FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

User-provided Networks: Relaying vs. Ad-hoc Routing

Luís Miguel Moreira de Carvalho

Master's Dissertation carried-out in the framework of the
Mestrado Integrado em Engenharia Electrotécnica e de Computadores
Major Telecommunications

Advisor: Dr. Rute Sofia (INESC Porto)

Co-Advisor: Prof. José Ruela (FEUP/INESC Porto)

July 2009

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES **2008/2009**

A Dissertação intitulada

"USER-PROVIDED NETWORKS: RELAYING VS. USER-PROVIDED NETWORKS: RELAYING VS. AD.HOC ROUTING"

foi aprovada em provas realizadas em 17/Julho/2009

o júri [illegible]

Presidente Professor Doutor José Alberto Peixoto Machado da Silva
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Artur Miguel do Amaral Arsénio

Professor Doutor Artur Miguel do Amaral Arsénio
Professor Auxiliar do Departamento de Engenharia Informática do Instituto Superior técnico da Universidade Técnica de Lisboa

[illegible]

Professor Doutor José António Ruela Simões Fernandes
Professor Associado do departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da universidade do Porto

Helena Rute Esteves Carvalho Sofia

Doutora Helena Rute Esteves Carvalho Sofia
Investigadora INESC Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Luís Miguel Moreira de Carvalho

Autor - LUÍS MIGUEL MOREIRA DE CARVALHO

Faculdade de Engenharia da Universidade do Porto

Resumo

Este documento foi submetido como dissertação de Mestrado, como requisito parcial para conclusão do Mestrado Integrado em Engenharia Electrotécnica e de Computadores (MIEEC). O foco principal desta tese é a análise de performance de uma forma simples de relaying wireless (WiFi) em ambientes user-centric. Para alcançar resultados relevantes, a tese foi dividida em duas partes principais: i) análise de trabalhos relacionados; ii) implementação e validação.

No que diz respeito ao ponto i), este trabalho descreve a evolução das arquitecturas wireless, as principais abordagens de routing em wireless, bem como os principais aspectos de relaying. Relativamente ao ponto ii), a tese descreve as ferramentas utilizadas, tanto em implementação, como em validação.

A avaliação de performance é feita através de uma comparação significativa, i.e., relativamente a uma forma de routing multihop. Nesta tese são descritos os detalhes de implementação, contribuições, bem como os principais desafios enfrentados. Além disso, são também mencionadas as perspectivas de trabalho futuro.

Abstract

This document was submitted as the MSc dissertation, to partially fulfil the requirements to conclude the Mestrado Integrado em Engenharia Electrotécnica e de Computadores (MIEEC). The thesis is focused upon the performance analysis of a simple form of wireless (WiFi) relaying in user-centric environments. To achieve adequate results, the thesis work has been split into two main parts: i) analysis of related work; ii) implementation and validation.

In regards to i) the thesis goes over the evolution of wireless architectures, main routing approaches in wireless, as well as main aspects of relaying. In regards to ii), the thesis describes a set of tools that have been applied both in terms of implementation and validation.

Performance evaluation is provided against a significant benchmark, i.e., one form of multi-hop routing. Details concerning implementation, contributions, as well as challenges faced are provided in this thesis. Future work is also described.

“In theory, there is no difference between theory and practice.
But, in practice, there is.”

Unknown

Contents

List of Figures

List of Tables

Acronyms and Symbols

Acronym	**Meaning**
ARP	Address Resolution Protocol
AODV	Ad-hoc On-demand Distance Vector
AODV-PA	Ad-hoc On-demand Distance Vector Path Acumulation
AP	Access Point
CTS	Clear to Send
DAG	Direct Acyclic Graph
DARPA	Defense Advanced Research Projects Agency
DSDV	Destination-Sequenced Distance Vector
DSL	Digital Subscriber Line
DSR	Dynamic Source Routing
DV	Distance Vector
DYMO	Dynamic Manet On-demand
FOSS	Free Open Source Software
HAL	Hardware Abstraction Layer
LAN	Local Area Network
LS	Link State
LWAP	Light-Weight Access Point
MAC	Medium Access Control
MADWiFi	Multiband Atheros Driver for Wireless Fidelity
MANET	Mobile Ad-Hoc Network
MGEN	Multi-Generator
MPR	Multipath Relay
NAT	Network Address Translation
NIST	National Institute of Standards
NTP	Network Time Protocol
OLPC	One Laptop per Child
OLSR	Optimized Link State Routing
OSI	Open Systems Interconnection
PAN	Personal Area Network
RERR	Route Error
RREP	Route Response
RREQ	Route Request
RTS	Ready to Send
RTT	Round-Trip Time
SSID	Service Set Identifier
TC	Topology Control

TORA	Temporarily-Ordered Routing Algorithm
UPN	User-provided Network
VoIP	Voice Over IP
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network

Chapter 1

Introduction

Internet services and models have been going through a paradigm shift, product of three main factors: i) widespread wireless technologies; ii) increasing variety of user-friendly and multimedia-enabled terminals; iii) availability of open-source tools for content generation. Together, these three factors are changing the way that Internet services are delivered and consumed, first because the end-user has a particular role in controlling content as well as connectivity, based upon *cooperation*. Specifically focusing upon Internet access, Internet *connectivity* models that rely upon cooperation are already being commercially adopted in some networks.

In regards to Internet access (connectivity), *Wireless Local Area Networks (WLANs)* are now widely deployed at private homes and public spaces, with a tendency to spread further. At the same time, end-users expect to have access to broadband mobile services at low cost and in a way that is at the same time easy and intuitive. This is not a trivial deployment, given that new radio systems imply additional operational costs for access operators. Therefore, access operators are normally reluctant to invest on such type of systems. Moreover, the currently deployed private WLAN environments are normally not used at their full capacity. For instance, it is common nowadays to complement DSL access with one wireless *Access Point* (AP) per household, serving in average a few users and a small set of end-user wireless devices (in average, 2 or 3). On the one hand, such configuration implies that radio resources are not being fully used. On the other hand, due to the fact that it is common to have more than one wireless access point within the same range, there is strong spectrum overlap, which undermines the wireless coverage and disrupts the quality to be provided.

A potential way to overcome the mentioned problems is an emerging wireless architecture which is coined *user-provided* in [1]. *User-provided Network* (UPN) represents an emerging wireless architecture which can be created spontaneously based on cooperation incentives within a community of Internet users. The main distinction between UPNs and other wireless autonomic models is that a user surpasses its role as a simple consumer of services, to play a more active role in providing networking services, e.g., sharing his/her Internet access. This is a new concept that attempts to integrate the new role of a user both as a provider and consumer of services in the network. It should be noticed that user-provided networking is not a new proposal of a wireless network; instead, it is a concept that attempts to categorise a phenomenon that is today in an embryonic but already commercial state. For instance, companies such as Whisher [2] or FON [3] already provide the possibility for an end-user to share his/her Internet access on-the-fly by recurring to different methodologies. UPNs are further detailed in section 2.1.4.

Being based upon end-user wireless-enabled devices, UPNs are based upon privately owned WLANs. This implies that the mode applied may be ad-hoc or infrastructure. In addition, UPNs integrate as part of the network, equipment that is not, as of today, capable of performing routing. Instead and as described in [1], UPNs may rely on a user-centric relaying model, where connectivity sharing is placed in the end-user device, thus allowing other trusted nodes to take advantage of such connectivity. Therefore, UPNs may include routers but normally the devices simply relay connectivity [1]. There are both advantages and disadvantages to this. With relaying, connectivity has a short-range, but the global status and system is simple. Furthermore, the option on whether to relay or to route is strongly associated to the topology and to the available cooperation mechanisms, both of which in UPNs are expected to change dynamically and frequently.

This thesis represents a first step towards understanding the potential advantages and disadvantages of relaying when compared to multihop routing. The thesis work contemplates state-of-the-art analysis, concept development, as well as performance evaluation on a realistic testbed.

This report corresponds to the outcome of the thesis work and is organised as follows. Chapter 2 gives an overview of the related work and state-of-the-art analysis, explaining all the related technologies and concepts necessary to fully understand the problems raised by this thesis. It goes over related work, focusing on the concept of ad-hoc networks (single-hop and multihop), features and information dissemination, i.e., main routing protocols, and related work concerning UPNs and wireless relaying techniques.

Chapter 3 goes over the specific problem this thesis relates to, detailing the problem's motivation and goals. In addition, it gives a brief description of the tools necessary to the performance tests, as well as mentioning what we had to start with.

On chapter 4 we detail the contributions and improvements made to the tools used throughout the thesis work. This chapter describes the problems faced due to working on existing implementations of every tool necessary to the thesis main experiments, followed by a detailed description of what was improved and the methodology applied.

Chapter 5 details the particular parameters of the experiments. We go over the purpose of every experiment, as well as the most important parameters, along with a description of the methodology followed. This chapter is concluded by the presentation and discussion of the results achieved. Chapter 6 concludes this document, summarising work performed as well as steps to take in the follow-up of this work.

Chapter 2

Related Work

The notion of wireless communications being supported by the end-user (end-user empowerment) is a recent trend that has been steadily evolving. This was possible due to the evolution of wireless technologies. This section provides an overview of such evolution and emphasis is provided into the available routing mechanisms.

After this introduction, section 2.1 gives an overview of wireless (WiFi)[1] architectures evolution, starting with the infrastructure mode until the most recent user-centric paradigms. The section will cover main features, drawbacks and advantages, as well as main application scenarios for each of the architectures. Special attention is given to describing the concept of UPNs, living examples, challenges that this trend currently faces, and how this thesis contributes to such topic.

Section 2.2 describes multihop routing, features, drawbacks, advantages, as well as application scenarios, given that one of the main goals of this thesis is to analyse the advantages and best scenarios for using simple relaying techniques instead of multihop routing, assuming that nodes frequently move and that several nodes are willing to share Internet access.

Section 2.3 provides more detail into the form of multihop routing chosen to be applied as the benchmark for the performance evaluations, i.e., AODV/DYMO. Finally, in section 2.4 there is an analysis of relaying methods that can prove useful to the wireless networks in study, either on layers 2 as 3 of the OSI model. The chapter concludes in section 2.5, which gives a summary of the related work analysis.

2.1 Wireless Architectures Evolution

Two different models for wireless networks exist today: *infrastructured* and ad-hoc. These two models take different approaches about who controls the network as well as how connections between nodes are made.

[1]The term wireless will be, in this document, used to refer to WiFi.

2.1.1 Infrastructure Mode

In a managed network, control is centralised, which implies that there are specific nodes, called APs, which manage connections on a point-to-point, wireless basis. It should be highlighted that a unicast session in wireless still implies broadcast transmission, due to the permissive wireless mode.

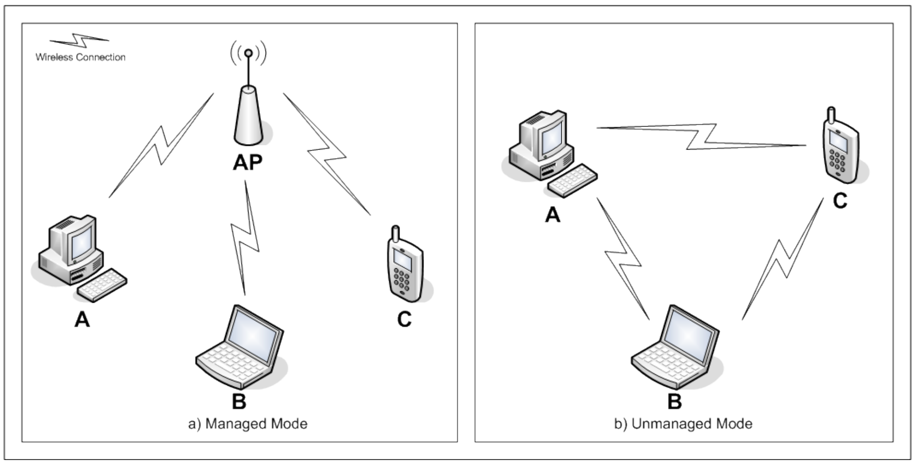

Figure 2.1: Wireless models.

In the managed model (cf. Figure 2.1.a), the AP is responsible for controlling radio resources. In this mode, a station wishing to transmit sends an *Request to Send* (RTS) packet to the AP, which responds with a *Clear to Send* (CTS) if no collision is detected. The AP sends periodic beacons so that every station keeps its clock synchronised. This means that, in a managed network, every communication involves the AP, i.e. a station transmits its packets to the AP and the AP routes it to the desired destination, making it possible for a station to communicate with another, even if it is not at its direct reach.

Advantages of this model are the possibility for the network (and consequently the access operator) to better manage resources. The flip-side to this is that it requires some specific hardware and some previous planning of the network, undermining the possibility for users to start communication sessions spontaneously, where and whenever they want.

2.1.2 Single-Hop Ad-hoc

The next evolutionary step in wireless architectures towards an autonomic behaviour is the unmanaged, or ad-hoc, model (cf. Figure 2.1.b). Ad-hoc networks are made by nodes directly connected to each other without a centralised control AP, where the broadcast nature of wireless communications is used as an advantage to discover new nodes within reach. To begin transmission, a station sends a RTS to the destination and waits for a CTS, just like in the managed mode.

Other stations know the contention periods by hearing the RTS/CTS packets. To keep clocks synchronised, every station tries to send a beacon, but just one gets access to the medium at any given time slot. They all try again in the next available period.

The nature of ad-hoc grants that there is no centralised management over the network, allowing participants to create networks at will, based upon their wireless equipment capabilities, requiring each node to act like both a client and a server, leaving the network control distributed equally over each node.

In a basic ad-hoc configuration, communication is *single-hop*, which means that every node in the network communicates directly with the others. This topology has the advantages of requiring no routing whatsoever, as the broadcast nature of wireless technologies is used to get packages to their destinations. However, to achieve this, every node must be within reach of every other station in the network, which is not a scalable solution. As a consequence of this scalability problem, single-hop ad-hoc networks can only be implemented on small networks with a very limited number of users, distributed over a small area, and low bandwidth. Nonetheless, its simplicity in configuration makes it ideal for some basic setups such as home networks (e.g. to share files, printers, play multi-player games), *Personal Area Networks* (PANs) with technologies like Bluetooth [4], and many other uses whenever a small group of users need to establish connectivity among all the elements of such group.

2.1.3 Mobile Ad-hoc Networks

In order to solve the scalability problem, ad-hoc networks evolved to become *multihop* thus allowing nodes to communicate with peers that may not necessarily be directly connected.

As we can see in Figure 2.2, node A is not directly connected to node D, which means that if node A wants to send data to node D, it will have to be forwarded by other nodes (for instance by node B). The path through which the data travels is determined by the protocol and metrics used.

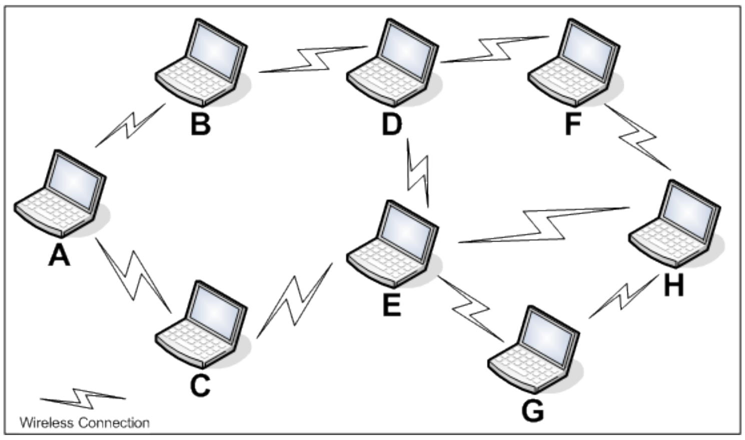

Figure 2.2: Example of a multihop network.

The main applications of this ad-hoc model are *Mobile Ad-hoc Networks* (MANETs). These are multihop networks, where nodes are mobile terminals communicating via some wireless technology (usually IEEE 802.11). In a MANET, there are routing algorithms to take care of computing the best paths (according to specific metrics) from an end-to-end perspective, allowing nodes that are not directly connected to communicate in an efficient way. In order for MANETs to scale, a robust protocol is needed, as with the mobility of the nodes, in conjunction with the unpredictability of the wireless links makes it more difficult to maintain a path from two distant nodes in a MANET.

The versatility and ease of deployment of these networks makes them ideal for a large variety of scenarios [4], such as:

- **Military operations.** It actually was the military that triggered the development of ad-hoc networks, with the DARPA Packet Radio project, in 1972. In a battlefield, there may be no opportunity to deploy the infrastructures necessary to a managed network, and shall these infrastructures be destroyed, communications are compromised. In addition, the constant mobility of units call for a less rigid solution, so a robust and reliable multihop network is a perfect solution for the modern battlefield.

- **Emergency situations.** In case of a natural disaster that takes out communication infrastructures in an urban area, emergency ad-hoc networks can be easily and rapidly deployed, allowing communication between rescuing agencies.

- **Communication between vehicles.** VANETs (vehicular ad-hoc networks) are a new emerging application of multihop ad-hoc. Vehicles in traffic can communicate with each other, passing information about crashes, traffic jams and emergencies.

- **Internet distribution in underdeveloped countries.** The One Laptop Per Child (OLPC) program uses low-cost laptops distributed in underdeveloped countries. These laptops connect with each other, forming mesh-networks, even when the laptops are turned off. Some central points, like schools, have wired Internet connections and share these connections with the OLPCs. This way Internet connection can be spread to a large area without the need to set up infrastructures.

- **Easy sharing of files or connectivity.** Because these networks don't required planning nor additional hardware, they are ideal for a group of users that suddenly need to connect their equipments to share files, play network games, etc.

2.1.4 User-provided Networks

This section presents an analysis of the UPN model, having [1] as main reference.

The main idea behind UPNs is the ability of a user to become more than just a simple consumer of Internet services and instead also be a provider, as happens today in the field of energy with microgeneration models. Users willing to provide some form of service to a community (e.g. to share their already subscribed connectivity) are coined *micro-providers* in UPNs, as they will be providing a networking service (e.g. connectivity, in a network that spreads around them). This provisioning has to be made simple to the end-user, requiring little or zero configuration, and, at the same time, the user must be completely aware of what the implications of such doing are: such provisioning has to obey to the rules stipulated by the access towards customers.

The UPN architecture is therefore tightly related to trust management and to cooperation incentives. This architecture in intertwined with human aspects such as mobility patterns and also such as trust management, given that nodes in a UPN are normally nodes that humans rely on for personal use. Being based upon user equipment, UPNs are highly heterogeneous, with elements ranging from powerful desktop computers, wireless access points, and laptops, to simple mobile terminals like mobile phones and other multimedia equipment with wireless capabilities. This diversity also has implications on the network topology, being the underlying architecture expected to mix both ad-hoc and infrastructure mode segments.

Another relevant feature of UPNs is its autonomic behaviour. The diversity of the network allied with the lack of a centralised control over node mobility and behaviour makes it crucial for the network to self-organise. This self-organisation is necessary to keep traffic loads balanced between micro-providers. Challenges currently being faced by this architecture are:

- **Security concerns.** As with multihop networks, security concerns come mainly from the fact that there is little to no control over who enters the network and who is forwarding traffic. The presence of a malicious or misbehaved user can cause massive damages o the network. However, in contrast to ad-hoc, cooperation incentives are the basis for a strong network of trust. In [1] the authors claim that UPNs do not require tight security control mechanisms and instead, should integrate innovative trust management schemes that mimic human trust behaviour.

- **Building networks of trust.** The model of connectivity sharing and relaying in user-provided is highly dependent upon trust between peers. These trust mechanisms should follow human relationships and socialisation.

- **Preventing access subscription violation by micro-providers.** There is usually nothing to prevent a user from sharing its Internet connection with other users. However, having multiple users accessing through the same connection can bring other issues, like exceeding traffic limitations imposed by a specific subscription agreement.

- **Routing/forwarding schemes.** The diversity of equipment within the network may include elements incapable of routing traffic. Furthermore, simple relaying techniques may be best suited in some topologies.

2.1.4.1 Living-Application Scenarios

There are already some commercial applications of an embryonic UPN architecture. That is the case of the communities of FON [3], OpenSpark [5], Whisher [2], among others.

FON is a *virtual operator*, i.e., an entity that simply coordinates Internet access by means of a UPN where the incentive to share Internet access is simply broader roaming. FON users are encouraged to share their Internet access with users of the worldwide FON community in order to obtain global WiFi coverage, i.e. a user sharing its connection will have free access to the shared network of other FON users. FON users share their connection by means of a wireless router, which is configured with two different SSIDs. One is used for private purpose; while the other is open to all FON members (registration in FON is free). Initially, the FON routers (Foneras) were distributed freely to new user, but are now sold at low-cost. In addition to this basic "I-share-you-share model", FON also incorporates the regular models of pre-payed access (a liens) and of hotspot model. The key aspect to highlight here is that FON is not a vendor nor an operator. FON does not own any type of infrastructure. Instead, it simply keeps a database of its freely registered members. The hardware provided by FON is simply resold. The infrastructure to provide Internet access is in fact already owned by each of the FON users. Hence the notion of virtual operator.

OpenSpark was in fact the first entity to implement and commercialise the concepts shared by these communities. FON inherited most of the aspects of OpenSpark. This is a decision of the micro-provider, it might be willing to share connection with visitors for free, or profit a little with that. Again like FON, OpenSpark uses access points to allow users to share connectivity, but users are also offered the possibility of downloading firmware for a compatible router they already have, instead of purchasing a new one.

The approaches in FON and OpenSpark are rather static, in the way that connection sharing is done by a fixed wireless access point. This way, mobility is only ensured by keeping connected users in range of the access point. Whisher takes a different approach in the way that users share connectivity. It presents a solution that is dependent on software alone. This allows a user to install Whisher software on any platform with a wireless card already available, immediately starting providing connectivity anywhere, resulting in greater mobility possibilities and a more dynamic network. These first applications of user-provided concepts are still very limited. None of them uses ad-hoc network modes, ignoring the possibilities and versatility that unmanaged networks can bring. None makes any attempt at cooperating with existing concurrent access points, i.e., if two access points exist in nearby locations, they are still two separate networks concurring for spectrum, not delivering full potential to connected users.

2.2 Multihop Routing

The previous sections described briefly the evolution of wireless architectures. This section is dedicated to the debate of multihop routing, given that it is one of the main building blocks in this thesis.

To overcome the scalability drawbacks imposed by single-hop ad-hoc, whilst keeping the unmanaged nature of ad-hoc and the simplicity of configuration, research was undertaken to make multihop ad-hoc feasible.

Multihop ad-hoc means that information is *forwarded* (hopped) across multiple points of the network, making it possible for nodes not in direct reach to communicate, expanding the limitations of ad-hoc. This implies that elements of the network will have to forward each other's packets, following some kind of paths to destinations. Because information about routes is not available when a node joins the network, these paths will have to be computed in a later stage, in a distributed manner, allowing for traffic to be routed through the network.

As every node is randomly positioned in the network area, each node may not be within transmission range of all others. In addition, nodes are mobile, so the interconnections between nodes may change very rapidly through time. This is a very problematic situation in terms of *routing* and packet delivery, as static routes are hard to implement. To solve this problem, every node in the network is required the additional effort to forward the packets of its neighbouring nodes. The specifics of routing algorithms used will be addressed later in this document.

This brings a set of unique conditions that raises problems that do not exist in the wired and managed equivalents. Problems like robustness, scalability, vulnerability, capacity, are more aggravated in ad-hoc networks. Main issues that arise are:

- The fact that nodes share the same medium without a centralised control over radio resources means that there will be scalability problems. When node concentration is high, nodes will have trouble sending their packets because of the shared medium and IEEE 802.11 RTS/CTS contention windows. Besides that, RTS/CTS does not solve the exposed terminal problem, which will increase as the network grows larger [6].

- As each nodes' traffic is forwarded wirelessly through other nodes, the number of transmissions generated per packet is higher (remember that every transmission is broadcasted). This drops the total capacity available in the network [7].

- Results show that the maximum per-node capacity of a large network with N nodes in random positions is of $O(1/\sqrt{N})$ [8]. Which means that the capacity approaches zero as the network grows.

- Nodes detect link failures and so, must re-compute the routes to destinations. For that purpose nodes must exchange messages to compute new paths, which in turn causes bandwidth overhead that decreases network capacity. [6]

- Because nodes have to forward each other's traffic, these networks are more vulnerable to security threats, besides the security issues resulting from wireless technologies. This is aggravated by the fact that there is usually no control over who is connected to the network [9].

Transmission of information from an end-to-end perspective and having as basis an ad-hoc network requires, from the network elements, capability to choose on-the-fly paths to follow. Hence, in ad-hoc multihop networks (from now on referred to as multihop), each single node participates in path computation so that paths can be dynamically established, ensuring reliability in packet delivery end-to-end, even among nodes that are not within the direct range of each other. It should be noticed that ad-hoc has some features which make the implementation of normal routing protocols used in wired networks unfeasible. For instance, wireless equipments are often battery operated and have a very limited bandwidth (when compared to their wired equivalents), so routing protocols should be made specifically for these networks, keeping energy consumption and generated traffic overhead to a minimum and at the same time tolerate a high degree of node mobility.

The next section goes over the different existing categories of multihop routing, highlighting aspects such as categories related to the way propagation of information is performed, or when to compute routes.

2.2.1 Categories

Existing routing protocols for multihop networks follow different assumptions on the network and very different strategies to compute paths. As such, routing protocols have different classifications, depending on what parameters we look at. The following classification is provided based upon the survey done in [10]. Such classification is highly relevant when considering routing schemes to be applied, and even new schemes to be devised. Main categories are:

- **Propagation of Information: Link state vs. Distance Vector Routing.** In what concerns propagation of information, multihop routing can be characterised as link state (LS) or distance vector routing (DV).

  These categories are the ones also present in conventional (wired) networks. In LS, a node exchanges information based on a packet containing link state information about all of its neighbours, each time there is a topology change. This way, every node knows the complete topology of the network (the network is flooded each time) and can compute routes to any destination it wants. The problem with this approach is greedy behaviour, which results in flooding of the network. Even in regular ad-hoc scenarios topology changes are expected to occur frequently and therefore, flooding is bound to happen frequently thus resulting in high routing overhead.

  As for DV, nodes only store information about the next hop and distance of any given destination. The route is then constructed on-the-fly, with every node forwarding it to the next hop in its table. The problem with this scheme is slow convergence and the occurrence of routing loops, as wells as of count-to-infinity.

- **When to Compute Routes: Pre-computed vs. On-demand Routes.** Pre-computed strategies usually demand periodic updates from nodes so that any node knows the network topology in any given time. The upside is that every node keeps up to date tables with entries for all routes in the network. The downside is that periodic updates take constant bandwidth, whether a node needs to transmit or not.

On-demand routing means that nodes will only request nodes from the network when they need, keep only routing tables with paths they need. This decreases the overhead in the network; however, when the node first needs the route, it isn't there, so there is a high initial delay.

- **How to Disseminate Routing Updates: Periodical vs. Event-driven Updates.** Information about network topology and node connections is broadcasted in the network, periodically or only when specific events occur (such as nodes coming and going). When using periodical updates, the period must be controlled carefully, finding a good compromise between keeping information up-to-date and bandwidth overhead. Even-driven schemes only disseminate information when needed, but if the network is too dynamic this may spend too much bandwidth.

- **Type of Ad-hoc Topology: Flat vs. Hierarchical Structure.** In a flat structure every node is treated equally and takes the same part in path computation. This is easy to implement, but does not scale very well. A large flat network may generate too much routing traffic.

  In a hierarchical structure, nodes are aggregated into small groups of nodes (clusters), and then these groups are aggregated into larger groups, and so on. Then, routing information is usually disseminated differently inside a cluster and outside the cluster border. This solves the problem of flat hierarchies, but is harder to implement and maintain.

- **How and Where to Compute Routes: Decentralised vs. Distributed.** In a decentralised scheme, every node must maintain sufficient topology information so that it can compute paths to any destination, using only information that it has.

  Distributed route computation means that nodes store only information about local links. When a route is needed, several nodes cooperate in order to form the path.

- **Metrics: Source Routing vs. Hop-by-Hop.** In source routing, nodes send the complete path (i.e. every node the packet must pass from source to destination) on the packet header. The down side is that sending all this information in the header will increase routing bandwidth usage. The advantage is that path computation is completely done by the source node.

  Hop-by-hop routing uses the opposite strategy. Nodes only store information about the next hop of the path (who to send the packet to). This brings traffic overhead down, but has the disadvantages that every node must keep routing information and may lead to loops.

- **Resilience: Single path vs. Multiple Paths.** Designing a protocol that obtains a single path to a destination is simpler but a protocol that finds multiple paths to a destination is more robust. Having multiple paths to a destination, a node can choose the best route to follow and alternative paths in case of failure.

2.2.2 Main Multihop Routing Protocols

The previous section gave general classification aspects of multihop routing protocols. This section provides a classification and brief overview of the main protocols used in multihop networks, namely, DSR [11, 12], OLSR [13], TORA [14, 12], AODV [15, 12] and DYMO [16]. These protocols are capable of enjoying loop freedom in ad-hoc, which is a crucial aspect to take into consideration given to the highly dynamic topologies in ad-hoc.

2.2.2.1 DSR

The *Dynamic Source Routing protocol* (DSR) falls into the on-demand, flat structure, source routing and multiple path protocol categories. An operational example for DSR computation is provided in Figure 2.3, where node A (source) wants to communicate with node F (destination). In order to compute the path to F, A starts by flooding the network with a message of type *Route Request* (RREQ). These packets are flooded to the destination node (F), or to a node that has a valid route to the destination. Upon reaching the destination, a *Route Reply* (RREP) packet is transmitted from destination to source, through the shortest (hop-count) path. This packet is source routed, which means that every node in the route will also learn routes to the source and destination.

Each DS packet contains the route it plans to use to reach a destination. In addition, each request has a unique identifier (sequence number) to prevent nodes from repeating the same request. If a node is unable to reach the next node, an error message must be sent back to the sender so the route can be destroyed and recomputed. A more detailed explanation of DSR can be found in [11].

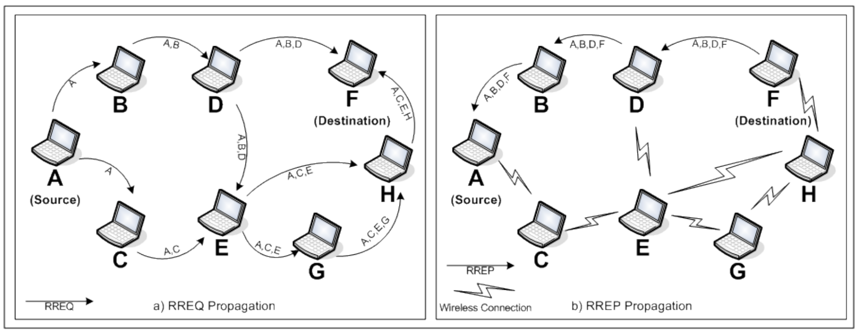

Figure 2.3: DSR example.

An advantage of DSR is that, by reacting on-demand, it does not require periodic route updates. A second advantage is that it relies on path accumulation to perform topology discovery and hence avoids the need to set a routing table.

There is an overhead associated with the process of discovery, but this overhead can be minimised by using adequate caching. However, there are a few disadvantages associated with DSR. First, it implies bi-directional links, i.e., the reply must be sent in-path, on the route previously traversed by the RREQ message. Second, when a node is not capable of forwarding a message and if there is no alternative path on its cache, then the packet is dropped. It should be noticed that there is an optimisation to deal with this situation which takes advantage of having the node's identity in the source route and also of the fact that nodes overhear packets sent to neighbours.

2.2.2.2 OLSR

The Optimized Link State Routing protocol (OLSR) [17] is a LS proactive protocol. Being derived from its LS counterparts, OLSR (such as OSPF [18] or IS-IS [19]) uses a greedy approach to perform topology discovery, i.e., LS normally requires flooding of the topology information across all nodes in a network, to keep the topology perspective of each router synchronised. Albeit being LS based, OLSR prevents such flooding to happen fully using a two-step approach. First it relies on the regular HELLO exchange to notify adjacent neighbours. Second, it relies on multipoint distribution relay (MPR) nodes to perform topology information distribution. MPR nodes then source and forward TC messages which contain the MPR selectors. This functioning of MPRs makes OLSR unique from other link state routing protocols in a few different ways: The forwarding path for TC messages is not shared among all nodes but varies depending on the source, only a subset of nodes source link state information, not all links of a node are advertised but only those which represent MPR selections. This way, MPRs assist OLSR to reduce signalling overhead (which is common in LS approaches), by reducing the need for redundant transmissions.

A scheme to its basic functioning is provided in Figure 2.4. Every node sends periodic HELLO packets to its one-hop neighbours, with a list of its neighbours. Nodes receiving a HELLO packet must reply with their own HELLO messages. This will allow every node to select a list of nodes to be its *Multi Point Relay* (MPR), so that every two-hop neighbour is connected through a MPR. MPRs are used to rebroadcast periodical *Topology Control* (TC) messages that will allow every node to get sufficient topology information to construct routes to any given destination. Detailed information concerning OLSR can be found in [13].

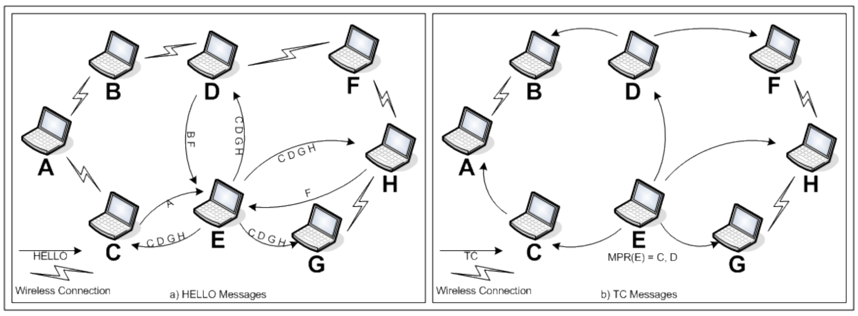

Figure 2.4: OLSR example.

OLSR main advantage is that, by being LS, it prevents routing loops. Main disadvantages of OLSR relate to its lack of security, given that any node can inject malicious information; reasonable overhead related to keeping routes to all nodes (in most cases, this is not necessary) as well as not supporting multicast.

2.2.2.3 TORA

The *Temporally Ordered Routing Algorithm* (TORA) can be classified as a reactive protocol. Two main aspects that make it different from the previous approaches is that TORA is based on link-reversal algorithms, while the other approaches derive from single-source shortest-path routing. Furthermore TORA was devised to work on flat addressing environments while the remainder approaches suit hierarchical addressing environments. TORA focuses on finding a working route, not necessarily optimal (shortest-path for instance). When building a route, TORA forms a *Directed Acyclic Graph* (DAG) rooted at the destination, using the notion of heights in every node. Information can only flow from a 'higher' node to ones with smaller metrics. TORA's height metric is composed by: logical time of a link failure; the unique ID of the node that defined the new reference level; a reflection indicator bit; a propagation ordering parameter; the unique ID of the node. Upon a link failure, nodes flood reference levels to form a new DAG.

TORA has the main disadvantage of requiring synchronisation of all nodes. A more detailed description of the protocol is in [14].

2.3 Ad-hoc On-demand Distance Vector

AODV is a DV reactive multihop routing protocol. It is intended for scenarios where mobile wireless nodes are connected in an ad-hoc fashion, being the topology characterised by a mix of both static and mobile nodes, frequent changes in link connectivity, frequent movement of nodes, low processing, low network utilisation as well as low memory overhead. In addition, AODV mitigates loop occurrences by relying on destination sequence numbers.

Having the capability to compute paths along multiple hops, at the same time keeping a low control overhead in the network allows AODV to be scalable well. The mechanisms of route recovery, in case of link failure due to node mobility, give it the capability of sustaining highly dynamic networks. Control messages are kept simple, requiring low processing, keeping CPU requirements low and increasing the life of battery-operated equipments.

2.3.1 Protocol Overview

At first a node must find its neighbouring nodes. This can be done recurring to the periodic broadcast of HELLO messages (cf. Figure 2.5). If a node receives a HELLO message it adds the sender of the message as its neighbour. If it fails to receive a pre-determined number of HELLO messages, it deletes this entry from the list.

HELLO messages can generate a great deal of traffic overhead, so, if another method to monitor link connectivity exists (like link layer feedback), these messages are not used.

When a node joins the network, its routing table is empty, so, in order to be able to establish connections, it must find paths to the desired destinations. A node initiates a route discovery

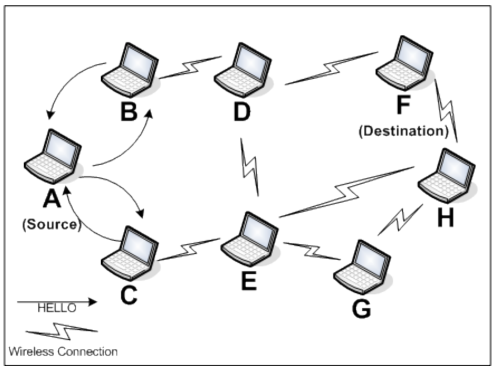

Figure 2.5: AODV example - HELLO broadcast.

process only when it needs a route to that specific destination and routes are only maintained while they are needed (each route has an expiration timer to ensure they are kept fresh). This means that when a process first needs a route, it will not be present in the nodes' routing table, so a greater initial delay exists. It is worthy to note that a routing table entry consists of: destination; next hop; destination sequence number; number of hops; active neighbours for the route; expiration time. It is not stored the full path to the destination.

To mitigate routing loops, AODV uses a sequence number mechanism similar to DSDV. Each node keeps an independent sequence number for each route it knows. A node requesting a route compares the sequence number of the destination sequence number in the route request with its own sequence number, updating it to the highest of these values.

The route discovery process is started by a source node broadcasting a RREQ packet to its neighbours. Any node that receives this RREQ either replies to the RREQ if it has a valid route to the destination (or is itself the destination) or rebroadcasts the RREQ. Along with the rebroadcasts, every node receiving the RREQ starts constructing a reverse path to the source node of the RREQ, granting that when the destination replies, they will have a route to the source node.

As illustrated in Figure 2.6 let us consider the case where node A wants to start a session with station F. As A does not have a valid route entry to F, it broadcasts a RREQ packet. This packet is flooded through the network, reaching F.

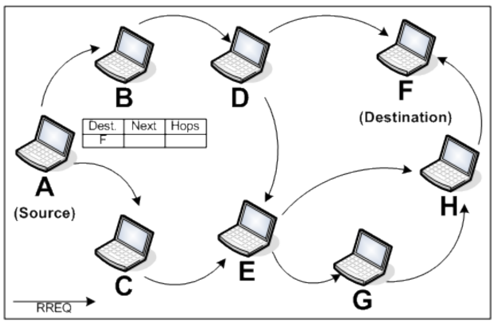

Figure 2.6: AODV example - RREQ broadcast.

A node with a valid route to the destination replies with a RREP packet. Every node by which this RREP packet passes adds a forward route in its routing table, to the destination node. This packet is forwarded back to the source and so, routes from destination to source and source to destination are constructed.

In Figure 2.7, we see that, after receiving the RREQ packet, node F must reply with a RREP, using the shortest-path that was used by A to reach it. The nodes that take part in the route will use table entries that were acquired during the RREQ flooding. In the figure we can only see a table entry consisting of destination, next hop and hop count, but, as referred before, there is more information kept for each route.

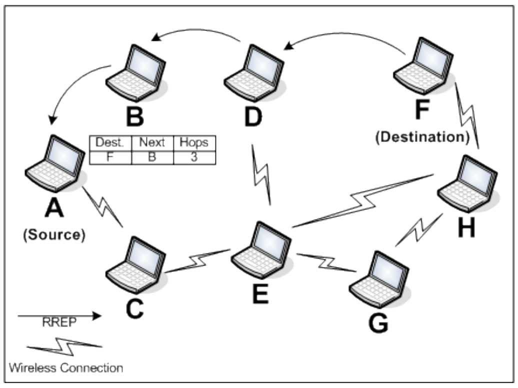

Figure 2.7: AODV example - RREP propagation.

In the event of loss of communication between two nodes that are part of a path (due to mobility, for instance), the node that detects the break sends back a RERR message, signalling every node in the path that the route is now invalid. The source then repeats the route discovery process if it still need to maintain the connection.

In our example, in Figure 2.8, when station F leaves the network (or moves to another location), node D senses this link loss and sends a RERR packet back to station A. A will then discard the route to F and flood another RREQ if he still needs to transmit to F.

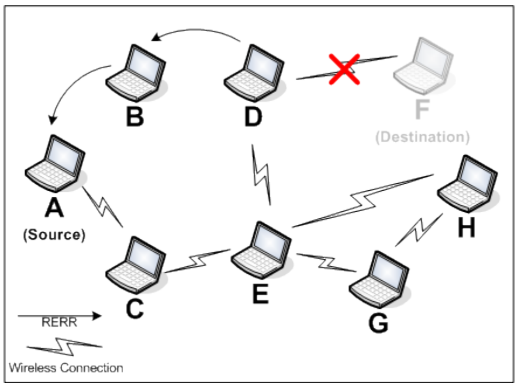

Figure 2.8: AODV example - RERR propagation.

2.3.1.1 DYMO

Dynamic MANET On-Demand (DYMO) is the IETF MANET working group standards-track version of AODV [20], currently in its 17th revision. It is the updated version of AODV. It uses some improvements made to AODV in previous works, such as AODV-PA [21] and AODVjr [22], but retains the core functionalities and mechanisms as AODV. The route setup and maintenance mechanisms are basically the same, being that DYMO, compared to AODV, was stripped of some features that were considered non-essential, for instance, HELLO messages are optional and gratuitous RREP are removed. Also, using the work in AODV-PA, DYMO relies on path accumulation, so that the resulting overhead is reduced: intermediate nodes take advantage of overheard RREQs and RREPs to build paths. In addition to the AODV cleanup, DYMO messages conform to the Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format as described in [23].

2.4 Relaying in Wireless

The previous section described related work concerning multihop routing. This section deals with relaying by providing notions as well as addressing related work. Relaying techniques can be used in wireless networks to enhance some parameters of these networks, such as: expanding the range; expanding coverage; better throughput.

2.4.1 Expanding Network Range

The basic method of relaying can be used to expand the range of a network, through the use of a *wireless repeater*. A repeater can be a normal AP or node configured in such way that it will repeat every signal that it receives, usually using directional antennas (cf. Figure 2.9). A node wishing to transmit to other node that is out of its reach will transmit to the repeater. The repeater will then replicate every frame, reaching for the destination node.

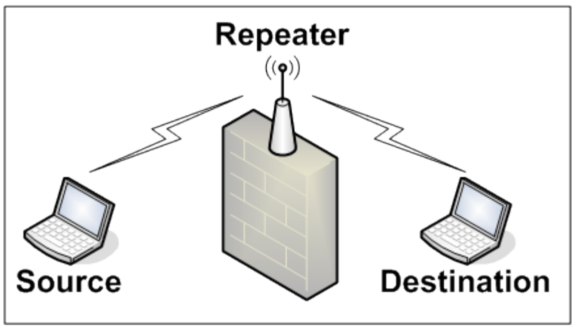

Figure 2.9: Relaying by repetition.

This technique is widely used in managed networks, configuring several APs as repeaters, so that the network can be expanded to a large area, allowing for better special diversity of nodes.

2.4.2 Expanding Coverage

This is different from the previous point, in the way that the range of the network is not increased through the simple repetition of frames, but rather by having nodes expressly forwarding each others' packets. This can be achieved through mechanism such as *store-and-forward* or *decode-and-forward*.

Store-and-forward and decode-and-forward mechanisms work by having a node transmit packets to an intermediate node, instead of directly to the destination node. This intermediate node has the role of receiving every packet from the source, put them in its own network queue, and retransmit them to the destination node when possible.

Store-and-forward techniques are used in ad-hoc networks to allow nodes to forward each others packets and thus increase the network diameter and coverage.

2.4.3 Improving Throughput

Cooperation techniques can prove to achieve better throughput in some multihop scenarios. This works by having an assistant node to forward transmissions from source to destination, very similar to store-and-forward schemes. The difference is that this approach takes advantage of nodes being very close together in dense networks and uses shorter transmissions, between close nodes, instead of transmitting directly to a far away node.

Shorter transmissions mean that the medium will be used for less time, with lower probability of error in the transmission, which, in its turn means higher throughput, lower delay and reduced interference.

In Figure 2.10 we can see an example of *cooperative communication*. A is transmitting a packet to C. t_n are the times it take to transmit the packet. In this scenario, if $t_1 + t_2 \leq t_3$ then this scheme is actually worthy and would perform better than direct transmission.

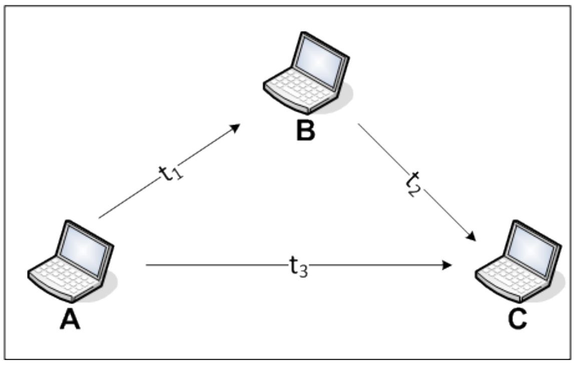

Figure 2.10: Cooperative communication.

This is the principle of CoopMAC [24], a cross-layer scheme that proposes a modification of the MAC layer of 802.11 to allow cooperative communication.

2.5 Summary

This chapter went over global notions and related work analysis, which assisted both the implementation and evaluation performed in the thesis work. The chapter starts by providing an overview on the evolution of wireless architectures, culminating with the most recent user-centric paradigms. Then, multihop routing is described, being the main routing protocol operation summarily explained. Specifically, we provided details of DSR, DSDV, TORA, AODV and DYMO. Table 2.1 summarises the characteristics of these protocols.

Table 2.1: Main characteristics of DSR, OLSR, TORA and AODV.

	DSR	OLSR	TORA	AODV
Propagation of Information	Distance-Vector	Link-State	Link-Reversal	Distance-Vector
When to Compute Routes	On-Demand	Pre-Computed	On-Demand	On-Demand
How to Disseminate Routing Updates	Event-Driven	Periodic	Event-Driven	Event-Driven
Type of Ad-hoc Topology	Flat	Flat	Flat	Flat
How and Where to Compute Routes	Distributed	Decentralized	Distributed	Distributed
Metrics	Source Routing	Source Routing	Hop-by-hop	Hop-by-hop
Resilience	Multiple Path	Single Path	Multiple Path	Single Path

Relaying techniques in wireless was then the topic covered, explaining the different advantages and evolution of relaying.

The description presented in this chapter represents the initial related work analysis which helped to understand the status of the different architectures and protocols, as well as advantages and disadvantages, always having in mind the main goal of this thesis, i.e., to provide an initial comparison of a simple form of wireless relaying to an adequate form of multihop routing.

Chapter 3

Problem Space, Notions and Tools

This chapter relates to the problem space addressed in the thesis, covering both notions and the main tools relied upon, as well as the main thesis building blocks. After this brief introduction, section 3.1 goes over the thesis main building blocks, namely, relaying, multihop routing, as well as radio resource aggregation, explaining main aspects, advantages and disadvantages, as well as implementation status.

Section 3.2 addresses additional tools that were used throughout the thesis work, namely, traffic generators, and a local wireless testbed, highlighting implementation considerations. A summary is then provided in section 3.3.

Being user-centric, UPNs do not have a clear splitting between network and end-user devices, as is common in the regular Internet, according to its end-to-end principle. Instead, UPNs will contain elements with the capability to route traffic, but some that are not routers as well. Furthermore, for specific scenarios attaining a few nodes, routing may be an unnecessary feature. Relaying is an intrinsic MAC feature which up until very recently did not allow more than one hop transmission. However, there are clear cases where relaying (as described in section 2.4) may serve the same purpose and in fact assist in improving overall network conditions.

The work developed throughout the thesis aimed at understanding if relaying could indeed be a feasible solution, and under which conditions can simple packet relaying between the nodes be a better solution. To achieve this under the most realistic conditions possible, a specific testbed was installed and several experiments have been performed both for relaying and for multihop routing in the form of AODV.

Under the described assumptions, some initial questions have been considered and the work to be developed in this thesis aimed at providing answers to:

- What is the "best" scenario possible for relaying (vs. routing), by varying the diameter of the network and the number of available micro-providers? (scalability).
- What is the average and maximum number of hops supported for the case of relaying? (reliability).

- Can relaying be the basis for a service as robust as the ones provided currently based upon ad-hoc routing? (robustness).
- In UPNs, what is the impact of having nodes dynamically changing between ad-hoc and managed networks? (auto-configuration).

In order to attempt to answer the mentioned questions, four main building blocks were considered:

- To implement and evaluate a simple form of relaying in our testbed;
- To rely on a form of multihop routing as benchmark for the validation aspects;
- To consider aggregation of radio resources, as an optimisation for relaying.
- To design a simple testbed that could serve as proof-of-concept for relaying.

These aspects are further detailed next.

3.1 Main Building Blocks

Figure 3.1 illustrates a generic scenario for a simple UPN and which will assist the explanation of the building blocks of this thesis. In such scenario, node D corresponds to an AP owned by a specific user, who is willing to share his/her Internet connection with other users within his/her community. Therefore, node B and node C can automatically connect to node D, given that they are in the range of D and also given that the access to D is open, or there are shared credentials.

However, nodes A and E are not in the direct range of D but they would profit to have access to the Internet connection of node D.

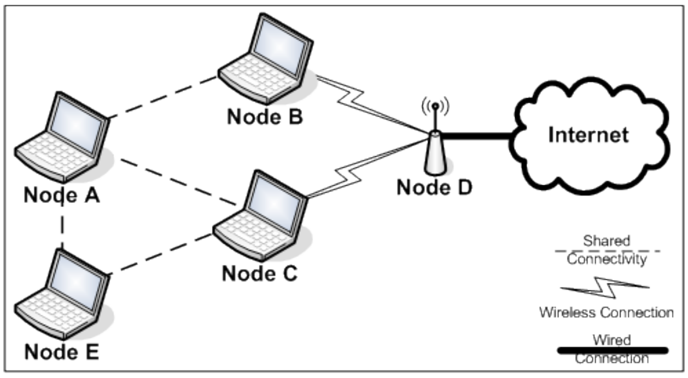

Figure 3.1: Developed testbed.

In order for nodes A and E to be able to access the Internet connection of node D, one possibility is for all the nodes involved to have routing capabilities and therefore, be able to perform multihop routing. However, that is not always possible. In the illustrated scenarios, the nodes represent personal computers, which by default are not routing enabled. One alternative to multihop routing is relaying. With relaying, node E must select a neighbour to be its default neighbour, i.e. the neighbour it will send packets to, in order for them to reach to the Internet. This neighbour will also have a default neighbour and will forward the packets to it. The routing tables have only one entry, the default gateway, and there is only one possible path for the packets to go through. Therefore, the main difference of relaying to routing is that there is no path computation involved. The second aspect is that there is no need to keep a routing table. A third relevant aspect is that there is no need to interconnect nodes in ad-hoc: the network may be managed, or ad-hoc.

The relaying scenario becomes even more interesting if node A and E could simply consider that all successor nodes (B and C) are a single node on the way. In other words: if nodes A and E would be able to simultaneously connect to nodes B and C and, at the same time, take advantage of load-balancing, then there are overall network benefits. This is feasible if a virtual node incorporates nodes B and C as illustrated in Figure 3.2.

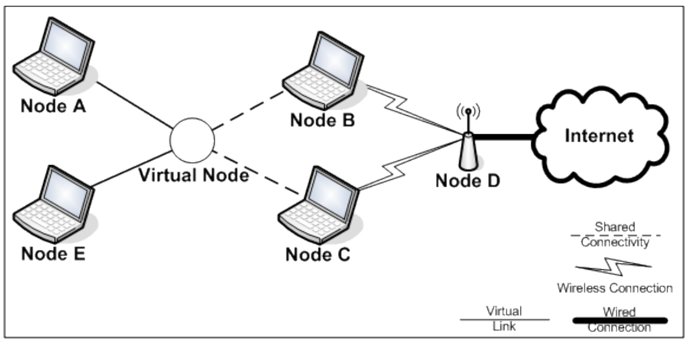

Figure 3.2: Radio resource aggregation example.

Then, with radio aggregation tools as the ones described in section 3.1.3.2, nodes A and E could easily optimise resources on the radio link.

3.1.1 Relaying in Wireless

The ability to do relaying and packet forwarding is readily available in any operating system either by means of bridging on the MAC Layer, or by means of *IP masquerading* (known as *Internet Connection Sharing* in Windows Systems [25]).

To exemplify relaying on Layer 2 (by means of bridging), let us consider Figure 3.3 and nodes A, B, and D. Node B is in the range of D and therefore can take advantage of its Internet connection. Just looking into options of OSI Layer 2, node B has to become a relay, i.e., a software bridge has to be established between his two wireless devices wlan0 and wlan1. This allows both WLANs (the one connected to wlan0 and the wlan1) to be seen as a single segment for the upper Layers and therefore, allows traffic from D to be seen by A.

Two remarks are significant in this case: nodes A, B, and D are not necessarily connected to form an ad-hoc network. Several configurations are possible, being one of them having an infrastructure mode (which is more common for the case of user-centric wireless architectures).

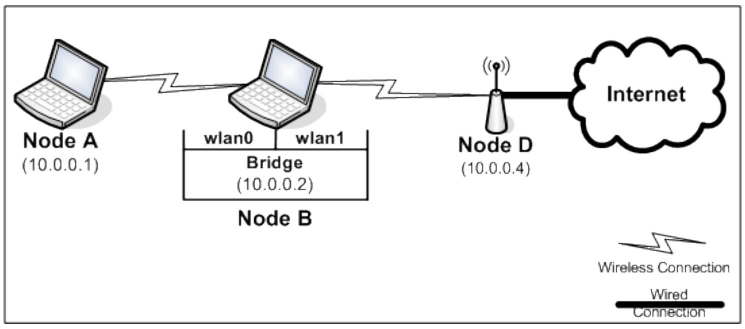

Figure 3.3: Relaying in layer 2.

In addition, it is not necessary for node B to integrate two different wireless cards: it is only necessary to support virtualization. Today, virtualization of the wireless interface is an integrated feature of most operating systems, due to the use of softmac implementations. For instance, in Unix systems, virtualization is today part of mac80211 [26], a softmac implementation which is open and compatible with most wireless drivers as of today. In Windows systems, MAC layer virtualization is integrated starting from Windows 7 [27].

As mentioned, relaying can also be performed by relying on features of the OSI Layer 3, i.e., IP masquerading. IP masquerading (or IP masquerade) is a networking addressing filtering function which looks like a one-to-many *Network Address Translation* (NAT). In its basic form, IP masquerading allows nodes that have no valid (routable) IP address outside a specific LAN (or WLAN) to communicate to the outside. For our specific relaying case, IP masquerading allows a specific wireless interface to act on behalf of N different wireless interfaces. To better exemplify how relaying is performed by means of IP masquerading, let us again consider Figure 3.3 and nodes A, B, and D. B is set to perform IP masquerading between wireless interfaces wlan0 and wlan1. It should be noted that it is not necessary to consider two interfaces in order to get masquerading. The iptables rules also apply to a single interface as well. When A sends traffic to an Internet destination the packets sent are in fact treated by B which is the default gateway to A. When B gets packets from A, it sends them to D as if they were its own traffic. Hence, D is actually not aware that A exists. The term masquerading is derived from the fact that B actually masquerades traffic from A. This is actually different from routing, and in user-provided networking has implications in terms of traceability [1].

In technical terms, IP masquerading requires [28]:

- Kernel support for IP masquerading.
- Filtering rules setup, by means of iptables.
- IP forwarding enabled in the Kernel.

Relaying in *Open Systems Interconnection* (OSI) Layer 2 has the advantage of forwarding the packets in an earlier stage of the TCP/IP stack, meaning that in a network card which has the MAC layer logic implemented directly in its firmware, forwarding is done with no additional processing overhead. On the other hand, if done in OSI Layer 3, packets to be forwarded must traverse the stack until reaching the IP layer, which can bring additional computational overhead. This additional overhead becomes insignificant when using a network card with a softmac layer as, in this case, the processing of the MAC information is already done by software, which makes it insignificant the processing of the IP information.

Relaying at different OSI Layers also has implications on the possible network topologies. Again using the example in Figure 3.3, if one relies on a OSI Layer 2 technique, all the nodes A, B and D must be in the same IP address range, i.e., as only the MAC information of the packet is changed, nodes A and D must still be in the same IP network.

This changes when we use an OSI Layer 3 approach. As the relay node now changes the IP information on the packets as well as the MAC information, sending the packets as if they were their own, it is now possible to have nodes A and D in different IP networks. Of course, this is only possible as long as node B has an interface in each of these networks.

3.1.2 Choosing a Benchmark: Multihop Routing

As explained in chapter 1, one of the main expected goals of the thesis proposed is to analyse the performance that relaying has on a specific testbed, being the benchmark a specific form of multihop routing.

From the analysed multihop approaches (which correspond to the ones that are most popular today), all of them succeed in maintaining loop freedom in wireless networks by employing different techniques. TORA is a reliable approach but requires a significant amount of routing information to be cached. DSR achieves a good trade-off in terms of efficiency vs. resources, but does not react well to highly dynamic environments. OLSR is a pretty solid protocol, but its proactive nature makes it have too much overhead to keep routes that may be unnecessary.

Each of these protocols takes a different approach to maintaining loop freedom. AODV, DSR and TORA focus on minimising protocol overhead and reducing network memory. As a result they adapt well to the changing topology of wireless ad hoc networks. However for wireless networks with little communication AODV has a clear advantage since it generally does not remember links which have not been used recently. This makes it less susceptible to outdated routing information which is common in mobile wireless networks.

Comparative studies, be it simulations or field experiments, show that in face of node mobility, reactive protocols have better performance than their proactive counterparts [29, 4, 30, 31]. In addition, in a UPN, a user (or several) is sharing its connectivity to a broader network or network service (the main use being the Internet). This implies that a user connecting to such network will usually require a highly dynamic way to compute routes to the gateways in said network, and routes to other users may not be significant. As such, to analyze which protocol could perform better in user-provided scenarios (and hence which will become the choice for our benchmark approach), special attention was given to the two main protocols in consideration by the IETF MANET workgroup, namely AODV and DSR.

AODV and DSR have some similarities, which results in both having similar performance results when directly compared. However, depending on traffic load, network diameter and node mobility, there are situations in which one outperforms the other.

In large networks AODV performs better than DSR [29, 32]. This is because of the routing overhead on DSR, due to its source routing nature. For this same reason, AODV also performs better in high traffic loads [32], requiring less bandwidth for route discovery packets.

In addition, both these protocols have been shown to be scalable to large networks, especially when a low number of routes are requested (as every node with a route to a destination can reply to a route request). This makes AODV seem ideal in user-provided scenarios, being expected that it performs very well in the given scenarios.

3.1.2.1 Implementation Status

There are several implementations of both AODV and DYMO publicly available, mainly for Linux systems. Such implementations either fall into user-space or kernel-space, and such categorisation has a significant impact in terms of performance and ease of use. Keeping everything in kernel-space can cause compatibility issues, as well as causing potential system instability due to buggy implementations. User-space implementations can be easier to implement and to use, but normally achieve a lower performance. Tests have shown that a user-space daemon can take up to 1 order of magnitude more in packet forwarding than its kernel-space equivalent [33].

Some of the most popular implementations in use, and which have been thoroughly analysed in [33], are: Kernel-AODV [34], AODV-UU [35], AODV-UIUC [36]. These implementations are already a few years old and since their release, AODV has evolved into DYMO. Moreover, DYMO implementations are still scarce (due to its embryonic stage of standardisation) and the only DYMO worthy implementation we could find was NIST-DYMO [37]. Other implementations of DYMO included DYMOUM [38] (inspired in AODC-UU) and DYMO-AU [39], however, these implementations did not have all the features in NIST-DYMO, and were not kernel-level.

AODV-UU was developed at the Uppsala University (hence the UU suffix). It uses a userspace daemon, where most of the protocol's logic resides. It has, therefore, worse performance than Kernel-AODV but has the advantage of having its code ported to the ns-2 simulator, allowing the simulation of the exact same code in use in real-world implementations. This implementation has also been included a number of improvements to the AODV specification that allows it to achieve slightly better performance, namely in keeping lower bandwidth overhead through the improvement of the HELLO messages.

AOVD-UIUC is something of a hybrid implementation. It uses the Ad-hoc Support Library (ASL) and keeps the forwarding logic separated from the routing functions, the forwarding functions reside in kernel-space, while the later is kept in a user-space daemon. This causes it to handle received packets faster, as only a few packets are subject to the overhead due to the use of the user-space.

Kernel-AODV is, as its name indicates, implemented directly in the Linux Kernel, as a module. This results in Kernel-AODV having the best performance, in terms of packet handling, of all three implementations because, having all the protocol logic in kernel level, the packets do not need to reach to the user-space, thus refraining from this computational overhead. Kernel-AODV was implemented by the National Institute of Standards, NIST. It has the big disadvantage of being for the 2.4.X Linux kernel only.

NIST-DYMO is the evolution of Kernel-AODV. It comes from updating the Kernel-AODV source code to the 2.6.X Linux kernel and polishing the protocol according to the DYMO draft. Similarly to Kernel-AODV, NIST-DYMO uses kernel code, through netfilter hooks, and seemed to integrate some interesting features, such as: gateway routing; sub-netting; multiple interface support.

In order to get a suitable implementation to serve as a benchmark in the performance tests, such implementation must have a set of necessary features, such as gateway routing and a good implementation of the route discovery and maintenance mechanisms. In addition, it must comply with some non-functional requisites: fast packet forwarding implementation (performance); open-source (availability); stable version (reliability).

As the core functionalities of DYMO are identical to the ones previously described to AODV (cf. section 2.3), NIST-DYMO should have identical operational behaviour as the one described for AODV in section 2.3. However, there are some particularities that deserve a more detailed explanation.

Figure 3.4 provides an example for the operation of NIST-DYMO, highlighting its main features.

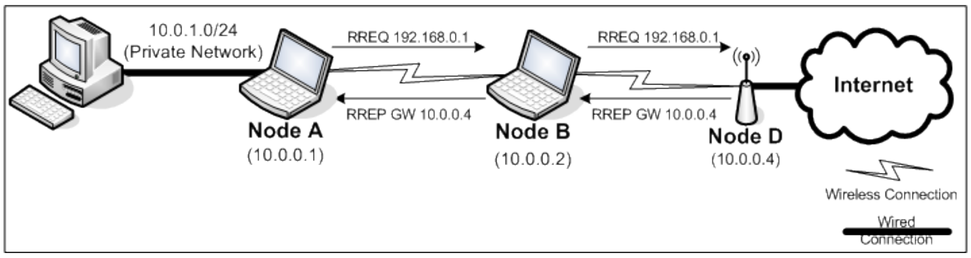

Figure 3.4: Example NIST-DYMO operation.

As one can see from the figure, the route acquisition mechanisms are the same as AODV. What is different in this situation is that the route request is for an address outside of the range of the local network. In this case, it should be possible to configure a node as a gateway (node D in this example). These nodes will reply with their own address for this type of request.

One other important feature NIST-DYMO must perform correctly is the definition of private networks that should be ignored by DYMO. This will not only keep every other node in the network from being aware of these networks, but more importantly, will refrain NIST-DYMO from handling addresses belonging to these networks. It is important to have DYMO ignoring these networks, as they will be used to access the nodes through a wired interface, for configuration.

3.1.3 MAC Virtualization and Radio Resource Aggregation

As described in section 3.1.1, relaying can profit from the notion of radio resource aggregation. Specifically, in UPNs, if the concentration of users is high (cf. Figure 3.5), i.e., if the AP coverage is *dense*, then instead of attempting to fight back the media interference, a better bandwidth and resource utilisation could be achieved if nodes simultaneously connect to more than one of the available APs, as illustrated in Figure 3.6.

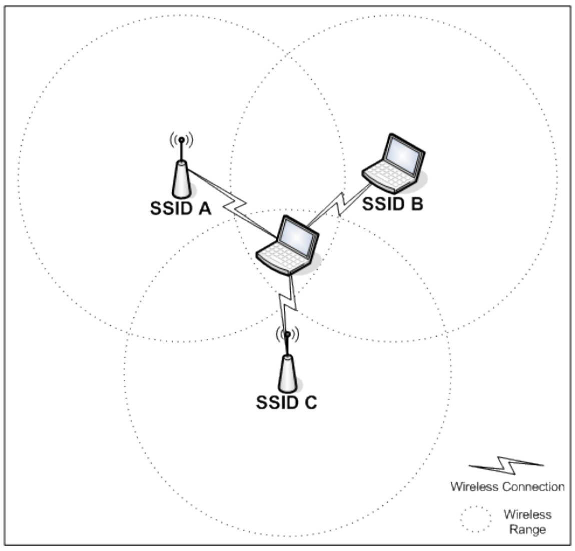

Figure 3.5: Node in the range of three overlapping networks.

One possible way to perform this is to allow the node to consider a set of APs as a virtual AP and hence take advantage of wireless resource aggregation. As also debated in section 3.1.1, simultaneous connections to N APs requires, from the endterminal, support for two different wireless interfaces. However, in some equipment it may not be possible to install a new card, and even if it is possible, it represents additional costs and battery drain.

3.1.3.1 MAC Virtualization Tools

An alternative approach in connecting to several APs at the same time is to use virtual wireless interfaces. The madwifi project (*Multiband Atheros Driver for Wireless Fidelity*) [40] was one of the first public-domain attempts to perform virtualization of the MAC layer. The project developed the first open-source kernel device driver for Atheros-based chipsets.

The driver is an open-source driver, however, it depends on the *Hardware Abstraction Layer* (HAL), which is proprietary software. MAC virtualization became an integrated feature of the ath5k version of the Linux kernels, then superseding madwifi, which became legacy. However, due to its stability, it is currently still in use in several tools.

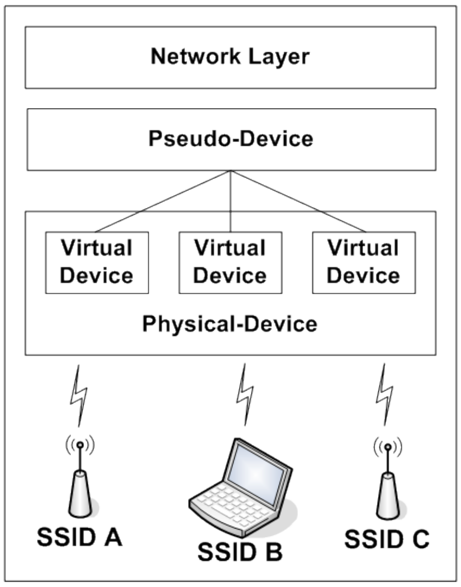

Figure 3.6: AP aggregation scenario.

Ath5k is released under a *Free Open Source Software* (FOSS) license for Atheros-based chipsets and part of the wireless-2.6 tools. Main differences relates to the fact that while madwifi had to interoperate with a closed HAL, ath5k calls hardware functions directly. Another main difference is that ath5k relies on a fully softmac layer, mac80211.

Mac80211 actually is capable of supporting most of the common drivers available today, not only Atheros.

Furthermore, in madwifi the virtualization functionality was implemented in the driver itself and therefore, could only be applied to the Atheros chipsets. In contrast, for ath5k, the virtualization is not integrated in ath driver; instead, it is part of mac80211. This means that for mac80211, virtualization of any wireless device is possible.

This new module makes way for aggregation in conjunction with virtualization, simplifying things and making madwifi tools unnecessary. Also, mac80211 is a completely open-source softmac layer, in use by many different drivers, which translates into more versatility and easier deployment.

Virtualization per se only allows simultaneous connectivity to different WLANs. This suits the purpose of, for instance, multi-homing. However, it is not transparent to higher layers of the client network stack. This means that, when connected to two APs, these will be seen like two different network connections by the host. This is a disadvantage in relation to using aggregated AP resources as it does not allow for load balancing.

3.1.3.2 Radio Resource Aggregation Tools

Radio resource aggregation is a recent trend in wireless and has its origin tracing back to 3G, cell aggregation. In regards to WiFi, today there are three main successful tools.

VirtualWiFi [41] allows a wireless card to switch between different networks by relying on a network hopping scheme that is transparent to both the user and applications. Therefore, the user seems to be simultaneously connected to different wireless networks. VirtualWiFi holds two main components: a NDIS intermediate driver, and user-level tools (for Windows XP) which perform the switching logic. Albeit interesting, VirtualWiFi is not optimised in terms of resource aggregation. Furthermore, there was, at the time of this report, no public-domain implementation of VirtualWiFi. By the time of this report, only two implementations of radio resource aggregation were available: Juggler [42, 43] and FatVAP [44, 45].

Juggler is a refinement of VirtualWiFi which also incorporates a fast switching scheme between different APs. The fast switching scheme works as follows. Juggler assigns an adjustable fraction of the duty cycle to specific WLANs. Assuming two APs, AP1 and AP2, let us consider that Juggler assigns 60% of the time for AP1 and 40% for AP2. Upon completion of the first part of the duty cycle, Juggler sends to the virtual device connected to AP1 a null IEEE 802.11 frame with the PSM bit set, i.e., it tells AP1 that the terminal went to powersave mode. It then toggles to AP2 by performing regular MAC functions, i.e., scanning. If AP2 is on a different channel than AP1, it adjusts the MAC parameters such as SSID, MAC address.

The distribution of time between different APs is static in Juggler. Again considering AP1 and AP2 and assuming a duty cycle of 100ms, AP1 would get a weight of 60 and AP2 a weight of 40.

Code is publicly available for Juggler, for the Realtek 8185 chipset. Such chipset relied on a specific open-source driver implementation, the rtl-wifi. Furthermore, rtl-wifi uses a specific softmac layer implementation net80211 (derived from netbsd). Albeit potentially portable to other platforms, the code of Juggler is however, old.

FatVAP (Fat Virtual AP) is an 802.11 driver built on the madwifi driver developed by the MIT. FatVAP relies on a network hopping scheme giving, to both end-user and applications, the notion of simultaneous attachment to different WLANs. The interesting reasoning from FatVAP relies on the components of such hopping scheme, namely: i) selection of an optimal set of APs to connect to; ii) fast switching between APs on the chosen set, thus preventing packet loss; iii) load-balancing scheme that maps traffic to APs according to their available bandwidth. FatVAP works as follows. Load-balancing between APs is performed according to the available bandwidth of each AP. It should be noticed that FatVAP does not require any probing; Available bandwidth of each AP is estimated based upon a simple function which encompasses link losses, driver's rate selection, and competition from other stations on the same AP.

Switching between APs is performed based on a notion of station duty-cycle, for which the authors provide a fixed value of 100ms, justified as the minimum necessary to ensure stability for one TCP connection. During the duty cycle, toggling between APs is performed in a way (based on available bandwidth) to ensure that a station gets in average the maximum wireless rate allowed any time. Therefore, if a station is already achieving such rate on a specific AP, FatVAP will not perform toggling.

Albeit recent and showing interesting features due to its highly dynamic behaviour, FatVAP is not a public domain tool[1].

Table 3.1 summarises the main features of FatVap vs. Juggler.

Table 3.1: Feature comparison between FatVAP and Juggler.

Feature	Chipset Support	Same Channel Support	AP Connection Duration	Fast Switching Support	Packet Loss Avoidance	Load-Balancing Support	LWAP Support	AP Selection
FatVAP	Atheros + madwifi	Yes	Based on varying time slicing, depends on # APs, w and e	Yes, dynamic function	Yes, based on internal queues set per virtual interface	Yes, flow based	Yes	Based on maximum allowed throughput
Juggler	Realtek (all software, relies on open-source rtl-wifi)	Yes	Based on fixed time slicing, 10% dedicated to secondary connections		Yes, based on 10% allocated to secondary connections (previously established)	Yes, flow based	No	Static

3.2 Additional Tools

In addition to the main building blocks that correspond to main goals to attain in the work that lead to this thesis, we relied on other specific tools to assist our work. The main relevant were: traffic generators, as well as a local wireless testbed.

3.2.1 Traffic Generators

In order to be able to estimate the network conditions and parameters in the experiments carried, we must put the network under different conditions of traffic and load. Two options could be considered: i) to rely on real data traffic, e.g. VoIP traffic; to emulate such traffic by relying on realistic parameters.

In order to allow achieving results with a high level of confidence, the option was to consider a traffic generator, as this would give us more control over the traffic parameters.

Some traffic generation tools available were therefore analysed, having in mind to understand the ones that could best suit the purpose. Specifically, we chose to model VoIP traffic as close to reality as possible. The choice for VoIP traffic relates to:

- VoIP is one of the most common real-time applications used in user-centric communities.
- VoIP mimics bursty traffic, being more interesting to the network.
- VoIP stands for a realistic example of UDP traffic.

Due to time constraints it was not possible to consider the impact of other types of traffic, e.g, TCP. This is therefore a task for future work as described in chapter 6.2.

The following tools were considered: iperf [46], netperf [47] and MGEN [48].

[1] In the FatVAP paper, the authors claim software is available. However, we have contacted MIT and the answer was that the code was property of a Korean company with whom the software was developed.

Iperf and netperf are quite similar. Both are more focused on measuring the network available bandwidth, not leaving much flexibility. They both are capable of measuring bandwidth and datagram loss and iperf also presents the results of jitter and RTT. It is possible to specify a traffic type, TCP or UDP, although it is not possible to specify the traffic pattern.

MGEN is a more versatile tool that allows shaping the test traffic in a more detailed way. It allows us to specify the type of traffic, as well the pattern of the traffic. It works with a server-side application and a client-side one. The logs registered at the server-side, keep all the information about all the packets received, such as the time at which the packet was sent and received, size, sequence number and flow identifier. This way, one can easily develop a simple program or script to analyze these logs and extract all the parameters, like end-to-end delay, jitter, throughput, packet loss.

We chose to use MGEN in the benchmarks, because of its versatility.

3.2.2 Testbed

The experiments to be carried out relied on a local wireless testbed set up at INESC Porto, composed of six nodes, namely, four laptops and two embedded Access Points. The equipment was diverse to simulate the conditions of a UPN scenario. Table 3.2 gives a detailed description of the equipment.

Table 3.2: Testbed equipment.

Node	OS	Kernel	Architecture	Processor	RAM	WiFi Chipset
A	OpenWRT 8.09	2.6.26.5	mips	Atheros AR2315	16MB flash	
B	OpenWRT 8.09	2.6.26.5	mips	Atheros AR2315	16MB flash	
C	Open SUSE 11.0	2.6.27.21	x86	Intel Core2 Duo T7350	3GB	Intel 5100
D	Open SUSE 11.0	2.6.27.21	x86	Intel Core2 Duo T7500	2GB	Intel 4965
E	Ubuntu 8.10	2.6.27.14	x86_64	Intel Core2 Duo T7350	3GB	Intel 5100
F	Ubuntu 9.04	2.6.28.13	x86_64	Intel Core2 Duo T7700	4GB	Intel 4965

All of the laptops have been configured with the latest version of the Linux wireless system (compat-wireless 2.6.30) [49], so they required a kernel version of at least 2.6.27. This was to ensure the correct behaviour and best performance possible of the wireless cards in ad-hoc networks, as the previous drivers were having some faulty behaviour in ad-hoc environments.

The two embedded devices were kept with the default kernel packed with OpenWRT 8.09 [50], because it already comes with the necessary drivers for its hardware (madwifi), so there was no need to install compat-wireless and thus no need to install a newer kernel.

Some changes had to be made to the default configuration in order to remove the bridge that is created by default in OpenWRT, connecting the Ethernet and wireless interfaces. This had to be done because such a bridge was causing malfunctioning of the wireless driver when the wireless interface was put on ad-hoc mode. Specifically and due to the relaying functionality implemented (changes to the iptables, cf. section 4.1), broadcasts on the wireless link were being interpreted as traffic to unknown destinations by the Ethernet interface.

The ad-hoc network used to connect all devices was configured to be open (unencrypted) to avoid unnecessary encryption overhead and possible problems. It was also important to choose a channel that was not used by any other of the surrounding wireless networks to minimise exterior interference and avoid collisions. Nodes relied on 802.11g, channel 8, as the channels 1, 6 and 11 were highly populated. In terms of bit rate, nodes were configured to use the maximum allowed speed in 802.11g, i.e., 54Mbps.

In terms of spatial correlation, all the nodes in the network have been kept close and hence could overhear every other node's wireless signal. However, this was not the desirable case in the experimentation scenarios. Therefore, connections between nodes are simulated through the use of iptable filters, dropping packets received by the nodes we don't want to be connected to. This proximity between the nodes has the disadvantage of causing more MAC contention.

One final aspect of the testbed concerns the *Network Time Protocol* (NTP) server. The server used was a part of the lab's network, as it had already an existing server. It was then necessary to prepare the nodes involved in the transmission of packets to connect and synchronise with this time server. This was done using the ntpdate and sntp tools in Ubuntu and OpenSUSE, respectively as these are the default tools readily available in the repositories of each distribution. The steps taken in each experiment to keep the clock synchronised are described with more detail in section 5.1.3.

3.3 Chapter Summary

This chapter was dedicated to the explanation of the thesis main building blocks, describing notions, tools, as well as architectural options. The work presented here allowed us to adequately develop a testbed for relaying which stands for a proof-of-concept that is expected not only to be re-used in the future, but also to be extended by integrating additional network conditions. This aspect is further discussed in Chapter 6.

Chapter 4

Contributions

The previous chapter described the thesis main building blocks, explaining the choices that were made in terms of tools to rely upon. This section is exclusively dedicated to the contributions of the thesis both in terms of concepts, implementation, and analysis. The section goes over specific changes to software, parameter tuning, installation options, and configurations.

4.1 Relaying Contribution

In order to evaluate relaying, the option followed was to consider OSI Layer 3 relaying, given that it was felt that it would be better to compare relaying vs. routing at the same layer.

Relaying was implemented recurring to iptables to make each node use IP masquerade in its wireless interface. It is worthy to note that there was only one existing interface in each node. In this case, nodes simply relay packet through the same interface they got them from. Masquerade works by having each node change the source IP of each packet it forwards, to its own IP address, like NAT. When a packet comes in the reverse path, nodes do a similar procedure, changing the destination address instead. We can see this behaviour in the example of Figure 4.1, where node 10.0.0.1 is sending a packet to a destination outside the local network, and then receives a reply.

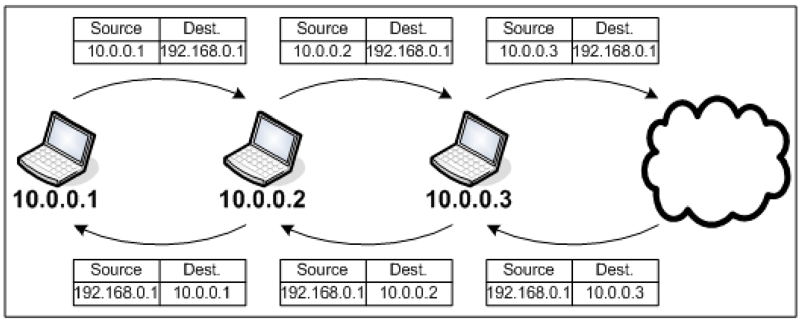

Figure 4.1: Relaying implementation.

By relying on IP masquerading and as explained in section 3.1.2, each node “hides” its predecessors from its neighbours. Given that no path computation mechanism is involved, relaying requires a specific selection of one or more successors. For the sake of simplicity, we configured the preferential neighbour by default. In a real-world implementation a specific mechanism related to trust management and cooperation incentives would have to be the basis for such a selection.

The configuration of each node has been performed by running scripts which are provided in Annex A.

4.2 DYMO Contribution

As DYMO was to be used as the benchmark in the performance evaluations, it was important to have a fast and reliable implementation covering all the required features. As mentioned in section 3.1.2, NIST-DYMO is currently the only option for such an implementation. The implementation seemed the perfect candidate for the job, as it is announced as having all the necessary features and sufficient code maturity.

Although the NIST-DYMO code is indeed mature, already integrating several enhancements and bug fixes, it is also an old implementation, meaning that it was coded for Linux kernel 2.6.8. Our testbed required at least version 2.6.27 due to, among other issues, virtualization support. So the first task performed was to port NIST-DYMO to kernel 2.6.27, replacing the deprecated function calls with their recent equivalents. As the 2.6 kernel evolves very fast, there were significant changes, e.g., in the way memory is managed, and even in regards to filters and hooks related with the forwarding aspects. Furthermore, there is no support whatsoever. We found the code not to be well document and no forum is active.

With NIST-DYMO up and running, severe implementation issues were detected, including features that were wrongly implemented. Some of these issues might have been caused by the fact that we were using a newer version of the kernel, which may do things a little different than the version the authors were using.

The first issue noticed related to the capability of nodes to acquire a route to a gateway. In DYMO, such feature is performed by its gateway discovery mechanism. Specifically, for dymo-default, relies on the regular route discovery features (broadcast of RREQ and unicast RREPs) to create a single default route, i.e., to discover a route to one gateway. NIST-DYMO uses its own routing table, however, it still needs the kernel to have valid routes. Remember NIST-DYMO uses netfilter hooks to process the incoming and outgoing packets that traverse the TCP/IP stack, but still, when a packet is generated to a destination that is not in the kernel’s routing table, it is discarded before even reaching those hooks. This is because a check for a valid kernel route is done before passing the packet to netfilter. To overcome this problem without changing any kernel code, the solution was to go around the problem by tricking the kernel into thinking it does have a valid route. We added a dummy gateway route, with destination 127.0.0.1, with a high metric value (1000) in each node. This route is used to force the packets to traverse the stack, reaching the netfilter hooks. When NIST-DYMO acquires a new gateway route, it comes with a metric value of 0, meaning that it will be the one used by the kernel and the dummy one will be ignored.

Once nodes started acquiring gateway routes, we could see that there were a lot of race conditions existent in the code, in the handling of gateway routes, that had to be taken care of. On top of that, the route maintenance and route discovery mechanisms were not fully implemented and not according to the IETF draft. Specifically, nodes that were forwarding packets for destinations they didn't know were not generating RREQs, even if they didn't have a valid route to the destination. This problem was fixed by changing the hook function that handles the received packets.

Additionally, the mechanism of route timeout and route delete timeout was implemented, but was not being used in the code. This resulted in routes being deleted from the kernel as soon as the valid route timeout expired, but these same routes were never deleted from DYMO's own routing table. To fix this problem it was necessary to revise all the logic involved in the routes timeout and deletion.

We improved these mechanisms by eliminating the race conditions and setting the route maintenance operations accordingly to the draft.

The final challenge in getting NIST-DYMO to work correctly was to get it to work with the mips and x86 architectures interconnected. While x86 machines are big endian architectures, mips falls in the little endian category. This should not raise any problem given that current kernels allow automatic cross-compiling. However, NIST-DYMO relies on bitfield type structures. ISO C does not specify how bitfields are arranged in memory, causing it to be compiler and architecture dependent [51]. This is even more problematic when we have a bitfield type variable that crosses over multiple bytes, as it will certainly be stored differently in different architectures. As it was, the message headers were being interpreted differently on different architectures and the protocol wouldn't work.

In order to completely solve this problem in a quick way (due to time constrains), our proposal was to remove all bitfields from the code, which means a rewrite of a big portion of it. Our option was to just minimise the usage of bitfields, removing only the ones that spread across multiple bytes. Implications of this relate to the fact that the implementation provided by us for little-endian systems is not compliant with the current state of work of DYMO and therefore, requires a future revision. However, there are no performance issues whatsoever.

NIST-DYMO configures DYMO's parameters statically in the code. The values used were the ones that came preconfigured with NIST-DYMO:

- HELLO interval: 2s.
- Route used timeout: 5s.
- Route delete timeout: 25s.

Summing up, we turned NIST-DYMO into a more robust, cross-platform and updated version that can easily be deployed and used in real-world scenarios. We tested this implementation in the testbed, with 6 nodes of different architectures and the results were good. The code is available at [52].

4.3 Radio Resource Aggregation Contribution

For the third building block, radio resource aggregation, the first option followed was to consider the integration of the only open-source tool available, Juggler. Juggler was, however, developed for kernel version 2.6.14 and only for the Realtek chipset as described in section 3.1.3. A first contribution to this module was porting Juggler to kernel 2.6.27. After this task, we realised that there were still several problems:

- Juggler was for a specific wireless chipset (Realtek 8185). We only had one laptop available with that chipset.
- Juggler relied on an old driver and, most relevant, an outdated softmac layer. This could result in performance issues and problems in porting the code.
- Juggler required the application of a patch and subsequent kernel recompilation. This made it harder to port and upgrade, as the kernel is changing fast.
- Last but not the least, several performance issues were detected, perhaps due to kernel changes in the more recent version and the usage of low-performance hardware.

Due to the drawbacks described, our decision was to put aside Juggler and consider building from scratch a specific aggregation tool. The new Linux wireless subsystem [49] already comes with a softmac layer (mac80211), which is fully open-source and used by many drivers, including ath5k. This MAC layer implementation can be used as the core to an aggregation feature in a more straightforward way than the Juggler implementation, as described in section 3.1.3.1.

Having in mind an aggregation system that could support any wireless driver, the core of an aggregation scheme based on mac80211 has been built and which is illustrated in Figure 4.2.

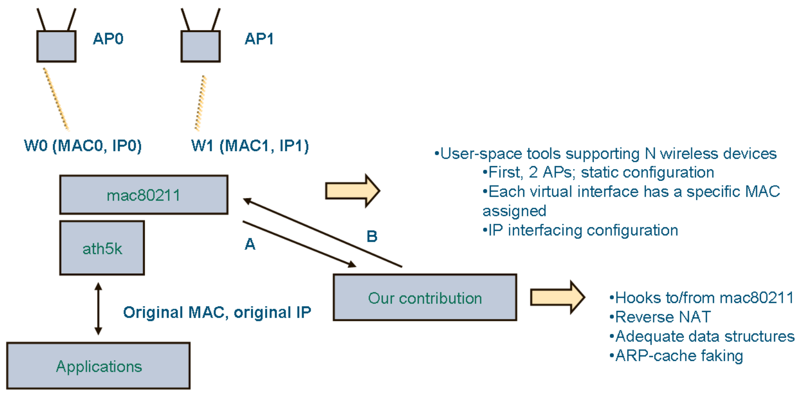
AP0
AP1
W0 (MAC0, IP0)
W1 (MAC1, IP1)
mac80211
ath5k
A
B
Original MAC, original IP
Applications
Our contribution
•User-space tools supporting N wireless devices
•First, 2 APs; static configuration
•Each virtual interface has a specific MAC assigned
•IP interfacing configuration
•Hooks to/from mac80211
•Reverse NAT
•Adequate data structures
•ARP-cache faking

Figure 4.2: Our contribution to the aggregation module.

Considering Figure 4.2, we have implemented most of the functionalities represented. There is a framework for the necessary virtual interfaces, that allows assigning a specific AP to a virtual interface, as well as the IP configuration of the virtual interfaces. The data structures needed to keep track of AP associations are also implemented, along with the necessary mac80211 hook for the inbound and outbound reverse-NAT mechanisms. The *Address Resolution Protocol* (ARP) cache mechanisms are not present, as it wouldn't even be possible to test them.

Due to time constraints, it was not possible to implement everything. So, the tool is not functional and therefore it will not be considered as part of the evaluation described in chapter 5. Reasons for not having it fully functional relates to the driver's toggling mechanism. Specifically, in ath5k and assuming virtualization of the same wireless interface, mimicking a simultaneous connection between N APs means that the driver must be able to toggle in real-time the SSID present in the physical device, among other parameters (e.g. channel). It should be highlighted that this is currently a documented bug specific to the ath5k that is not related to our specific contribution. However, due to time constrains, it was not possible to assist in solving it.

Having the possibility to apply aggregation of resources to relaying would be a very powerful tool in user-provided scenarios, especially when trying to use some relaying scheme that considers several nodes willing to simultaneously relay data for N sources. However, not having it present does not undermine in any way the comparison to be provided between relaying and DYMO. Nonetheless, aggregation is left as future work implementation and testing.

4.4 Additional Contributions

In addition to the contributions already described, it is worthy to note that, while we were using MGEN, a bug in the way it handles packet sequence numbers was detected that could potentially increase the number of packets identified as lost. It happened that the packet sequencer was sometimes being initialised at a random value, instead of zero. That being, the first packet of a flow could be marked with a high sequence number, instead of zero. As MGEN is open-source, this problem was solved by fixing the source code, and a patch will be sent to the MGEN development team, in addition to being provided at [52].

An additional contribution that resulted from this thesis is the script that models realistic VoIP traffic according to codec G.711. Settings are described in section 5.1.2, and the script is provided in Annex B.

4.5 Chapter Summary

This chapter describes the implementation contributions performed to each of the building blocks described in chapter 3 and which are here summarised according to each building block of the thesis.

In regards to multihop routing:

- Ported NIST-DYMO to the most recent available kernel at the time of the testbed setup (version 2.6.27).
- Improvement of NIST-DYMO in terms of robustness of implementation (solved main bugs).
- Implementation of capability for a node to act as a gateway.
- Implementation of capability to allow NIST-DYMO support both in x86 and mips architectures.
- Improvement of NIST-DYMO in terms of route discovery and maintenance.

In regards to relaying:

- Development of scripts that allow any node to perform relaying, based on mac80211 and on iptables rules.

In regards to aggregation:

- Development of a mechanism capable of performing network hopping between available WLANs, thus mimicking simultaneous connections to more than one WLAN.

In regards to the proof-of-concept:

- Development of MGEN scripts (python) to model concurrent sessions of VoIP traffic with different rates, according to codec G.711.
- Testbed implementation containing 6 nodes, including embedded and x86 architectures.

It should be noticed that this represents only the partial outcome of the thesis, given that this simply relates to the implementation aspects. The global overview on contributions is provided in chapter 6.

Chapter 5

Performance Evaluation

This chapter is dedicated to the performance evaluation of the two main building blocks of this work, namely, relaying vs. multihop routing in the form of DYMO, attempting to answer the questions that led to this work and that can be aggregated into 3 main aspects:

1. Scalability issues. Is there a “best” scenario possible for relaying (vs. routing), or a maximum number of hops where relaying performance degrades?
2. How reliable is relaying when compared to routing? Can relaying be the basis for a service as robust as the ones provided currently based upon ad-hoc routing?
3. Can MAC contention affect the performance of relaying?

Answers to these questions were provided by relying on several experiment sets, where the number of hops, sources, and also of concurrent flows was varied. Such analysis was performed for a variable modelling of VoIP traffic. Then, jitter, packet loss, as well as end-to-end delay was measured.

The chapter starts by detailing goals and the methodology followed. A generic description of the evaluation parameters and scenarios is then provided, followed by a description of the topologies implemented and of traffic settings. Finally, section 5.2 explain in a detailed manner the results obtained during the experiments and which refer to packet loss, packet jitter, as well as end-to-end delay.

5.1 Evaluation Objectives, Methodology, and Parameters

The experiments presented in this section have as main goal to analyse relaying vs. DYMO (as a representative approach standing for on-demand multihop routing) in terms of end-to-end delay, packet jitter, as well as packet loss. For a specific IP datagram x, the end-to-end delay $d(x)$ is defined as the time it took for the packet to travel from source to destination, i.e. the interval between the time the packet was sent and the time it was received, as in equation (5.1).

$$d(x) = t_r(x) - t_t(x) \tag{5.1}$$

Where $t_r(x)$ and $t_t(x)$ represent the time at which packet x was received and transmitted, respectively. For this measure to be accurate, the source and destination of the packet must have synchronised clocks. This is a problem we try to minimise through the use of an NTP server, however, this doesn't completely eliminates it, as NTP has an error margin as well.

The inter-packet delay is defined as the variation between the delays of two consecutive packets. For two consecutive packets, x and y, x being the first packet received, the inter-packet delay is defined in equation 5.2:

$$D(x,y) = d(y) - d(x) \tag{5.2}$$

The term jitter is used here to denote packet jitter, i.e., the inter-packet delay variation. Packet jitter is therefore computed relying on the inter-packet delay between packet x and its predecessor $x-1$. Hence, packet jitter of packet x, $j(x)$, is computed as described in equation (5.3):

$$j(x) = |D(x,x-1)| \tag{5.3}$$

Then, the total jitter, J, of a specific flow with N packets is defined as the average of every packet jitter, according to equation (5.4):

$$J = \frac{\sum_{x=2}^{N} j(x)}{N-1} \tag{5.4}$$

Packet loss, P, is here defined as the ratio between the number of lost packets and the number of packets that were sent, not counting the packets received out-of-order. This computation is performed based on the sequence number of the packets received. For every flow, the expected packet sequence number is kept, and if the received sequence number is higher than the expected, then the total number of lost packets is incremented. Packet loss is expressed in percentage, according to equation (5.5):

$$P = \frac{L}{N} \times 100 \tag{5.5}$$

Where N is the number of total packets that were sent by the source node and L is the number of packets that didn't make it to the destination node.

5.1.1 Main Topologies

The experiments run considered three different topologies as basis for the different scenarios developed. The first topology considered (Topology I) is illustrated in Figure 5.1. Topology I corresponds to a worst-case example, as it includes the longest possible path in our testbed. The topology contemplates 6 hops, which is considered to be a bad scenario for VoIP in multihop [53].

For the mentioned topology, nodes F to A are connected by means of an ad-hoc network but only hearing its direct neighbours. The purpose of this configuration related to the need to place the 6 devices in a way that they would not overhear each other. Due to space constraints, one way to emulate such behaviour was the one which we opted to implement. This was done by having a rule in iptables which dropped all packets coming from specific MAC addresses, i.e., the ones that weren't neighbours.

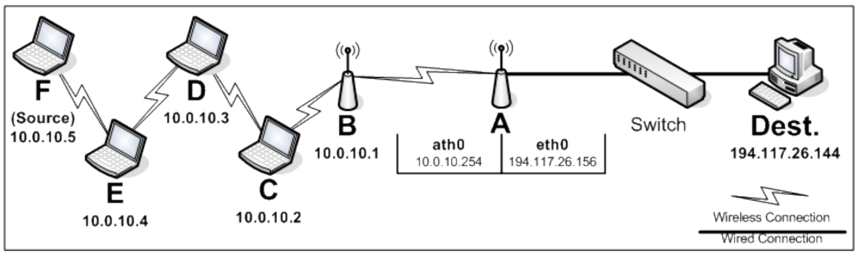

Figure 5.1: Topology I.

The second topology considered (Topology II) is represented in Figure 5.2. Topology II is in fact similar to I, being the only difference the number of hops present, which was now reduced to 4. Such reduction will assist in trying to understand the impact of the path size in the performance evaluation. Due to time constraints it was not possible to test every possible network diameter we could get on the testbed, so we chose the biggest we could get, in the previous scenario, and a medium sized one, on which both DYMO and relaying were expected to have good performance.

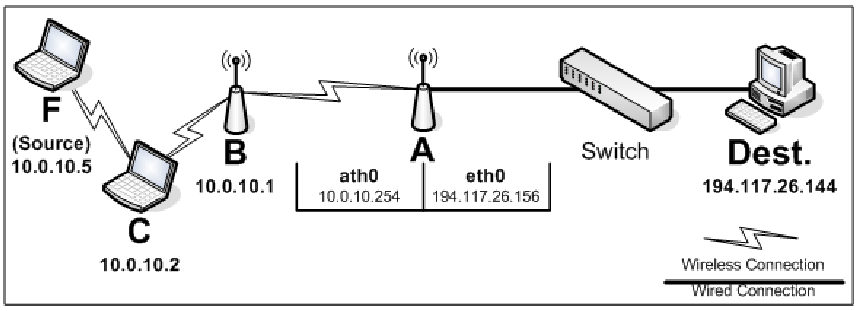

Figure 5.2: Topology II.

Topology III is illustrated in Figure 5.3. It consists of paths that are four hops long, but now there are stations contending for the same media. Specifically, nodes E and F correspond to source nodes. Our expectations were to increase media competition expecting to see how it affects the overall performance of the network.

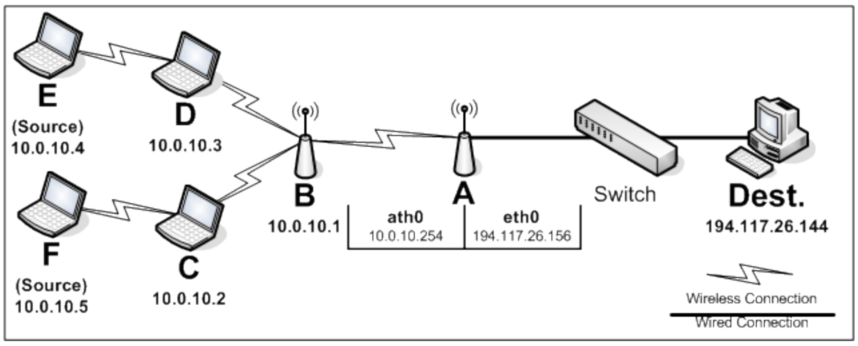

Figure 5.3: Topology III.

5.1.2 Traffic Settings

The traffic used in the simulations was generated by relying on MGEN. The purpose was to mimic concurrent VoIP sessions as close to reality as possible. Therefore, an MGEN script was set to model an approximation of VoIP traffic, according to G.711 [54]. G.711 codec defines the following:

- Unidirectional support for a minimum of 64 kbps.
- Average of 50 packets per second, being the payload size averaged on 160 bytes, thus resulting in a total of 226 bytes per packet.
- An average rate of 11KBps, including TCP and IP headers (approximately 90kbps).

With the aid of MGEN, a script was created to generate a varying number of UDP bursty flows. A specific set of flows follows a Poisson distribution with constant interarrival starting times. For each specific flow (which models one VoIP session), bursty traffic with an ON period based on an exponential average of 4s was considered. The average interval between bursts of the same flow was set to 1.5s, as to get a duty cycle with an ON period of approximately 70% [55]. The Poisson process allows us to model VoIP in a realistic way. Keeping the average number of flows constant in the system results in emulating concurrent VoIP sessions that contribute to an average load of the network. Furthermore, flow generation is performed throughout the experiment duration.

The generation of flows with starting times following a Poisson distribution is not handled by MGEN. As such, a python script that models a Poisson process was used to generate, on-the-fly, the MGEN scripts for each experiment. This script is provided in Annex B.

5.1.3 Methodology

A set of Linux shell scripts to trigger the experiments were created that would perform every step necessary to get the experiments going in a way that can easily be replicated. Scripts are run on a sender due to some practical aspects. Firstly, the sender node also had a wired connection to the server destination, as well as to the NTP server, to allow it to synchronise its clock and control the client side of MGEN without compromising the experiments. In the scenarios in which there was more than one node generating traffic, these would be connected through a switch, which would also be connected to the servers through a router.

Running the performance evaluation on a testbed required synchronisation of all nodes, which was performed by recurring to an NTP server on INESC Porto's LAN.

Therefore, before each experiment, the clocks of all nodes involved on the simulations were synchronised by relying on NTP. Furthermore, every experiment was set to run for a period of 5 minutes, with an additional 40 seconds of dedicated warm-up time. This warm-up period was particularly important when using DYMO, as it allows for all nodes to discover their neighbours and their routing tables to stabilise.

MGEN logs were kept only at the destination node, as the only potentially useful information stored in the sender node log is the number of transmitted packets. However, this value can be derived from the sequence numbers of the packets received at the destination, so the receiver log is sufficient.

Although MGEN generates useful logs, it does not have any suitable tool to analyze them. As such, in order to extract and interpret the information stored on the logs, a python script was developed to run the necessary calculations. The raw data thus obtained with this script is then used in a spreadsheet to be treated and put in a way that can be easily understood.

Each of the experiments performed relies on a set of different parameters, ranging from different topologies with different number of network elements, to different numbers of concurrent flows. The purpose was to create a meaningful subset of scenarios which would include a representative subset of parameters.

Then, each experiment set was run 10 times with adequate independent random seeds for traffic generation. A 95% confidence interval for each averaged result is presented in the form of tables in Annex C.

The number of flows on the system was varied in order to simulate different loads of traffic, starting from 5 flows to 60 flows. The experimentation scenarios are therefore a product of the three topologies illustrated in section 5.1.1, together with the traffic settings described in section 5.1.2.

5.2 Results

In this section we present the results obtained. For each scenario run, results concerning packet loss, delay, as well as jitter are presented and explained. Each of the values has been computed as the average of the 10 runs and the levels for a 95% confidence interval are shown.

An additional remark relates to the fact that the testbed was not completely isolated as shown in section 5.1.1. Due to the time taken by each experiment (in average one hour), the networking environment may have slightly changed, e.g., we may start an experiment during work hours, when there are other people around using wireless, and end it at night, when there is no one around. This may add a little variation to the final results. Due to time constraints, it was not possible to repeat each experiment under different conditions, being this a task for future work.

5.2.1 Experiment 1

The first experiment run relies on Topology I (cf. section 5.1.1) which represents a linear topology, with an average six hop path. In the topology, node F corresponds to a sender and Node Dest corresponds to the destination. The sender generates traffic according to the settings described in section 5.1.2. Flows vary as 5, 10, 20, 30, and 40, in order to allow the representation of different loads. This corresponds roughly to rates of 40KBps, 80KBps, 160KBps, 240KBps and 320KBps, respectively, excluding headers. An additional remark is that each flow corresponds to a single VoIP session and hence, despite the fact that only one source is present on this scenario, there are concurrent sessions active.

Starting by the analysis of the average delay, Figure 5.4 plots the average delay achieved in milliseconds (y-axis) for the different load represented by the number of concurrent flows on the system in each instant (x-axis). The average delay is plotted both for relaying and DYMO, to allow a comparison to be adequately performed.

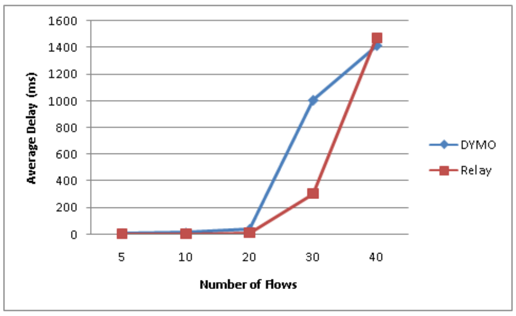

Figure 5.4: Experiment 1 average delay.

In regards to average delay, when the network has a low load, both DYMO and relaying show a low delay, even though relaying is the approach that attains the lowest values. Around 30 flows, both DYMO and relaying see a significant increase in delay, which is simply a product of the network becoming congested. In such case, relaying actually outperforms DYMO. Our interpretation is that while DYMO attains overhead in terms of control messages, relaying does not. Therefore, in case of congestion, DYMO shows a worse performance. Around 30 flows DYMO shows a significant increase in delay. The experiment as been repeated to ensure correctness of the results. Our hypothetical explanation for this relates to high network congestion, showing that DYMO does not react well to these conditions. We believe that due to such level of congestion, some of the required path setup signalling is lost, thus resulting in more variability.

Figure 5.5 plots packet loss, being the y-axis dedicated to packet loss in percentage, and the x-axis related to the average number of flows in the system.

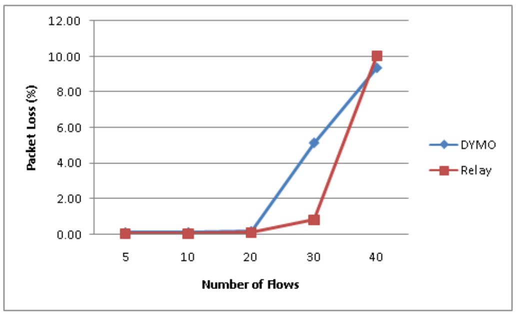

Figure 5.5: Experiment 1 packet loss.

The increase in packet loss closely follows the increase in delay. In this case, both parameters grow exponentially with more than 20 flows. This happens because as stated before, the network starts becoming saturated. The results obtained above 20 flows show a high delay, which is not compatible with VoIP traffic.

In regards to the behaviour of relaying when compared to DYMO, relaying shows a slightly better behaviour, both in terms of delay and packet loss. Again, this relates with the DYMO control overhead.

Jitter results are depicted in Figure 5.6, where jitter is provided by the (y-axis) vs. the number of concurrent flows (x-axis). The results show an increase as the transmission rate increases at the source, both on DYMO and relaying. The difference between DYMO and relaying is more noticeable in this result, which shows that DYMO attains more variability. This is a consequence of building on-demand routes.

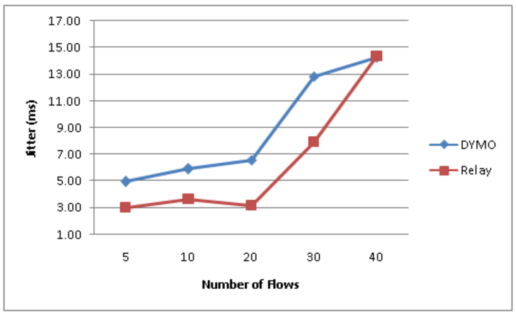

Figure 5.6: Experiment 1 jitter.

Some additional remarks relate to the variability observed in this scenario. For each run of the same experiment there was significant variation, especially in terms of delay and packet loss. This was due to the high number of hops in the scenario, which in turn causes a lot of unpredictable MAC contention, in particular due to the fact that the topology is an ad-hoc network with blocking being performed on the MAC layer. Furthermore there is no centralised control over the network, there is no guaranty that every node will get a fare share of transmission times.

The variation in the results was even more noticeable with a high number of concurrent flows (higher than 20), which is coherent with an increase in MAC contention.

5.2.2 Experiment 2

The previous scenario was repeated but now by relying on Topology II, which has only four hops instead of 6 as in Topology I. As in the previous experiment, node F was the single source transmitting concurrent flows according to the traffic settings described in section 5.1.2 to the destination node. Given that the experiment relies on a topology with an average shorter path (when compared to Topology I), the number of flows was varied from 10 (80KBps) to 60 (480KBps). The results for average delay and packet loss are respectively depicted in Figure 5.7 and Figure 5.8.

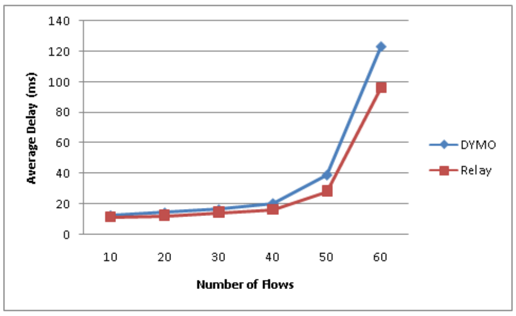

Figure 5.7: Experiment 2 average delay.

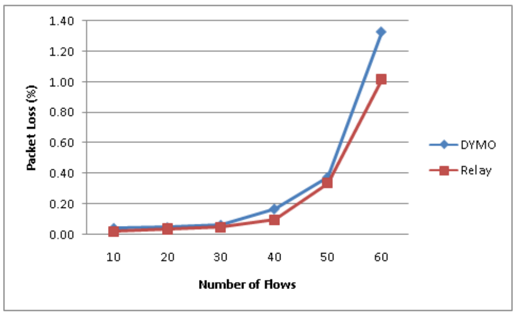

Figure 5.8: Experiment 2 packet loss.

For both DYMO and relaying the first observation to draw is that relaying still attains slightly lower values than DYMO. Saturation of the network is now only reached around 50 flows and hence the more abrupt raise in the delay. This is coherent with the fact that DYMO attains routing message overhead while relaying does not.

In regards to Experiment I, results now obtained attain significantly lower values in average. Only with 60 concurrent flows do we get a significant increase in delay and in the number of lost packets. Still, even with as many as 50 concurrent flows, the results are good enough for real time traffic constraints [53], with a maximum packet loss of around 0.4% and 40ms of maximum delay. This is simply a consequence of the lower number of hops in the topology used.

As illustrated by Figure 5.9, DYMO again shows more variability than relaying. Overall, jitter is also significantly lower than in the previous set of results. The value rises with the increasing number of concurrent flows, but does not vary more than 3ms.

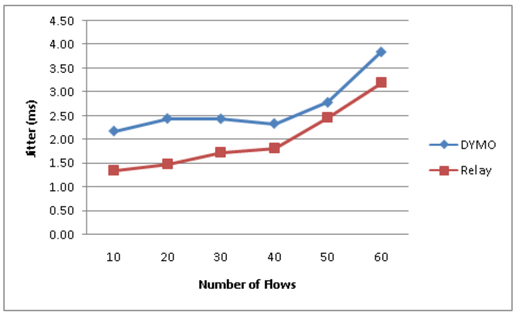

Figure 5.9: Experiment 2 jitter.

5.2.3 Experiment 3

The third experiment relies on Topology III and has the purpose to understand what could be the impact of having more than one source (more than one station) competing for the wireless media. In order to allow a fair comparison to the previous two scenarios, the same load is kept on the network, i.e., concurrent flows vary between 10 and 60. The average delay and packet loss are depicted in Figure 5.10 and Figure 5.11, respectively.

The first observation to draw from the results obtained is that DYMO and relaying show a closer behaviour with relaying presenting slightly lower values. There is therefore less variability in DYMO in contrast to the previous experiments run. Main reason for this is that the load on the network is being distributed by two sources and hence routes are possibly being kept active for a longer period. In contrast, for the linear topologies relied before, intermediate nodes would see the routes updated more frequently.

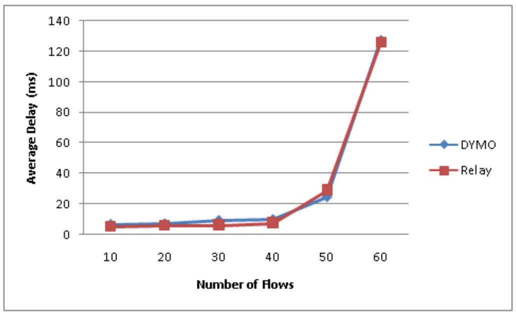

Figure 5.10: Experiment 3 average delay.

Let us now compare these results with the ones obtained in experiment 2, which holds the same average path length with a single source.

Average delay is lower for experiment 3 which happens because now two sources are contending for the media on the first hop. The same behaviour is observed for packet loss. In average, experiment 3 has half of the delay and half of the packet loss of experiment 2. This was expected, given that the network load is being generated by two sources instead of one. However, the most significant remark to be made is that DYMO shows more variability than relaying, even for these experiments where congestion is not an issue.

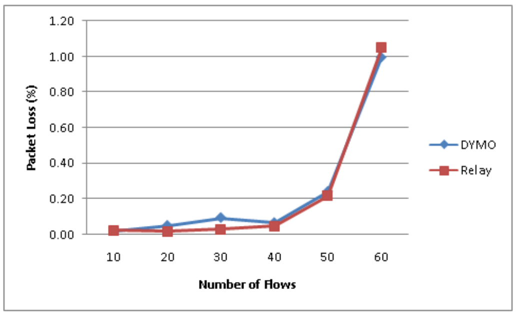

Figure 5.11: Experiment 3 packet loss.

The values obtained for jitter follow the behaviour of that on previous experiments, i.e., it increases almost linearly with the increase in transmission rate, as shown in Figure 5.12. When compared to the jitter values obtained in Experiment II, there is a slight increase. This was expected, as we increased the concurrency of flows. Relaying, which does not have the overhead of control packets as DYMO, performs a little better in terms of jitter, being always slightly below its routing counterpart.

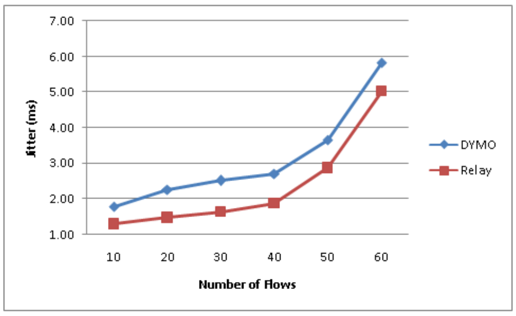

Figure 5.12: Experiment 3 jitter.

The results of this experiment show that the diversity of source nodes will impact the performance of both DYMO and relaying. As we double the number of sources, packet loss and delay decrease, while jitter is increased slightly.

Chapter 6

Conclusions and Future Work

This chapter relates to the summary and conclusions to be drawn from the work developed and which gave rise to this thesis. The thesis work was split into two main categories: analysis of the problem space to be addressed, as well as analysis of the performance of the proposed problem space.

In regards to the first part (analysis of the problem space), the thesis work related to the analysis of related work and the state-of-the-art concerning wireless architectures evolution, multihop routing, as well as relaying in wireless. In addition to related work, the problem space analysis also integrated available tools, potential advantages and disadvantages, as well as the justification for the choice of a specific set of tools.

In regards to the second part, analysis of the performance of the proposed problem space, the thesis work delved into the implementation of a local wireless testbed that could be used as proof-of-concept for relaying. This implied dealing with several implementation problems, in addition with contributing with specific configuration and implementation capabilities. The work was split into three main modules, namely, relaying, multihop routing, as well as aggregation of radio resources. The performance evaluation dealt with the two main aspects that were the focus of this thesis, namely, relaying vs. multihop routing and attempted to answer specific questions related to scalability and reliability. The purpose of such evaluation was to understand up to which point can relaying be an interesting concept to deal with, in comparison to multihop routing (which, for some scenarios, may turn out to be a complex solution). The extensive evaluation was performed on the testbed which was a contribution of this thesis, and evaluation parameters considered were end-to-end packet delay, average packet loss, as well as jitter. Traffic was modelled by MGEN, a well-known traffic generator, and incorporated realistic parameters.

Albeit being a first step towards the performance of relaying vs. routing, the evaluation performed allow us to conclude that relaying does have interesting features in the tested scenarios, showing less variability than the instance of multihop routing chosen (NIST-DYMO). Such lower variability is a result of the lack of signalling overhead which DYMO has to deal with, and also of the simpler filtering rules that are the basis to forward traffic. There is however the need to confirm the results achieved by means of scenarios attaining even more variability in terms of the different parameters, e.g., topologies, traffic matrix, node mobility, energy constrains. These are tasks that will be addressed as future work (cf. section 6.2).

6.1 Contributions and Challenges Faced

The main contributions of this thesis can therefore be listed as:

- Analysis of related work concerning wireless architectures evolution and relaying in wireless.
- Analysis of the status of current multihop on-demand routing implementations.
- Analysis of related work and of current implementations concerning radio resource aggregation.
- Development of a wireless testbed including both x86 and mips architectures. The development contemplated a full-fledged OS installation for all the nodes, including the embedded ones, as well as configuration required to model nodes spaced with and without overlapping in a contiguous, small space.
- Implementation of generic relaying tools (scripts) capable of turning any wireless node into a relay.
- Implementation of specific features for the NIST-DYMO (kernel space) in order to make it run adequately on the developed testbed:
  - ported NIST-DYMO to the most recent available kernel at the time of the testbed setup (version 2.6.27);
  - improvement of NIST-DYMO in terms of robustness of implementation (solved main bugs);
  - improvement of NIST-DYMO in terms of route discovery and maintenance;
  - implementation of the capability for a node to act as a gateway;
  - implementation of the capability to allow NIST-DYMO support both in x86 and mips architectures;
- implementation of specific features to perform aggregation of radio resources on the testbed nodes;
- implementation of kernel module capable of performing network hopping between different WLANs, based on related work;

- Performance evaluation of the implemented form of relaying vs. the improved NIST-DYMO for the developed testbed:
  - traffic modeling according to a realistic setting of VoIP (codec G.711);
  - on-the-fly configuration of different topologies that served as the basis for the different scenarios evaluated;
  - on-the-fly interpretation and plotting of the raw-logs obtained;

During the thesis work, the main challenges and lessons learned were:

- Testbed constraints (space, architectures, driver-tool mapping).
- Implementation issues and lack of online support for outdated tools.
- Performance evaluation:
  - NTP synchronisation issues. The measurements of end-to-end delay required the source and destination nodes to be kept synchronised during experiments, but we observed a slight variation on the clocks of both nodes.
  - DYMO did not always behave as expected. While running experiments, we noticed that the implementation of NIST-DYMO did not always obey to the standard. We corrected it as much as possible.
  - Experiment duration. By relying on a testbed, our experiments ran on real-time. Each individual run of one experiment lasted 6 minutes, thus, each experiment required at least 12 hours to extract results. This should be lower if we had considered a simulator, being the trade-off reliability based on realistic scenarios vs. time.
  - Variability due to lack of complete isolation of the testbed. We noticed some variability due to network load in the lab. In the future the testbed will be completely isolated.

6.2 Future Work

As mentioned, the current work represents an initial step concerning an adequate understanding of the performance that relaying may attain. A first step to consider next, relates to the integration of radio resource aggregation into the relaying module and to repeat the experiments performed so far considering this new module. A second step relates to the need to consider at least an additional multihop routing protocol, e.g. OLSR, to better understand the performance that relaying may attain. A third step relates to the need to ensure that all the implementations available and relied upon are in sync with the available standard version, even if still in a phase of work-in-progress.

In regards to performance evaluation, there is the need to add more scenarios to the experiments, namely, more variability in terms of topology and also in terms of traffic relied upon. Therefore, a next step is to consider other topologies and inclusive consider how can the testbed be plugged into a more scalable wireless testing platform, e.g. Emulab [56]. One other step to take in regards to the evaluation is to consider more realistic tools, e.g. ixChariot and real applications, e.g. Skype, near On-demand Video, as well as other types of traffic patterns. In addition, there is the need to consider more sophisticated forms of relaying and also, to consider other evaluation parameters, e.g., energy efficiency or adaptability to node movement.

Annex A: Testbed Configuration Scripts

This is the shell script to connect to the ad-hoc network. We need to specify the interface to use, the IP we want to assign and an optional IP for the gateway.

```
#! /bin/sh
#usage: connect.sh <interface> <ESSID> <N ip> [<N relay gw>]

if test -x /usr/bin/sudo ; then
SUDO=/usr/bin/sudo
echo "sudo in: $SUDO"
else
# FONERA
SUDO=""
echo "No sudo"
fi
if test -x /sbin/iptables ; then
# Ubuntu
IPT=/sbin/iptables
else
# OpenSUSE
IPT=/usr/sbin/iptables
fi
/bin/echo "iptables in: $IPT"

if test -x /usr/sbin/iwconfig ; then
# SUSE
IWC=/usr/sbin/iwconfig
else
# Ubuntu
IWC=/sbin/iwconfig
fi
echo "iwconfig: $IWC"
$SUDO killall NetworkManager

$SUDO /sbin/ifconfig "$1" down
$SUDO $IWC "$1" mode ad-hoc
$SUDO $IWC "$1" essid "$2"
$SUDO /sbin/ifconfig "$1" up
$SUDO /sbin/ifconfig "$1" 10.0.$3/24

if test $# -gt 3; then
/bin/echo "Relay Mode"
$SUDO /sbin/route add default gw 10.0.$4
```

```
$SUDO $IPT -t nat -A POSTROUTING -o $1 -j MASQUERADE

else
/bin/echo "DYMO Mode"
$SUDO /sbin/route add default gw 127.0.0.1 metric 1000
fi
```

This is the shell script to configure the network on the nodes. We need to specify the interface to use and, optionally, some MAC addresses of whom we want to block traffic from.

```
#! /bin/sh

# usage: script.sh [<interface> [MAC to block]]
# use with no parameters to clear all iptables rules

PWD="`/bin/echo $0 | sed "s/\/[^\/]*$//g"`"

# The file ``mac_list'' contains a list of the names of the nodes, followed by their MAC addresses
LIST=mac_list

if test -x /usr/bin/sudo ; then
SUDO=/usr/bin/sudo
/bin/echo "sudo in: $SUDO"
else
# FONERA
SUDO=""
/bin/echo "No sudo"
fi

if test -x /sbin/iptables ; then
# Ubuntu
IPT=/sbin/iptables
else
# OpenSUSE
IPT=/usr/sbin/iptables
fi

/bin/echo "iptables in: $IPT"

# Flush all existing rules
$SUDO $IPT --flush
$SUDO $IPT --delete-chain
$SUDO $IPT --table mangle --flush
$SUDO $IPT --table mangle --delete-chain
$SUDO $IPT --table nat --flush
$SUDO $IPT --table nat --delete-chain
$SUDO $IPT --table mangle -P PREROUTING ACCEPT
# Default rule is to accept all connections
$SUDO $IPT -P INPUT ACCEPT

if test $# -ne 0 -o "$1" ; then
# Default gateway interface from routing table
GW=`/sbin/route -n | grep -E "^0.0.0.0" | head -n1 | sed "s/.*\ \([^\s]\+\)$/\1/g"`

if test "$GW" != "" -a "$GW" != "lo" ; then
```

```
# There is a default gateway, so we must perform NAT
/bin/echo -e "Configuring IP Masquerade\nGateway Interface - $GW"
$SUDO $IPT -t nat -A POSTROUTING -o $GW -j MASQUERADE
$SUDO $IPT -A FORWARD -i $GW -o $1 -m state --state RELATED,ESTABLISHED -j ACCEPT
$SUDO $IPT -A FORWARD -i $1 -o $GW -j ACCEPT
if test `uname -m` = "mips" ; then
$SUDO $IPT -A FORWARD -i $GW -o br-lan -m state --state RELATED,ESTABLISHED -j ACCEPT
$SUDO $IPT -A FORWARD -i br-lan -o $GW -j ACCEPT
fi
else
/bin/echo "No gateway detected"
fi
# Block all traffic from MAC addresses in given arguments
ARGS=`/bin/echo $* | sed "s/\ /\\\\\|/g"`
MACS=`cat $PWD/$LIST | grep "$ARGS" | sed "s/.*[\t\ ]\+\(.*\)/\1/g"`

for MAC in $MACS
do
/bin/echo "Blocking $MAC"
$SUDO $IPT --table mangle -A PREROUTING -m mac --mac-source $MAC -j DROP
done
$SUDO /bin/sh -c "/bin/echo "1" > /proc/sys/net/ipv4/ip_forward"
else
/bin/echo "All iptables rules cleared"
if test `uname -m` = "mips" ; then
/etc/init.d/firewall restart
else
$SUDO /bin/sh -c "/bin/echo "0" > /proc/sys/net/ipv4/ip_forward"
fi
fi
```

Annex B: Traffic Generator Script

This is the python script used to generate the MGEN scripts.

```
from numarray.random_array import poisson
import sys

file = open(sys.argv[3], 'w')

# Number of flows to generate
N = int(sys.argv[1])

# UDP or TCP
type = sys.argv[2]

file.write("TXLOG\n")

start = 0

for n in range(1,N+1):
start = start + poisson(2)
out = str(start) +  " ON " + str(n) + " " + type
+ " DST 194.117.26.144/" + str(5000+n) + " \\\nBURST [RANDOM 4.0 PERIODIC [50 226] EXPONENTIAL 1.5]\n"
file.write(out)

for n in range(1,N+1):
out = "300 OFF " + str(n) + "\n"
file.write(out)

file.close()
```

Annex C: Confidence Intervals For Experimental Results

Table 6.1: 95% Confidence interval for experiment 1.

		DYMO		Relaying	
Flows	Param.	min	max	min	max
5	delay	5.42	7.11	1.29	3.59
	loss	0.07	0.10	0.02	0.06
	jitter	4.67	5.18	2.33	3.64
10	delay	9.37	11.36	2.41	5.36
	loss	0.08	0.11	0.04	0.06
	jitter	5.63	6.15	2.64	4.62
20	delay	27.12	49.73	5.36	9.40
	loss	0.13	0.18	0.08	0.10
	jitter	5.62	7.48	2.68	3.61
30	delay	568.34	1438.16	157.41	449.93
	loss	2.59	7.67	0.26	1.36
	jitter	9.47	16.22	6.21	9.63
40	delay	1257.64	1566.62	1258.97	1684.61
	loss	8.16	10.52	5.62	14.37
	jitter	13.26	15.27	12.03	16.67

Table 6.2: 95% Confidence interval for experiment 2.

		DYMO		Relaying	
Flows	Param.	min	max	min	max
10	delay	12.27	12.76	10.88	11.56
	loss	0.03	0.04	0.01	0.02
	jitter	2.07	2.27	1.25	1.44
20	delay	13.86	14.77	11.49	12.92
	loss	0.03	0.05	0.03	0.04
	jitter	2.27	2.60	1.30	1.64
30	delay	15.46	17.24	12.95	15.40
	loss	0.05	0.07	0.03	0.06
	jitter	2.20	2.67	1.45	1.98
40	delay	16.01	24.89	14.16	17.72
	loss	0.08	0.25	0.07	0.12
	jitter	1.78	2.88	1.49	2.12
50	delay	23.39	54.70	22.11	34.18
	loss	0.13	0.62	0.20	0.47
	jitter	2.19	3.37	1.98	2.95
60	delay	77.49	168.79	57.90	134.06
	loss	0.26	2.40	0.73	1.30
	jitter	3.01	4.70	2.50	3.88

Table 6.3: 95% Confidence interval for experiment 3.

		DYMO		Relaying	
Flows	Param.	min	max	min	max
10	delay	4.56	8.21	4.54	5.30
	loss	0.01	0.02	0.01	0.04
	jitter	1.70	1.86	1.01	1.55
20	delay	6.49	7.11	5.28	5.78
	loss	0.02	0.08	0.01	0.02
	jitter	2.13	2.37	1.21	1.71
30	delay	8.00	9.74	5.50	6.42
	loss	0.04	0.15	0.02	0.04
	jitter	2.23	2.78	1.29	1.97
40	delay	8.86	10.18	5.60	9.07
	loss	0.05	0.08	0.03	0.05
	jitter	2.61	2.79	1.42	2.31
50	delay	16.00	32.77	8.31	49.22
	loss	0.09	0.39	0.15	0.28
	jitter	3.43	3.86	2.14	3.59
60	delay	81.91	173.17	26.90	224.47
	loss	0.40	1.59	0.83	1.27
	jitter	5.39	6.25	3.89	6.17

References

[1] R. Sofia and P. Mendes. User-provided networks: consumer as provider. *Communications Magazine, IEEE*, 46(12):86–91, December 2008.

[2] Whisher - building the world's largest wifi network. `http://www.whisher.com/`.

[3] Fon. `http://www.fon.com/`.

[4] I. Chlamtac, Marco Conti, and J. N. Liu. Mobile ad hoc networking: imperatives and challenges. *Ad Hoc Networks*, 1(1):13 – 64, 2003.

[5] Openspark - better than free. `https://open.sparknet.fi/index.php`.

[6] P. Chung and S. Liew. Throughput analysis of ieee802.11 multi-hop ad hoc networks. *IEEE/ACM Trans. Networking*, 15:309–322, 2007.

[7] J. Li, C. Blake, S. J. De Couto, H. I. Lee, and R. Morris. Capacity of ad hoc wireless networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 61–69, New York, NY, USA, 2001. ACM.

[8] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Trans. on Info. Theory*, pages 388–404, 2000.

[9] F. Stajano and R. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. 1999.

[10] X. Zou, B. Ramamurthy, and S. Magliveras. Routing techniques in wireless ad hoc networks — classification and comparison. In *Proceedings of the Sixth World Multiconference on Systemics, Cybernetics, and Informatics (SCI 2002*, 2002.

[11] D. B. Johnson, D. A. Maltz, and J. Broch. Dsr: The dynamic source routing protocol for multi-hop wireless ad hoc networks. In *In Ad Hoc Networking, edited by Charles E. Perkins, Chapter 5*, pages 139–172. Addison-Wesley, 2001.

[12] E. M. Royer and C. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications*, 6:46–55, 1999.

[13] P. Jacquet, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *IEEE International Multi Topic Conference*, pages 62–68, 2001.

[14] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *IEEE Computer and Communications Societies, Annual Joint Conference of the*, pages 1405–1413, 1997.

[15] C. E. Perkins. Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.

[16] C. E. Perkins and I. D. Chakeres. Dynamic manet on-demand (dymo) routing, ietf draft (work in progress).

[17] T. H. Clausen and P. Jacquet. Optimized link state routing protocol (olsr), ietf rfc 3626.

[18] J. Moy. Ospf version 2, ietf rfc 2328.

[19] D. R. Oran. Osi is-is intra-domain routing protocol, ietf rfc 1142.

[20] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector (aodv) routing, ietf rfc 3561.

[21] S. Gwalani, E.M. Belding-Royer, and C.E. Perkins. Aodv-pa: Aodv with path accumulation. In *Communications, 2003. ICC '03. IEEE International Conference on*, volume 1, pages 527–531 vol.1, May 2003.

[22] I. D. Chakeres and L. Klein-Berndt. Aodvjr, aodv simplified. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(3):100–101, 2002.

[23] T. H. Clausen, C. M. Dearlove, J. W. Dean, and C. Adjih. Generalized mobile ad hoc network (manet) packet/message format, ietf rfc 5444.

[24] P. Liu, Z. Tao, S. Narayanan, T. Korakis, and S. Panwar. Coopmac: A cooperative mac for wireless lans. *IEEE Journal on Selected Areas in Communications*, 25(2):340 – 354, February 2007.

[25] Windows xp internet connection sharing. `http://www.practicallynetworked.com/sharing/xp_ics/`.

[26] mac80211 - linux wireless. `http://wireless.kernel.org/en/developers/Documentation/mac80211`.

[27] Windows 7 with native wireless support. `http://www.istartedsomething.com/20090516/windows-7-native-virtual-wifi-technology-microsoft-research/`.

[28] Ip masquerading. `http://www.comptechdoc.org/independent/networking/guide/netipmasq.html`.

[29] J. Per, L. Tony, H. Nicklas, M. Bartosz, and D. Mikael. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 195–206, New York, NY, USA, 1999. ACM.

[30] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *In Proc. Mobicom*, pages 85–97, 1998.

[31] R. S. Gray, D. Kotz, C. Newport, N. Dubrovsky, A. Fiske, J. Liu, C. Masone, S. McGrath, and Y. Yuan. Outdoor experimental comparison of four ad hoc routing algorithms. In *MSWiM '04: Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 220–229, New York, NY, USA, 2004. ACM.

[32] S. R. Das and C. E. Perkins. Performance comparison of two on-demand routing protocols for ad hoc networks. 2000.

[33] I. Chakeres and Belding-Royerm E. Aodv implementation design and performance evaluation. *International Journal of Wireless and Mobile Computing*, 2005.

[34] Kernel aodv. `http://w3.antd.nist.gov/wctg/aodv_kernel/index.html`.

[35] Aodv-uu. `http://sourceforge.net/projects/aodvuu/`.

[36] Aodv-uiuc. `http://sourceforge.net/project/showfiles.php?group_id=56798`.

[37] Nist-dymo. `http://sourceforge.net/projects/nist-dymo/`.

[38] Dymoum. `http://masimum.dif.um.es/?Software:DYMOUM`.

[39] Dymo-au. `http://www.daimi.au.dk/~rolft/wp/?page_id=17`.

[40] Madwifi. `http://madwifi-project.org/`.

[41] R. Chandra and P. Bahl. Multinet: connecting to multiple ieee 802.11 networks using a single wireless card. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 882–893 vol.2, March 2004.

[42] A. J. Nicholson, S. Wolchok, and B. D. Noble. Juggler: Virtual networks for fun and profit. Technical Report CSE-TR-542-08, University of Michigan, April 2008.

[43] Juggler project at university of michigan. `http://www.eecs.umich.edu/~tonynich/juggler/`.

[44] S. Kandula, K. C. Lin, T. Badirkhanli, and D: Katabi. FatVAP: Aggregating AP Backhaul Capacity to Maximize Throughput. In *5th USENIX Symposium on Networked Systems Design and Implementation*, San Francisco, CA, April 2008.

[45] Fatvap project at mit. `http://nms.lcs.mit.edu/~kandula/projects/fatvap/`.

[46] iperf. `http://sourceforge.net/projects/iperf`.

[47] netperf. `http://www.netperf.org/netperf/`.

[48] Multi-generator (mgen). `http://cs.itd.nrl.navy.mil/work/mgen/`.

[49] Linux wireless. `http://linuxwireless.org/`.

[50] Openwrt. `http://openwrt.org/`.

[51] Low level operators and bit fields. `http://www.cs.cf.ac.uk/Dave/C/node13.html`.

[52] Ian software server. `http://ian.inescporto.pt/software.html`.

[53] P. Hofmann, C. An, L. Loyola, and I. Aad. Analysis of udp tcp and voice performance in ieee 802.11b multihop networks. *European Wireless*, 1-4, April 2007.

[54] Traffic analysis (voip) - cisco systems. `http://www.cisco.com/en/US/docs/ios/solutions_docs/voip_solutions/TA_ISD.html`.

[55] K. Chen, C. Huang, P. Huang, and C. Lei. Quantifying skype user satisfaction. In *SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 399–410, New York, NY, USA, 2006. ACM.

[56] Cmulab. `http://boss.cmcl.cs.cmu.edu/`.