

On [37] is made an experimental evaluation of two protocol routing solutions: HWMP and B.A.T.M.A.N. In both cases, the mesh network has not been implemented with mobile devices. The tests performed examine three characteristics of the nodes: path and throughput stability and recovery after a node failure. The authors chose these three aspects because they are quite critical in real scenarios.

The results and conclusions show that the level of stability on path and throughput is better in BATMAN. However, in case of node failure, the HWMP recovers faster than BATAMN. The authors do not define any winner because both protocols have advantages and disadvantages.

In [38] an experimental analysis is performed between the following routing protocols: HWMP, B.A.T.M.A.N and OLSR, using a real world testbed. The paper presents some disadvantages in HWMP comparatively to OLSR and B.A.T.M.A.N.

The tests performed show that the maximum throughput of HWMP is significantly less than OLSR and B.A.T.M.A.N. The results also demonstrate that B.A.T.M.A.N had the highest value of throughput. The average round trip delay is shorter in HWMP due the need of discover a new route when a packet is lost. The authors conclude that the routing protocol defined in 802.11s standard [2] need to be refined.

The testbeds performed so far demonstrate that the routing protocols were not analyzed in mesh networks with mobile devices. The analysis of mesh networks performance in mobile environments requires special characteristics which are not taken into account in these tests, such as: mobility effects, battery life management, low debt and hardware limitations.

3. Solution and implementation

This chapter specifies the solution proposed within the current research work. Then, it also explains how this solution has evolved towards its final implementation into a real testbed for a mesh network with mobile devices. In the beginning of each section, a high-level description of a specific subject is always presented and, finally, at the final part of that section, all the necessary steps to deploy that functionality are finally discussed.

The solution proposed in the current work is based on the design and integration of a solution that incorporates mobile devices within a mesh networking infrastructure. Besides that, a new metric was also developed and implemented considering as a novelty some battery information (remaining energy, energy threshold, temperature, status and voltage) before any decision about packet forwarding through the mesh topology.

The proposed solution was developed under these four following steps:

i) Mesh network protocol selection (Section 3.1)

ii) Mesh networks in mobile devices (Section 3.2)

iii) New energy-aware routing metric for IEEE802.11s (Section 3.3)

iv) Integration of a new metrics in mobile device (Section 3.4)

The Figure 3.1 shows a typical mesh network formed by mobile devices and routers.

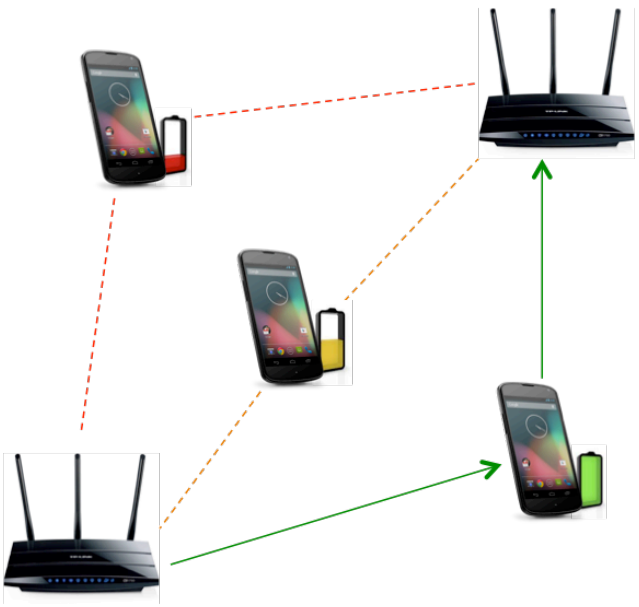

Figure 3.1 – Mesh network topology with mobile devices and energy-aware metric.

To save the battery energy in a fair way among all the mobile devices we advocate the usage of an energy-aware metric for the routing path decision. This metric could be particularly useful if it enables the exchange of any message between the two routers of Figure 3.1 through always the intermediate mobile device with the highest battery level. In this way, our main aim is to enhance the network lifetime.

The first step in the development of our solution was the choice of the OSI layer to deploy the solution, which is described in the following section.

3.1. Mesh network protocol selection

First reason for choosing protocols to be tested was based on the OSI model layer where the protocol would work. Therefore, on a first stage, a comparative survey of the existing protocols and their respective functioning layers was made (Section 2.2). The study was carried out and it was possible to conclude that the most adequate projects were Open802.11s and B.A.T.M.A.N-advanced.

Some reasons for choosing Open802.11s and B.A.T.M.A.N-advanced were: each of these routing protocols is implemented at Layer 2, and it is possible to implement it at the Linux kernel. In this way there is more information, available from the physical and MAC layers, which can be used by the routing algorithm. This allows more flexibility to extend the routing protocol and the metric to be used. Additionally, operating at Layer 2 has strong benefits, by reducing the time delay to react to unexpected problems and to take routing decisions, as compared with implementations at higher layers.

In order to determine which is the most suitable routing protocol and metrics for the implementation of a mesh network with mobile devices, we have studied two mesh projects: Open802.11s and B.A.T.M.A.N-advanced, described below.

The Open802.11s [39] [40] project is an open-source implementation of the ratified IEEE 802.11s wireless mesh standard. Its main objective is implementing 802.11s to run on any Linux device. The Open802.11s has been part of the Linux kernel since version 2.6.26. The 802.11s, proposes the Hybrid Wireless Mesh Protocol (HWMP) [41] as the default mandatory routing protocol, and Airtime Link Metric as the default cost metric to select the most convenient path among several nodes.

Open802.11s is a stable solution and implements the mesh routing protocol at Layer 2. It also enables greater flexibility to extend the routing algorithm and associated metrics, in order to improve the performance for mobile devices.

The B.A.T.M.A.N-advanced [16] has been designed by Freifunk, a wireless community based in Germany. The goal of this project is to replace OLSR with B.A.T.M.A.N. The protocol operating principle is to spread the knowledge about the best destination paths among all nodes.

B.A.T.M.A.N-advanced can be implemented on OpenWRT with the installation of a new kernel package. This project, proposes a proprietary routing algorithm (B.A.T.M.A.N) and routing cost metric (TQ - Transmission Quality). B.A.T.M.A.N-advanced also implements the mesh routing protocol at Layer 2.

In order to verify which of the two projects is the most suitable for integration into mobile nodes of a mesh, we have carried out a set of tests over a real testbed. That testbed was implemented with different topologies and some relevant results are presented and discussed for path discovery time, bootstrap time, throughput and recovery time after a node failure on 4.2. From the obtained results we can conclude that Open802.11s shows a better performance than B.A.T.M.A.N-advanced.

3.2. Mesh networks in mobile device

Following the choice of the mesh network protocol, the integration with the mobile device was made. As the initial step, after identifying the hardware requirements, we have made a comprehensive survey among the available mobile devices to compare their requirements. From this comparison, it was then selected the mobile device to be used in our research. We have also identified the main changes required for the integration of the mesh network protocol in that mobile device. These changes were the following ones:

i) Configure and build a new Android Kernel

ii) Configure and build wcn36xx driver

iii) Root the device

iv) Load the new Kernel and driver files into device

v) Configure device and create/join mesh network.

In Figure 3.2 the Android OS architecture is described, including all its four main layers, as follows: applications, application framework, libraries, and kernel.

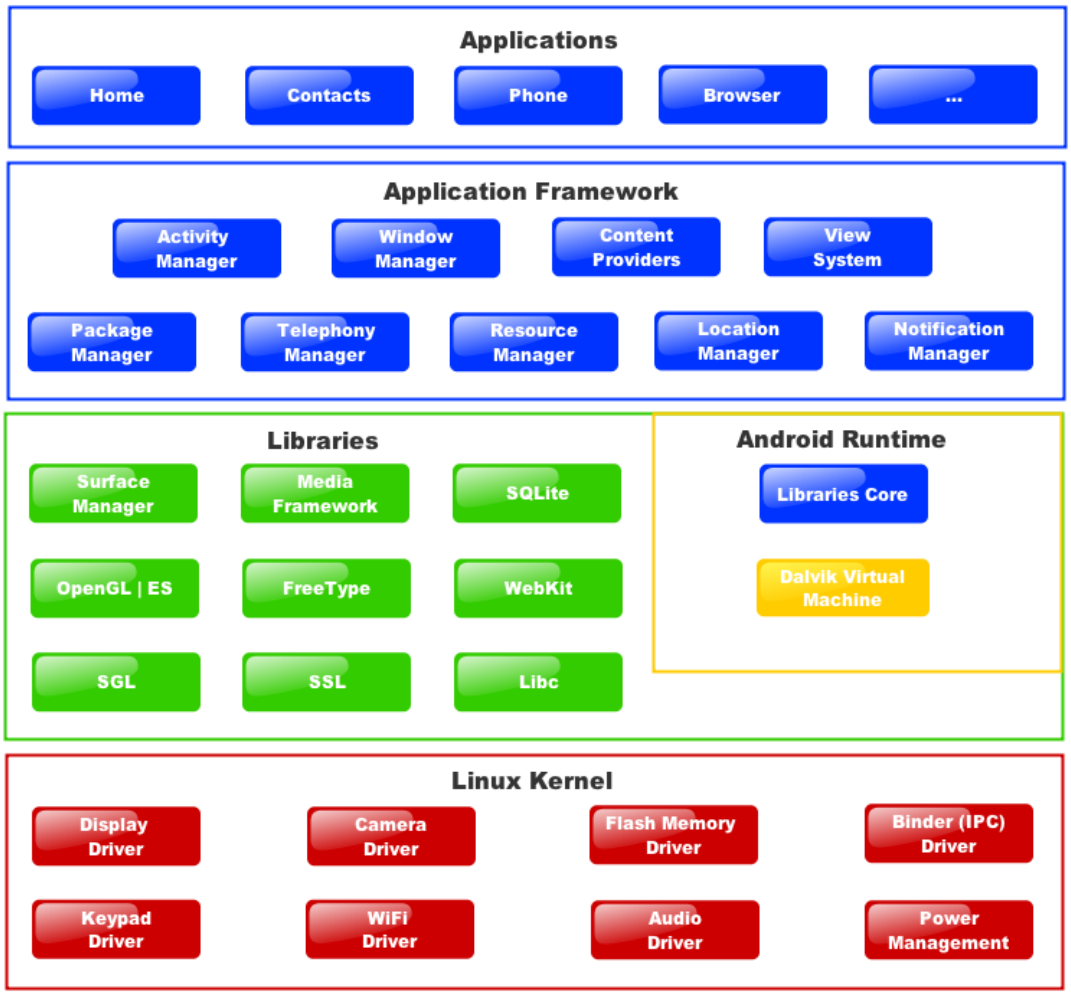

Figure 3.2 – Android OS architecture. [42]

As the above figure also shows, each layer of the Android architecture is composed by several modules. In particular, the Linux Kernel is the layer where the entire system is based because it is the bottom layer of this architecture.

In this case, this layer was the one that was repackaged and, after that, replaced. The main goal of this rebuilding and replacement is to allow Linux Kernel layer to be more flexible to manipulations. We have changed the WIFI Driver module to deploy new mesh parameters.

To modify the WIFI driver (see Figure 3.2), the manufacturer must provide the code to compile and so be able to build a new version of it with some different functionality. In this case, the WIFI driver selected was the wcn36xx (Figure 3.3), because it fulfills the above requirement.

Figure 3.3 helps to understand where in the Android architecture we can locate the WIFI driver wcn36xx is (i.e. Kernel layer) and its dependencies with other sub-modules.

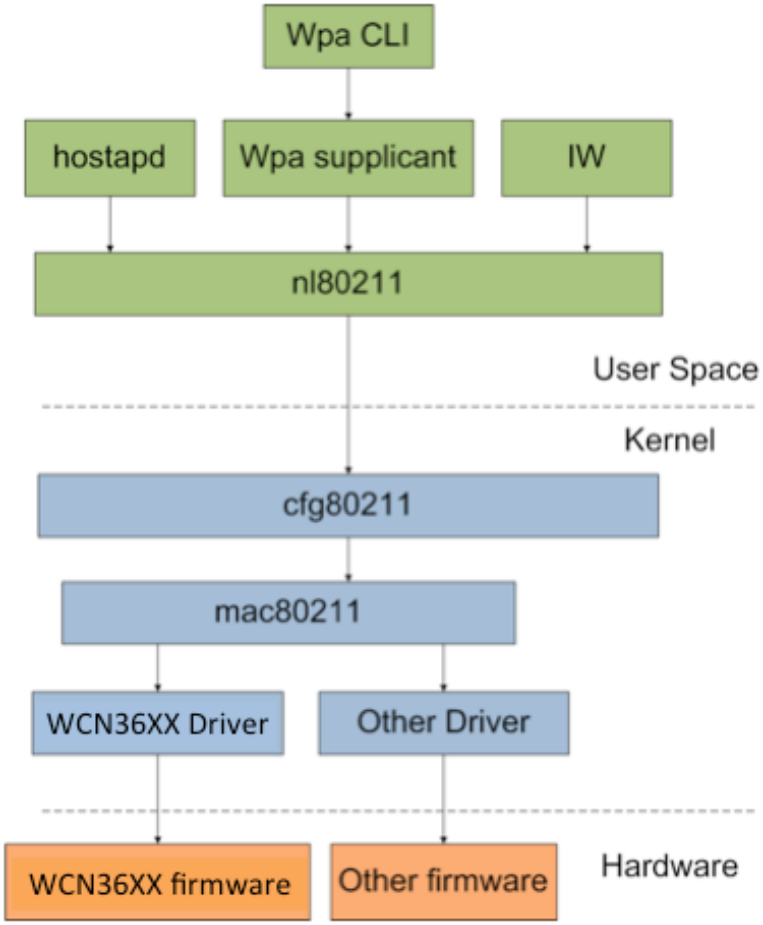

Figure 3.3 - Low level Android architecture

From Figure 3.3, it is possible to find again a strong dependency between the bottom WIFI Driver (i.e. hardware/firmware layer) and the upper Kernel layer. It is possible to see that, making use of this kind of driver, there is a limitation concerning the hardware level, which implies that the physical interface device must have a chipset supporting this driver type of the Kernel layer (i.e. wcn36xx).

It is also possible to identify the modules to be replaced in the mobile device. The modules wcn36xx, mac80211 and cfg80211 were replaced with new corresponding versions produced after a new build.

Based on this limitation, we have made a study among several mobile devices to identify the ones that have a chipset compatible with the driver wcn36xx. Table 3.1 shows the results of this study

Table 3.1 – Comparison between possible devices to implement a mesh network.

	Nexus 4	**Nexus 7 (2013)**	**Samsung Galaxy tab 3**	**Samsung Galaxy Tab 4**	**Sony Xperia Z**
Chipset	Qualcomm APQ8064 Snapdragon	Qualcomm Snapdragon S4Pro	Marvell PXA986	Marvell PXA1088	Qualcomm APQ8064 Snapdragon
CPU	Quad-core 1.5 GHz Krait	Quad-core 1.5 GHz Krait	Dual-core 1.2 GHz	Quad-core 1.2 GHz	Quad-core 1.5 GHz Krait
RAM	2 GB	2 GB	1 GB	1.5 GB	2 GB
Memory	8/16 GB	16 GB	8/16 GB	8/16 GB	16 GB
Driver Available	Yes (wcn36xx)	Yes (wcn36xx)	Yes (mvl8787)	No	Yes (wcn36xx)

This study was necessary because the mesh network is implemented at Layer 2 of the OSI model and this implies to change the device kernel properties, as we explained above. Thus, the study carried out shows that not all the devices allow the implementation of the mesh network at Layer 2. Among the devices that allowed a mesh implementation at Layer 2, as one can see in Table 3.1, the devices Nexus 4, Nexus 7, Samsung Galaxy tab 3 and Sony Xperia Z have a chipset driver that can be recompiled.

The device chosen for the test scenarios was the Nexus 4 due to its availability. For this device the wcn36xx is the mac80211 driver. This driver supports wcn3660 and wcn3680 chips found on Qualcomm SoC. The driver wcn36xx supports some features like the following ones:

- HW rate control
- AP mode support
- Ad-Hoc
- 80211s mesh

Once selected the device, the necessary steps to integrate the protocol configuration were made. The configuration of this device was defined, and a new Android Kernel has been built. This final step consists in the replacement of the default manufacturer

kernel by a new one that enables the configuration of a mesh network. The Kernel version used was the Android Mako 3.4 KitKat. When building a new image, the “load module support” was activated, to allow the driver wcn36xx to be loaded posteriorly.

The next step was the configuration and build of the Wifi Driver (wcn36xx). During this particular procedure, it was necessary to make a portability of the Linux version used on driver. Linux Kernel backports were required for that. It should be noted that Linux Kernel backports provide drivers released on newer kernels backported for usage on older kernels. In the driver build configuration was necessary to enable the option “Enable mac80211 mesh networking (pre-802.11s) support”.

The new build of wcn36xx also implies the build of other two modules already identified in Figure 3.3, inside the Kernel Space, i.e. : mac80211 and cfg80211. This new configuration makes it possible to create a mesh network with mobile devices. This configuration is further detailed in the Appendix C.

The third step was to root the mobile device. There are several ways to root the mobile device. In our case, a specific software for Nexus devices (Nexus Root Tool Kit) was used. With this software, it is possible to root the device and load the new generated kernel image. Finally, with all the necessary files generated and loaded into the mobile phone we can then configure the device to join or create a new mesh network.

With the implementation of a mesh network with mobile devices, the battery limitation is a hot topic and the energy management becomes crucial to enhance the battery lifetime of mobile nodes and hopefully the lifetime of the entire mesh. Thereby, a new metric proposal was studied for IEEE 802.11s networking environments.

3.3. New energy-aware routing metric for IEEE802.11s

After concluding that the existence of a metric type taking into consideration battery information it is a fundamental demand, two proposals were, following that sense, designed for IEEE 802.11s networking environments. One of the proposals is based on the device’s battery remaining energy. The other uses the battery voltage to calculate the metric value. Besides remaining energy and voltage, other metrics were

also studied, such as current, temperature and battery status. The main aim behind all these metrics is the minimization of the overall energy consumption during the normal operation of the mesh network. This also means we aim to maximize the time until a mobile node turn off due to its battery turns completely drained out.

In order to create an energy-aware routing metrics there are several approaches and possible strategies. Possible approaches are:

- Residual Values: It consists in the use of the remaining battery or the actual voltage of each node as the main metrics for the routes selection process.
- Energy Drain Rate: This approach, as the name suggests, is based on the energy drain rate of the device, which represents how fast energy is being consumed.
- Expected Energy Consumption: It consists of predicting the energy consumption generated with transmission.

In this case, both following proposals follows the first approach. Namely, it is based on energy remain and voltage values to calculate the metric.

a) Energy level approach

First approach is based on the remaining energy of the device. Therefore, the designed formula to represent energy penalty is inversely proportional to the energy level of the device.

The energy penalty, E_p is expressed as indicated in (35).

$$E_p = \frac{1}{E_r} \quad \textbf{(35)}$$

With the goal of assuring that devices do not reach a low battery level, a threshold level was added to the previous formula.

$$E_p = \frac{1}{E_r - T_h} \quad \textbf{(36)}$$

Parameter T_h brings about that the expression will tend to infinite as the E_r value comes closer to threshold value. In order for the expression not to produce an infinite value and a subsequently negative one, a mechanism has to be placed assuring that formula calculation of (36) will only be evaluated for a value of $E_r > T_h$, resulting in the expression indicated on (37).

$$(E_r > T_h \ \rightarrow E_p = \frac{1}{E_r - T_h}) \quad \textbf{(37)}$$

With the aim of standardize the final formula and lead the value to have a higher significance level, maximum battery level value was added. So, the energy penalty final formula is is available in (38).

$$(E_r > T_h \ \rightarrow E_p = \frac{E_{max}}{E_r - T_h}) \quad \textbf{(38)}$$

where E_{max} is the maximum battery level in percentage (from 0 to 100), E_r is the remaining energy in decimal values (from 0 to 1), and T_h is the threshold value (from 0 to 1).

Energy penalty final formula obtained was subsequently added into the Airtime Link equation (1), because it is the used metric on Open80211s and has already been implemented on the mobile device. Finally the new metric C'_a is expressed as illustrated in (39).

$$C'_a = \left[O_{ca} + O_p + \frac{B_t}{r} + \frac{E_{max}}{E_r - T_h}\right] x \ \frac{1}{1 - e_{pt}} \quad \textbf{(39)}$$

With this addition, the Airtime Link metric also considers the battery energy level and the routing protocol can make decisions with this new information. The energy penalty parameter (38) is added using the same magnitude to the already existing components (channel access overhead - O_{ca}, protocol overhead - O_p, number of bits in test frame - $B_t$and current bit rate - r).

During our energy-aware metric tests, other metrics were also studied, like the battery voltage of the mobile device. This decision was made because in spite of the voltage behavior of the battery mobile device is strongly correlated with its remaining energy, the voltage trend is different from the one shown by the battery remaining energy.

b) Voltage level approach

We have also decided to study the voltage level approach to our energy-aware metric. Therefore, previous formula (38) was adapted for the main component to

incorporate the battery voltage value. According to this, the voltage penalty is described in (40).

Therefore, previous formula (38) was adapted for the main component to become device voltage value. According to this, voltage penalty is described on (40).

$$(V > T_h \ \rightarrow V_p = \frac{1000}{V - T_h}) \quad \textbf{(40)}$$

where V is the voltage in volts and T_h is the threshold value (from 0 to 5). Similarly to the case of remaining energy, the decreasing voltage value leads to a metric penalty, which makes the equation inversely proportional to V. The value of 1000 on the upper part of the equation plays the role of standardizing the value to the same order of magnitude used in the metric dependent on the battery energy.

In order to make a correct fit between the voltage value and the remaining energy, a mathematical analysis has been performed to enable a fair comparison between both metrics. In this sense, the formula in (40) was changed to the one shown in (41).

$$(V > T_h \ \rightarrow V_p = \frac{1000}{(A + B{\times}V) - T_h}) \quad \textbf{(41)}$$

In (41) it is possible to observe two fitness parameters A and B, with respectively the values of -3.6458 and 1.09698.

After obtaining the final expression for the voltage, it was then added that expression to the final metric formula, resulting in the equation described in (42).

$$C'_a = \left[O_{ca} + O_p + \frac{B_t}{r} + \frac{1000}{(A + B{\times}V) - T_h}\right] x \ \frac{1}{1 - e_{pt}} \quad \textbf{(42)}$$

With this addition, the Airtime Link metric considers the voltage instead of the remaining energy. The tests and comparison between these two metrics is available in section 4.4.

c) Other metric parameters

In addition to the value of energy level, other battery parameters were also added. The final metric combined diverse information such as the battery operation, status, and temperature. Figure 3.4 shows the pseudo code with all the values being used to evaluate the energy-aware metric.

```
If(not charging) then
        If(Er<=Th)
                Return MAX_METRIC
        If(overheat or overvoltage or unspecific failure) or (temperature > X)  then
                Return C'a + Penalty
        Else
                Return C'a
Else
        Return Ca
```

Figure 3.4 – New energy-aware metric pseudo code

The pseudo code of Figure 3.4 shows the metric implementation with all the parameters we have previously referred. The first step in this algorithm is to verify if the mobile device is not charging and if this statement is false (device is charging) the metric return the default Airtime Link metric. Otherwise, the code verifies if the remain energy value is equal or less than the threshold value and if its true returns the maximum metric value (99999). Another verification is made in order to verify if the device is overheat, overvoltage, unspecific failure or the temperature is greater than a given value. If any of these abnormal situations occurs then the algorithm calculates the new metric value with a penalty. Otherwise, the algorithm returns the value calculated by the new metric value without any penalization.

3.4. New energy-aware routing metric implementation on mobile device

After metrics design and respective formulas creation, implementation of those formulas was set on the mobile device with the aim of making functional tests, comparing both metrics and identifying possible improvements.

First step for implementing the new metrics was analyzing the available metrics for mobile devices. The *dumpsys*, by way of example, is an android tool that runs on the device and dumps relevant information about the status of system services. With this tool is possible to dump information about the CPU, RAM, Battery, WIFI, Memory. For this case the used command was the “dumpsys battery”, which provides the following information.

Table 2 - Values and definitions of each parameter from battery information API.

Parameter	**Value**	**Definition**
AC powered	True | False	Power source
USB powered	True | False	Power source
Status	1-Unknown | 2-Charging | 3-Discharging | 4-Not charging | 5-Full	Status battery information
Health	1-Unknown | 2-Good | 3-Overheat | 4-Dead | 5-Over Voltage | 6-Unspecified Failure | 7-Cold	Health battery information
Present	True | False	Indicating whether a battery is present
Level	[0,100]	Battery level as a percentage
Scale	[0,100]	Maximum battery level
Voltage	[0,4195] (mV)	Battery voltage level
Current	[-1000000,1000000] (uA)	Battery current level
Temperature	[0,380] (ºC x 10)	Battery temperature level
Technology	Li-ion	Technology of the current battery

Regarding Table 2, data parameters used on metric building were:

- Status: It allows us to identity if the mobile device is charging or not and, if not, news metrics is calculated.
- Health: It allows us to check battery status, knowing if it has any kind of malfunction or it is in critical status.
- Level: This parameter is used on energy metrics calculation and defines remaining energy.
- Scale: Battery maximum capacity is one of the other parameters used on energy approach.
- Voltage: Voltage value is used for metrics calculation based on device voltage.
- Temperature: It allows us to perceive battery temperature and, via this data, to create an extra penalty in case of temperature exceed a limit value.

Next step was metrics implementation on device as well as changing module mac80211 (as shown in Figure 3.3). This module contains all functions used for AirTime Link metrics calculation. Then, variable related to the new metrics was added to AirTime Link formula. Due to restrictions concerning decimal calculations at a Kernel Space level, it was necessary to make use of the penalty calculation on User Space and, then, make use of results on Kernel Space. The described tests in secton 4.4 shows that both metrics have been successfully implemented.

4. Tests and Results

This chapter presents and discusses the evaluation reults of our mesh proposal. The tests have been executed sequentially using a real mesh testbed. The relevant information extracted from a specific test was used to proceed with the next one.

For the realization of these tests were used 4 routers TP-LINK WDR4300, 1 router TP-LINK WR841N and 1 mobile device Nexus 4. Their characteristics are described below.

Table 3 - TP-Link TL-WDR4300 specifications

System-On-Chip	AR9344 (MIPS)
CPU/Speed	560 MHz
Wireless No1	SoC-integrated: Atheros AR9340 2x2 MIMO for 2.4GHz 802.11b/g/n
Wireless No2	separate Chip: Atheros AR9580 3x3 MIMO for 5GHz 802.11a/n

Table 4 - TP-Link TL-WR841N specifications

System-On-Chip	AR9341
CPU/Speed	535 MHz
Wireless	Atheros AR9341

Both devices (TL-WDR4300 e TL-WR841N) come with a proprietary firmware installed by default. The OpenWRT distribution used in both TP-LINKs is the Barrier Breaker 14.07. This version was released in 2014 and is based on the version of Linux Kernel 3.10 that already includes Open802.11s.

Table 5 – Nexus 4 specifications

Chipset	Qualcomm APQ8064 Snapdragon
CPU/Speed	Quad-core 1.5 GHz Krait
RAM	2 GB
Battery	Non-removable Li-Po 2100 mAh battery

4.1. Open802.11s testbed

The scenario of our first test was made throughout implementation of Open802.11s. The initial goal of this test was to understand how mesh networks work in a real scenario. Another goal is to verify if scenarios and their corresponding topologies are appropriate, allowing us – or not - to conclude about network performance. Finally, other objective was to confirm if implementation was made correctly.

These tests were executed with four TP-LINK TL-WDR4300 and one TP-LINK TL-WR841N. Each device was installed with the firmware OpenWRT, version Attitude Adjustment 9.12. This kernel (version 3.3) supports by default 802.11s. The mesh network operated in Channel 1 (2.412 GHz) with 27dBm transmission power. The modulation was 802.11g/n, High Throughput mode, 20MHz. The arrangement of the devices was random and in the same room. To perform tests, two topologies have been used, Line and Multi-Path, as following described.

Line Topology

The Line topology, shown in Figure 4.1, allows performing tests that take into account the number of hops and its impact in the time to discover a path between nodes as well as the maximum achievable throughput.

During these tests all nodes have been in line of sight. For the definition of this topology, and due to the fact that HWMP is a Layer 2 protocol, it was then necessary to use the *iw* command to define the necessary topology, blocking the communication between some peers:

iw dev <mesh_interface> station set <destination_mac> plink_action block

To measure the discovery of a node, a ping was issued between two nodes (A and E) and the HWMP messages have been analysed with a sniffer (Wireshark). The time for path discovery was the elapsed time since node A sent an ARP request message until node A updated the ARP table and sent the ICMP ping request to node E.

Figure 4.1 - Line Topology with five routers.

The throughput tests have also been carried out with the Line topology. To generate traffic IPerf [43] was used, which allows TCP and UDP measurements of bandwidth between two nodes. For the following test scenarios was only generated TCP traffic

Results, with the line topology, for path discovery time and throughput in relation to the number of hops of each path are presented in Figure 4.2.

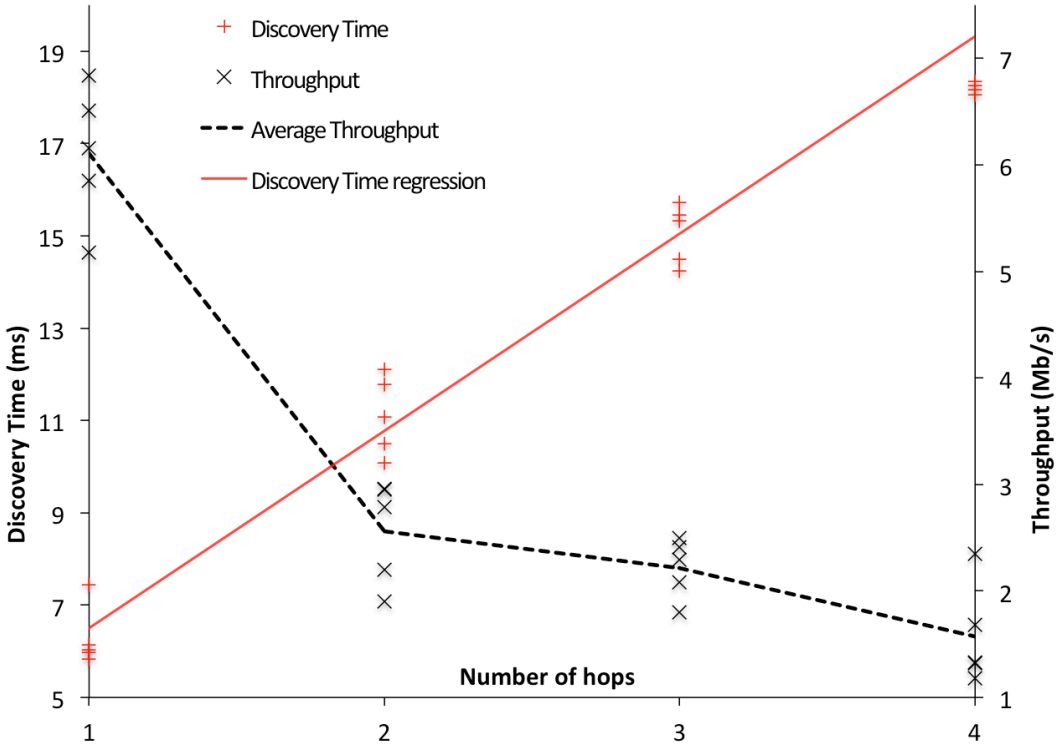

Figure 4.2 – Discovery time and throughput vs. number of hops.

These results were obtained from five runs of the above described tests. On one hand, the discovery time shows a linear-increasing trend as expected. It increases on average 4.2ms for each additional hop. On the other hand, the throughput shows a non-linear decreasing trend, starting from an initial maximum value of 6.1Mb/s (1 hop). Results in Figure 4.2 are similar to the ones of a previous work [40] but in relation to the throughput, we have obtained slightly better results.

Multi-Path Topology

The scenario in Figure 4.3 illustrates the case when multiple paths exist between two nodes, e.g. A and C, with no direct communication between them.

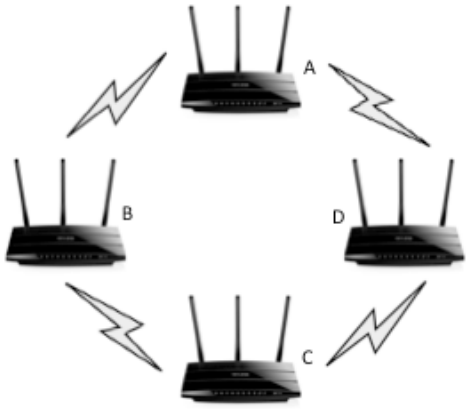

Figure 4.3 – Multi-Path Topology with four routers.

This scenario was used to measure the recovery time after a node failure. When this test was initiated there was a continuous ping from node A to C. During the test, the intermediate router, which was been used to establish the communication path, was disconnected. The time for the routing protocol to discover an alternative path was then measured, since ping fails after the router was disconnected, and it stayed failed until the mesh found an alternative path, recovering in this way the ping.

Figure 4.4 shows the messages exchanged between 4 nodes to establish an alternative path, when using a Multi-Path topology. Initially, when analysing the PING Request and PING Reply messages between the node A and C, it was verified that the connection was made through node D. After the node D was switched off, it was no longer possible to perform the PING Request.

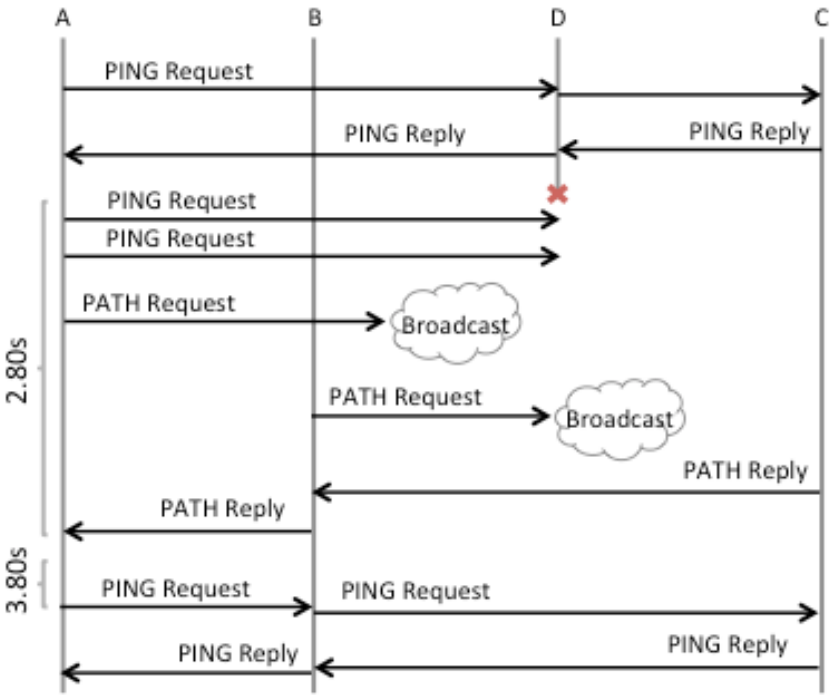

Figure 4.4 – Message diagram for detailing recovery time after a node failure.

After node D failure, the nodes exchanged a set of messages to find an alternate path between nodes A and C. For this, a PATH Request message from node A was sent in Broadcast; the Node B has sent a PATH Request Message in Broadcast as well. The node C responded to node B with a PATH Reply. Then, node B responded to node A with a PATH Reply, re-establishing the connection between nodes A and C. Since the breaking of node D until the new path was selected, the elapsed time was 2.8027s. After the new path was established between nodes A and C (via node B), it was then possible to exchange again PING Request and Reply messages. The total time between the node failure and the first subsequent successful PING Request was 3.7997s. Results with this topology have been obtained after eight runs.

On the basis of these obtainedresults , we can conclude that those are quite similar to the ones achieved on a previously study [40]. This shows that the implementation of a mesh network was successfully made. According the chosen topologies and the analyzed test scenarios, we can see that both could lead us to a conclusive analysis of the network performance. Nevertheless, the throughput results were not completely reliable because it was affected by the interference among routers. This occurred because each router was within the range of others.

4.2. Real testbed with Open802.11s and B.A.T.M.A.N-advanced

After the initial verification test where we have checked that our testbed was operating in a correct way, we have performed a second test to compare Open802.11s and B.A.T.M.A.N-advanced as well as to perceive which one of these methods could lead to a better performance according to several aspects such as the time to discover a node, throughput and time to recover from a node failure.

The hardware and software used was the same as the previous testbed. The version of B.A.T.M.A.N-adv was 2012.3.0 and it was obtained from OpenWRT packages (kmod-batman-adv). The mesh network operated in Channel 1 (2.412 GHz). The modulation was 802.11g/n, High Throughput mode, 20MHz. The MCS rate was fixed at 3 and 0with the command “iw <interface> set bitrates mcs-2.4 4“. The devices were placed at strategic points inside a house and the power transmission reduced to 0 dBm to reduce their coverage range. To compare both mesh implementations, two topologies have been used, Line and Multi-Path, as following explained.

Line Topology

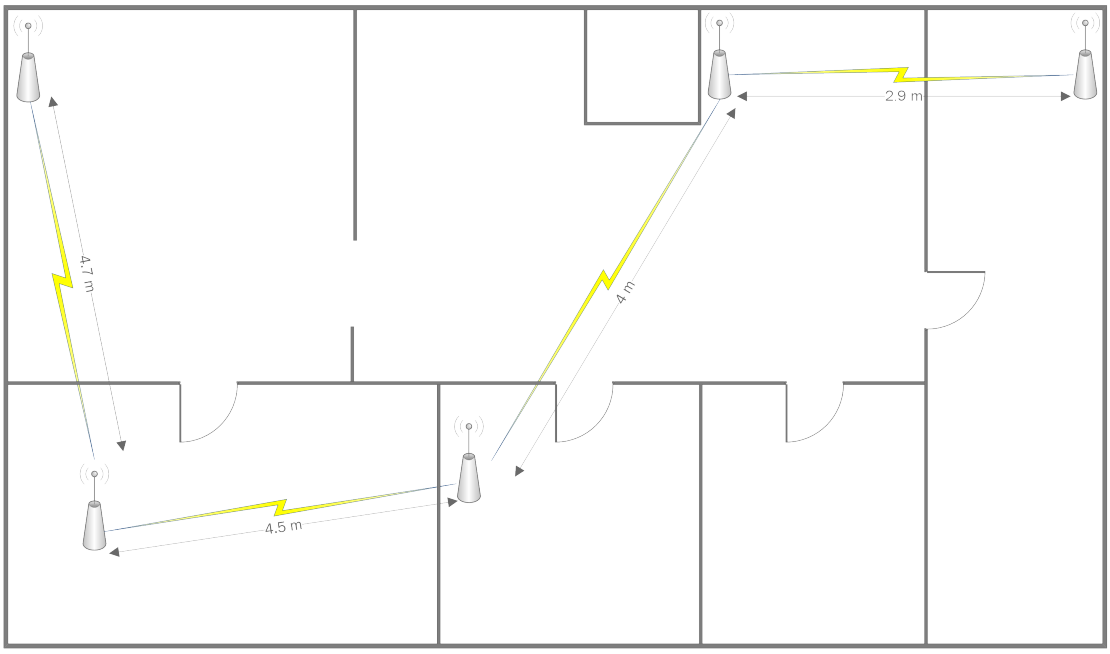

Figure 4.5 – Line topology real testbed setup.

The strategy we have used to measure the discovery time of Open802.11s was the same of the one explained in section 4.1.

In the case of B.A.T.M.A.N, due the fact that it is a proactive protocol does not make sense measuring the time discovery of a node, but the network bootstrap time. To measure this time, one router was added to the existing network and calculated the time to discover all the remaining nodes. The throughput tests have also been carried out with the Line topology. To generate TCP traffic IPerf was used again.

Throughput

Results, with the line topology, for throughput with Open802.11s and B.A.T.M.A.N-advanced in relation to the number of hops of each path are presented in Figure 4.6.

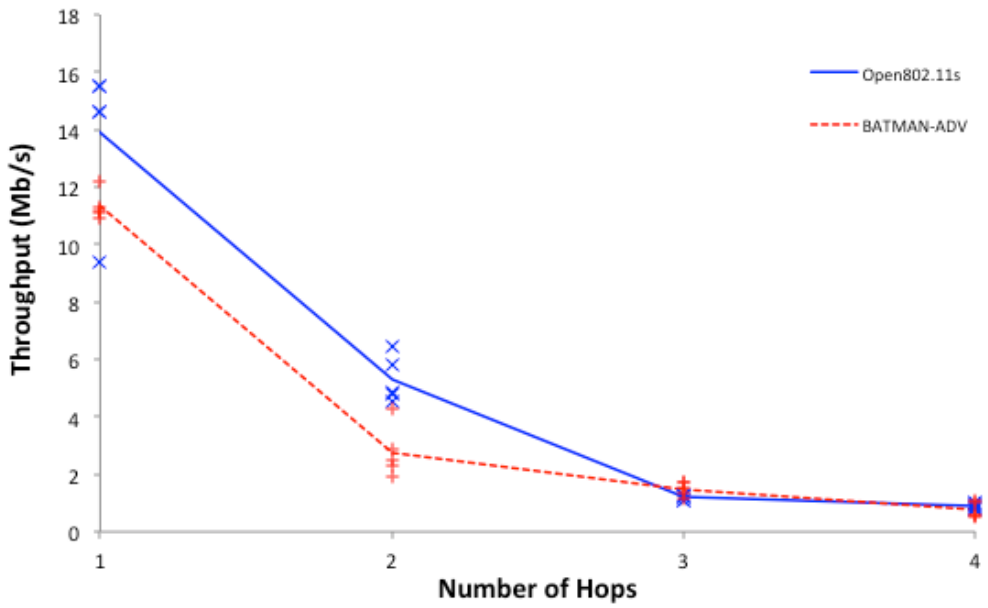

Figure 4.6 – Throughput for Open802.11s and B.A.T.M.A.N-advanced

These results in Figure 4.6, were obtained from five runs of the above described throughput test for both projects. For both projects the transmission rate for the first two hops was fixed at 28.9Mbs/s with MCS 3. For the last two hops (three and four) the MCS was reduced to 0, which corresponds to a transmission rate of 6.5Mb/s. The throughput test for both projects shows a non-linear decreasing trend, starting from an initial maximum value of 15.5Mb/s for Open802.11s and 12.2Mb/s for B.A.T.M.A.N. With the increasing number of nodes the results of both proposals tend to converge and the minimum throughput for both projects was 745Kb/s related with four hops. Figure 4.6 shows slight better results for Open802.11s than B.A.T.M.A.N., but this advantage is only visible for a small number of hops (i.e. for one and two).

HWMP Discovery Time and B.A.T.M.A.N-advanced Bootstrap Time

Figure 4.7 shows the Open802.11s discovery time in function of the number of hops.

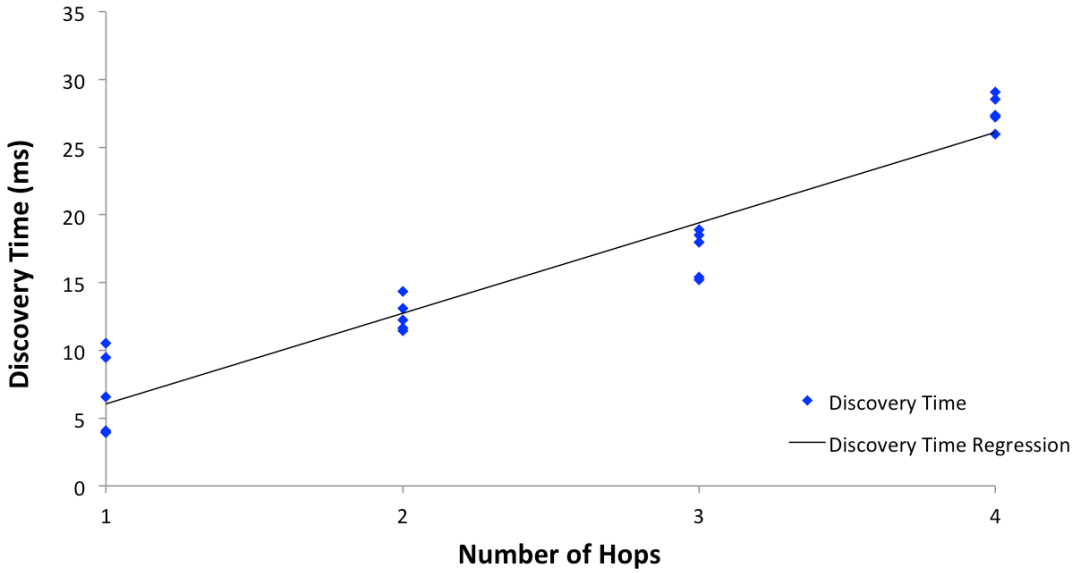

Figure 4.7 – Open802.11s discovery time vs. number of hops.

The discovery time for Open802.11s shows a linear-increasing trend as expected. It increases approximately on average 6ms for each additional hop. These results show that the time to find a particular node in a mesh network with Open802.11s is very appealing, achieving for example to discover an entire network with 5 routers in approximately 27ms.

Figure 4.8 shows the B.A.T.M.A.N. bootstrap time in function of number of hops.

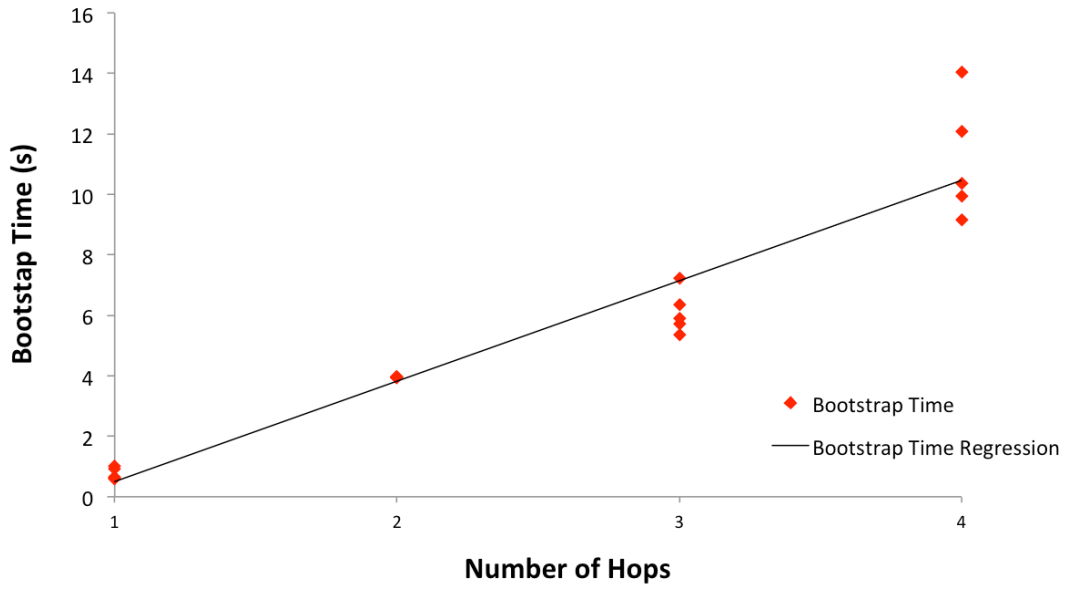

Figure 4.8 – B.A.T.M.A.N-advanced bootstrap time vs. number of hops.

B.A.T.M.A.N. is a proactive routing protocol, and for that reason the nodes discover the network topology by its own initiative. Figure 4.8 shows a similar trend to what happens with the Open802.11s node discover. It shows a linear-increasing behavior. The bootstrap time increases on average 3s for each additional hop.

Analyzing the above two graphs (Figure 4.7 and Figure 4.8), it is observed that the bootstrap time for B.A.T.M.A.N. is superior to the discovery time for Open802.11s, the former in the order of seconds and the latter in order of milliseconds. However the bootstrap time is measured when the node turns on in the network while the discovery time is only calculated when the first PING request is sent to a specific node.

A proactive routing protocol such as B.A.T.M.A.N. it is expected to show up a considerable initial bootstrap time for the entire network creation. Nevertheless, a value around 10 seconds for discovery a network with 5 nodes is a factor to be considered for networks that require low response times.

With the knowledge of the entire network guaranteed by B.A.T.M.A.N. would be predictable that the PING was sent faster than in the case of Open802.11s; however this did not occur in our evaluation. In fact, the B.A.T.M.A.N. showed a lot of instability and high delay times for PING between nodes. These problems could be associated to the version of B.A.T.M.A.N. being tested.

Multi-Path Topology

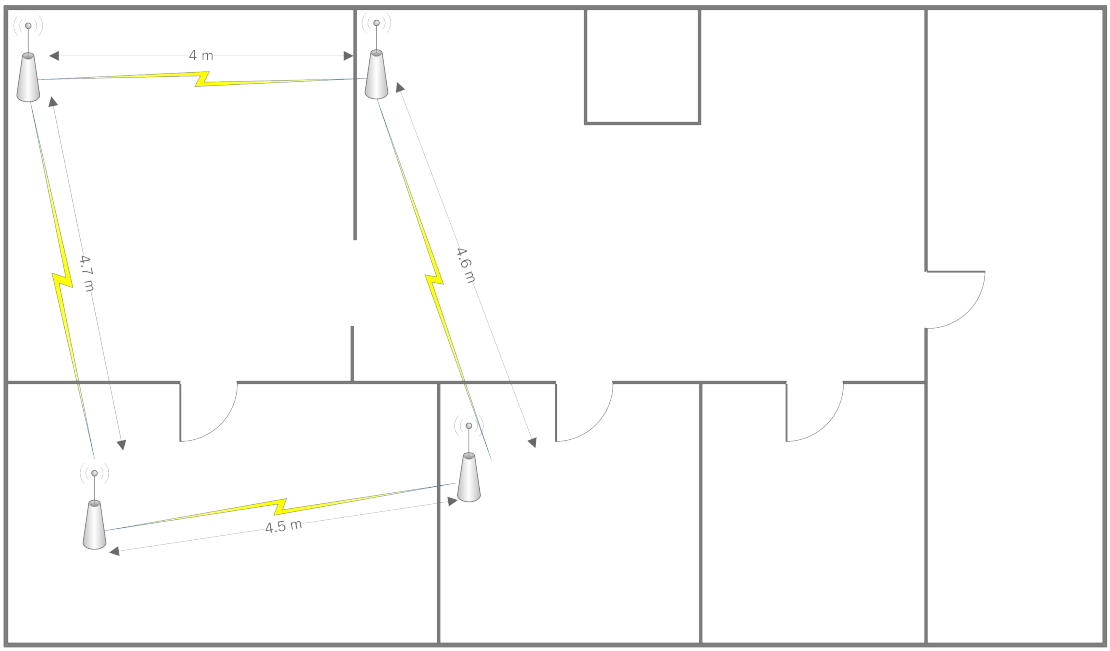

Figure 4.9 – Multi-path topology real testbed setup.

The scenario in Figure 4.9 illustrates the case when multiple paths exist between two nodes, e.g. A and C, with no direct communication between them. This scenario was used to measure the recovery time. When this test was initiated there was a continuous PING from node A to C. During the test, the intermediate router, which has been used to establish the communication path, was disconnected. The time for the routing protocol to discover an alternative path was then measured. This measurement was the time elapsed between ping failure event and it subsequent recover.

Recovery Time

Figure 4.10 shows the time to recovery in case of node failure for B.A.T.M.A.N. (red line) and Open802.11s (blue line). The node failure event can occur in several circumstances, for example: battery exhaustion, poor link quality, interferences or obstacles/obstructions.

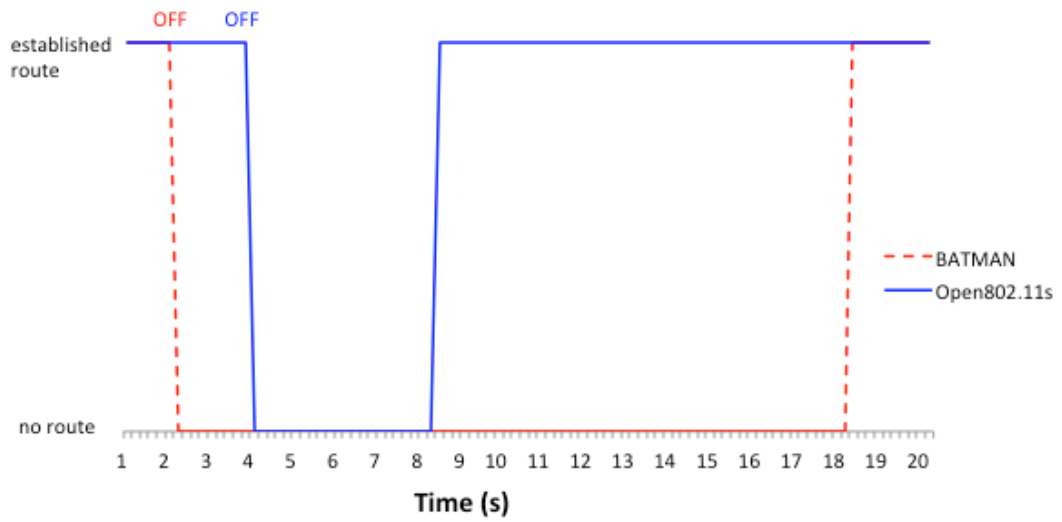

Figure 4.10 – Recovery time in case of node failure for B.A.T.M.A.N. and Open802.11s.

For this scenario both projects had the default settings. In the case of B.A.T.M.A.N-advanced, the OGM interval (a single message that each node periodically generates and send in broadcast on all hard interfaces) it is one second. The average results for this test have been obtained after ten runs.

As shown in Figure 4.10 the recovery time is much faster in Open802.11s, losing communication during only 4.31 seconds. In the case of B.A.T.M.A.N. this time is much higher, running out of communication for about 16.18 seconds on average. In a global analysis of the evaluation results, these suggest that the Open802.11s had advantage in all the performed tests.

4.3. Real testbed with mobile device and routers

After analyzing previous results (i.e. section 4.3), one can conclude that Open80211s showed better results that B.A.T.M.A.N-advanced. Consequently, we have selected Open80211s to be implemented in the mobile device, as shown in chapter 3.2. The main goal of this test is to verify if there is any kind of limitation of the mobile device when operating on a mesh network. Another goal is to confirm if the implementation of the mesh network was made correctly on the mobile device.

The hardware and software of the routers that we have used in the current scenario was the same as the previous testbeds. The version of Open802.11s installed on mobile device was the same as the one in the router. The mesh network operated in Channel 1 (2.412 GHz). The modulation was 802.11n, due to the limitation of WIFI device board (only supports 802.11n). It was also used High Throughput mode, 20MHz. The devices were placed at strategic points inside a house and the power transmission reduced to 0 dBm (routers and mobile device) to reduce their coverage range. To compare both devices, two topologies have been used, Line and Multi-Path.

Line Topology

In order to compare the performance of mobile devices and TP-Links in a Line Topology mesh configuration, we have performed tests for time discover and the maximum achievable throughput. To perform those tests, we have considered two scenarios, as demonstrated in Figure 4.11 and Figure 4.12.

Figure 4.11 - Line topology with five routers

Figure 4.12 – Line Topology with four routers and one nexus 4.

Discovery Time

Figure 4.13 shows the TP-Links and mobile device discovery time in function of number of hops.

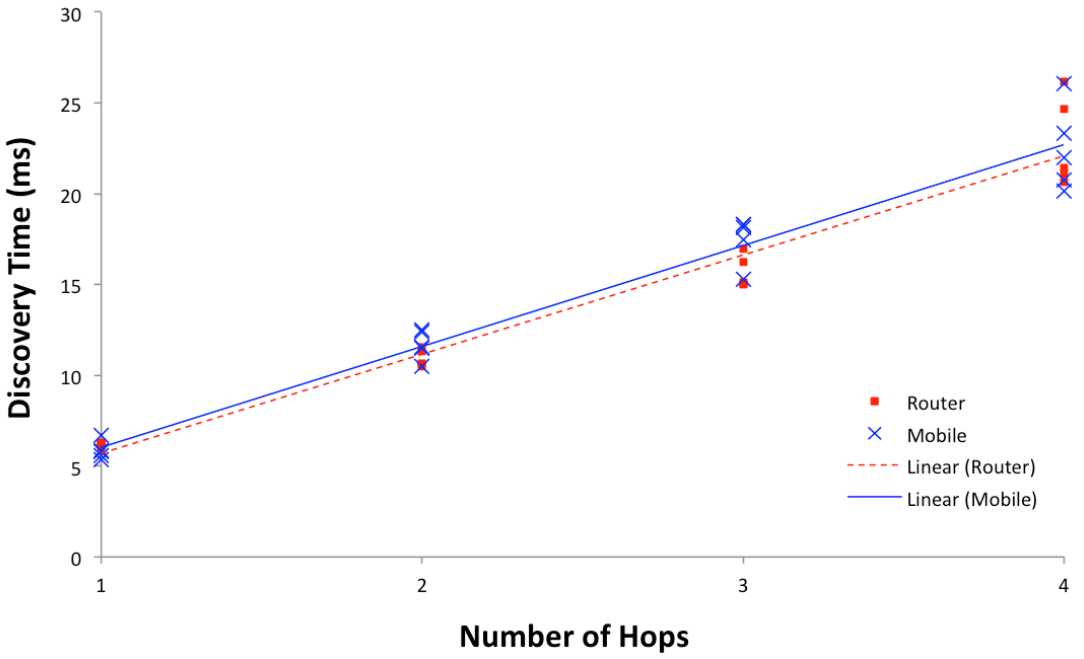

Figure 4.13 – Discovery time for TP-Links and mobile device

The discovery time for both cases shows a linear-increasing trend. In Figure 4.13 it is clear that the behavior of both is very similar. For both devices, the line increases approximately on average 5.8ms for each additional hop. Both, TP-Link and mobile device, achieving the entire network in approximately 23ms. Therefore, based on Figure 4.13 we can conclude that the results are similar in spite of using the mobile device.

Throughput

The throughput tests have also been carried out with the Line topology. To generate TCP traffic, IPerf was used and the source node was the mobile phone.

Figure 4.14 shows the throughput in function of number of hops for TP-Links and mobile device.

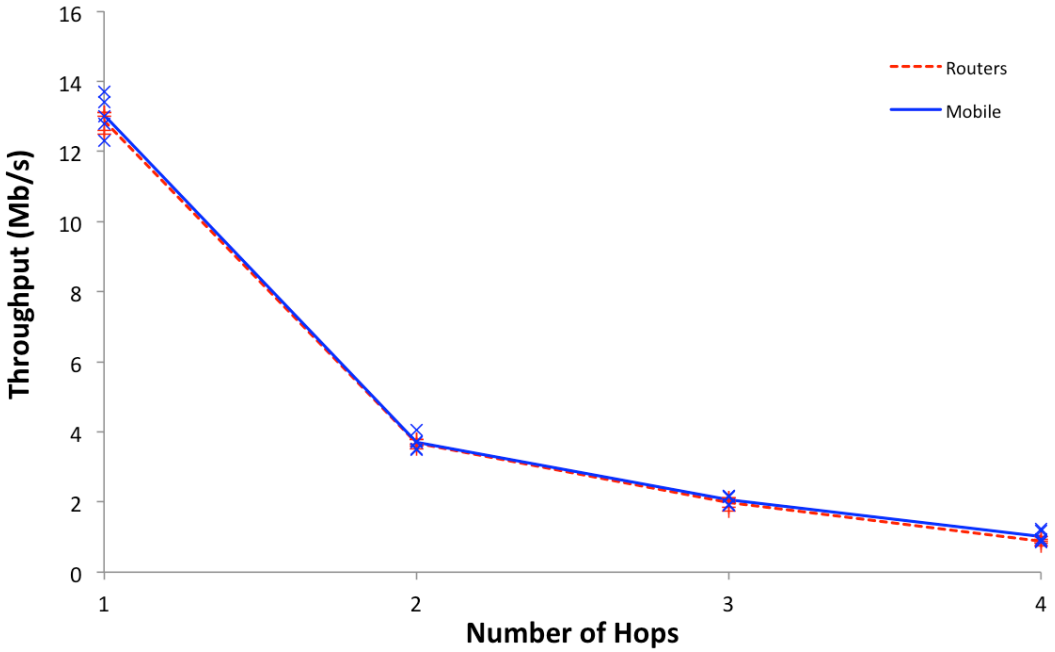

Figure 4.14 - Throughput for Open802.11s and B.A.T.M.A.N-advanced

Figure 4.14 shows the throughput test for both tests with different devices (TP-LINKs and mobile device). Both results shows a non-linear decreasing trend, starting from an initial maximum value of 12.9Mb/s. The minimum values for throughput converge to the value of 950Kb/s related with four hops. Figure 4.14 shows that there is no limitation in terms of throughput imposed by the usage of a mobile device.

Multi-Path Topology

Figure 4.16 and **Figure 4.15** shows the two topologies used to measure the time to recover only with routers vs. when using a mobile device.

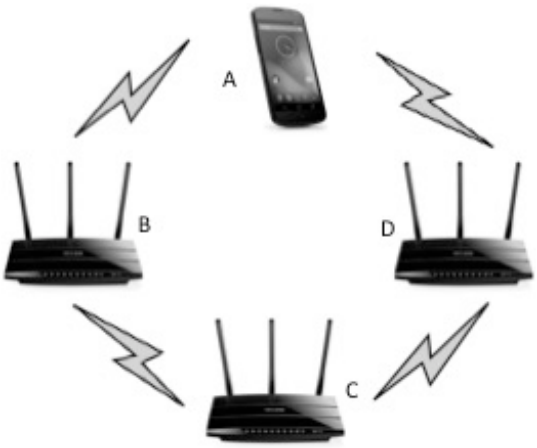

Figure 4.16 – Multi-path topology with three routers and one nexus 4.

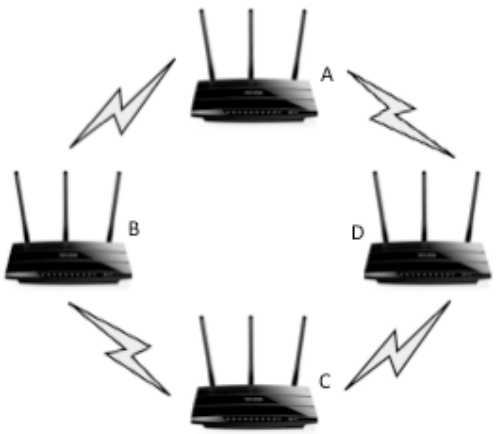

Figure 4.15 - Multi-Path Topology with four routers and one nexus 4.

In **Error! Reference source not found.** the mobile phone is placed in one of the ends of the mesh topology because it is intended that. It makes the handover decision in case of node failure.

Figure 4.17 shows the recovery time after a node failure for TP-Links and mobile device.

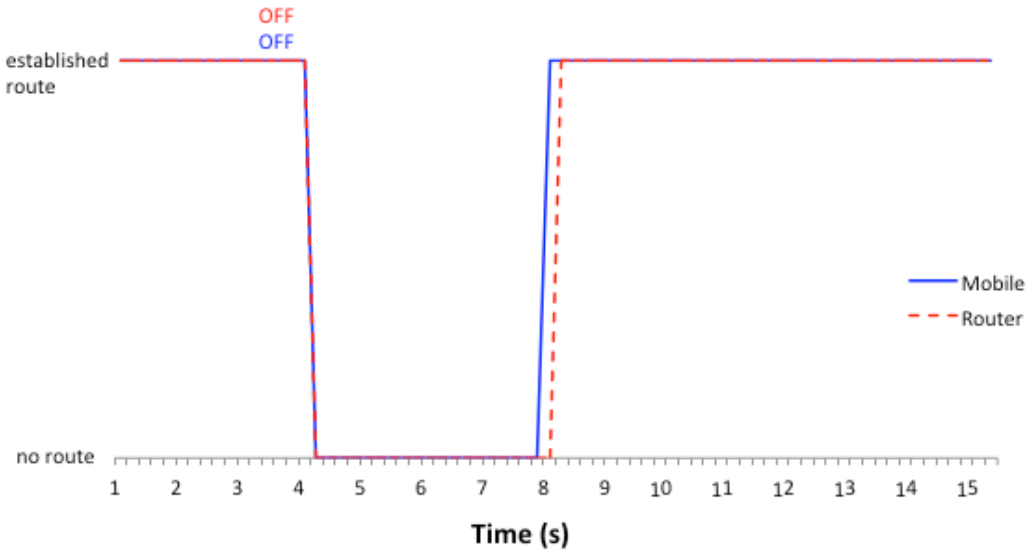

Figure 4.17 - Recovery time in case of node failure for routers and mobile phone.

Figure 4.17 shows the time to recovery in case of node failure for the topology with 4 routers (red line) and the topology with 3 routers and the decision of handover on mobile device (blue line). The figure above illustrate the recovery time after a node failure during a communication. For this scenario both devices had the default settings and the results for this test have been obtained after ten runs.

As shown in Figure 4.17 the recovery time is very similar in both tests, with an average of 4 seconds. In the case of mobile device the time without communication was 4.01 seconds, for routers the time for recover was 4.3 seconds. The 0.3 seconds in this case can be negligible due the scale of these values.

After analyzing the obtained results for discovery time, throughput and time to recover in case of node failure items it is possible to come to the conclusion that the mobile device has no limitations according to all studied parameters. These results also show that the implementation of the mesh network on a mobile device has been correctly done.

4.4. Real testbed with new energy-aware routing metric

We have proceed with a new set of tests to study the behavior of some battery information values. These tests also makes it possible to understand the expected advantages over the mesh operation for using new energy-aware metrics instead of the original Airtime Link metric, as discussed in section 3.3. In addition, these tests give us the opportunity to make functional tests of the implemented new metrics.

The routers hardware and software used was the same as the previous testbeds. The version of Open802.11s installed on mobile device was is the same of the router. The mesh network operated in Channel 1 (2.412 GHz). The modulation was 802.11n. It was also used High Throughput mode, 20MHz. The devices were placed at strategic points inside a house and the power transmission reduced to 0dBm (routers and mobile device) to reduce their coverage range. In order to ensure that in the ideal case (without error rate) Airtime Link value is equal on both devices, transmission rate was set to 1 Mb/s. *Fast Discharge* application was used in tests, allowing several tasks to be executed and enabling discharge to be made constantly and as faster as possible.

In this specific case, only the multi-path pathology (Figure 4.18) was used, as it allowed a decision-making process between two different nodes, allowing us as well to understand how the metrics could affect the decision about the more convenient path through the mesh .

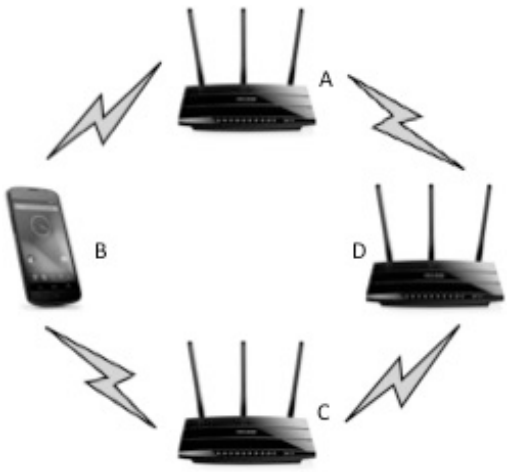

Figure 4.18 - Multi-path topology with three routers and one nexus 4.

Metrics implementation was only applied on mobile device B and, on router D (Figure 4.18). A value of 70% of the remaining energy was also set for this test.

In a first test, energy metrics were used to calculate the final metrics. In a second stage, battery voltage value was used to verify if there are advantages in using the voltage as a metric.

a) Energy level approach

Metric Value vs. Time

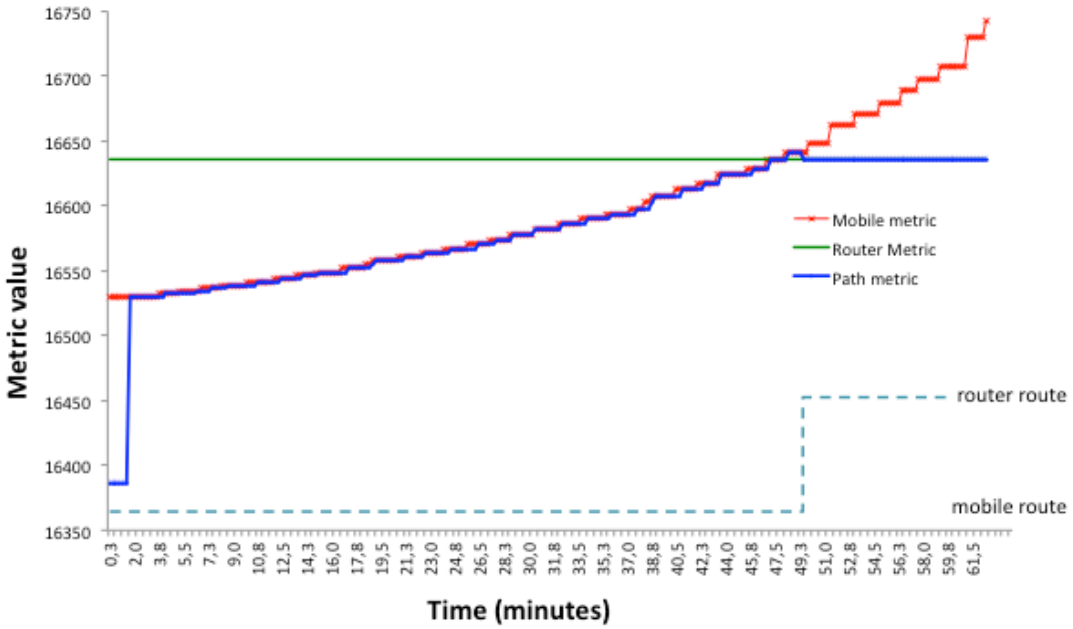

Figure 4.19 – New energy-aware metric path selection and metric values

In Figure 4.19 it is possible to note values obtained considering, over time, metrics for the mobile device and for the two routers. It is possible to identify, too, router D (dotted line) presenting a constant metric value, 16636, corresponding to 70% of the remaining battery. Metrics value for chosen path until 1.5 minutes is 16386, corresponding to Airtime Link value without considering energy and with a transmission rate set to 1 Mb/s. At minute 1.5, the mobile device ceases to charge and passes to a discharge battery mode. At this moment, can be verified a significant increase of metrics, related with energy decreasing. Over time, it is possible to observe that metrics value rises up, as might be expected. It is still possible to observe that metrics values measured on router appear with a slight delay concerning device measurements. This happens because router does not recalculate metrics value of path at the same time metrics value is obtained from device. The Handover occurs at 49 minutes, time value which corresponds to the moment when

the mobile device battery decreases from 70% to 69%. Therefore, the mobile device keeps a remaining energy value lower than the one of the router. Until the end of the test period, metrics value for the path remains constant, as might be expected.

New energy-aware metric vs. Energy

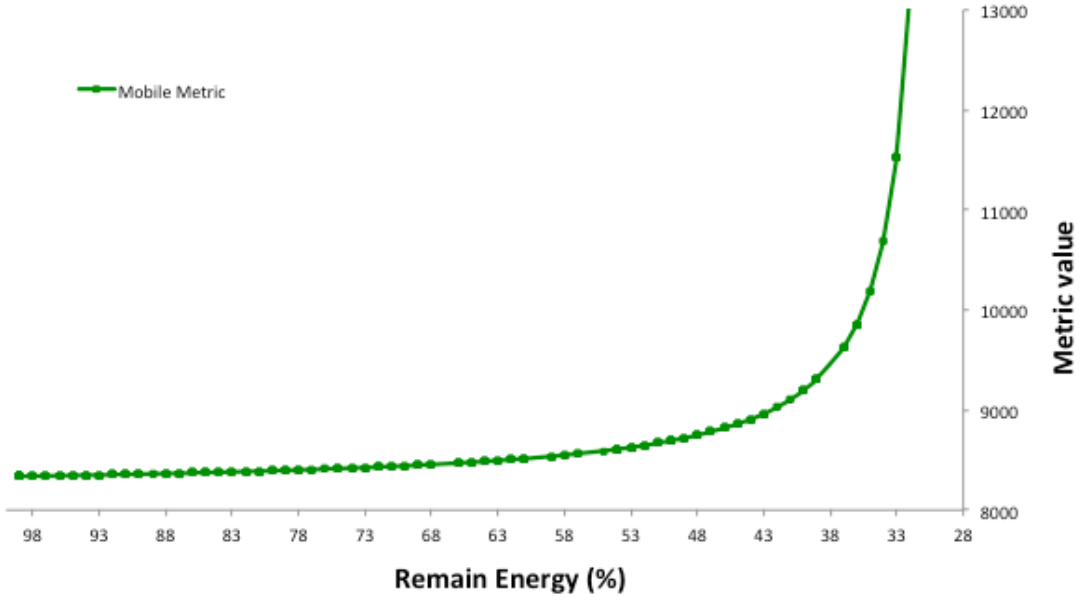

Figure 4.20 – Metric value vs. remain energy

Figure 4.20 shows collected data on the mobile device for the new energy-aware metric as a result of the remaining energy. Knowing that Airtime Link value for a transmission rate set to 1 Mb/s is 8913 it can be seen that, for a 99% battery value, new metrics gets a value of 8337, resulting in a loss of 144 according to final result. This value, as might be expected and as well as described in (39), is inversely proportional to the value of the remaining battery, increasing with the loss of energy percentage. It can also be noted in the previous figure that metrics values describe an exponentially growing range. Until approximately 55% of battery charge, curve growth is practically constant, increasing about 45 in metric value for each 10%. Metrics value tends to infinity reaching the imposed threshold value, assuring by this way that devices near this value are heavily penalized. Reaching threshold value (in this case, defined as 30%), metrics is defined as a constant value of 99999.

Temperature vs. Energy

Data was also collected from temperature values during the test. Those values are presented in Figure 4.21.

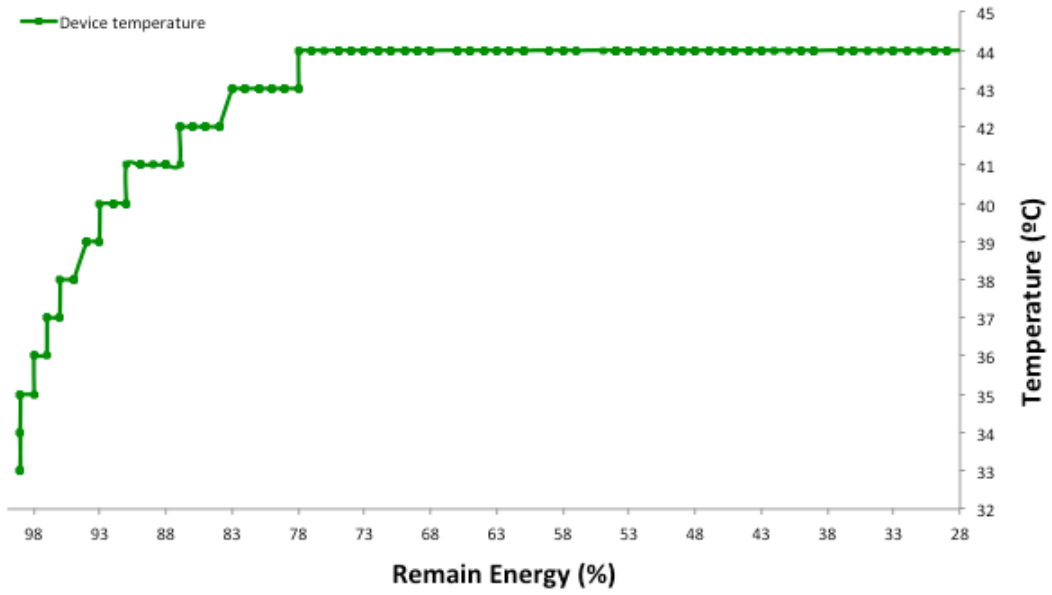

Figure 4.21 – Temperature vs. remain energy

Figure 4.21 allows us to understand temperature behavior as a result of the device remaining battery level. It is observable that, in the beginning of the test, temperature was set on 33º. This value sharply increases when the used application to discharge battery is opened, registering the value of 35º. Temperature values gradually rises until adjusting itself on 44º. Based on figure, it can be concluded that temperature metrics is related with device use but does not allow us to distinguish the remaining battery level or the battery drain rate. In addition to the temperature, they were also collected voltage values over the test scenario.

Voltage vs. Energy

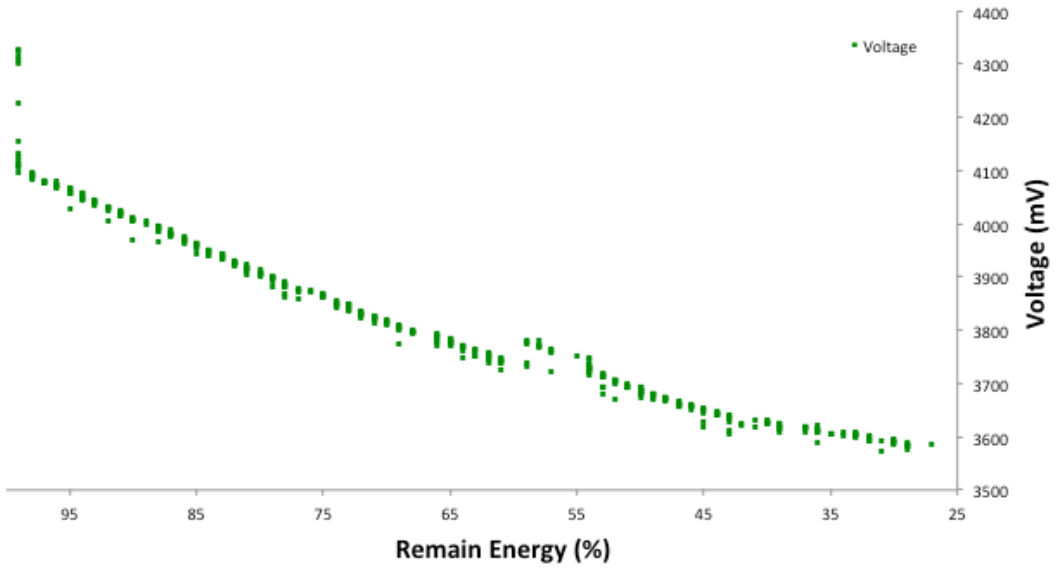

Figure 4.22 – Voltage vs. remain energy

Figure 4.22 corresponds to voltage values according to remaining energy evolution. Considering the above chart, we can see that there is a direct connection between measured voltage on device and remaining energy. As remaining energy decreases, Voltage values also decrease on a straight-line basis. One of the issues concerning usage of remaining energy is granularity, because collected data do not present decimal format digits. Throughout this research, it can therefore be concluded that voltage metrics could be replaced by remaining energy and, by that, metrics will obtain more granularity on produced results. Nevertheless, it is possible, too, given the above chart, to observe that voltage values variance has to be considered because there are relevant variances on collected data.

Throughout previous charts consideration, it can be seen that metrics implementation was successful and the handover to the router was made at the right moment. It was seen, too, that temperature values could serve as a complement to the final formula metrics, assuring that a device delivering a high temperature suffers a penalization.

b) Voltage level approach

After observing metrics results considering remaining energy, it was possible to note that there was a straight relation with that data and the voltage. In that sense, a study about voltage usage instead of a remaining energy approach is relevant. The main aim of this test was to verify if voltage usage brings advantages due to the fact of making use of a granularity higher than remaining energy.

Voltage and Energy metric vs. time

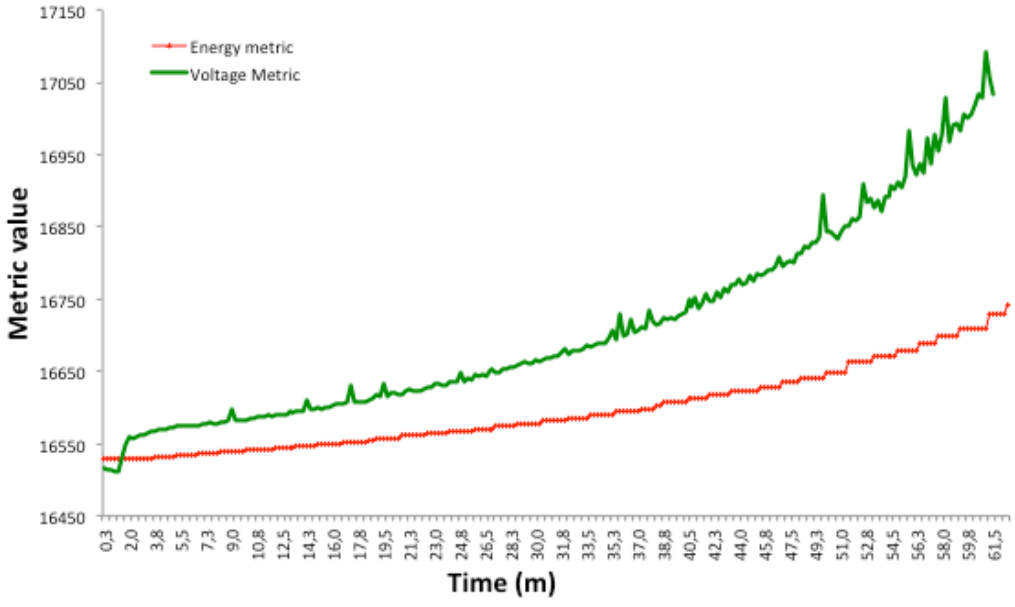

Figure 4.23 – Energy and voltage metric vs. time

In Figure 4.23, it is possible to observe both metrics behavior (Voltage and Energy) during the time of the test. In this Figure, it can be seen that, in the beginning of the test, voltage metrics shows a sharp behavior. This period corresponds to when the mobile device stopped charging and turned to a discharging status. Both curves behavior shows an increase. Besides that, voltage metrics exponential increase is clearer.

An interesting particularity to observe through Figure 4.23 is the fact that results obtained for energy are “stairs shaped”, while, according to voltage, this behavior is not verifiable. This could be explained by the fact that the granularity of voltage values are much higher than those of energy. It is still observable that voltage values present a much higher variance than those of energy.

New energy-aware metric vs. voltage

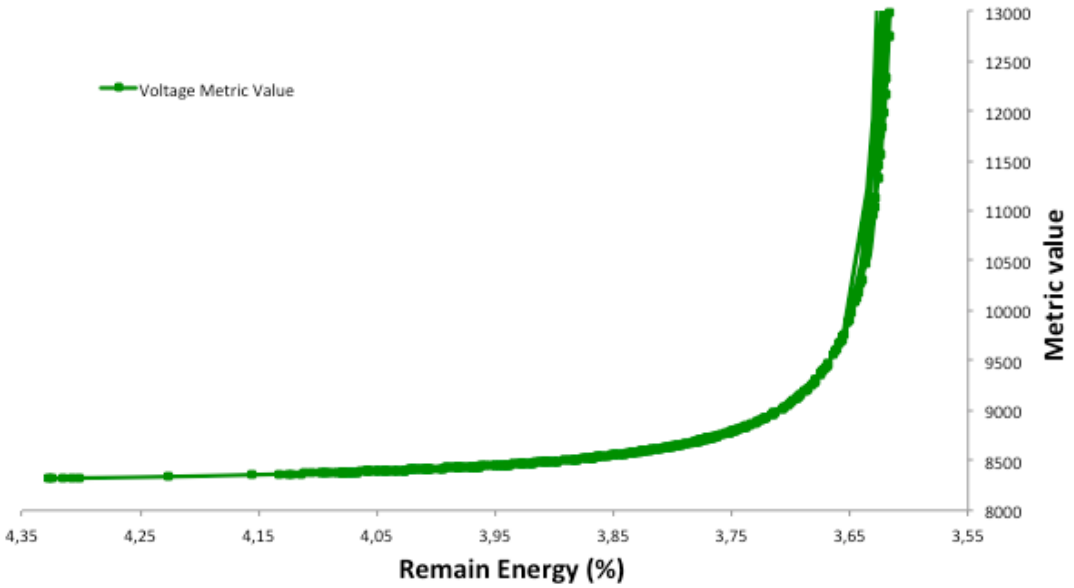

Figure 4.24 – Voltage metric Value vs. remain energy

Figure 4.24 shows collected data on the mobile device for the new energy-aware metric as a result of the voltage values. It is noted by the above figure that similar to what happened to the remaining energy, the metric values are inversely proportional to the voltage value, as described in (42).

It can also be noted in the previous figure that metrics values describe an exponentially growing range. The curve behavior is very similar to that obtained in the energy metric, practically continuous until 3.80v. It can be seen that the voltage values from 3.70V have wide variations, resulting in oscillations in final metric value.

Metrics value tends to infinity reaching the imposed threshold value, assuring by this way that devices near this value are penalized. Reaching threshold value (in this case, defined as 30%, corresponding to 3.597V), metrics is defined as a constant value of 99999.

After analyzing both metrics (voltage and energy) during time, it can be concluded that the main advantage of using voltage metrics is related with a higher data granularity. However, an obvious disadvantage of using voltage metrics is a significant variance of values. In order to overcome this disadvantage it can be applied a smoothing iterative function to decrease the variation range, e.g. exponential smoothing.

5. Conclusions

The current work discusses important solutions and enhancements for path selection and routing metrics in mesh networks with mobile devices [4]. We have initially discussed some relevant mesh networks projects for mobile devices. Nevertheless, none of these implements the mesh solution at Layer 2. Then, we have analyzed relevant proposals related with the path selection and routing protocols for mesh networks. We have concluded that the analyzed protocols do not consider energy consumption as a relevant metric for routing. The energy management is one of the main critical aspects in the design and operation of mesh networks with mobile devices, namely to increase the network lifetime. In this way, we have decided to reconsider the metrics used in this type of protocols and finding solutions that take into account the battery energy of each mobile device. Consequently, we had two important objectives to achieve with our research work based on energy-aware metrics: mobile node battery optimization, and limiting the path recovery delay.

A real testbed scenario with two different projects (B.A.T.M.A.N.-advanced and Open802.11s) was made in order to discuss the results for path discovery time, bootstrap time, throughput and recovery time after a node failure. In a global analysis of our evaluation results, these suggest that the Open802.11s had advantage in all the performed tests (delay and throughput) and was the chosen protocol for progressing with other research work.

The main objective of our work was the implementation of a mesh network using handheld mobile devices and it was successfully achieved, including the mobile node battery optimization and the handover controlled by an energy-aware metric. Based on tests in the mesh network deployed with mobile devices we can conclude that the mobile device has no limitations according to all the studied parameters.

In addition, as a novelty, we have successfully deployed and evaluated two energy-aware metrics solutions in a real mesh testbed.

For future work, we suggest a simulation work that will allow the study of the impact on the mesh performance of combining a set of metrics available in each mobile terminal to control the path selection and routing. With this type of tests, it is possible to evolve the metrics suggested in our work. The set of metrics in each mobile device can be, as an example, the following ones: current, temperature or the expected energy consumed.

6. References

[1] S. M. Faccin, C. Wijting, J. Kneckt, and A. Damle, “Mesh WLAN networks: Concept and system design,” *IEEE Wirel. Commun.*, vol. 13, pp. 10–17, 2006.

[2] “802.11s - IEEE Standard for Information Technology.” 2011.

[3] R. N. M. Carlos Meralto, José Moura, “Mesh Networks for Handheld Mobile Devices,” in *Conftele*, 2015.

[4] R. N. M. Carlos Meralto, José Moura, “Wireless Mesh Sensor Networks with Mobile Devices: A Comprehensive Review,” in *Research on Advanced Wireless Sensor Network Applications, Protocols, and Architectures*, IGI Global, 2015.

[5] “Open Garden,” 2014. [Online]. Available: http://opengarden.com/. [Accessed: 10-Dec-2014].

[6] “FireChat,” 2015. [Online]. Available: https://opengarden.com/firechat. [Accessed: 10-Dec-2004].

[7] “FireChat in Hong Kong: How an app tapped its way into the protests,” 2014. [Online]. Available: http://edition.cnn.com/2014/10/16/tech/mobile/tomorrow-transformed-firechat/. [Accessed: 10-Dec-2014].

[8] “VPN issues on KitKat (version 4.4),” 2013. [Online]. Available: https://code.google.com/p/android/issues/detail?id=62714. [Accessed: 10-Dec-2014].

[9] “VPNService can only provide connectivity for routes that are reachable without VPN,” 2013. [Online]. Available: https://code.google.com/p/android/issues/detail?id=62588. [Accessed: 10-Dec-2014].

[10] “Android 4.4: TCP advertises incorrect MSS over VPN (using VpnService),” 2013. [Online]. Available: https://code.google.com/p/android/issues/detail?id=61948. [Accessed: 10-Dec-2014].

[11] “Serval Project,” 2013. [Online]. Available: http://www.servalproject.org/. [Accessed: 10-Dec-2014].

[12] “Serval Mesh Extender,” 2014. [Online]. Available: http://developer.servalproject.org/dokuwiki/doku.php?id=content:meshextender:main_page. [Accessed: 10-Dec-2014].

[13] “The SPAN Project,” 2014. [Online]. Available: https://github.com/ProjectSPAN. [Accessed: 10-Dec-2014].

[14] “Commotion,” 2013. [Online]. Available: https://commotionwireless.net/. [Accessed: 10-Dec-2014].

[15] C. Perkins, “[RFC 3561] Ad hoc On-Demand Distance Vector (AODV) Routing.” 2003.

[16] “Open Mesh,” 2006. [Online]. Available: http://www.open-mesh.org/projects/batman-adv/wiki/Doc-overview. [Accessed: 10-Dec-2014].

[17] C. E. Perkins and P. Bhagwat, “Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers,” *ACM SIGCOMM Computer Communication Review*, vol. 24. pp. 234–244, 1994.

[18] P. J. T. Clausen, “[RFC 3626] Optimized Link State Routing Protocol (OLSR).” 2003.

[19] J. Chroboczek, “[RFC 6126] The Babel Routing Protocol,” 2011.

[20] D. Johnson, “[RFC 4728] The Dynamic Source Routing Protocol (DSR).” 2007.

[21] M. S. Islam, M. M. Alam, M. A. Hamid, C. S. Hong, and S. Lee, “EFT: A high throughput routing metric for IEEE 802.11s wireless mesh networks,” *Ann. des Telecommun. Telecommun.*, vol. 65, pp. 247–262, 2010.

[22] S. G. and L. T. Rosario G. Garroppo, “A joint experimental and simulation study of the IEEE 802.11s HWMP protocol and airtime link metric,” *Int. J. Commun. Syst.*, 2012.

[23] M. A. Bin Ngadi, S. Ali, A. H. Abdullah, and R. H. Khokhar, “A taxonomy of cross layer routing metrics for wireless mesh networks,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2012. p. 177, 2012.

[24] Linus Lüssing, “Transmission Quality,” 2011. [Online]. Available: http://www.open-mesh.org/projects/batman-adv/wiki/OGM. [Accessed: 20-Jul-2015].

[25] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, “A high-throughput path metric for multi-hop wireless routing,” in *Wireless Networks*, 2005, vol. 11, pp. 419–434.

[26] C. E. Koksal and H. Balakrishnan, “Quality-aware routing metrics for time-varying wireless mesh networks,” *IEEE J. Sel. Areas Commun.*, vol. 24, pp. 1984–1994, 2006.

[27] R. Draves, J. Padhye, and B. Zill, “Routing in multi-radio, multi-hop wireless mesh networks,” in *Proceedings of the 10th annual international conference on Mobile computing and networking - MobiCom ’04*, 2004, pp. 114–128.

[28] A. Subramanian, M. Buddhikot, and O. Miller, “Interference aware routing in multi-radio wireless mesh networks,” in *2006 2nd IEEE Workshop on Wireless Mesh Networks*, 2006, pp. 55–63.

[29] K. Scott and N. Bambos, “Routing and channel assignment for low power transmission in PCS,” *Proc. ICUPC - 5th Int. Conf. Univers. Pers. Commun.*, vol. 2, 1996.

[30] S. Singh, M. Woo, and C. S. Raghavendra, “Power-Aware Routing in Mobile Ad Hoc Networks,” in *MOBICOM ’98*, 1998, pp. 181–190.

[31] A. S. Arezoomand and M. Pourmina, “Prolonging network operation lifetime with new maximum battery capacity routing in wireless mesh network,” in *2010*

The 2nd International Conference on Computer and Automation Engineering, ICCAE 2010, 2010, vol. 4, pp. 319–323.

[32] C. K. Toh, “Maximum battery life routing to support ubiquitous mobile computing in wireless ad hoc networks,” *IEEE Commun. Mag.*, vol. 39, no. 6, pp. 138–147, 2001.

[33] Jae-Hwan Chang and L. Tassiulas, “Maximum lifetime routing in wireless sensor networks,” *Networking, IEEE/ACM Trans.*, vol. 12, no. 4, pp. 609–619, 2004.

[34] J.-H. Chang and L. Tassiulas, “Energy Conserving Routing in Wireless Ad-hoc Networks,” in *Proceedings IEEE INFOCOM 2000, The Conference on Computer Communications, Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Reaching the Promised Land of Communications*, 2000, vol. 1, pp. 22–31.

[35] Y. Jin, H. Miao, Q. Ge, and C. Zhou, “Expected transmission energy route metric for wireless mesh senor networks,” *Int. J. Digit. Multimed. Broadcast.*, vol. 2011, 2011.

[36] M. Zogkou, A. Sgora, and D. D. Vergados, “Energy Aware Routing in IEEE 802 . 11s Wireless Mesh Networks,” *Int. Conf. Wirel. Inf. Networks Syst.*, pp. 215–220, 2013.

[37] R. G. Garroppo, S. Giordano, and L. Tavanti, “Experimental evaluation of two open source solutions for wireless mesh routing at layer two,” in *ISWPC 2010 - IEEE 5th International Symposium on Wireless Pervasive Computing 2010*, 2010, pp. 232–237.

[38] J. C. P. Wang, B. Hagelstein, and M. Abolhasan, “Experimental evaluation of IEEE 802.11s path selection protocols in a mesh testbed,” in *4th International Conference on Signal Processing and Communication Systems, ICSPCS’2010 - Proceedings*, 2010.

[39] “Open 802.11s.” [Online]. Available: http://open80211s.org/open80211s/. [Accessed: 05-Nov-2014].

[40] G. R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke, “IEEE 802.11s: The WLAN Mesh Standard,” *IEEE Wirel. Commun.*, vol. 17, pp. 104–111, 2010.

[41] K. Yang, J. Ma, and Z. Miao, “Hybrid Routing Protocol for Wireless Mesh Network,” *2009 Int. Conf. Comput. Intell. Secur.*, pp. 547–551, 2009.

[42] Http://www.eazytutz.com/android/android-architecture/, “Android OS Architecture,” 2015. [Online]. Available: http://www.eazytutz.com/android/android-architecture/. [Accessed: 10-Jun-2015].

[43] “Iperf.” [Online]. Available: https://iperf.fr/. [Accessed: 10-Mar-2015].

7. Appendix A – Open802.11s configuration on OpenWRT

The settings used for open802.11s was the default. In this case a new interface was created (mesh0) wtih type “mp” (mesh point) and mesh id equal to "cmmesh". All the access points use the channel 1 to communicate (2.412 MHz).

The file “/etc/rc.local” (runs at the boot time) was changed and added the following commands:

iw dev wlan0 interface add mesh0 type mp mesh_id cmmesh

iw dev mesh0 set channel 1 HT20

ifconfig wlan0 down

ifconfig mesh0 up

ifconfig mesh0 192.168.10.2

8. Appendix B – B.A.T.M.A.N-adv configuration on OpenWRT

1. Install Batman Advanced module

   $ opkg update

   $ opkg install kmod-batman-adv

2. Create a new interface “mesh” with the protocol “batadv”

   $ vi /etc/config/network

   ```
   config interface 'mesh'
           option ifname 'adhoc0'
           option mtu '1528'
           option proto 'batadv'
           option mesh 'bat0'
   ```

3. Change settings of wireless card

   $ vi /etc/config/wireless

   ```
   config wifi-device  radio0
           option type     mac80211
           option channel  1
           option macaddr  64:66:b3:16:48:27
           option hwmode    11ng
           option htmode    HT20
           list ht_capab    LDPC
           list ht_capab    SHORT-GI-20
           list ht_capab    SHORT-GI-40
           list ht_capab    TX-STBC
           list ht_capab    RX-STBC1
           list ht_capab    DSSS_CCK-40
           option disabled 0
   config wifi-iface mesh
           option device   radio0
           option ifname   adhoc0
           option network  mesh
           option mode     adhoc
   ```

```
        option ssid     batman-mesh
        option encryption none
        option bssid 02:CA:FE:CA:CA:40
```

4. Edit Batman-adv configuration file

```
$ vi /etc/config/batman-adv
config mesh 'bat0'
        option interfaces 'adhoc0'
        option 'aggregated_ogms'
        option 'ap_isolation'
        option 'bonding'
        option 'fragmentation'
        option 'gw_bandwidth'
        option 'gw_mode'
        option 'gw_sel_class'
        option 'log_level'
        option 'orig_interval'
        option 'vis_mode'
        option 'bridge_loop_avoidance'
```

5. Add the interface than the batman-adv should use to build the mesh network

```
$ batctl if add adhoc0
```

6. Configure IP and NetMask for batman-adv interface

```
$ uci set network.bat0=interface
$ uci set network.bat0.ifname=bat0
$ uci set network.bat0.proto=static
$ uci set network.bat0.mtu=1500
$ uci set network.bat0.ipaddr=192.168.11.3
$ uci set network.bat0.netmask=255.255.255.0
$ uci set batman-adv.bat0.interfaces="adhoc0"
$ uci commit
$ reboot & exit
```

9. Appendix C – Open802.11s configuration on Android

1. Install latest updates and reboot the system

$ sudo apt-get update

$ sudo apt-get dist-upgrade

$ sudo reboot

2. Download and install the Java 6 JDK

$ cd ~/Downloads

$ sudo chmod +x jdk-6u33-linux-x64.bin

$ ./jdk-6u33-linux-x64.bin

$ sudo mv jdk1.6.0_33 /usr/lib/jvm/jdk1.6.0_33

$ sudo update-alternatives --install /usr/bin/javac javac /usr/lib/jvm/jdk1.6.0_33/bin/javac 1

$ sudo update-alternatives --install /usr/bin/java java /usr/lib/jvm/jdk1.6.0_33/bin/java 1

$ sudo update-alternatives --install /usr/bin/javaws javaws /usr/lib/jvm/jdk1.6.0_33/bin/javaws 1

$ sudo update-alternatives --install /usr/bin/jar jar /usr/lib/jvm/jdk1.6.0_33/bin/jar 1

$ sudo update-alternatives --install /usr/bin/javadoc javadoc /usr/lib/jvm/jdk1.6.0_33/bin/javadoc 1

$ sudo update-alternatives --config javac

$ sudo update-alternatives --config java

$ sudo update-alternatives --config javaws

$ sudo update-alternatives --config jar

$ sudo update-alternatives --config javadoc

$ java –version

3. Install all required packages to play with Android.

$ sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl zlib1g-dev libc6-dev lib32ncurses5-dev ia32-libs x11proto-core-dev libx11-dev lib32readline5-dev lib32z-dev libgl1-mesa-dev g++-multilib mingw32 tofrodos python-markdown libxml2-utils xsltproc

4. Download the Kernel

Link: https://android.googlesource.com/kernel/msm/+/android-msm-mako-3.4-kitkat-mr0

$ mkdir ~/android

$ cd ~/android

$ mkdir kernel

$ cd ~/Downloads

$ cp msm-android-msm-mako-3.4-<VERSION> ~/android/kernel

$ cd ~/android/kernel

$ tar -zxvf msm-android-msm-mako-3.4-<VERSION>

5. Download Android arm-eabi tool chain

$ cd ~/android

$ mkdir tools

$ cd tools

$ git clone https://android.googlesource.com/platform/prebuilts/gcc/linux-x86/arm/arm-eabi-4.6

$ sudo gedit ~/.profile

Add: export PATH=:~/android/tools/arm-eabi-4.6/bin:$PATH

$ source ~/.profile

$ echo $PATH

$ export ARCH=arm

$ export CROSS_COMPILE=arm-eabi-

6. Build the Kernel

$ cd ~/android/kernel

$ make mako_defconfig

$ make menuconfig

In confguration menu follow this steps:

i. Enable option “Enable loadable module support” (using space key)
ii. Open “Enable loadable module support” menu using <Select>
iii. Enable option “Module unloading”
iv. Back to main menu
v. Open “Networking support” menu

vi. Open sub menu “Wireless”
vii. Disable option “cfg80211 - wireless configuration API”
viii. Back to the main menu
ix. Open menu “Device Drivers”
x. Open sub menu “Staging drivers”
xi. Disable option “Qualcomm Atheros Prima WLAN module”
xii. Close and save the configuration menu

Finnaly make the zimage:

$ make zImage

7. Build Backports driver

Download backport 3.16.2-1: http://drvbp1.linux-foundation.org/~mcgrof/rel-html/backports/

$ mkdir ~/android/driver

$ cd ~/Downloads

$ tar -xf backport-3.16.2-1.tar.xz

$ mv backport-3.16.2-1 ~/android/driver/

$ cd ~/android/driver/backport-3.16.2-1

$ make defconfig-wcn36xx

$ make menuconfig

And enable “Enable mac80211 mesh networking (pre-802.11s) support”

Comment “clk_disable” and “clk_enable”. Edit and Exit

$ gedit backport-3.16.2-1/compat/compat-3.6.c

//EXPORT_SYMBOL_GPL(clk_disable);

//EXPORT_SYMBOL_GPL(clk_enable);

$ make KLIB=~/android/kernel KLIB_BUILD=~/android/kernel

8. Build wcn36xx_msm driver

Link: https://github.com/KrasnikovEugene/wcn36xx

$ mv wcn36xx-master ~/android/driver

$ cd ~/android/driver/wcn36xx-master/wcn36xx_msm

$ make KLIB=~/android/kernel KLIB_BUILD=~/android/kernel

9. Root Device (Nexus 4)

To root the device (Nexus 4) was used Nexus Root Toolkit.

Download Link: http://www.wugfresh.com/nrt/

Follow this video: https://www.youtube.com/watch?v=K25n7i6zgb0

10. Install Android SDK

Original Tutorial Link: https://androidonlinux.wordpress.com/2013/05/12/setting-up-adb-on-linux/

Download SDK: http://developer.android.com/sdk/index.html

$ cd ~/Downloads/android-sdk-linux/tools

$ ./android

Select “Android SDK Tools” and “Android SDK Platform-tools” and install the two packages

$ gedit ~/.bashrc

export PATH=${PATH}:~/Downloads/android-sdk-linux/tools

export PATH=${PATH}:~/Downloads/android-sdk-linux/platform-tools

$ sudo reboot

11. SetUp ADB

$ sudo gedit /etc/udev/rules.d/51-android.rules

#x-x-x-x-x-x

SUBSYSTEM=="usb", ATTR{idVendor}=="xxxx", ATTR{idProduct}=="xxxx",

MODE="0666", GROUP="adbandy"

$ android update adb

$ cd ~/Downloads/android-sdk-linux/platform-tools/

$ ./adb kill-server

$ sudo ./adb start-server

Next, Open “Home” Directory and configure it to show hidden files and folders by pressing Ctrl + H. A bunch of Files and Folders will show up. Find the .android folder and open it. In there, you will find a file called adb_usb.ini, open it and add this line:

0xxxxx

Replace the red Xs with your Vendor ID and save the file

$ cd ~/Downloads/android-sdk-linux/platform-tools/

$ sudo service udev restart

$ ./adb kill-server

$ sudo ./adb start-server

$ ./adb devices

12. Get old boot.img from Nexus 4

Original Tutorial: http://forum.xda-developers.com/showthread.php?t=2131953

$ adb shell
$shell@mako: su
$root@mako: dd if=/dev/block/mmcblk0p6 of=/sdcard/boot.from_n4.img
$root@mako: exit
$shell@mako: exit
$ cd ~/android/
$ mkdir nexus
$cd nexus
$ adb pull /sdcard/boot.from_n4.img

13. Create new boot.img with ABOOTIMG

Link: http://www.ubuntuupdates.org/package/core/precise/universe/base/abootimg

$ cp boot.from_n4.img myboot.img
$ abootimg -u myboot.img -k ~/android/kernel/arch/arm/boot/zImage
$ adb boot myboot.img

$ adb shell
$shell@mako: cd sdcard
$shell@mako: mkdir mesh_driver
$shell@mako: exit

14. Copy the new boot.img to device

To copy the new boot.img to device (Nexus 4) was used Nexus Root Toolkit.
Follow this video: https://www.youtube.com/watch?v=bHktiqjXbDk

15. Move drivers to Nexus 4

Drivers location:
backports-3.16.2-1/compat/compat.ko
backports-3.16.2-1/net/wireless/cfg80211.ko
backports-3.16.2-1/net/mac80211/mac80211.ko
backports-3.16.2-1/drivers/net/wireless/ath/wcn36xx/wcn36xx.ko
wcn36xx-master/wcn36xx_msm/wcn36xx_msm.ko

Use ADB to push the drivers to device:

$ adb push compat.ko sdcard/mesh_driver

$ adb push cfg80211.ko sdcard/mesh_driver

$ adb push mac80211.ko sdcard/mesh_driver

$ adb push wcn36xx.ko sdcard/mesh_driver

$ adb push wcn36xx_msm.ko sdcard/mesh_driver

16. Install IW

Download IW compiled for Android -> www.onlyxool.net/wp-content/uploads/2012/06/iw.tar.gz

$ adb push ~/Downloads/iw /sdcard/

$ adb shell

$shell@mako: su

$root@mako: cp sdcard/iw data/local/tmp

$root@mako: chmod 755 data/local/tmp/iw

$root@mako: mount -o rw,remount /system

$root@mako: cp data/local/tmp/iw /system/bin

$root@mako: mount -o ro,remount /system

17. Install IPERF

Download Iperf compiled for Android -> http://sourceforge.net/p/iperf/patches/23/

$ adb push ~/Downloads/iperf /sdcard/

$ adb shell

$shell@mako: su

$root@mako: cp sdcard/iperf data/local/tmp

$root@mako: chmod 755 data/local/tmp/iperf

$root@mako: mount -o rw,remount /system

$root@mako: cp data/local/tmp/iperf /system/bin

$root@mako: mount -o ro,remount /system

18. Create a script to load drivers and setup the mesh network

#!/bin/bash

lsmod /sdcard/mesh_driver/compat.ko

lsmod /sdcard/mesh_driver/cfg80211.ko

lsmod /sdcard/mesh_driver/mac80211.ko

lsmod /sdcard/mesh_driver/wcn36xx.ko

lsmod /sdcard/mesh_driver/wcn36xx_msm.ko

iw dev wlan0 interface add mesh0 type mp mesh_id cmmesh
iw dev mesh0 set channel 1
ifconfig wlan0 down
ifconfig mesh0 up
ifconfig mesh0 192.168.10.6
iw dev mesh0 set power fixed 0

10. Appendix D – New Artime Link Metric with energy consideration

1. Disable USB charging

$ adb shell

$shell@mako: su

$root@mako: echo 0 > /sys/class/power_supply/usb/device/charge

2. Install Fast Discharge App

Download Link: www.apk20.com/apk/192596/start

$ cd ~/Downloads/

$ adb install jp.gr.java_conf.taketake.KyusokuHouden.apk

3. Change Airtime Link Metric calculation

For change the Airtime Link metric calculation:

$ vi backport-3.16.2-1/net/mac80211/mesh_hwmp.c

4. Re-build kernel modules

$ cd ~/android/driver/backport-3.16.2-1

$ make defconfig-wcn36xx

$ make menuconfig (enable “Enable mac80211 mesh networking (pre-802.11s) support”)

$ make KLIB=~/android/kernel KLIB_BUILD=~/android/kernel