NTNU
Innovation and Creativity

Perspectives to Ad-hoc Extensions of Cellular Networks

Andreas Schjønhaug

Master of Science in Computer Science
Submission date: September 2006
Supervisor: Alf Inge Wang, IDI
Co-supervisor: Thomas Heide Clausen, LIX, Ecole Polytechnique

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

With more and more cellular phone handsets being introduced with short-range wireless communications interfaces (Bluetooth and, lately, wifi), an alternative to using the purely cell-based mobile phone networks infrastructure can be envisioned: mobile phone cells can be extended via ad-hoc relaying of traffic from out-of-coverage handsets towards in-coverage handsets, and mobile phones might be able to establish communication paths between them without passing through the cellular infrastructure. Furthermore, with wifi-enabled handsets and low-bandwidth voice-over-IP, traditional cellular network coverage may easily be extended by multi-fabric access to e.g. GSM, UMTS and wifi access points.

This presents a number of challenges including, but not limited to, billing, security, multi-fabric routing, hand-over and quality of service. One task of this project is thus to look at the necessity of providing brokering network access, addressing variables influencing its success.

The other task for this project is to construct the necessary mechanisms for establishing the ad-hoc part of an extension to cellular networks. This involves analysing ad-hoc networking and implementing OLSRv2, one of several experimental solutions for ad-hoc networking that exist within the Internet Engineering Task Force (IETF). To emulate such a system, but also for debugging and evaluating the implementation, a test-bed is required to be constructed.

Assignment given: 15. March 2006
Supervisor: Alf Inge Wang, IDI

Perspectives to Ad-hoc Extensions of Cellular Networks

Andreas Schjønhaug

September 21, 2006

Abstract

This master dissertation describes perspectives to an ad-hoc extension of cellular networks such as GSM or UMTS, but the ideas may also be valid for 802.11 hotspots. By extending cellular networks with ad-hoc networking, the range of the cellular network can greatly be extended. Other than the technical infrastructure to provide connectivity, a policy enforcing mechanism - a broker - is envisioned to administer and maintain connections, and handle issues such as billing, security, hand-over and quality of service. This thesis describes a thus far little explored application domain for mobile networks.

The dissertation analyses ad-hoc networks and the application areas in general, and in particular the Optimized Link-State Routing protocol version 2 (OLSRv2), which is one of the standards track ad-hoc routing protocols developed within the Internet Engineering Task Force (IETF).

The results obtained during this project are the very first implementation of the OLSRv2 specifications, and also the very first practical experiments with the protocol. The implementation has been tested and validated using a small test-bed. Furthermore, a brokering network access mechanism is suggested to accommodate an ad-hoc extension of cellular networks, discussing factors that should be considered before a successful deployment.

Preface

This master's dissertation is submitted to the Norwegian University of Science and Technology (NTNU) in partial fulfilment of the requirements for the degree Master of Science. It is written in the period from 15th of March to 21st of September 2006 for the Department of Computer and Information Science (IDI). The actual work has taken place at the Département d'Informatique (DIX) at Ecole Polytechnique in Paris, France.

I first made contact with Thomas H. Clausen at DIX during fall 2005, conveying my interest in mobile networks. The features of a possible dissertation were discussed in the following months, and further refined during spring 2006. The work has been completed over a period of 26 weeks, with main focus on implementing the new version of the Optimized Link-State Routing Protocol.

I have been following the mobile market closely the last years, astonished by the continuous change and innovation. The trends in the market point towards a convergence between traditional cellular based networks and networks with short range interfaces such as wifi, manifested through the increasing number of handsets proposing both traditional UMTS/GSM and wifi interfaces. This trend may suggest a potential deployment of ad-hoc networks, and I feel privileged to have had the chance of finding myself on the bleeding edge of ad-hoc networking for the last months, working with some of the best researchers in the field. If this alone has not been a motivating factor, my tasks have been both interesting and demanding, and they have been met with the greatest enthusiasm.

I would like to thank Thomas H. Clausen at DIX at Ecole Polytechnique, for his invaluable help and support throughout the project, as well as the entire Hipercom team. I would also like to thank Alf Inge Wang at IDI, without whom my exchange programme in France would not have been possible, and Svein Leidland for providing me with an office in the last weeks of my work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter gives an introduction and overview of the motivating application scenario. It will describe perspectives to an extension of cellular based networks where ad-hoc networking is used, as well as describing the structure and scope of the dissertation. Thus, section 1.1 will take a look at the background and motivating factors. Section 1.2 describes the problem domain, while section 1.3 takes a closer look at which of these problems the dissertation will treat.

1.1 Background and motivation

With more and more cellular phone handsets being introduced with short-range wireless communications interfaces (Bluetooth and, lately, wifi), an alternative to using the purely cellular based mobile phone networks infrastructure can be envisioned: mobile phones might be able to establish communication paths without passing through the cellular infrastructure, via ad-hoc relaying of traffic from out-of-coverage handsets towards in-coverage handsets. Furthermore, with wifi-enabled handsets and low-bandwidth voice-over-IP, traditional cellular network coverage may easily be extended by multi-fabric access to e.g. GSM, UMTS and wifi access points.

This presents a number of challenges including, but not limited to, billing, security, multi-fabric routing, hand-over, quality of service and so on.

1.2 Problem domain description

Cellular based networks such as the **Global System for Mobile Communications (GSM)** has been around for decades. Even though there has been an increase in bandwidth and greater coverage over the years, the fundamental principle stays the same. Users communicate with each other

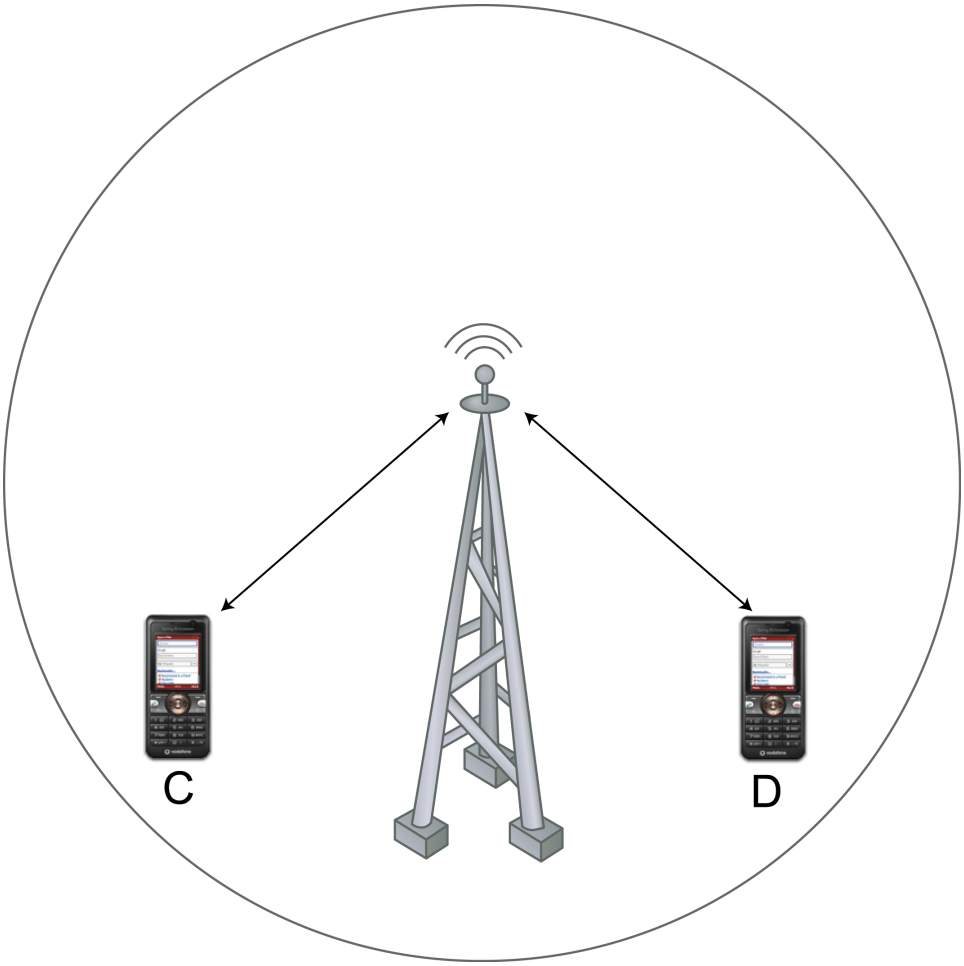


Figure 1.1: A traditional cellular mobile network. Users communicate with each through the use of cellular towers when within range (big circle) but not directly.

through cellular towers when in coverage. The traffic must pass through the towers, acting as **base stations**, and users are not able to communicate directly (figure 1.1).

In recent years, **Personal Digital Assistants (PDAs)**, but also mobile phones have been equipped with wifi network interfaces, in addition to the normal support for cellular networks such as the **Universal Mobile Telecommunications System (UMTS)**. This opens the door to a co-existence between cellular and **mobile ad-hoc networks (MANETs)**, broadening the coverage area and reducing the load on the cellular network.

However, this merge is far from trivial when it comes to billing. In purely cellular networks, mobile operators control the network and can easily monitor the traffic. Using different billing schemes, either the sender or

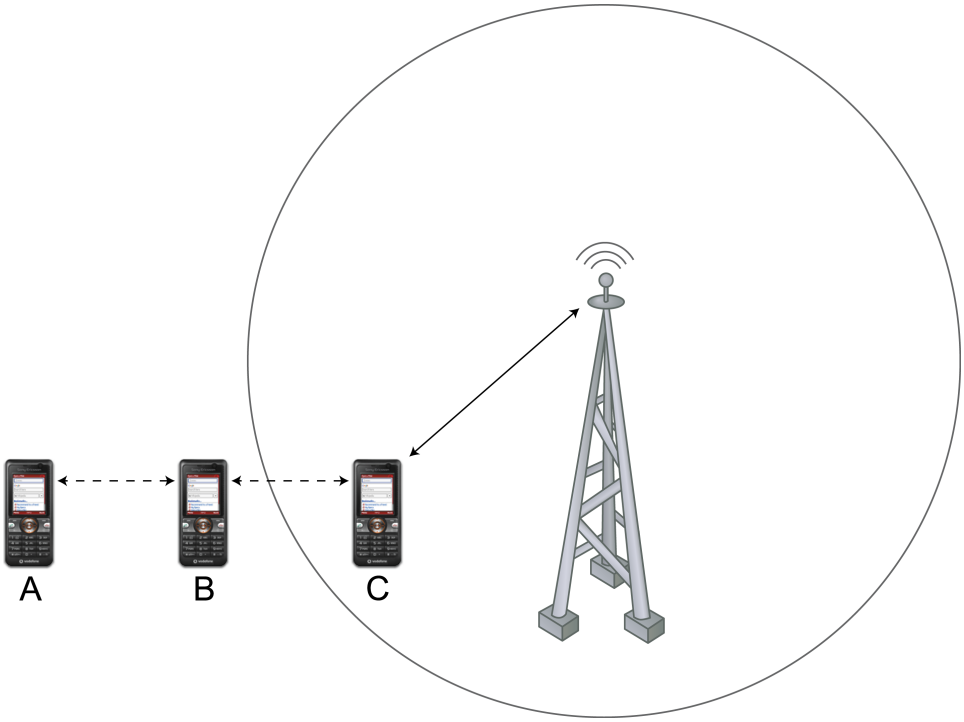


Figure 1.2: A scenario with both a MANET and a cellular network in action. The two headed arrows indicate connections, where the dashes lines are MANET connections while the solid lines are cellular network connections. The big circle denotes the coverage area of the cellular tower. Only user C is within coverage of the cellular tower, and in our scenario user A wishes to gain access to its network.

receiver, or both, can be charged.

The introduction of MANETs within cellular networks makes things more complicated. Figure 1.2 shows an example where users, equipped with handsets supporting both cellular and MANETs are in the same network. A user outside the coverage area of the cellular tower wishes to gain access to the network provided by the tower. Obviously, he cannot use the built-in cellular network support, since he is not within range. However, intermediate nodes relay the traffic, a connection is possible. This connection arises a number of questions. Who will pay for the connection? After all, the sender is not the one directly connected to the cellular tower. Why would the intermediate nodes relay traffic? By doing so, there are nothing but disadvantage as the relaying is expensive in terms of process power and power consumption. Should there be some kind of reward for bandwidth sharing users?

Developing the previous scenario, the billing gets more complicated. In

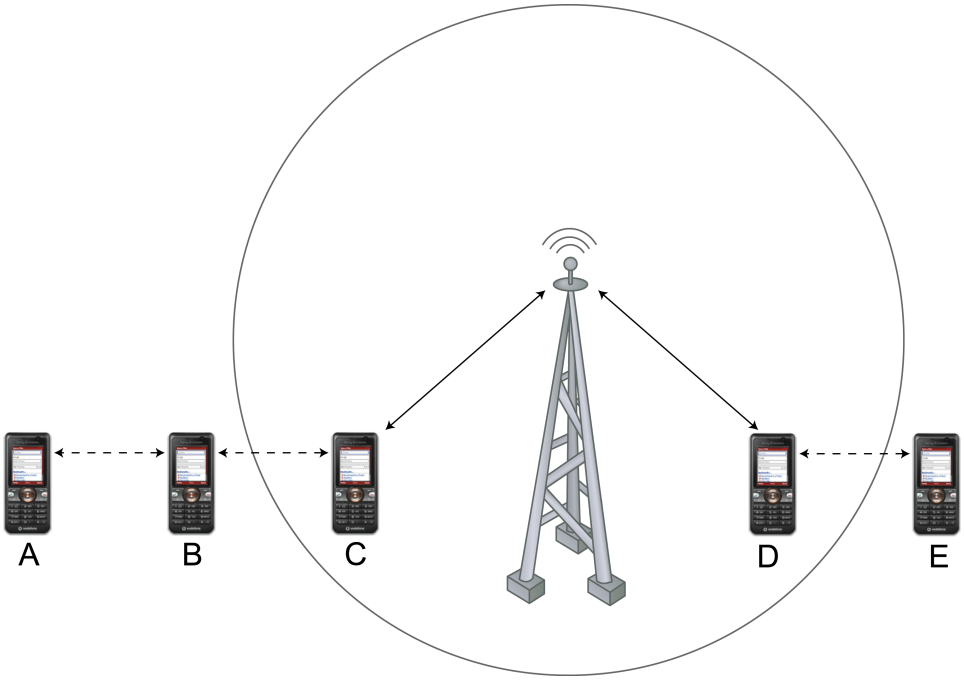


Figure 1.3: A scenario with both a MANET and a cellular network in action. The two headed arrows indicate connections, where the dashed lines are MANET connections while the solid lines are cellular network connections. The big circle indicates the coverage area of the cellular tower. In this scenario user A wants to connect to user E, but none of them are within the coverage area of the cellular tower. However, using a combination of MANET and cellular network connections there exists a path over which they can communicate.

figure 1.3, two users outside the coverage area of the cellular tower wish to communicate. Like in the previous scenario, the path from the sender to the receiver involves both the MANET and cellular network. Who should pay for the connection through the cellular tower? Should the expenses be shared between the sender and receiver, and if so, how should it be done?

Besides billing, other issues may also be addressed. In networks where users relay traffic for each other, security becomes important. How can users be sure that messages they send are not read or altered by others when relayed? How can they know that the messages actually reach the destinations? There are also questions related to **hand-overs**, that is, when users switch between the networks. Should the transition be automatic, or should the user disconnect from one network and then reconnect to another? Finally, **quality of service (QoS)** is important. What are acceptable packet losses and bandwidth? How can a system make QoS guarantees in a multi network environment, possibly involving intermediate nodes forwarding traf-

fic? How can the system decide whether to use the ad-hoc or cellular network with a given set of conditions?

All these questions need to be accounted for, for an ad-hoc extension to cellular networks to work as intended. There is thus a need for a mechanism providing brokering network access, handling issues such as billing, security, hand-over and QoS.

1.3 Objectives and scoping

With the last section in mind, this dissertation deals with a certain number of the issues mentioned. The proposed system (figure 1.3) can be divided in two main parts: the cellular and ad-hoc network. Cellular networks have been around for decades, are well known, and do not offer many technical challenges. Ad-hoc networks, on the other hand, is still at an early stage of development, and not very explored in this context. These observations made ad-hoc networking the most interesting approach to take.

The tasks among others of this project are thus to construct the necessary mechanisms for establishing the ad-hoc part of the system. This involves analysing ad-hoc networking and implementing a routing protocol. Several experimental solutions for ad-hoc networking exist, but while working with the team behind OLSR, a quite natural choice was to implement the new version currently under development.

The proposed system (figure 1.3) includes handsets capable of both ad-hoc and cellular based communication. To emulate such a system, but also for debugging and evaluating the implementation, a test-bed is required to be constructed.

Furthermore, the necessity of providing brokering network access will be discussed, looking at motivating factors for such a service.

1.4 Overview and structure

The remainder of this dissertation is structured as follows.

Chapter 2 describes mobile ad-hoc networks, and looks at what they do, where they are used and how they work.

Chapter 3 describes one of several ad-hoc routing protocols, OLSR. The general characteristics and algorithms are explained before detailing its anatomy, explaining how the different components work together. Finally, a comparison between OLSR [6] and its coming successor OLSRv2 is given, highlighting the improvements.

Chapter 4 takes a closer look at the work done during the implementation of OLSRv2. The chapter provides background for the programming

language used, focusing on its strengths and weaknesses regarding an implementation of an ad-hoc routing protocol. Then, an overview of the code structure is given, followed by an overview of the test-bed, looking at why it was set up and what it was used for.

Chapter 5 describes the perspectives of a brokering network access service, highlighting its motivating factors.

Chapter 6 discusses future work.

Chapter 7 concludes this dissertation, by summarising the goals achieved and the contributions made to the OLSRv2 standardisation effort.

1.5 Summary

Limited coverage in cellular networks may be extended by deploying ad-hoc networks in the out-of-coverage areas, connecting these networks to cellular towers. This chapter has described such a system, discussing some its challenges concerning billing, hand-overs and quality of service. It has also scoped the work in this dissertation to focus on the ad-hoc part, implementing and testing an ad-hoc routing protocol, as well as discussing the use of a broker service.

Thus, the following chapter will detail mobile ad-hoc networks, giving an overview of their application domains as well as analyse different routing protocol taxonomies.

Chapter 2

Mobile Ad-hoc Networks

Focusing on the ad-hoc part of the proposed system in section 1.2, this chapter will in section 2.1 provide an overview of **mobile ad-hoc networks (MANETs)**. Then, applications of MANETs are discussed in section 2.3, followed by a presentation of different routing protocol taxonomies in section 2.4.

2.1 Overview

A MANET is a collection of **mobile nodes** communicating over **wireless links**. Mobile nodes (see figure 2.1), hereafter nodes, are wireless network devices that freely can change their position over time. A wireless link, hereafter link, is an arbitrary direct connection between two mobile nodes. The nodes' possible random movements, combined with their appearances and disappearances affects the **network topology**, the configuration of connections between nodes in a network.

Wireless communication implies that nodes may have limited range and constrained bandwidth. This means that destinations may be out of range, and that nodes may have to relay data traffic. The data traffic may be relayed multiple **hops** before it reaches the destination. A hop means an intermediate connection in a chain of links, and a chain of links is also known as a **path**.

These facts place two fundamental requirements on the **routing protocol**, the mechanism used to provide each node with topological information allowing it to calculate the appropriate paths over which data is transmitted. First, the calculation must be decentralised, since there is most likely no single node connected to all other nodes. Second, since the topology changes can occur frequently, nodes must be self-configuring and the routing protocol must compute multiple, loop-free routes while keeping the **overheads** to a minimum. Overhead means any resource, such as computation time, memory or bandwidth, that is utilised by a routing protocol.

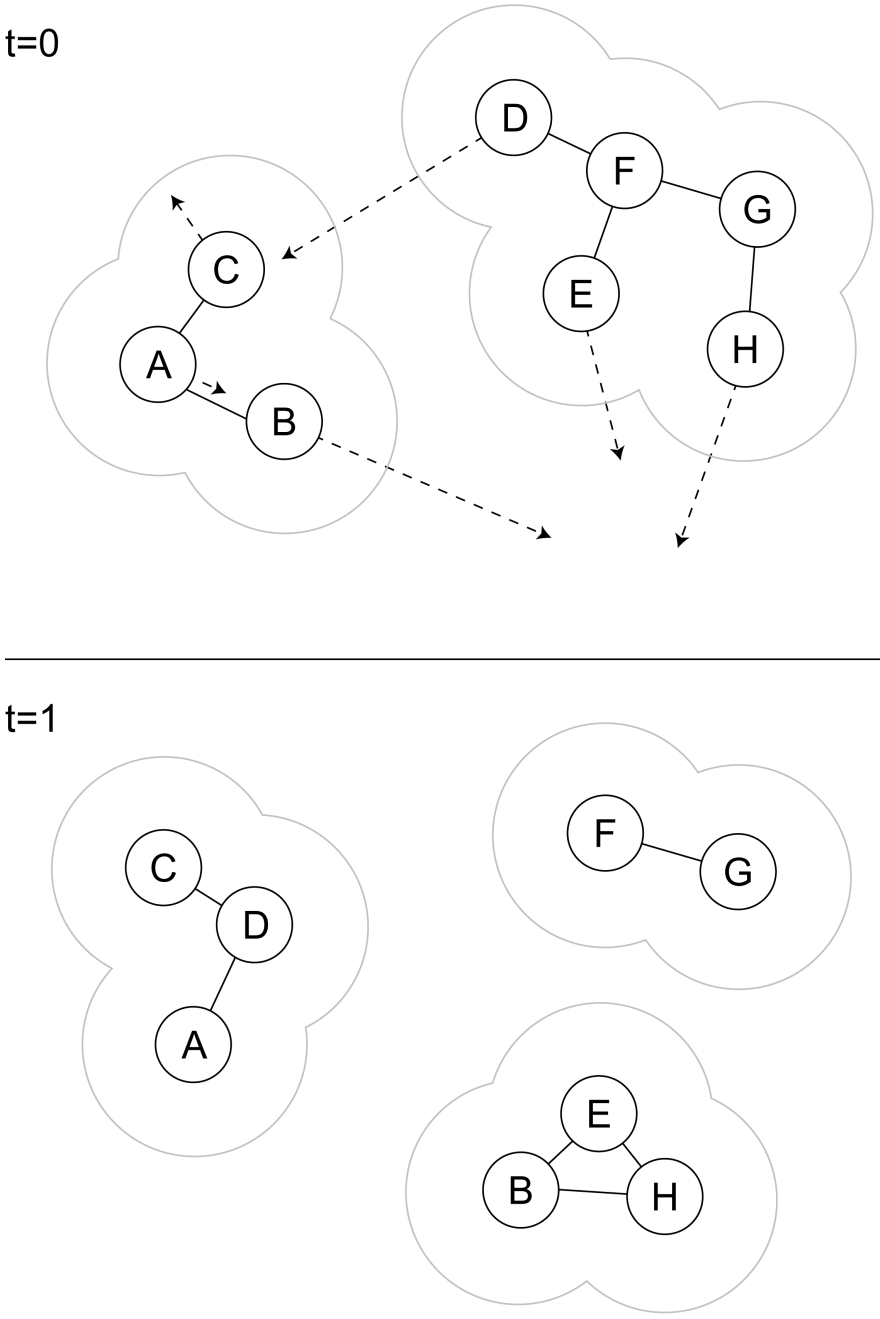


Figure 2.1: Mobile nodes can change their position randomly and at any time, creating different topologies and networks. In this example, at t=0, there are two MANETs, formed by nodes A, B and C and nodes D, E ,F, G and H, indicated by the lines between them. The dashed lines with arrows indicate the movements the nodes are about to make. At t=1 the original two networks have split into three, all having new topologies.

2.2 Wireless networking

A wireless network is any network where there exist no physical wired connection between the nodes, but where the nodes are connected using radio waves. There exists two types of wireless networks, differentiated by how many hops there are along the paths in the network.

2.2.1 Single hop networks

In the single node architecture connectivity is provided by access points (see figure 2.2). All communication goes through the access point, acting as a router. This is how the ever popular 802.11 access points, as well as cellular phone systems function. The access point determines the range and size of the network, and may also act as a static backbone to other networks.

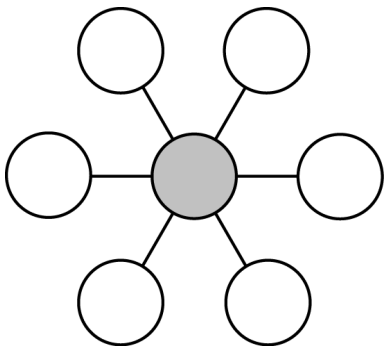


Figure 2.2: An example of a single hop network, where the lines indicate links. All nodes relay traffic through the access point (indicated in grey). Even if they may be within range of each other, they do not communicate directly.

2.2.2 Multi hop ad-hoc networks

In multi hop ad-hoc networks all nodes act as routers. There is no common access point, and thus no reliance on any fixed infrastructure. Since all nodes may not be within radio range of all other nodes, destinations may be multiple hops away, meaning that traffic must be forwarded by intermediate nodes if it is to reach the destination. This implies that every node is as important as the next, as each node may be an intermediate hop on a path towards the destination. Figure 2.3 shows such a network. The range of multi hop ad-hoc networks is typically larger than single hop networks, as every participating node expands the coverage.

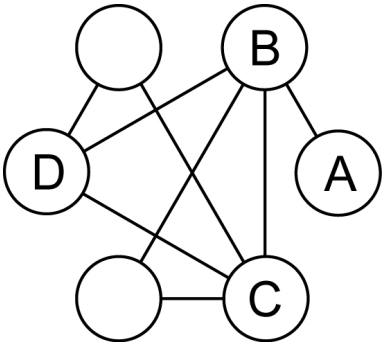


Figure 2.3: An example of an ad-hoc network with an arbitrary topology, where the lines indicate links. All nodes may act as routers, forwarding traffic. In this example, traffic from node A to D can be forwarded by B or through other paths such as through B and C.

2.3 Application domain

Applications for MANETs are many and varied. The classic example is crisis areas where the infrastructure, such as cellular towers, power supplies and fibre optic networks, has been fully or partially destroyed, and communication links has to be set up quickly and without central coordination. Such scenarios may include many participants relocating at will. These characteristics suit MANETs, with their ability to allow for rapid changing topologies and number of participants.

MANETs may also find use in military operations for closed, in-team communication and in areas with little or no centralised and organised connectivity.

Another industry example is **wireless sensor networks (WSN)**, where spatially distributed devices using sensors monitor physical or environmental conditions such as sound, vibration, temperature and pressure. Originally motivated by military operations such as battlefield surveillance, WSN has found its way into civilian application areas as environment surveillance and traffic control.

MANETs may however also be used in so-called hybrid networks, where parts of the network are fixed. For example, in urban areas, the range of hotspots could greatly be expanded without costly pulling of fibres etc., by deploying MANET nodes. These nodes would then allow users out of hotspot range to get connected by relaying traffic.

2.4 Routing protocols

The purpose of routing protocols is to dynamically communicate information about paths between nodes in a network. When several paths lead

to the same destination, **metrics** are used to determine whether one path should perform better than another. Metrics can express any values, such as bandwidth, hop count, load, delay, reliability and communication cost. The simplest is hop count, implying that the transmission cost over each link is 1.

2.4.1 Distance-vector routing protocols

Distance-vector routing protocols determine the best path on how far away the destination is, choosing the path with the lowest "distance", which can, again in the simplest case, be as simple as an aggregation of hop counts. Each node only knows the next hop and distance to its destination, the so-called distance-vector. This differs from link-state routing, where each node knows the full topology.

How it works

At first, a node searches for other nodes within range. It then learns who its **neighbours** are, that is, nodes to which it has direct links. Using metrics, it assigns a distance to each of its neighbours. Then, each node sends its distance-vectors to its neighbours, and this is done every time a change is detected in the distance-vectors. The neighbours compare these distance-vector with their own, possible updating their own if they find better paths, i.e. paths with lower distance. This process continues until no more updates are detected when processing the distance-vector of neighbours. The network is said to have converged, and no more messages are sent until a change is detected. The most commonly used algorithm for this task is the Bellmann-Ford algorithm [12]. Over time, all nodes discover the best hop towards all destinations. If a node becomes unavailable, its links will remove it from their distance-vectors. On the following updates, this information will propagate and new paths will eventually be found to all nodes still reachable.

The collection of distance-vectors make up the **routing table** on a node, in form of destination, distance, next hop entries, and this information is used when relaying normal data traffic.

Pros and cons

Distance-vector routing protocols are simple and efficient in small networks, and require little, if any management. However, they do not scale well, since each node sends all its distance-vectors on each update, creating a significant communication overhead in larger networks. These protocols may also suffer from convergence issues, such as the classic count-to-infinity-problem (see appendix B), which is a direct result of the asynchronous announcement

scheme. These factors, including the fact that distance-vector routing protocols only use local information, such as distance, and not path, have led to the development of more complex, but more scalable link-state routing protocols for use in large networks.

2.4.2 Link-state routing protocols

The basic concept of link-state routing is that every node acquire **topology maps**, containing partial topologies. These maps are used to calculate routes to all destinations. The status of the links is updated at all times, hence the name link-state. This approach differs from distance-vector routing, where all destinations are sent, but to neighbours only.

How it works

After searching for nodes within range, every node floods the network with a topology map, listing their neighbours only. This information is assembled on every node and used to create a **graph**, showing the full topology of the network. Each node then independently calculates, generally using a shortest path algorithm such as Dijkstra's algorithm [12], the paths to every other node. The resulting part is finally stored in the routing table.

Pros and cons

The advantage of link-state routing protocols is that they react more quickly to topology changes [7], making them more fit for dynamic environments. The communication overhead is generally smaller than for distance-vector routing protocols [7], since only information about the neighbours are transmitted, while the entire routing table is sent in distance-vector routing protocols.

The main disadvantage of link-state routing protocols is the continuous calculations and memory burdens. These may be too heavy, especially for small devices.

2.4.3 The MANET approach

MANET routing protocols differ from wired routing protocols, mainly due to the transmission particularities as limited range, restrained capacity and more frequent packet collisions. The MANET Working Group (see appendix C) classifies two types of MANET routing protocols, a classification perpendicular to the classic classifications of distance-vector and link-state routing protocols. Based on route discovery time, MANET protocols are either proactive and reactive. There is also a third category, known as hybrid routing protocols, which tries to encompass the advantages of the two former.

Proactive

Proactive or table-driven protocols are routing protocols which are updated with the state of the network at all times. The advantage of such protocols is that there will be no delay when a route is needed, since all routes already are known to all nodes. The communication overhead is also constant, but smaller than for reactive ones except in special cases [7].

The Optimized Link-State Routing Protocol (OLSR) [6] is an example of a proactive link-state protocol, which uses partial topology maps and optimised flooding, while the Destination-Sequenced Distance-Vector routing (DSDV) [11] is an example of a proactive distance-vector protocol. OLSR will be further discussed in chapter 3.

Reactive

Reactive or on-demand protocols will try to find a route to a destination only when required, i.e. when a node has a packet to deliver to a given destination. The memory and processing overhead is smaller than for proactive protocols, as there is less state to maintain per node. The communication overhead is traffic and mobility dependent, and will only in special cases [7] be smaller than proactive protocols. Another drawback is the delay before a link to the destination is established, as the route must be discovered first.

The Ad-hoc On-demand Distance-vector (AODV) [10] is an example of a reactive distance-vector protocol.

Hybrid

Hybrid protocols are, as the name implies, a mixture of proactive and reactive protocols. The idea is to take the advantages from both types of protocols to create a fusion which works well in different situations.

An example of such a protocol is the Zone Routing Protocol [8]. The basic idea is to split the network into different zones. Within a zone, a proactive protocol is used while a reactive protocol is used between zones. This means that a node will know the topology of its imminent neighbourhood while not having to know the topology of the whole network, thus taking the favourable sides of both proactive and reactive routing protocols. The protocol is modular in the way that any proactive and reactive protocols can be used. However, since hierarchical routing is used, the path to a destination may be suboptimal. While the research community is active within the area of hybrid protocols, this class of protocols is less mature and, thus, not currently candidates for standardisation.

2.5 Summary

Routing protocols have classically been described as either distance-vector or link-state. MANETs introduce the perpendicular classes proactive and reactive protocols. Using these classifications, the existing MANET protocols under development can be classifieds as shown in table 2.1.

The next chapter will detail one of these routing protocols: OLSR.

	Distance-vector	Link-state
Proactive	DSDV	OLSR
Reactive	AODV	DSR

Table 2.1: Table showing the relationship between distance-vector and link-state routing protocols on one side, and proactive and reactive protocols on the other side with examples of protocols.

Chapter 3

OLSR

This chapter discusses the Optimized Link-State Routing Protocol (OLSR), by first giving an overview in section 3.1. OLSR has three main characteristics which will each be discussed consecutively. There is the neighbourhood discovery in section 3.2, then the multipoint relay flooding in section 3.3 and finally the topology diffusion in section 3.4. Then follows a closer look at how the protocol is constructed and how the different components work in section 3.5. Section 3.6 looks at differences between the original OLSR and the new version under development.

3.1 Overview

OLSR [6] is a proactive link-state routing protocol used in MANETs. Being a link-state protocol, each node continuously collects control traffic diffused in the network by other nodes, in order to calculate a routing table.

OLSR can be described as three functional components, which will be further explained in the following sections.

- The neighbourhood discovery, the process of looking for nodes within range.
- Multipoint relay (MPR) flooding, a technique used such that every node can receive a given message in an optimised way.
- Link-state diffusion, the process of distributing topology maps used shortest paths calculations.

3.2 Neighbourhood discovery

Neighbourhood discovery is a continuous process, whereby nodes are determining the set of nodes which are within transmission range. This is done

through each node periodically transmitting HELLO messages. The purpose of these messages is to inform potential receivers about the node and its neighbours, populating the receivers' one and two hop neighbourhoods. The **one hop neighbourhood** is the set of nodes one hop away, while the **two hop neighbourhood** is the set of nodes two hops away.

3.2.1 One hop neighbourhood population

Following figure 3.1, once a node B receives a HELLO message from node A, it will add A to its one hop neighbourhood, which is a list of addresses along with timestamps telling when entries expire. This is illustrated at t=0. At t=1, B broadcasts a HELLO message, telling that A is an **asymmetric link**, meaning that B does not know if A can receive its messages. When A receives this message, it knows that there is a **symmetric link** between the two nodes, since messages have been sent and received in both directions. At t=2, A will announce this information. When B receives this message, it will update the status of its link to A to symmetric, broadcasting this information in the next HELLO message at t=3.

3.2.2 Two hop neighbourhood population

Continuing with the example in figure 3.1, assume that a new node appears next to node B (see figure 3.1). At t=4, node B receives a its HELLO message. The next HELLO message from node B will thus include information about the symmetric link to node A and the asymmetric link to C, as shown at t=5. Node A does regard C to be a two hop neighbour at this point, as the link between B and C is asymmetric. At t=6, node C confirms its symmetric link to node B. At t=7, node A receives a HELLO message from B, with a confirmation that node B has a symmetric link to node C. Node A will now add node C to its two hop neighbourhood, which along with the one hope neighbourhood makes the basis for the multipoint relay flooding explained in the next section.

3.3 Multipoint relay flooding

OLSR uses a technique called **multipoint relay (MPR) flooding** (figure 3.3) to diffuse partial maps of the topology. This is an optimisation over classic flooding, where information propagates outwards from the originating node (figure 3.2) by being relayed once by each node receiving the information. The idea of MPR flooding is to only let certain nodes perform the relaying. The advantage compared to classic flooding is clear [9], since it generates less transmission and thus fewer collisions in a wireless network, while ensuring a network wide spread.

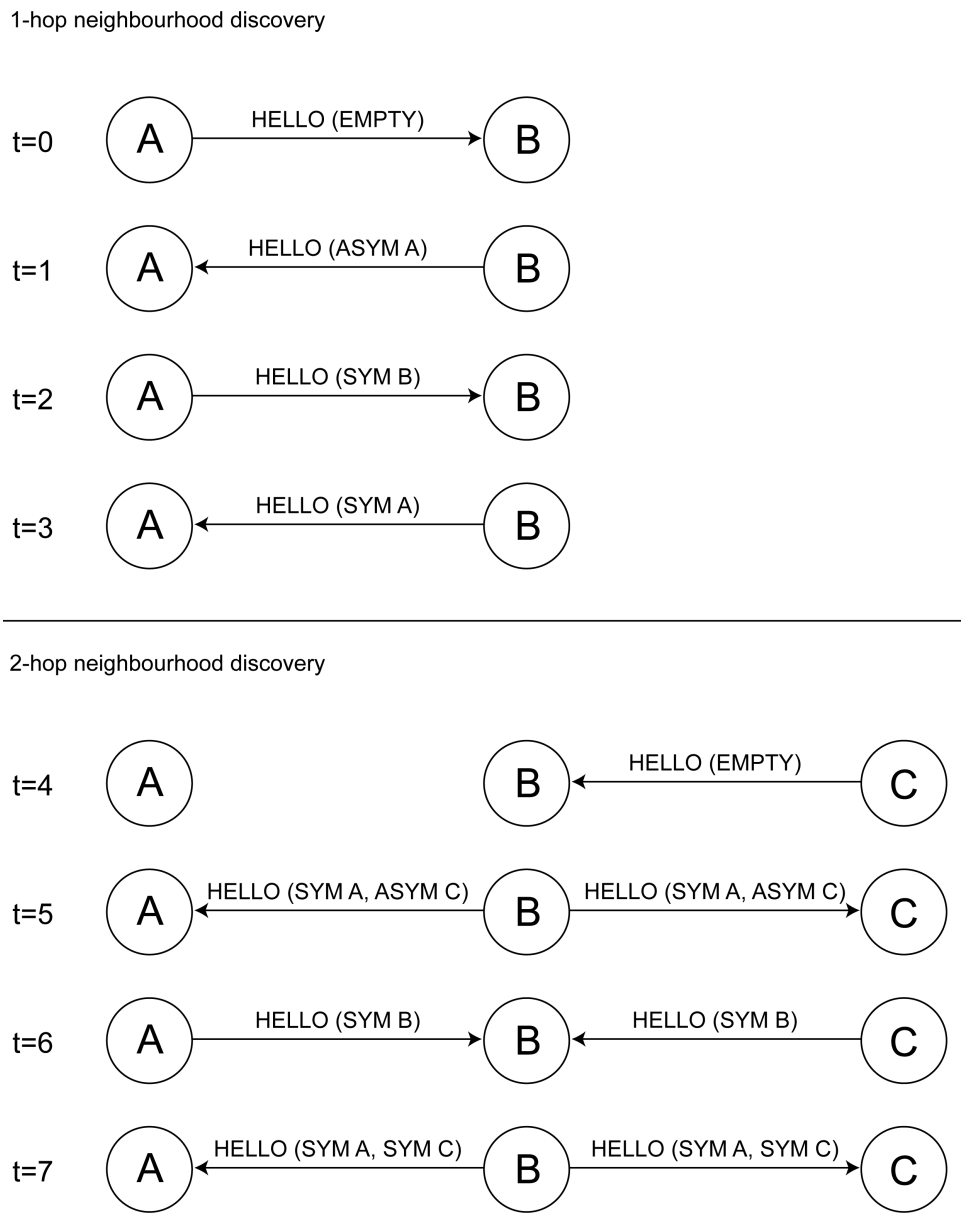


Figure 3.1: A typical example of one and two hop neighbourhood discovery. The arrows indicate broadcasts.

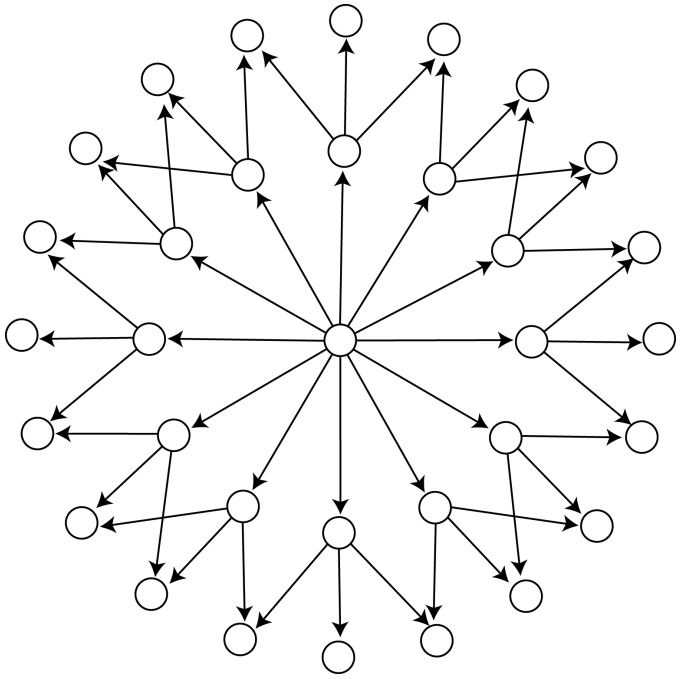


Figure 3.2: An example of classic flooding. The arrows indicate only the way control traffic is passed, not all transmissions. Every node in the network relays data outwards from the centre node.

A node does not choose to be an MPR itself, but is elected as such by its neighbours. By selecting a neighbour as MPR, the selector node asks the MPR to relay any broadcasted message received from the sender. The MPR selector also asks the MPR to advertise the link between them in **topology control (TC)** messages.

The purpose of the MPR selection procedure is to find those one hop neighbours that best cover all two hop neighbours. This is done to ensure that all neighbours two hops away will receive the information the node has gathered (see section 3.4). MPR selection is done as soon as a node has started populating its two hop neighbourhood, and is a continuous process that reflects the changes in the topology. For more about the one and two hop neighbourhood discovery procedures, see section 3.2.

3.4 Topology diffusion

A node selected as MPR has a special responsibility in OLSR: to diffuse link-state information describing the links between itself and the MPR selectors throughout the network. A node will do this through TC messages which by default are transmitted with fixed intervals. These messages, containing the

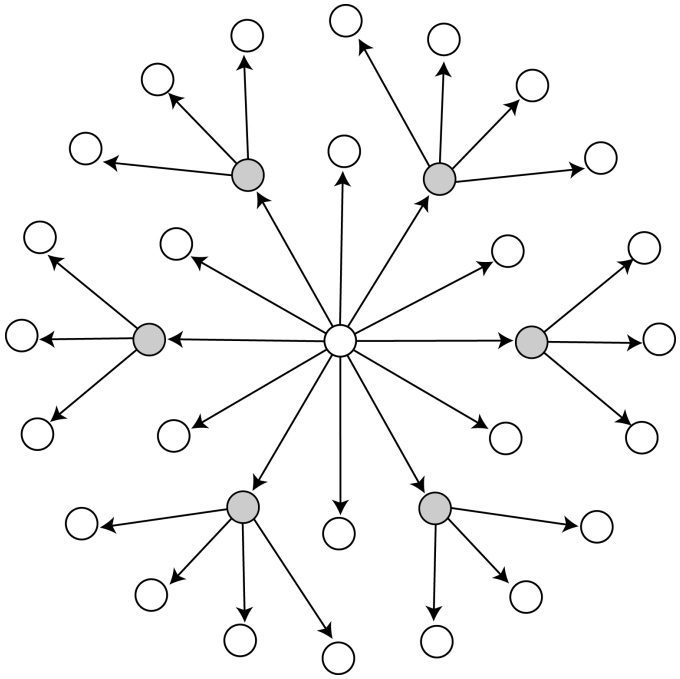


Figure 3.3: An example of MPR flooding. The arrows indicate only the way traffic control is passed, not all transmissions. The centre node selects nodes as MPR such that the all 2-hop neighbours are reached. By selecting carefully, the amount of communication overhead used for topology diffusion can severely be decreased. MPR nodes are depicted in grey.

addresses of the nodes that have selected it as MPR, and possibly also other nodes, are diffused throughout the network through other MPRs. With topology maps included in TC messages, each node can build a full topology graph. This information is then used to calculate the shortest paths as described in chapter 2. The results are recorded in the routing table. Then, the **Internet Protocol (IP)** [13] stack can use this information when doing forwarding of normal data traffic. An example of topology diffusion is given in figure 3.4.

3.5 Protocol anatomy

This section will give a detailed description of the anatomy of OLSRv2. The relationship between OLSR and OLSRv2 will be discussed in detail in section 3.6.

3.5.1 Overview

Looking at the protocol from an abstract point of view, there are two sorts of input and outputs in each node: HELLO and TC messages. Incoming messages are used to update the information repositories, from which outgoing messages are built. The relationship between the information repositories and messages in OLSRv2 is shown in figure 3.5. Finally, an optimised routing table is calculated from the information gathered.

3.5.2 Information repositories

The information repositories contain two subrepositories, the neighbourhood information base and the topology information base. Some of the sets include information about when entries should expire, the so-called **time-to-live** value. This value is important but also difficult to decide. If entries are kept too long, there is a risk of redundant information, and packets can get lost in the relaying process. On the other hand, if the time-to-live is too short, information will appear and disappear from the sets, even if the node in questions is healthy and alive.

Neighbourhood information base:

The link set contains all the neighbours that are within range, e.g. nodes that are 1-hop away. Information such as the neighbours' IP addresses are recorded along with the link status. The link status can be one of three; SYMMETRIC meaning a bidirectional link, HEARD meaning an unidirectional link or LOST denoting an expired link.

The symmetric neighbour set contains all 1-hop neighbours with link status SYMMETRIC.

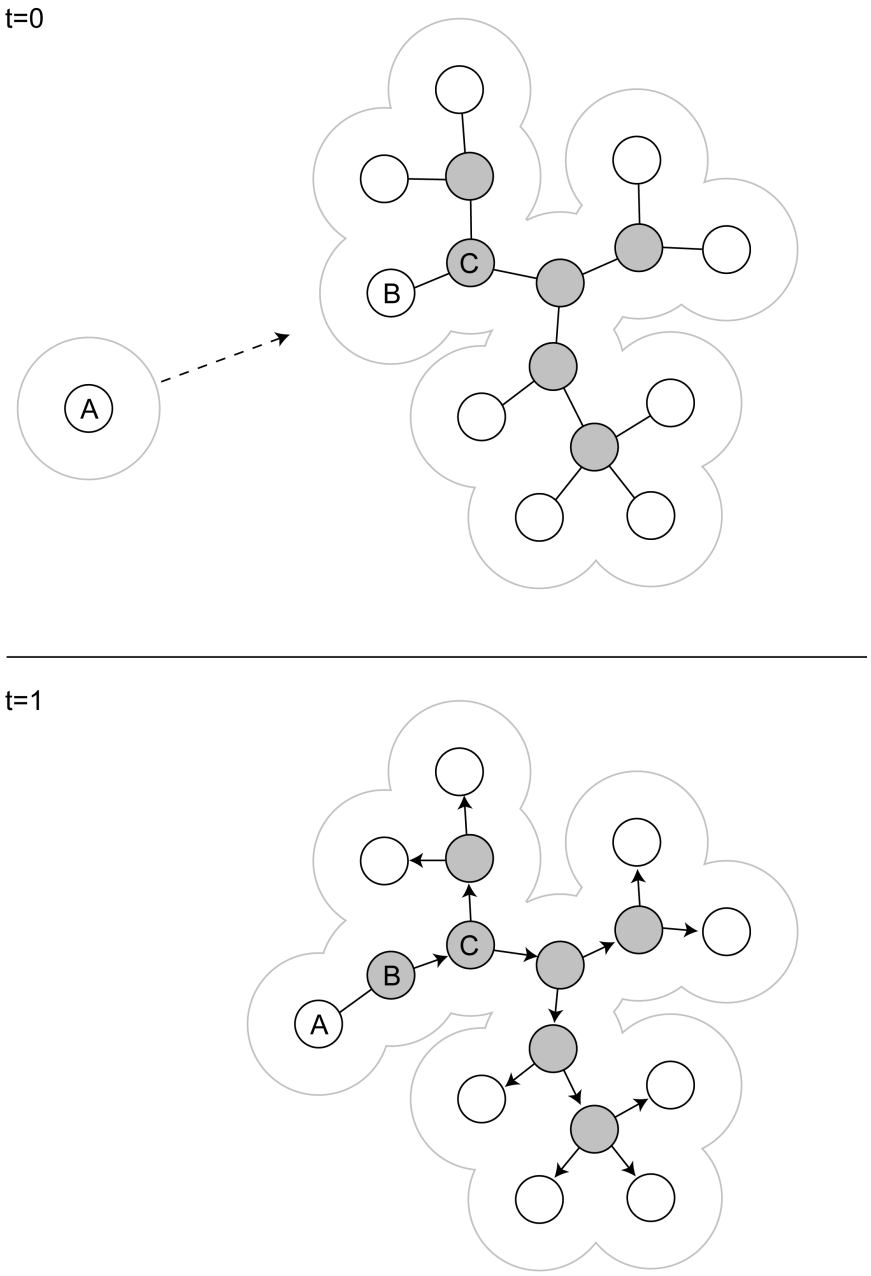


Figure 3.4: An example of topology diffusion. At t=0, node A moves towards the network, connecting to B. Node A chooses B as MPR, asking B to advertise the link between them. At t=1, this information is diffused through the MPRs in the network, depicted in grey. The arrows indicate the way the topology map are passed.

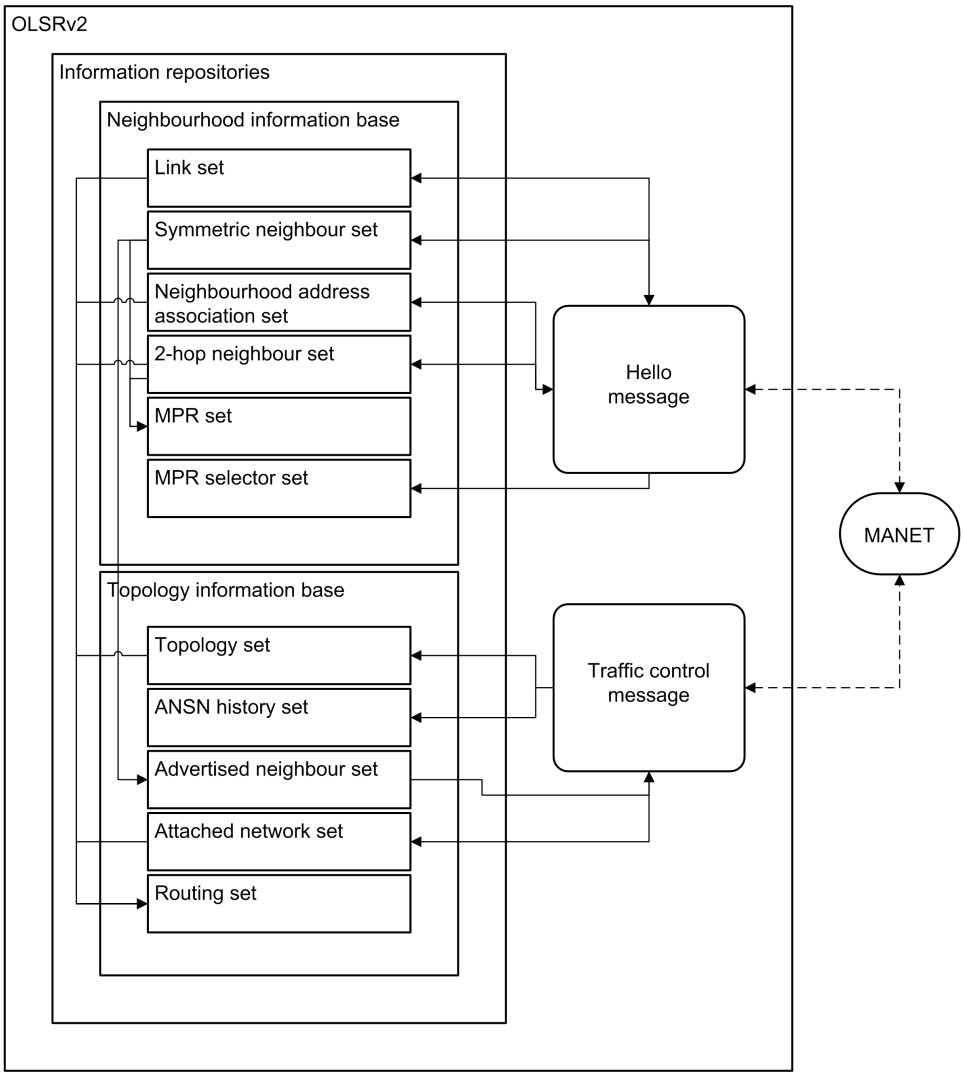


Figure 3.5: Figure showing how HELLO and TC messages are used to update the different repositories on a node running OLSRv2. The dashed lines indicate message transmissions between a node and the MANET. The solid lines indicate how the different messages and sets update one another. A single headed arrow indicate a one way update, while a double headed arrow indicate a two way update.

The neighbourhood address association set records information about the MANET interface configuration in the 1-hop neighbourhood.

The 2-hop neighbour set contains information about nodes 2-hops away. Only SYMMETRIC links are recorded.

The MPR set lists which nodes the node has selected as MPRs.

The MPR selector set lists which nodes have selected the node as MPR.

Topology information base:

The topology set contains the topology maps acquired from diffused TC messages. The maps are the basis for the full topology graph and shortest path calculation.

The advertised neighbour set sequence number (ANSN) history set records information about the freshness of the topology maps received.

The advertised neighbour set maintains a set of addresses of symmetric 1-hop neighbours, which are to be advertised through TC messages.

The attached network set maintains information about attached networks and through which nodes they can be accessed.

The routing set describes the selected path to each destination in the network for which a route is known.

3.6 OLSRv2 - What's new

The original version of OLSR [6] became an IETF Request for Comments (RFC) in October 2003. Provided the feedback and continuous work on the original protocol, a new version of OLSR has emerged, essentially retaining the fundamental algorithms (link-state, proactive, MPR flooding), while making several other improvements. At the time of writing, OLSRv2 is still an Internet-Draft, but is expected to be published as standards track RFC early 2007.

3.6.1 Protocol composition

The previous one protocol specification is now three independent documents. The main specification [5] is supplemented by a document [3] specifying a generalised MANET packet/message format and a protocol [4] specifying a MANET neighbourhood discovery.

3.6.2 Neighbourhood discovery

The process of neighbourhood discovery has been extracted from the main document and can thus easier be used as reference for other applications. The independent protocol [3] is expected to published as standards track RFC in 2007.

3.6.3 Packet format

The packet format used in OLSRv2 has been revamped and is now specified in an independent document [3] that may be used by other MANET protocols. The packet format now uses **time-value-length (TLV)** blocks for conveying additional information in messages. This implies that third party extensions can be developed and used within an OLSRv2 network, without affecting pure OLSRv2 nodes.

An optimisation has been implemented for listing addresses. Since parts of addresses on the same network often are common, they can efficiently be compressed, by listing the common parts only once. As an example, the IP addresses `[10.1.2.48], [10.2.3.48], [10.22.4.48]` can be compressed to `[10],[1.2,2.3,22.4],[48]`, later being rearranged to the original addresses, resulting in less communication overhead.

3.6.4 Messages

The five different message types in OLSR have been cut down to two. They now also have the same structure of address blocks and tlv blocks, leaving a single parser for all OLSRv2 messages. This means that there is a cut in parser code size by approximately 33%. Still, these two message types are able to do the same tasks previously done by five.

3.6.5 Multiple interfaces

Multiple interface support is changed. Previously, multiple interface declaration (MID) messages were used. These are no longer needed as all interfaces are listed in all packets. Even though this may increase the size of individual packets, it leads to a decreased chance of collisions and lost packets since fewer packets are sent.

3.6.6 Gateway handling

The **host and network association (HNA) messages** from OLSR are eliminated. HNA messages were used to inform about **gateways**, access points to external networks such as the Internet. The information contained in HNA messages is now found in TC messages, and this is possible due to the use of TLVs including the same information. This change makes the

protocol more efficient, as it generates less control traffic while doing the same job. TC messages has sequence numbers, allowing explicit expiration of information. HNA information was non-expirable, possibly making gateway information redundant. The integration of TC and HNA messages means that gateway information can be expired.

3.6.7 Processing and forwarding

Processing and forwarding of incoming packets are now cleanly separated in the specification, making implementation easier and improves code readability. Essentially, processing and forwarding are two entirely independent processes, are specified as such, and can be implemented as such.

3.6.8 Extensibility

The independent specification of the message exchange format [3], provides a way to allow for third-party applications to be developed and used within an OLSRv2 network, without interfering with the normal use. Through the use of **internal** and **external** extensibility, third-party extensions are able to let normal nodes diffuse their information. Internal extensibility implies that information is injected using the TLV mechanism, while external extensibility implies that a protocol extension may specify and exchange new message types, which can be forwarded and delivered correctly. OLSR only allowed for external extensibility.

Examples of internal extensibility is adding checksums to messages, to protect the integrity of the data by detecting errors, or by signing messages to authenticate the sender. External extensibility can for example be messages diffused in the network for quality of service surveillance, measuring the delay from sender to destination.

3.7 Summary

From the MANET protocols introduction in chapter 2, this chapter has detailed one - OLSR. Existing in two version, OLSR [6], a standards track RFC since 2003 and OLSRv2 currently undergoing standardisation, this chapter has extracted and presented the common core algorithm functions to both versions, as well as described how OLSRv2 has evolved from OLSR.

The next chapter will look at the implementation work of OLSRv2.

Chapter 4

Implementation and Testing

Based on the specification of OLSR, which was analysed in chapter 3, this chapter will

1. set up the requirements to an implementation,
2. justify, based on the requirements, choice of programming language and implementation model, as well as present the code design,
3. present the test-bed and the testing conducted on OLSRv2.

4.1 Requirements

Other than implementing OLSRv2, the factual requirements are:

Compliance to the specifications. The implementation should function as a live demonstration of the work being done on the specifications, acting as a reference implementation.

Easy-to-read code is a priority, since the specifications work will not completed before the code is to be delivered. As a result, other students should be able to continue, implementing the changes between the Internet-Drafts and the final RFCs.

Platform independence , as it allows the same implementation being tested on numerous devices without changing the code or doing recompilations.

Develop my programming skills - this is a pedagogical requirement, as I knew little about network programming before starting.

With the OLSRv2 specifications still being Internet-Drafts, the implementation process would be gradual. Each time a new draft emerges, the code

should be altered to apply to the modifications. More importantly, this allows for direct feedback from the implementation experiences to be used and taken into account for improving the specifications.

4.2 Java

Java is a well known and powerful object oriented programming language, developed by Sun Microsystems since the early 1990s. Java is designed to be not only cross platform in source form, but also in compiled form. This is done through the use of intermediate byte code, which is interpreted by the **Java Virtual Machine (JVM)**. Java quickly became popular, much due to its simplicity compared to other languages (e.g. implicit garbage collection and memory management), and is used today in a variety of platforms spanning from embedded devices and cellular phones on the low end to enterprise servers and super computers on the high end. Another interesting aspect of Java is its ubiquity on mobile phones, which makes it the de facto industry standard when developing software on such devices. Java also comes with **Javadoc** for generating API documentation from source code.

Java comes in three different flavours, so called Java Runtime Environments (JRE), which are each implemented with a special kind of platform in mind.

Java ME (Micro Edition) is a limited implementation for small devices such as mobile phones and PDAs, considering their restricted capacity both in storage and processor power.

Java SE (Standard Edition) is a general purpose implementation for PCs, servers and similar devices.

Java EE (Enterprise Edition) is Java SE, with special purpose APIs for multi-tier client-server applications.

As a result of Java's many and strong features with the requirements in section 4.1 in mind, the latest incarnation of the standard edition, version 5.0, was chosen for this project.

4.2.1 Challenges

By using intermediate code, Java should in theory be "write once run everywhere", and this is mostly the case. However, life is not always that simple. Even though Java has become more powerful over the years in terms of accessing system functions, the current version (5.0) cannot alter the systems' routing tables. This being a crucial part of any routing protocol, a workaround was required through the use of native code, written in C++. This

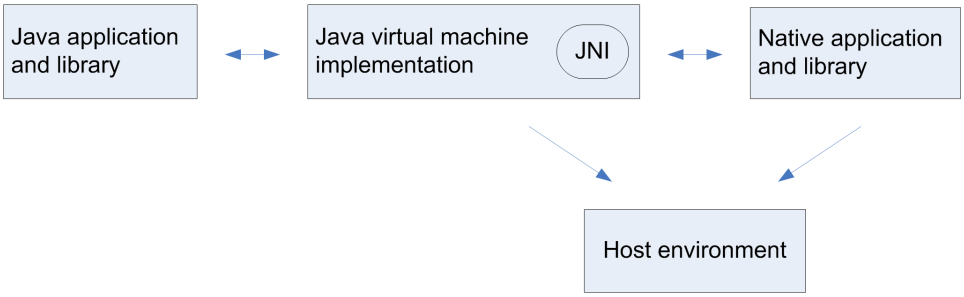


Figure 4.1: The Java Native Interface serves as the glue between Java and native applications, allowing Java to invoke native code and vice versa.

means that for each platform the code is to run on, a set of native methods performing the task of altering the routing tables must be developed. Hopefully, future version of Java will address this limitation, but for the time being it is an adequate workaround.

One way native code can be used within Java applications is through the use of the **Java Native Interface**. JNI lets Java code use code and code libraries written in other languages, such as C or C++, and allows for calling Java code from within native code (figure 4.1).

4.2.2 Logging

The implementation make use of the Java Logging APIs, meaning that log reports can be recorded sequentially and in chronological order. The log reports can include from nothing at all to full and detailed information about received and sent packets or system failures. Information about the different repositories are also provided at will, and the information can easily be output to screen or written to a file. The level of logging can be changed at runtime, which means that no recompilation of the code is necessary. Apart from being a tool for seeing what is going on behind the scenes, the logging framework also serves as a tool for debugging the code.

4.3 Algorithms and data structures

The specification [5] does not put any constraints on the data structure nor algorithms used in the calculations on the information repositories.

Regarding the use of algorithms, there are two main challenges in OLSRv2.

The first one is the calculation of MPRs, which is a variant of the NP-complete set cover problem, in which each node tries to find the set of one hop neighbours that best covers all two hop neighbours. A greedy heuristic is implemented as suggested, where one hop neighbours are chosen by the

number of two hop neighbours they cover, choosing the one hop neighbours covering the most two hop neighbours first.

The second challenge is packing the messages as small as possible. A message may contain any number of address blocks and TLV blocks, both having contents that may be arranged in a number of ways. This is a variant of the bin packing problem, which is NP-hard. A variant of the suggested heuristic is implemented, where all addresses in a HELLO message other than those in the local information base is put in one single address block, sorted by link status. In TC messages, all advertised neighbours are put in one address block while gateways are put in another.

The packet format specification [3] used in OLSRv2 suggests the use of address compression in address blocks. IP addresses on the same network often have common parts, meaning that they efficiently can be compressed by listing the common parts only once. By splitting addresses into head, middle and tail parts, there are essentially four ways to categorise a given set of addresses:

- common head, individual middle, common tail (`[a.b.b.c]`, `[a.d.d.c]`)
- common head, individual middle, no tail (`[a.b.b.b]`, `[a.c.c.c]`)
- no head, individual middle, common tail (`[a.a.a.b]`, `[c.c.c.b]`)
- no head, individual middle, no tail (`[a.a.a.a]`, `[b.b.b.b]`)

The implementation finds where to make the splits, by searching for differences from both sides of the addresses. The head is found by looking at the first bytes of all addresses, advancing one step to the right until a difference is found. The tail is found in the opposite way, starting with the last bytes of all addresses, working its way to the left while comparing until a difference has been detected. This is illustrated with an example in figure 4.2.

The data structures are implemented in the most simple way, using dynamic arrays (`ArrayList`). The elements are listed consecutively, and there is no sorting of elements.

It is important to note that these solutions are chosen deliberately, as there is no functional requirement to performance. The priority was to make the implementation work, to demonstrate and validate the protocol's fundamental mechanisms. Code optimisation is listed as future work (chapter 6).

4.4 Code structure

While implementing the protocol, the specifications were followed closely for several reasons. Firstly, the implementation served to verify if the proposed structure in the specifications easily could be transformed into well

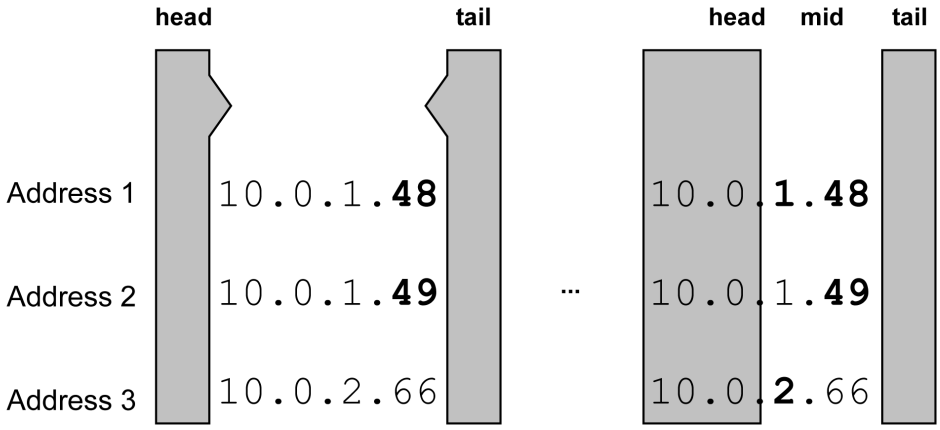


Figure 4.2: Figure showing how the head, middle and tail are found in a set of addresses. In this example, the search for the tail is stopped right away as the last bytes of the two first addresses differ. The head is found to be the two first bytes of the addresses, as the third bytes of address 1 and 3 differ. This means that the addresses can be packet as [10.0],[1.48,1.40,2.66], reducing communication overhead.

structured object oriented code. Secondly, issues not envisioned while writing the specifications could be highlighted when implemented. Thirdly, a close tie between specification and code is good in an educational sense, as it makes the code easier to follow. Such a tie could also be an indicator of specification readability and consistency.

As the complete system makes a rather complex figure, only a part of it is shown in figure 4.3, where focus has been put on showing the main components, excluding the information repositories. The anatomy of the information repositories, which is identical in code form, can be found in figure 3.5. This section will present some of the implementation's main components.

4.4.1 Node

The node class is the main object, and is the point of departure when running the implementation. A node can have one or multiple interface class, each of which has a sender and receiver class for packet transmissions. Each node has an originator address, which is used by the interfaces to tell that all control traffic is coming from the same node. The main threads also runs on the node (figure 4.4.1), alternating between purging the repositories, generating messages and processing incoming parsed packets on each Interface.

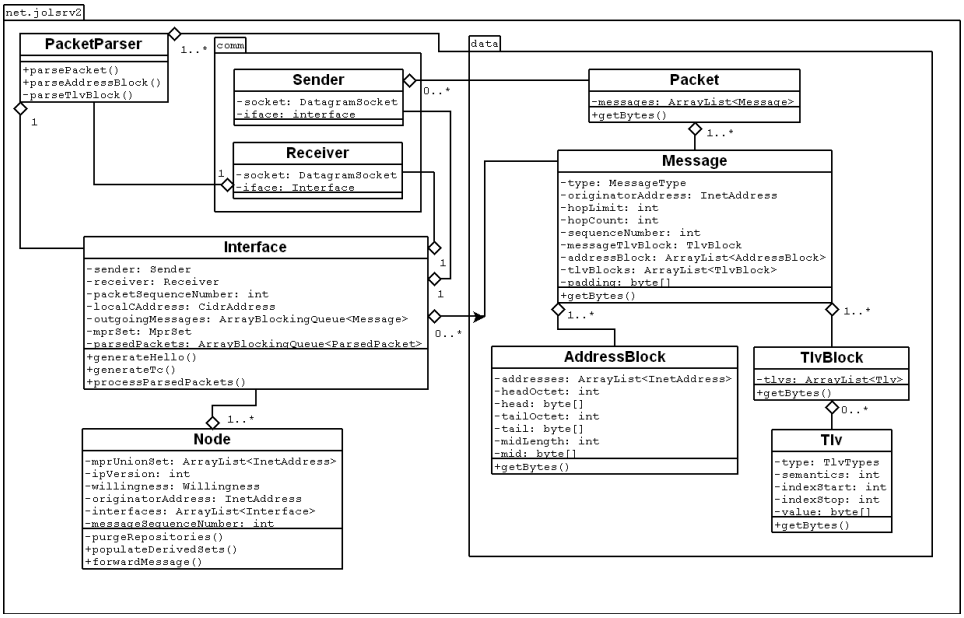


Figure 4.3: UML class diagram showing parts of the system

The HELLO flag is set to true every 2 seconds, while the TC flag is set to true every 5 seconds, as proposed in the specifications [5]. This means that the greater part of the time is used to keep the repositories fresh, either by removing expired entries, or adding or updating entries based on the information found in received messages.

The purging of the repositories goes through every repository, looking for expired entries. If old entries are found, they are deleted, possibly resulting in recalculations of MPRs and routing tables. The structure of the information repositories and their interaction with messages can be found in figure 3.5.

4.4.2 Interface

Each interface class has a local address used to separate interfaces from the same node. One of these addresses are chosen as originator address in the node. Messages are created periodically on each interface by the generate HELLO and generate TC methods. All messages include the local information block as the first address block, which includes all interface addresses on a given node. The messages are then put in the outgoing messages queue on the respective interfaces.

The generate HELLO method gathers its neighbours found in the link set, and sorts them by link status (which can either be SYMMETRIC, HEARD or LOST). It then packs these links in an address block with appurtenant TLVs indicating link status and information about links chosen

```
purgeRepositories();

if(flagHello){
  for (Interface i : interfaces)
    i.generateHello();
  flagHello = false;
}

if(flagTc){
  for (Interface i : interfaces)
    i.generateTc();
  flagTc = false;
}

for (Interface i : interfaces)
  i.processParsedPackets();
```

Figure 4.4: Figure showing the code running in the main loop. The code alternates between sending messages and purging and updating the repositories.

as MPR.

The generate TC method enquires the advertised neighbour set for addresses to include. It also includes gateway addresses if applicable.

Sender

The sender class will try to pack as many messages as possible into the same packet before transmitting. As the sender is scheduled to create and send packets with a fixed interval, the message generating methods may have had time to produce several messages, saving communication overhead.

Receiver

Incoming packets are received on each interface by the receiver class. It saves the incoming bytes and asks the packet parser class to parse them. If everything goes well, a parsed packet will be delivered to the interface receiving it, put in the parsed packets queue, ready for processing.

4.4.3 Packet

The variables of packets class are as follows:

```
private static final int zero = 0;
private static final int reserved = 0;
private int sequenceNumber;
private ArrayList<Message> messages;
```

Each packet includes thus one or more messages and a sequence number used to determine its freshness. The `zero` and `reserved` are special variables set to 0 as specified in [5]. In the specifications, external packets may include a packet TLV block, as shown in figure 4.4.3, but currently there is no use of packet TLV in OLSRv2, and are therefore not supported in the implementation. When a packet is ready to be transmitted, a nested loop is run making the contents of each message, address block, TLV block etc. into a stream of bytes.

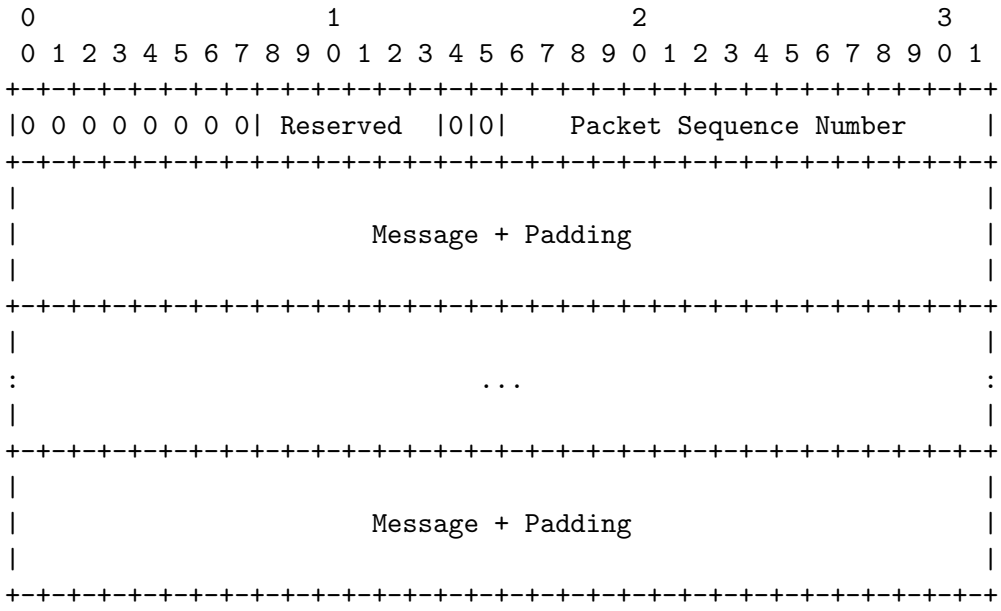
4.4.4 Message

The variables in the message class are as follows:

```
private MessageTypes type;
private int semantics;
private InetAddress originatorAddress;
private int hopLimit;
private int hopCount;
private int sequenceNumber;

private TlvBlock messageTlvBlock;
private ArrayList<AddressBlock> addressBlocks;
private ArrayList<TlvBlock> tlvBlocks;
```

The message class has a message type, which for OLSRv2 either can be HELLO or TC, and a message semantics which specifies the interpretation of the remainder of the message header. This means that a decision to process a message or not can be made simple by reading as far as this value. The originator address identifies the interface originating the message. Hop limit is the maximum number of hops the message is allowed to do, while hop count is the number of hops the message has been transmitted so far. The message also have a sequence number to determine its freshness. The fields described so far is the message header. The message body consists of a message TLV block and zero or more pairs of address blocks and TLV blocks. The blocks are collections of respectively addresses and TLVs. The complete message layout is shown in figure 4.4.4. The messages are 32-bit aligned by padding if necessary.



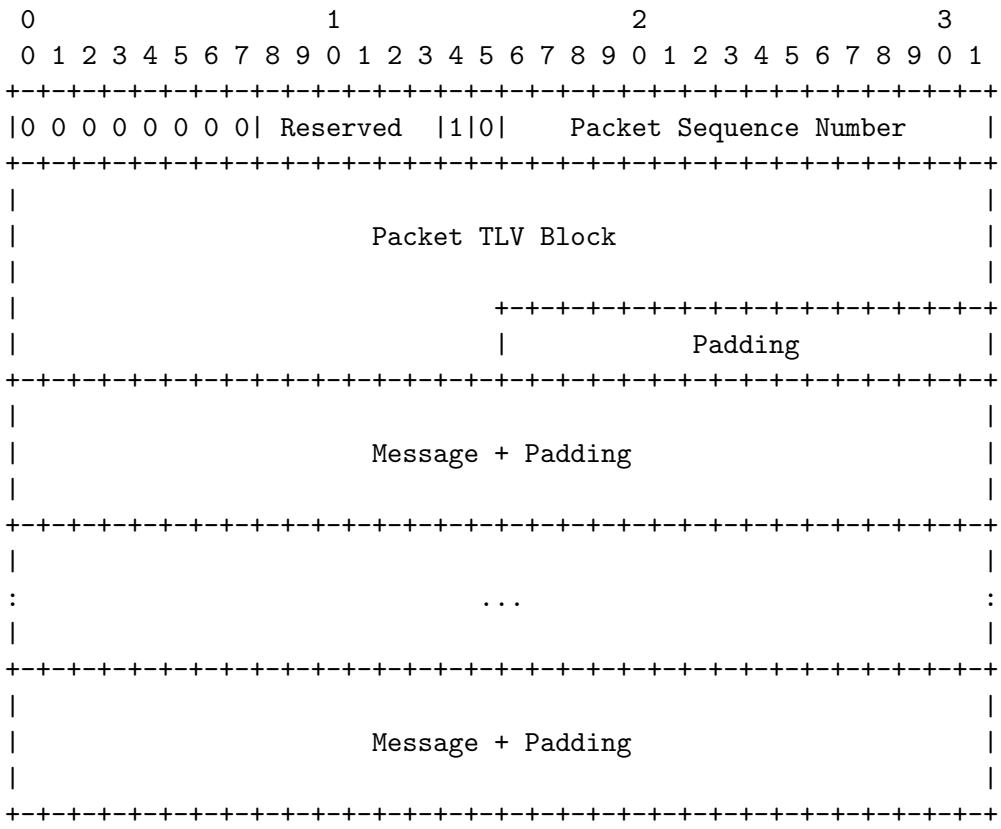


Figure 4.5: Figure showing the two alternative layouts of an OLSRv2 packet, the difference being the packet having a packet TLV block or not. For details about the different fields, see [3] [5].

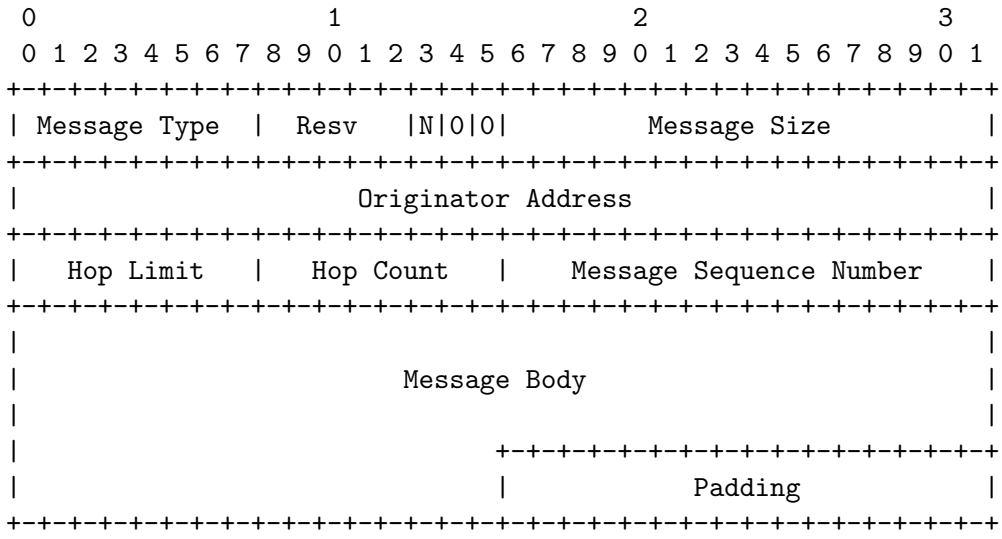


Figure 4.6: Figure showing the layout of an OLSRv2 message. For details about the different fields, see [3] [5].

4.5 Jitter

As discussed in [7], periodical emissions of control messages in a MANET may suffer from transient loss of routes to parts of the network. This can happen due to changes in the MPR set being announced at the same time from different nodes, leading to packet collisions. This synchronisation problem can be reduced by enforcing **jitter**, a random time value, making the messages either be sent before or after the schedule transmission. The jitter makes the messages less synchronised and more tolerant to getting dropped, leading to more stable routes.

The use of jitter is not required by the specifications for interoperability, but is nonetheless implemented with a random delay between -0.5 and 0 seconds on each Node.

4.6 Testing

The test-bed (figure 4.7) consists of four PCs and a laptop, all equipped with both Ethernet and wifi interfaces, as well as the latest version of Java (5.0). By using Secure Shell (SSH), all machines can be controlled from a single computer, creating logs for later analysis.

The set-up creates two fully connected networks, one on each of the interfaces. There are essentially two ways of creating other topologies. One is to simply enable/disable network interfaces. A more sophisticated way, which also allows for more complex topologies, is to filter incoming packets.

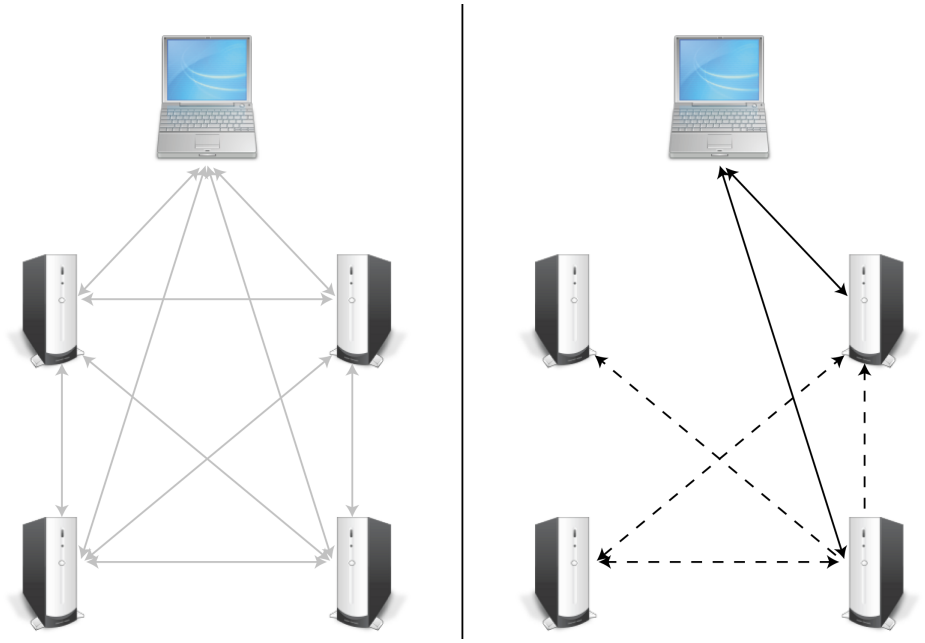


Figure 4.7: On the figure to the left, a schematic overview of the test-bed is shown. All machines are equipped with both wifi and Ethernet cards, creating fully connected networks on both interfaces, indicated by the grey two headed arrows. In the scenario to the right, the laptop is emulating a cellular tower through the Ethernet connection, indicated by the solid double headed arrows, while the PCs are running an ad-hoc network, indicated by the dashed arrows. Note that all the links in the ad-hoc network are not symmetric, indicated by the single headed arrows.

This is done by defining topologies in software, dropping packets as they arrive.

The test-bed is set up with two goals in mind. One is to allow iterative testing and debugging of the code while implementing the specifications. The value of having such equipment while developing is great, as it gives a direct indication to whether or not the implementation works and behaves as expected. This allows improving the quality of the code, making it more robust, experimenting with parameters such as mobility and willingness to relay traffic, making nodes handle packets with unexpected contents etc.

The second goal is to emulate the use of ad-hoc networks within cellular networks, as proposed in section 1.2. With all computers having two network interfaces, using the techniques described above to create different topologies, the scenarios found in section 1.2 can be emulated. One or more computers can emulate the cellular towers, while the others form an ad-hoc network. An example of such a scenario is shown in figure 4.7. These tests showed that such a system, from a technical point of view, can be developed.

4.7 Summary

This chapter has described the implementation of one of the MANET protocols proposed by the IETF: OLSRv2. A test-bed was set up for testing the code during the development, as well as to emulate an extended cellular network where ad-hoc networks are deployed.

The next chapter will look at the perspectives of this extension.

Chapter 5

Perspectives: Brokering Network Access

With an ad-hoc extension to cellular networks, the ad-hoc networking is assumed to be provided by the users themselves, through "lending resources of their handsets" to other, meaning that the functionality of the network depends heavily on users' co-operability. In a perfect world, everyone would relay traffic, making the network as robust and efficient as possible, and nobody would abuse the network provided by others by using excessive resources etc. This can be true for closed ad-hoc networks, such as military operations networks, or in networks where all users are working towards the same goal. However, in the consumer market, the scenario is quite different. One can no longer assume that users are willing to constantly share their connections for free, as the disadvantages of doing such are greater than the benefits. The users can be assumed to be selfish, as it is likely that they would want to send data without relaying themselves. These uncooperative nodes has a dramatic effect on the operation of the network. Thus, in order to motivate for bandwidth sharing and to administer such a system, brokering network access, i.e. ensuring security, billing, hand-over etc., should be considered.

5.1 Background

In the last years, mobile phones have not only been integrate with short range wireless interfaces such as Bluetooth, but also wifi technology such as wireless local area network (WLAN). This implies that a user can choose between deploying the ordinary cellular networks or wifi networks, or both.

In such multiple interface networks, the use of ad-hoc networking can be envisioned. Users out of reach of wifi access points or cellular towers, may reach them through intermediate nodes, e.g. other users and their handsets.

Both from the **network provider** and user point of view, this extension

has several advantages, but also introduces a new set of challenges, as the following sections will elaborate. In this context, a network provider is an organisation providing cellular networks and/or wifi access points.

5.2 Variables

An ad-hoc extended cellular network has several variables that needs to be considered when making network brokering decisions. This section will look closer at four of them: load distribution, cost distribution, coverage extension and quality of service, discussing their values from both the user and network provider's point of view. The main points of this discussion can be found in table 5.1. This is detailed further in the following sections.

Variable	Network provider		User	
	Pros	Cons	Pros	Cons
LD	Frees resources	Fewer users	Competition freedom	Locked devices
CD	Cheap	No device control	Device control	Expensive
CE	Broader audience	Uncontrolled coverage	Easier to connect	No QoS guaranty
QoS	Attract users	Infrastructure development	Higher bandwidth	No guaranty

Table 5.1: Table showing the advantages and disadvantages for both network providers and users regarding load distribution (LD), cost distribution (CD), coverage extension (CE) and quality of service (QoS).

5.2.1 Load distribution

With users employing their ad-hoc wifi connection to communicate, they may no longer need to use the network providers' equipment for their connections, opening for competition freedom in offering cheap hotspots or other access points. On the other hand, this may result in network providers locking their handsets to only allow for certain connections. For the network provider, this load distribution results in a decreased load, which frees resources allowing more cellular based connections. At the same time, fewer users may lead to decreased benefits.

5.2.2 Cost distribution

After the initial cost of developing and deploying such a system, the cost of growth will be on the user side, and not only in terms buying new handsets. Battery is depleted more quickly on small devices when enabling support for

wifi connections. Although battery capacity is an ongoing field of research, it has not kept pace with the rest of the industry. Processor power is consumed on processing other people's data, power that already may be limited on handsets. However, a user may choose between different types of mobiles, finding one with fitting its requirements to price, battery capacity etc.. The cost distribution also means that the network providers are not in control of the equipment used, possibly relying on underperforming nodes.

5.2.3 Coverage extension

For the users, coverage extension means that may be easier to connect to the network. However, this does not say anything about the quality of service once connected. Greater coverage through ad-hoc networking also means that network providers are able to reach more customers without the need of extra cellular towers or hotspots. However, since the nodes providing the extension are mobile, it is not possible to control the spatial distribution, possibly resulting in unbalanced topologies. Also, a larger coverage is harder to provide regarding quality of service (see section 5.2.4).

5.2.4 Quality of service

Current wifi technologies offer a greater bandwidth than that of cellular based networks, and could therefore be attractive to users demanding high speed connections. However, wifi connections have a nominal bandwidth, and make no guaranty to latency, dropped packets, errors and other quality of service issues. Ad-hoc networks, with their multi hop architecture, make the situation even worse. By sharing the bandwidth through relaying, the user's own data rate decreases as well. The network providers may advertise the higher bandwidth to attract more users, but may be forced to develop their infrastructure to guaranty a minimum quality of service.

5.3 Challenges

The main challenge for mobile operators is how to profit from customers using wifi enabled devices to communicate between each other in ad-hoc modus, escaping the need to utilise the mobile operator's equipment. There is little doubt that this is any mobile operator's worst nightmare technology, rendering them obsolete. However, an important aspect mentioned in the introduction is that it is not likely that users will share their wifi connections out of pure ideology. Thus, there is still a need for an organising party, which will be in charge of setting up, maintaining connections etc., as well as the cellular based network for traffic that does not fit ad-hoc networks, implying that brokering network access cannot be completely self-organised.

With the last section in mind, an approach to the problem is looking at the motivating factors, finding a balance between the advantages and disadvantages. The variables in table 5.1 can be treated as a case of cost-benefit analysis, maximising the advantages while keeping the disadvantages at a minimum, both for the mobile operators and users. The challenge is of course to find both qualitative and quantitative ways to measure these variables in terms of costs and benefits. The accuracy of the outcome is closely tied to the accuracy of the estimations, meaning that the estimations must be done with great care. Constructing plausible measures of the costs and benefits of variables is often both time-consuming and difficult, but a necessary mean to create an optimal system.

Once these factors have been decided, there is the questions of how network providers can make money in such a business model. The idea of users operating outside the network providers' jurisdiction is undoubtedly a controversial one, which requires a change of paradigm and a complete new way of thinking. For conservative network providers, these ideas may be compared to a revolution, threatening their position.

In [2] a system with virtual currency is proposed for stimulating users in an ad-hoc network to forward packets for each other. The idea of virtual currency is that each user must earn the right to have traffic forwarded by first forwarding for others, earning money, so-called "nuglets". The "nuglets" can later be used as a payment for getting packets forwarded. This may also stimulates users to a moderate use of the network, and that it will result in users leaving their devices turned on, so that they can be used as relays.

Such "nuglets-based" systems could be adopted by network providers, being the authority which authenticates and issues this currency to users. They could sell "nuglets" for real money, let users earn them through relaying for others, or use any other business model they see fit. Either way, this example shows that technically, the different components making the system envisioned in section 1.2 exist, and thus can be deployed.

5.4 New killer applications

As history shows, new technologies introduce new services, services that may not have been envisioned at the time of deployment. When **Short Message Service (SMS)** was introduced in the GSM system, it turned out to be an accidental success that took nearly everyone in the mobile industry by surprise. Piggybacking on the carrier's control channel, the same portion of the spectrum used for phone location and status updates, made, and still makes SMS very cheap for the mobile operators. With spin-off products like ring tones and wallpapers, SMS quickly became extremely profitable.

SMS is an example of a **"killer application"**, a service that creates a

sustaining market and proves the value of the underlying technology. SMS, with its asynchronous and time insensitive delivery characteristics, could be deployed in ad-hoc networks. In an airport, airline companies could send SMS messages about flight reschedules or cancellations to their passengers, either using the mobile operators' systems or transmitting through an airport ad-hoc network, saving money. All that the airline companies should have to do, would be to buy a handful of "nuglets" and be on its way.

A promising service for ad-hoc networks is **Voice-over-IP-over-Wireless (VoIPoW)** [14], which is the routing of voice conversations over wireless IP networks. However, during the design of WLAN, effort was made in making flawless transmissions over unpredictable and disruptive communications streams, by distributing the transmissions and transfer load. While this is not a problem for applications rearranging the packets back together, it may be critical for delay and bandwidth sensitive voice transmissions.

Another less demanding service is **Push-to-talk (PTT)**, where users alter to send and receives voice messages. The concept is similar to asynchronous instant messaging or SMS, and PTT is less sensitive to delay and bandwidth restrictions.

5.5 Summary

This chapter has shown the value of introducing a brokering network access service in an ad-hoc extended cellular network. A selection of the challenges has been explored, challenges not related to providing the infrastructure, but rather deploying and operating the infrastructure while being in both users and network providers' best interest. The deployment of such a system can lead to killer applications, proving the value of the underlying technology.

Given the limited time for this dissertation, the next chapter will look at future work.

Chapter 6

Future Work

While OLSRv2 has been implemented and tested, there are still tasks which could be undertaken. These fall into two broad categories: continuing developing OLSRv2 as it published as standards track RFC in 2007, and develop and implement the brokering network access service described in chapter 5. This chapter will briefly outline directions for future work on the foundation made in this dissertation.

6.1 Implementation of OLSRv2

This section looks at which implementation issues remain. The implementation includes Javadoc, the industry standard for documenting Java classes, which eases further development.

6.1.1 RFC compliance

The implementation was made with compliance to the Internet-Drafts [3] [4] [5] in mind. At the time of writing, the specifications are still Internet-Drafts, with [3] submitted to the IETF for standards track RFC publication. The implementation could be developed further side by side with [4] [5] towards a final 1-to-1 compliance with the RFCs.

One feature not implemented is packet TLVs, which are currently not used in OLSRv2, but specified in [5]. This feature may need to be implemented in the future.

6.1.2 Code optimisation

Other than being compliant to the specifications [3] [4] [5], there were no functional requirements regarding the use of overhead such as processor power or memory. The focus of the implementation was to have a practical way to continuously accommodate and test specific features being worked on in the specifications. The implementation can thus be seen as an educational

implementation, and there exist no other performance results other than stating that the implementation works as specified in [3] [4] [5].

This means that many of the implemented solutions both regarding heuristic algorithms and data structures are suboptimal. The Java SE API implements several well known data structures which should be considered for better performance, and different heuristic algorithms could for example be deployed depending of network characteristics such as mobility and topology.

6.1.3 Platform variety

As discussed in chapter 4, Java 5.0 does not allow routing table management, meaning that native code has to be written for each new system. Hopefully, support for routing table management will be included in future version of Java, making the implementation true cross-platform. It can be discussed how likely this is to happen, but in the meantime, the implementation could be developed to support more platforms, by the use of native code, as described in 4.2.1.

6.1.4 Degree of freedom using variables

The current implementation only allows certain variables to be specified by the user, others are defaulted to the values suggested in the specifications. To permit more customisation, all fixed values found in the specifications should be changeable, allowing a greater degree of freedom to the user.

6.2 Brokering network access service

OLSRv2 allows for third-party applications to be developed and used in pure OLSRv2 networks without disturbing the normal operation. This can be done using the message exchange format, specified in [3], taking advantage of the extensibility options described in 3.6.8. Thus, a brokering network access service could be implemented, being in charge of billing, security, hand-over, QoS etc. as suggested in section 1.2.

6.3 Summary

This chapter has looked at future work on implementing OLSRv2 as well as the broker system. The next and last chapter will conclude the dissertation.

Chapter 7

Conclusion and Contributions

With more and more handsets being introduced with wifi capabilities, ad-hoc extensions of cellular networks can be envisioned. This dissertation has described perspectives to such a system. It has been shown that several components must be present, and the main focus has been put on the ad-hoc networking. The proactive link-state protocol OLSRv2 [5], has been implemented and tested, focusing on platform independence, code readability and specifications compliance. The implementation has been tested for correctness using a small test-bed. The test-bed also served as a platform for emulating deployment of ad-hoc networking within purely cellular based networks, showing that such systems can exist.

For a merge between ad-hoc and cellular networks to work as proposed, the need for a brokering network access service has been identified, looking at variables that may determine its success. However, the main problem has been identified to be more of a political question than a technical one, as the mechanisms for such a system already exist. The business model used by most network providers may be in direct conflict with the ideas of ad-hoc networking, as users may be able to communicate without deploying network provider equipment. However, as new services emerge, the value of the underlying technology may be proven, creating a sustaining market.

The remainder of this chapter looks at contributions made through my work.

7.1 Graduate student criteria

At the Paris IETF meeting in 2005, my supervisor Thomas H. Clausen set forth a "graduate student criteria" for the specification of the new OLSR protocol:

> The specification should be such that a good graduate student can implement it without previous knowledge, and without spending 3 years studying the domain in advance.

Since March this year, I have been that guinea pig. Starting with no experience in mobile network programming, but with a profound interest in mobile networking, I have implemented and tested the new specifications of OLSR. I have worked very close to the authors, providing them continuous feedback as they released new Internet-Drafts, and I feel honoured having my name being mentioned in the acknowledgement of OLSRv2.

I reckon we did good regarding the graduate student criteria. The specifications are written in such a manner that they can be implemented without having a broad MANET background. It was a bit like following a recipe, and where things were inaccurate or ambiguous, I asked the authors for elaborations and clearing up. They took my questions into consideration, and the changes could be reflected in the next Internet-Draft.

I think the graduate student criteria is an interesting experiment, which lowers the barrier for people interested in MANETs. With the forthcoming OLSRv2 RFCs, it is in my opinion possible for a graduate student to do an implementation within the limited time period of a dissertation.

7.2 First implementation of OLSRv2 from scratch

When starting on the implementation work, I basically had two choices. The first was to develop an existing OLSR implementation, making updates to accommodate the changes. After all, even if there are several changes between the two versions, the basic mechanisms are the same. Alternative two was to implement OLSRv2 from scratch, and this was the path I followed, for several reasons. First, it would be more challenging, not relying on other's work. It also meant that I would have to come up with good solutions to the numerous challenges, which will be elaborated later. Second, doing an implementation from scratch also meant that I would have to dig deep into every aspect of the protocol, turning every rock, analysing the specifications down to every sentence. This was also the most interesting path from the authors' perspective, as they wanted a thorough analysis of the specifications. Running code is a significant part of decision making in the MANET working group, and during the dissertation, I have contributed to [3] [4] [5], all standards track RFCs currently under development.

The result is the very first implementation of OLSRv2 from scratch, compliant to the latest Internet-Drafts [3] [4] [5] at the time of writing. As mentioned, implementing specifications can be compared to following a recipe: you are told what to do, but not how to do it. So the challenges were in designing such a system, deciding the structure and figuring out how to make it all run together. Working on several Internet-Drafts at once was a

challenge in itself, as new versions emerged every now and then, forcing code updates, or even redesigns of whole sections. These tasks took more time than first anticipated, as the possible ripple effects of code changes had to be accounted for. It also forced me to adopt methodical habits for checking for changes. Apart from that, I found the implementation work to be quite straightforward, discovering new and better ways to solve the tasks along the way as my experience and knowledge grew. One thing that caused a bit of a headache was the lack of routing table support in Java as mentioned in section 4.2.1, which meant that the cross platform requirement could not be fulfilled as desired. However, the amount of native code required on the respective platforms is not significant.

7.3 MANET test-bed

Before starting the implementation work, it was apparent that there was a need to set up a test-bed. A test-bed, the very first MANET test-bed at Ecole Polytechnique, was assembled using computers equipped with both wifi and Ethernet interfaces. This was done with several goals in mind. First, the test-bed served as a demonstration laboratory running a live implementation of OLSRv2, validating the protocol's correctness as it was developed. Second, with section 1.2 in mind, the test-bed would emulate a cellular network extended by ad-hoc networking.

While implementing OLSRv2, the test-bed served, using the logging functionality (see section 4.2.2), as a great help for debugging the code. The test-bed showed several interesting aspects of the specifications. Features could be tested right away, giving valuable feedback to the specifications authors. With the proposed system from section 1.2 in mind, the test-bed showed that a multi network environment indeed is possible.

Other than serving as a demonstration of the protocol, the test-bed can be used for future works on OLSRv2, extensions to it, or for other MANET protocol testing.

7.4 Nokia 770 demonstration

The DIX at Ecole Polytechnique obtained twelve Nokia 770 Internet Tablets (see figure 7.1) for doing on site MANET experiments, and for extending the test-bed with mobile entities. Being merely 15 cm x 8 cm in size, these wifi enabled devices are perfect for the task. Running Linux, it should be possible to use Java on them, but at the time of implementation there was no JVM available supporting the most Java 5.0. However, the 770s are set up, and all that is needed for the implementation to run, is a compatible JVM and some native code handling the routing tables.

Figure 7.1: The twelve Nokia 770s used for testing OLSR.

The 7th of July 2006, the 2006 edition [1] of the operating system on the Nokias was released, unfortunately still without support for Java. At the time of writing, Nokia has not released a plan for incorporating it.

As a result, the demonstrations were carried out with a C++ implementation of the original OLSR, making different topologies by placing the Nokias in different constellations outside the department buildings. Mainly for demonstration purposes, running simple ping commands discovering the devices' range, we tested OLSR using a real world example.

Bibliography

[1] *Internet Tablet OS 2006 edition Feature Notes.*

[2] L. Buttyan and J.-P. Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized ad hoc networks. 2001.

[3] T. Clausen, C. Dearlove, and J. Dean. *Generalized MANET Packet/Message Format.* draft-ietf-manet-packetbb-02, 2006.

[4] T. Clausen, C. Dearlove, and J. Dean. *Neighborhood Discovery for OLSRv2.* draft-clausen-manet-olsrv2nd-00, 2006.

[5] T. Clausen, C. Dearlove, and P. Jacquet. *The Optimized Link-State Routing Protocol version 2.* 2006.

[6] T. Clausen and P. Jacquet. *Optimized Link State Routing Protocol.* RFC 3626, 2003.

[7] T. Clausen, P. Jacquet, and L. Viennot. *Comparative Study of Routing Protocols for Mobile Ad-hoc NETworks.* 2002.

[8] Z. J. Haas, M. R. Pearlman, and P. Samar. *The Zone Routing Protocol (ZRP) for Ad Hoc Networks.* draft-ietf-manet-zone-zrp-04, 2002.

[9] A. Laouiti, A. Qayyum, and L. Viennot. *Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks.* hicss2001, 2001.

[10] C. Perkins. Ad-hoc on-demand distance vector routing, 1997.

[11] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[12] M. Steenstrup. *Routing in Communications Networks.* Prentice Hall, 1995.

[13] W. Stevens. *TCP/IP Illustrated, Volume 1: The Protocols.* Addison-Wesley, 1994.

[14] Pawel Waszkiewicz. Voip versus voipow - novelty or standard? *Technology Review*, 2005.

Appendix A

jolsrv2

jolsrv2 is a stand alone application written in Java SE version 5.0, and is an implementation of the OLSRv2 link-state routing protocol. This chapter describes how to install, configure and run jolsrv2, as well as gives some examples log output. The implementation only supports the Windows platform to the full extent, and a brief explanation to how to make the code run on other platforms is included.

A.1 Configuration and running

Installation is meant to be easy, as the distribution only includes one jar file, ready to launch with the command:

```
java -jar jolsrv2.jar
```

However, some settings are required before launching. The parameters that need to be set can be found in the included `config.txt` file. The parameters are explained below.

`IPVERSION` specifies which IP version to use. Legal values are `4` or `6`. This parameter cannot be omitted.

`WILLINGNESS` denotes a notes willingness to relay traffic. Legal values are `NEVER`, `LOW`, `DEFAULT`, `HIGH` or `ALWAYS` as described in [5]. If omitted, the value `DEFAULT` is used.

`GATEWAY` specifies to which gateway a node is connected. In case of no gateways, the parameter can be omitted. The implementation only supports the use of one gateway.

`INTERFACE` specifies the name of the interfaces to use, separated by space.

This parameter cannot be omitted.

`PREFIX` is the length of the address prefixes to use, separated by space. Note that the order of prefixes must be the same as used for the interface addresses in `INTERFACE`. If omitted, it defaults to the number of bits in the belonging `INTERFACE`.

`BROADCAST` defines the broadcast addresses to use, separated by space. Note that the order of broadcast addresses must be the same as used for the interfaces addresses in `INTERFACE`. This parameter cannot be omitted.

Furthermore, logs can be recorded. There are several logging levels, depending on how much information about the system is requested:

`ALL` indicates that all messages are logged.
`CONFIG` is a message level for static configuration messages.
`FINE` is a message level providing tracing information.
`FINER` indicates a fairly detailed tracing message.
`FINEST` indicates a highly detailed tracing message.
`INFO` is a message level for informational messages.
`OFF` is a special level that can be used to turn off logging.
`SEVERE` is a message level indicating a serious failure.
`WARNING` is a message level indicating a potential problem.

A logging properties file is included with logging level set to `FINE`, which outputs log records to the console. The properties file, `logging.properties`, can be loaded using the command:

```
java -Djava.util.logging.config.file=logging.properties -jar jolsrv2.jar
```

For further instructions on how to set up logging, see:
http://java.sun.com/j2se/1.5.0/docs/guide/logging/overview.html
Before running the application, root access must be obtained, in order to manage the routing tables. Then, the wifi connection must be activated, and an ad-hoc network must be created or joined. For instructions on how to do this, please consult your operating system manuals.

A.2 Examples

This section gives some examples of log records created by running jolsrv2 with different logging levels. The first example shows the sending of the first HELLO message with logging level set to INFO.

```
INFO: jOLSRv2
```

```
INFO: Generating HELLO on interface 192.168.10.100/32
INFO: Packet 0 sent with 40 bytes containing 1 message(s) over
      interface 192.168.10.100/32
INFO: Packet received on interface 192.168.10.100/32 from /192.168.10.55
```

There is not much information given, but sufficient for normal use. In the second example, the logging level is set to `FINE`, continuing where the previous example ended.

```
INFO: Packet received on interface 192.168.10.100/32 from /192.168.10.55
FINE: ,------------------
FINE: |  PARSING PACKET
FINE: |------------------
FINE: | * Receiving interface: 192.168.10.100/32
FINE: | - Found zero and reserved
FINE: | * Packet seq number: 79
FINE: |    ,-------------------
FINE: |    |  PARSING MESSAGE
FINE: |    |-------------------
FINE: |    | * Message type:       HELLOV2
FINE: |    | * Message semantics : 0
FINE: |    | * Message size:       30
FINE: |    | * Originator address: /192.168.10.55
FINE: |    | * Hop limit:          1
FINE: |    | * Hop count:          0
FINE: |    | * Message seq number: 79
FINE: |    | - Parsing TLV block
FINE: |    | - Parsing address block
FINE: |    | - Parsing TLV block
FINE: |    `------------------
FINE: `-----------------
INFO:   POPULATING REPOSITORIES
FINE: - LinkSet: Added tuple from 192.168.10.100/32 to 192.168.10.55/32 (LOST)
FINE: - NAA set: Added tuple to [192.168.10.55/32]
```

Using logging level `FINE`, the contents of packets received are shown, giving a more detailed view than the previous example. The log record shows that the packet contains one HELLO message which then is parsed. The contents of the message is finally used to update the repositories. In the last example, logging level is set to `ALL`. The scenario is system start-up, sending the first packet.

```
INFO: jOLSRv2
```

```
FINE: - Using IPv4 address /192.168.10.100 on
      Dell TrueMobile 1300 WLAN Mini-PCI Card
CONFIG: * Operating system: Windows XP
CONFIG: * Port:        50000
CONFIG: * Interface:   192.168.10.100/32     Broadcast: /192.168.10.255
CONFIG: * Originator address: /192.168.10.100
CONFIG: * Willingness: WILL_DEFAULT
INFO: Generating HELLO on interface 192.168.10.100/32
FINE: No links in LinkSet
FINEST: Getting bytes from Tlv
FINEST: Tlv type: VALIDITY_TIME
FINEST: Semantics: 4
FINEST:   -noindex
FINEST: Bytes size: 4
FINEST: Getting bytes from Tlv
FINEST: Tlv type: INTERVAL_TIME
FINEST: Semantics: 4
FINEST:   -noindex
FINEST: Bytes size: 4
FINEST: Getting bytes from Tlv
FINEST: Tlv type: WILLINGNESS
FINEST: Semantics: 4
FINEST:   -noindex
FINEST: Bytes size: 4
FINEST: mid
FINE: Added 2 byte(s) of padding
INFO: Packet 0 sent with 40 bytes containing 1 message(s) over
      interface 192.168.10.100/32
```

The log record information about the configuration, followed by the generating of the first HELLO message, detailing its contents. This level is first and foremost used for debugging.

A.3 Multiple platforms

As explained in section 4.2.1, each type of operating system must use a piece of native code handling the routing table operations. This implementation only includes native code for the Windows platform, compiled in a library called `jolsrv2_rt.dll`. If the implementation is to be used on other platforms, the following method interfaces handling routing table management must be implemented:

```
CreateEntry(String destination, String netmask, String gateway,
            String iface, int metric)
```

where

`destination` indicates the destination interface address.
`netmask` indicates the network mask of the interface creating the entry.
`gateway` indicates the address of the next hop towards the destination from the interface creating the entry.
`iface` indicates the interface address creating the entry.
`metric` indicates the metric of the route.

and

```
DeleteEntry(String destination)
```

where

`destination` indicates the destination interface.

These methods are found in the `RoutingSet.java` file. In the same file,

```
static {System.loadLibrary("jolsrv2_rt");}
```

must be changed to reflect the native library in use. For a comprehensive guide to how this is done, see
http://java.sun.com/j2se/1.5.0/docs/guide/jni/.

Appendix B

The Count-to-infinity Problem

One of the most important problems with distance-vector algorithms is the "count-to-infinity" problem, which will be examined in this chapter.

Imagine a network with four nodes running a distance-vector algorithm. The cost between each node is 1, as shown in figure B.1

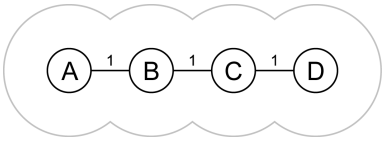


Figure B.1: A network containing four nodes running a distance-vector algorithm.

When the network has converged, the cost reaching each node is known. Table B.1 shows the total costs in the network.

from \ to	A	B	C	D
A	0,-	1,B	2,B	3,B
B	1,A	0,-	1,C	2,C
C	2,B	1,B	0,-	1,D
D	3,C	2,C	1,C	0,-

Table B.1: Table showing the routing tables of each node after the network has converged. n,X means that the total cost going through node X is n.

Imagine now that the link between nodes A and B breaks. Node B will reflect this in its routing table, and put infinity as link cost, but node C is unaware of the breakage. As node B receives C's routing table, it will see that node C has a link to node A costing 2, which is better than infinity,

and will believe that node C has an independent link to node A. Node B will thus update its link to A with cost 3 through C. When C receives B's routing table, it sees that B has updated its link cost to A from 1 to 3, and will update its own link to A to 4. This process loops until all nodes find out that the weight of link to A is infinity. These facts is the reason why distance-vector algorithms are said to have a slow convergence rate.

Appendix C

The MANET Working Group

An IETF working group is a branch within the Internet Engineering Task Force (IETF), working on a limited set of tasks. It is open to all who want to participate, and unlike other groups, like those in the IEEE, there is no voting procedure nor membership fees. The final arbiter of decision making is "rough consensus and running code". An IETF working group lasts as long as there are unsolved problems within that particular field of interest, and will normally be closed once the work described is finished.

The tasks of the MANET working group is summarised as follows on their website:

> The purpose of the MANET working group is to standardize IP routing protocol functionality suitable for wireless routing application within both static and dynamic topologies with increased dynamics due to node motion or other factors.

The members of the MANET working group are working on reactive and proactive protocols, and the working group's objective is to create standards for both. If however, they learn that the similarities between the too are significant, they may wish to follow the path of a converged approach. The neighbourhood discovery [4] and packet format [3] being factored out of OLSRv2 [5] are examples of such convergences. During my dissertation, I have contributed to the "running code" bit to [3] [4] [5], and the value of this is significant.