Routing and mobility in large heterogeneous packet networks.

Emmanuel Baccelli

▶ **To cite this version:**

Emmanuel Baccelli. Routing and mobility in large heterogeneous packet networks.. Networking and Internet Architecture [cs.NI]. Ecole Polytechnique X, 2006. English. NNT : . pastel-00001603

HAL Id: pastel-00001603
https://pastel.hal.science/pastel-00001603

Submitted on 28 Jul 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse présentée pour obtenir le grade de

DOCTEUR DE L'ECOLE POLYTECHNIQUE

en Mathématiques et Informatique

par

Emmanuel BACCELLI

Routage et Mobilité dans les Grands Réseaux Hétérogènes à Commutation de Paquets

Soutenue le 5 janvier 2006 devant le jury composé de :

Mario GERLA	UCLA, USA	Rapporteur
Kenichi MASE	Niigata University, Japon	Rapporteur
Stéphane AMARGER	Hitachi Europe, France	Examinateur
Laurent VIENNOT	INRIA Rocquencourt, France	Examinateur
Philippe JACQUET	Ecole Polytechnique, France	Directeur de Thèse

This work is dedicated to my friends and to my family,
to people who have been unconditional in their support,
my mother and father in the first place,
with a special dedication to Pierre-Jean Hullo,
for his inspiring courage.

Abstract

Integrating mobile ad hoc devices in the Internet brings a number of new challenges, both in terms of optimization of the routing protocols providing ad hoc networks, and in terms of integration of ad hoc mobility with the IP-based infrastructure. This thesis overviews these challenges. Some existing solutions are analyzed and compared while a number of new solutions are introduced.

Resumé

L'intégration d'appareils mobiles ad hoc dans l'Internet pose des problèmes intéressants aussi bien en terme d'optimisation des protocoles de routages utilisés pour fournir la connectivité ad hoc, qu'en terme d'intégration de la mobilité ad hoc dans l'infrastructure IP. Cette thèse passe ces problèmes en revue. Plusieurs solutions développées auparavant sont analysées et évaluées comparativement, ce pendant qu'un certain nombre de nouvelles solutions sont introduites.

Acknowledgements

I want to thank Thomas Heide Clausen, Philippe Jacquet, Pramila Mullan, Raju Rajan for their essential guidance.

I want to thank Hitachi Ltd for financing my PhD studies. In particular, I want to thank Stéphane Amarger, Masato Hayashi, Susumu Matsui, and Hiroki Satoh for their great support and help in many domains.

I also want to thank Mario Gerla and Kenichi Mase for accepting to review this document, and Laurent Viennot for taking part in the jury.

And finally, I wish to thank Lena Domröse for her special powers.

Table of Contents

List of Figures

Preface

As the Internet continues to span further, it meets the other network revolution currently taking place: the generalization of the use of mobile wireless devices. The present thesis deals with one aspect of this marriage: the integration of mobile ad hoc networks in the Internet.

Mobile ad hoc networks offer a new type of wireless connectivity that abolishes the need for mobile nodes or users to go through a predetermined central entity in order to communicate. This new freedom is extremely interesting in that it enables autonomous network functionalities for spontaneous fleets of mobile wireless devices. However, the use of wireless links, the absence of a central entity, and the mobility of the nodes impose the use of new, drastically optimized routing schemes in order to efficiently form and maintain these networks.

The idea of connecting such networks of mobile nodes to the Internet is very seducing, as it would both provide a natural mobile Internet access extension, and enable users to bypass dedicated Internet access entities when sensible, offering a service similar to multi-hop and multi-media Push-To-Talk. However, the specific characteristics of ad hoc networks bring a number of new issues in terms of their integration in the Internet infrastructure.

This thesis will therefore address on one hand designing optimized routing schemes in order to build efficient mobile ad hoc networks, and on the other hand designing solutions in order to integrate mobile ad hoc nodes in the Internet.

Routing optimizations must aim at reducing the amount of control traffic needed by the routing protocol in order to provide a functional network, as ad hoc nodes experience important issues in terms of scarce bandwidth and interferences (contrary to nodes in traditional networks). Such reductions may target different routing "fields". For example, *flooding optimizations* aim at reducing the amount of (re)transmissions needed network-wide to deliver some routing information to all the nodes in the network (a frequent operation that

produces most of the control traffic, for most routing protocols). On the other hand, *topology reduction* aims at optimizing the amount of routing information required by the routing protocol to provide satisfying routes through the network.

Other optimizations aim at adapting different routing mechanisms to a mobile ad hoc environment. For instance, most routing protocols employ specific "scalability" schemes in order to handle topologies with large numbers of nodes. However, the schemes that were designed for traditional networks are not adapted to mobile ad hoc networking. Thus, *scalability optimizations* aim at providing similar schemes enabling ad hoc routing protocols to handle large mobile ad hoc topologies.

Ad hoc nodes introduce a new type of node in the Internet: a node that may both form the network (*i.e.* a router) and use the network (*i.e.* a host). Thus, requiring a specific routing protocol to manage these new nodes at the periphery of the fixed Internet makes sense. However, ad hoc routing concepts are also considered for the design of router mobility schemes in the Internet. Although this is a rather new topic too, solutions in this field should on the other hand preferrably extend existing generic Internet routing protocols – in order for such generic protocols to continue being available, as they are found to be quite practical.

The solutions integrating ad hoc nodes in the Internet must aim at introducing ad hoc routing capabilities into the existing architecture of Internet protocols, while leveraging previous mobility solutions (such as MobileIP) as appropriate. In order to achieve this goal, new schemes must also be engineered aside of "pure" routing mechanisms, in order to cope with the specific nature of ad hoc networks, including mobile, multi-hop and wireless aspects. For example, MobileIP was not designed to handle multi-hop aspects, which yields the need for new *address auto-configuration* mechanisms able to deal with this particular characteristic. On another level, the introduction of mobile nodes and multi-hop wireless links brings a number of *new security issues* that have to be addressed in order to keep insuring a satistying network integrity.

Organization

This document is organized in three parts. The first part is an introduction to Internet in general and its functionning, with a particular focus on a specific component of this functionning: *routing*. The second part describes the specific routing issues and challenges that come when devices participating in the Internet are mobile. And finally, the third part introduces some solutions for Internet routing with mobile ad hoc devices.

Usage Guidelines

Advanced readers may skip Part 1, and start directly with the second part's introduction to routing challenges with ad hoc mobility in the Internet – which is the main focus of this document.

Thesis Contributions

The contributions of this thesis are mainly in the third part of this document. These contributions include the following:

- A comparison between existing flooding optimizations (MPR flooding and Gateway flooding), in Section 7.1, that was published in [5].
- An evaluation of the scalability optimization of ad hoc routing with Fisheye enhanced OLSR, in Section 8.2, that was published in [9].
- A new proposal for OSPF's extension in order to integrate router ad hoc mobility in the Internet, in Section 6.3, that was published in [11].
- A new solution for NEMO to integrate network ad hoc mobility in the Internet, in Section 6.4, that was published in [7].
- A new solution for ad hoc node IP auto-configuration, in Section 9.2, that was published in [8].
- A new scalability mechanism introducing clustering and hierarchical routing in OLSR (OLSR Trees), in Section 8.1, that was published in [12].
- New ways to introduce topology reduction optimizations for Overlapping Relays OSPF on ad hoc networks, in Section 7.2, that was published in [16].
- A new technique providing reliable synchronization of databases in mobile ad hoc networks (Database Signatures), in Section 7.3, that was published in [4].

Part I

Introduction to Internet Routing

Chapter 1

Internet History

Communication networks and industrial revolution have gone hand-in-hand since the invention of the telegraph in the 1830s and the telephone in the 1870s. The telegraph began with communications that could span nationwide, before the telephone achieved global coverage, maturing into the largest network in history, connecting billions of users all over the planet.

Today, we witness the advent of another global network: the Internet. Instead of simply connecting phones and being dedicated to voice communications, the Internet connects many kinds of devices and is used to communicate many different types of data: text, video, voice or audio in general – in short: anything that can be digitized. Already connecting almost a billion users in 2005, hundreds of millions of machines across the world, and an amount of traffic that doubles every year, the Internet quickly develops into what may soon be *the* network, universally used for all kinds of communication. This chapter presents a brief history of this development.

1.1 The ARPANET and Packet Switching

During the Cold War, in the late 1960s, the U.S. Defense Advanced Research Projects Agency (DARPA) initiated a research program aiming at developing a communications infrastructure that would be robust against even a large scale nuclear attack. Also influenced by their need to share expensive computing ressources

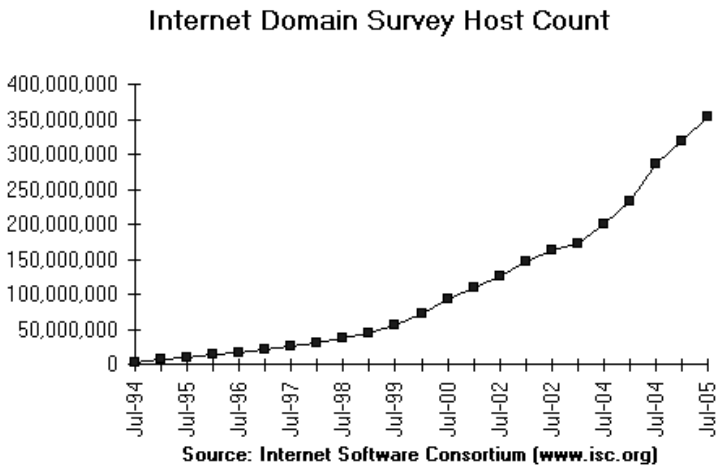


Figure 1.1: History of the number of machines connected to the Internet.

Date	Estimated Number of Internet Users
1995	15 million
1997	70 million
1999	250 million
2001	500 million
2003	700 million
2005	950 million

Figure 1.2: History of the estimated number Internet users over the last decade. (Source: Internet World Stats)

among themsleves, researchers have therefore conceived the successfull decentralized design of the Internet we know today.

This design is based on the concept of *packet networking* (also called packet switching), that appeared in the early 1960s. Packet networking namely differs from *circuit networking*, which was the main network technology at the time – being the base of telephony back then. The key difference between packet networking and circuit networking is that messages in a packet network are self-descriptive, whereas in circuit networks they are not. In packet networks, each message contains enough information about where it is from and where it is supposed to go, so that it can suddenly appear anywhere/anytime in the network, and still reach

its destination as expected. On the other hand, in circuit networks, messages do not contain self-descriptive information and are therefore totally dependent on being initially put on "the right tracks" at the right time, for them to reach their destination as expected.

In fact, this basic difference influences greatly the characteristics of the network. For instance, packet networks could be based on machines (called *multiplexors*) that are able to *store* messages and *forward* them when appropriate, whereas circuit networks could not work with multiplexors. This aspect made packet networks cheaper to build, maintain and operate compared to telephone networks. Indeed, multiplexors can share expensive transmission facilities more efficiently than the machines imposed by telephone networking. Moreover, this very same aspect made packet networks more reliable than telephone networks. Because a message in packet networks can "find its way", even if some machines on the path to destination are out of order, chances are that the message will find an alternative route to reach its destination. Even if no route is available at the moment, the message can be stored until a route is available again. On the other hand, in circuit networks, if a machine is unavailable on the path to destination, the message is lost.

Therefore, techniques and technologies for interconnecting packet networks of various kinds were investigated, the goal being to develop a system which would allow networked computers to communicate transparently across multiple packet networks. The DARPA project was called the "Interneting" project and the system of networks which emerged from the research was known as the "Internet". The system of protocols which was developed over the course of this research effort became known as the TCP/IP Protocol Suite, after the two initial protocols that were developed: Transmission Control Protocol (TCP) and Internet Protocol (IP) – see Chapter 2 for more details on Internet protocols. And in 1969 was born the earliest portion of the Internet – then called the "ARPANET" – connecting 4 machines in 4 southwestern US universities (UC Los Angeles, Stanford Research Institute (SRI), UC Santa Barbara and the University of Utah). From then on, the size of the Internet basically doubled every year for the next three decades.

In 1972, a large demonstration of the ARPANET at the International Computer Communication Conference (ICCC) was organized. This was the first public demonstration of this new network technology to the public. It was also in 1972 that the initial "killer" application, *electronic mail*, was introduced, motivated by the need of ARPANET developers for an easy coordination mechanism. Email quickly became the largest network application.

The ARPANET rapidly grew in population and capacity during the 1970s, linking several then "powerful"

supercomputers, and researchers and academics in many different fields had begun to make an increasing use of the network.

1.2 From NSFNET to the Internet

In the 1980s, while ARPANET's military funding progressively dried out (which led to its shutting down in 1989) the National Science Foundation (NSF) deployed a parallel network called NSFNET, which was based on the same TCP/IP technology, and created in 1985 a program that funded the connection of all educational facilities, academic researchers, government agencies, or international research organizations in the USA through this network, which was provisionned to handle the increasing traffic loads. In a few years NSFNET grew from a few nodes connected with 56 kbps links, to a backbone of 21 core routers with multiple 45 Mbps links, connecting over 50,000 networks on all seven continents, and even outer space.

Widespread development of local area networks (LAN) with Ethernet, personal computers (PC) and workstations in the 1980s also greatly participated in the flourishing of NSFNET. When NSF's funding stopped in 1995, the network that resulted was already known as the Internet since a few years in the academic community. Continuing explosive growth was then experienced when the introduction of HTTP documentation and the World Wide Web started to enlarge the interest for the Internet to a larger public. Many commercial computer networks (now known as Internet Service Providers, or ISPs) began selling connection to the Internet and data services to individuals, giving birth to the development we know of today.

1.3 Internet Standards

Part of the Internet's success is due to its open nature and its standardization process, which philosophically differs with the way standards are usually developed in the industry in general – and in the telephone industry in particular. Appendix A briefly overviews this special organization which produces many of the documents cited throughout the following chapters, on which the Internet is technically based.

The next chapter introduces to the techniques and standards employed to form the network of networks

that is the Internet.

Chapter 2

Routing in the Internet

In this chapter, we give a brief overview of Internet's general functioning, in order to introduce a particular functionality needed for networking: *routing*. Routing consists in the task of establishing paths through a network, enabling distant nodes to communicate. The challenges encountered in accomplishing this task with different Internet environments will be the main subject of the analysis presented in this document.

2.1 Internet Protocols Architecture

In order to solve the problem of creating a network, the Internet uses an old engineering technique: breaking a complex problem into smaller manageable pieces, and solving them one by one. Therefore, the Internet uses the abstractions of *layers* (similarly to other telecommunication systems) to separate networking into several functional areas. Each layer gathers *protocols*, and each protocol solves a specific problem inside the functional area of its layer.

A protocol is a set of rules and formats that govern some level of communication between machines, so that they can understand each other at this level. The separation into layers and protocols is hierarchical, in the sense that protocols of the highest layer, called *applications*, beneficiate from the services provided by the lower layers. The Internet layers and examples of protocols are pictured in Figure 2.1 and Figure 2.2. Their functions can be described as follows:

Layer 1: The Physical Layer – This is the lowest layer. It contains protocols responsible for moving the individual *bits* (the actual "0"s and "1"s forming digitized information) between machines connected by a communication medium (*i.e.* a link).

Layer 2: The Subnetwork Layer – This layer creates the abstraction of a *subnetwork*, reaching the machines connected by the same link. The protocols in this layer are responsible for (i) forming and monitoring frames (ordered sequences of bits) sent and received over the link, and (ii) arbitrating access to this medium if it is shared by several machines. This layer is also called the MAC (Medium Access Control), the LAN (Local Area Network) or the Data Link layer. Example of Layer 2 protocols include **Ethernet (IEEE 802.3)** for wired connection, or **WiFi (IEEE 802.11, also called WLAN)** for wireless connection. The subnetwork layer protocols of course use the services of physical layer protocols in order to function.

Layer 3: The Network Layer – This layer creates the abstraction of a *network*, reaching machines accross different subnetwork topologies and technologies. The function of the protocols in this layer is to logically concatenate sets of links to form the abstraction of a single end-to-end link. This allows a machine to communicate with any other machine in any subnetwork, by computing a route between these two machines, through intermediate machines if necessary (this computation is called *routing*). These protocols use the services of the subnetwork layer to carry packets over each link on these routes. For example, the most important layer 3 protocol used on the Internet is **IP (Internet Protocol)**. Aside from formating the information exchanged between different machines on the Internet, the crucial role of this protocol is to identify each machine (over all the subnetworks) with a unique name: *IP addresses*. This responsibility, and the employed addressing scheme are fundamental in the Internet, especially for routing, as we will see in next sections.

Layer 4: The Transport Layer– This layer usually provides the abstraction of reliable end-to-end links. Basically, it provides error control, flow control and congestion control services over the end-to-end links given by the network Layer. Error control deals with packet loss, corruption or duplication through detecting these errors, and discarding or retransmitting packets when needed (which is often the case). Flow and congestion control regulate the rate of packet transmissions of a source towards a destination, so that it matches the rate currently sustainable by the network and the destination. The transport layer is also responsible for distributing the right packets to to the right application. This distinction is done by adding to packets application-specific identifiers, called *ports*). Examples of Layer 4 protocols include **TCP (Transmission Control Protocol)**, that provides all the features we just mentionned, and **UDP (User Datagram Protocol)**, that does not provide error control or flow control, but just application distinction. UDP is used by applications that don't need error and flow control, or that provide their own error and flow control.

Layer 5: The Application Layer – This layer contains the protocols that directly serve the users that want

to communicate over the network, or access remote ressources. These protocols are also called applications. There are many examples of such protocols, including **HTTP (HyperText Transfer Protocol)** for the Web browsing, **FTP (File Transfer Protocol)** for file transfering, or **SMTP (Simple Mail Transfer Protocol)** for e-mails, *etc*.

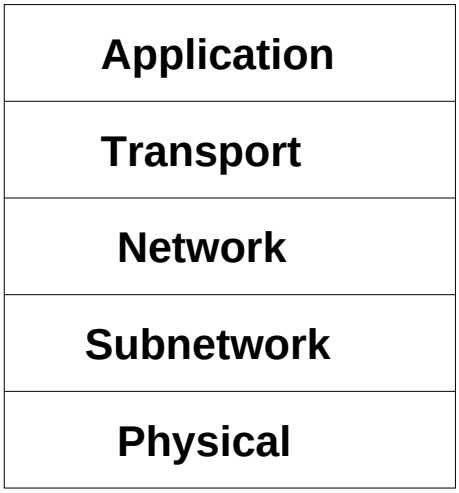


Figure 2.1: The layers in the Internet architecture.

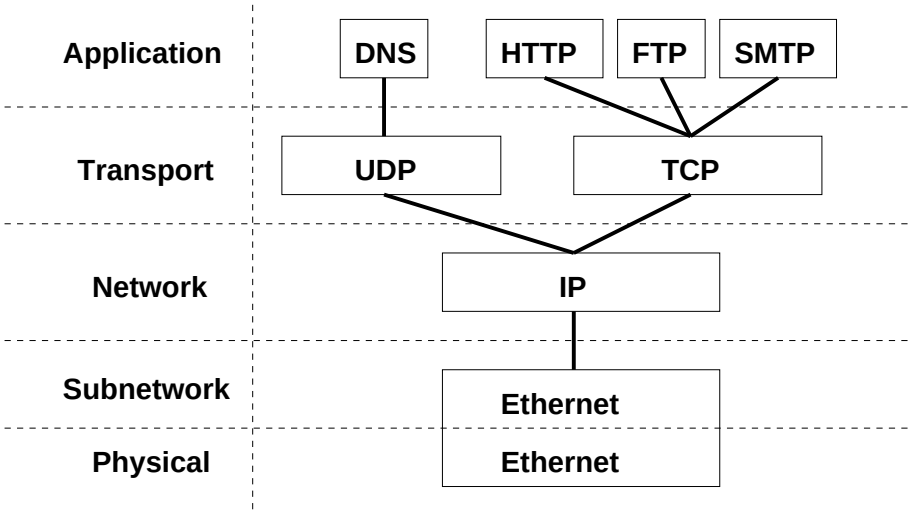


Figure 2.2: The Internet protocol stack.

Layers separations are not always respected. As shown in Figure 2.2 the Ethernet protocol spans over two layers at the same time, and this is not the only cross-layer example. However, it is has been the base of

Internet engineering for decades, and it is followed in general.

In the next section, we will focus on the layer 3 functionalities and more precisely, on IP and on the routing functionality.

2.2 IP Addressing

One of the fundamental pieces of the network layer is the addressing scheme it employs. In this section we will therefore describe it. As the Internet uses predominantly IP version 4 (IPv4), we will focus on this version of IP and in the following, IP will be synonymous with IPv4. The alternative version, IPv6, is not fully established yet, and furthermore, the principles described in the following apply also to this new version of IP.

IP addresses consist in 32 bits, and are typically represented with the dotted-quad notation, in which the four *bytes* (groups of 8 bits, which are also called *octets*) are separated by dots and written as decimal numbers that vary from 0 to 255. An example of this notation is:

$$10001111\ \ 01000110\ \ 00000010\ \ 00000111 = 143.70.2.7 \tag{2.1}$$

IP has knowledge only of this type of addresses. In particular, it has no knowledge of names and cannot deal with them. Therefore, an application called **DNS (Domain Name System)** is thereby needed to map usual web names such as *www.google.com* into the 32 bits binary addresses that IP deals with.

2.3 Hosts and Routers, Usual Internet Terminology

This section introduces some terms and definitions that will be used in this document.

Hosts and Routers – In the Internet, two distinct types of machines can be found: *hosts* on one hand, and *routers* on the other hand. Routers are machines (multiplexors) that are dedicated to forming the network, as well as storing and forwarding users' data: they are the *core* of the network. Hosts are machines that just use the network, from the *edge* of the core.

For example, when an individual connects to the Internet from home with his PC, his computer is a host. On the other hand, the machine to which his PC connects in order to go online is a router, that belongs to this individual's ISP (Internet Service Provider, see Section 2.6).

Interfaces and Subnets – Each host has typically one *interface* through which it is connected to a *subnetwork*. A subnetwork (often abreviated into a *subnet*) usually gathers a number of hosts and is typically connected to other subnets, via a router. A router has therefore typically several interfaces, at least one on each subnet it connects, and there may be many, in case of a big router.

Networks and Internet – An ensemble of subnetworks connected by a number of routers is called a *network* [*]. The Internet is then, physically, the interconnection of all the networks, routers and hosts around the world – that use IP.

A specificity of IP is that each router interface as well as each host interface must be assigned a unique IP address. With the exponential growth of the Internet, an IP address shortage was feared (though IPv4 specifies more than 4 billion different addresses), and this is initially why a new version of IP (IPv6) was envisionned, with bigger addresses: 128 bits long.

Nodes and Topology – Another common term in the networking terminology is: a *node*. In the Internet, and in the following, this term designates a machine that is part of the network in general. That is: a node in the Internet is either a host or a router. The specific state of a network at a given time, with respect to how nodes are inter-connected, is then called the network's *topology*.

Overhead and Control Traffic – In the following, we will call *overhead* or *control traffic* any message or part of message that is sent on a network, but that is not actual user data. In other words, this corresponds to the network load that is needed for the network to be functional.

Broadcast, Multicast and Unicast – The primary function of a network is to enable the delivering of some information to some nodes on the network. Different categories of delivery are defined: a *broadcast* is the operation of delivering a message to *all the nodes* on the network. A *multicast* is the operation of delivering a message to *a subset of nodes* on the network. And finally, a *unicast* is the operation of delivering a message

[*]Sometimes a subnet is also called a network, as it is indeed a small network connecting a number of machines. However, there is a difference between the two terms. On a subnet, nodes can usually communicate directly with each other over the same link, while on a network in general, nodes may have to communicate via one or more routers, as they may not be directly connected by the same link.

to *a single node* on the network.

2.4 IP Packets

The IP protocol defines the way information is sent over the network, between different machines. Information to be delivered is emitted as series of packets that share the same format and contain data that must be transmitted, typically at the request of an application, through TCP or UDP, and then down to IP (see Figure 2.2). This data is digitized, in form of a certain sequence of bits, and called the *user data*.

As mentionned in Section 1.2), the Internet is based on packet switching. Therefore packets also include self-descriptive information, *i.e.* where the packet comes from, where it is supposed to arrive *etc*. This information is contained in a header, called the *IP header*, in form of another sequence of bits, that is positionned in front of the user data in an IP packet.

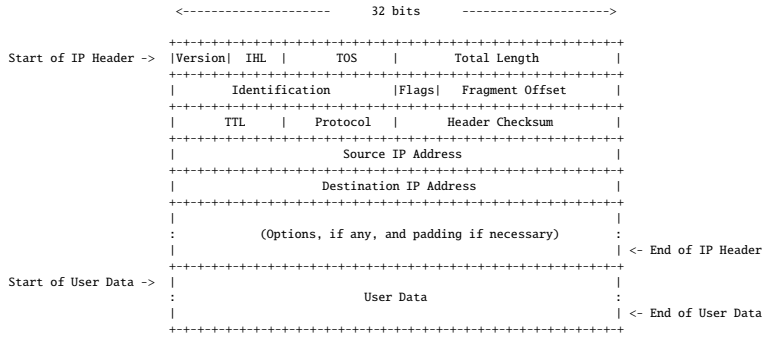


Figure 2.3: The IP packet format.

Figure 2.3 shows an IP packet in its most common graphic representation (in IETF documents). An IP packet is broken into pieces of 32 bits (the size of an IP address), and each piece is represented graphically by a row. The actual IP packet is then the series of bits represented by the concatenation of the rows, just like a text being read, from left to right on each line, and from the top, to the bottom line. The fields of an IP packet (the self-descriptive information) can be decribed as follows:

Version – *4 bits long*. This field identifies the version of IP specification to which the packet is formatted. As mentionned earlier the version is usually 4.

IHL – *4 bits long*. This field identifies the IP Header Length (IHL). It specifies the number of 32 bits words in the packet's IP header. Typically this number is 5.

TOS – *8 bits long*. This field identifies the Type Of Service (TOS). It was initially intended to enable IP to support different levels of service, but it is typically unused.

Total Length – *16 bits long*. This field specifies the total number of bytes in the packet, including the header.

Identification – *16 bits long*. This field is used as an aid for fragmenting and reassembling packets that are too big to be transmitted at once. Typically, this fragmentation function is not used.

Flags – *3 bits long*. This field is also used as an aid for fragmenting and reassembling packets that are too big to be transmitted at once. Typically, this flag is set to the *Don't Fragment* value.

Fragment Offset – *13 bits long*. This field is also used as an aid for fragmenting and reassembling packets that are too big to be transmitted at once. Typically, this fragmentation function is not used.

TTL – *8 bits long*. This field identifies the Time To Live (TTL) of the packet. It is used to control the length of time that a packet stays in the network. At each transfer on the path of the packet through the network, this field is decreased by 1. If this field reaches 0, the packet is considered as not valid anymore and will be discarded.

Protocol – *8 bits long*. This field indicates to which upper layer protocol it should pass the user data. Typically, this is either TCP or UDP.

Header Checksum – *16 bits long*. This field provides a protection of the packet header against corruption during transfer.

Source IP Address – *32 bits long*. This field identifies the sender of this packet with its IP address. This IP address is used to route the packet.

Destination IP Address – *32 bits long*. This field specifies where this packet should be delivered. This IP address identifies the destination of this packet, and is also used to route the packet.

Options – *Variable length*. This field allows further expansion of the IP header for special purposes. Most of the time it is unused, and this field is of length 0.

Padding – *Variable length*. This field is used if needed for alignment, to ensure that the header length of the packet is anyways a multiple of 32 bits words, even in case of "strange" option length.

User Data – *Variable length*. This field contains the data that was requested to be transmitted, usually by higher layer protocols such as TCP or UDP.

Some self-descriptive informations – source and destination IP addresses– included in the IP header are especially important for routing. The next section therefore introduces to routing in the Internet and how this routing is tied to IP addressing.

2.5 The Role of Routing

The Internet is basically a collection of thousands of packet subnetworks, large or small. Each of these subnetworks connects several hosts together (for instance several users' PC) via a layer 2 protocol – such as Ethernet or WiFi. On the other hand, each subnetwork is itself connected to other subnetworks via routers: machines that are connected to several subnetworks at the same time and whose primary task is to vehiculate data from one subnetwork to another, to create a network – that is, to its full extent: the Internet.

In order to achieve this abstraction, it must be ensured that any node in the Internet can reach any other node that is also online. If the two nodes are in the same subnetwork, the layer 2 protocol used on this particular subnetwork provides that the two nodes can communicate directly over the medium connecting them – for instance a wire in the case of Ethernet. Figure 2.4 shows a small subnetwork of hosts and one router connected by Ethernet. In this case, Host A can for example communicate with Host C with the Ethernet protocol, through their interfaces on the same wire. This wire (represented by the central solid line in Figure 2.4) is in the case of this subnetwork the shared communication medium that connects directly each interface to every other interfaces on the network.

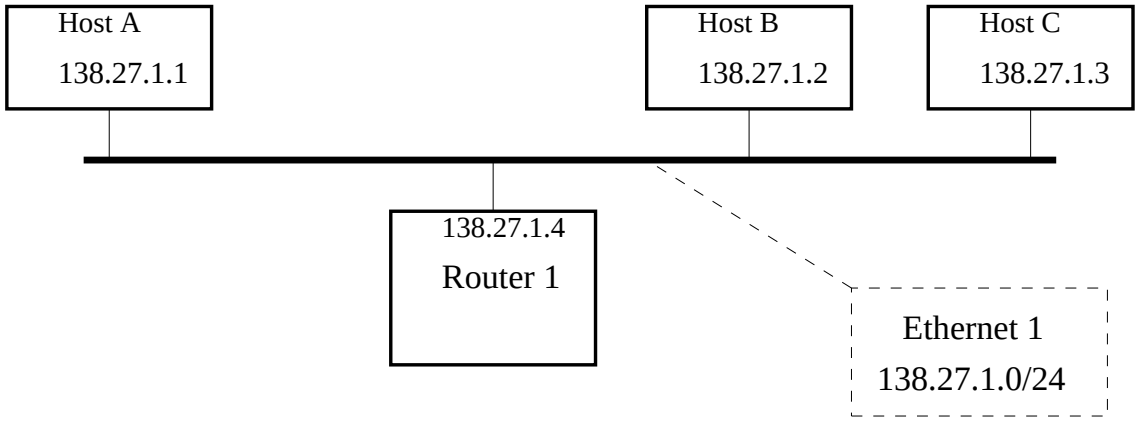


Figure 2.4: Subnetwork connected by Ethernet.

Note that, as shown in Figure 2.4, the usual IP addressing scheme is to have all the interfaces in the same subnetwork share the same *prefix* (also called *subnet mask*). Indeed, hosts A, B and C, as well as Router 1, each have an interface on the subnetwork, and all these interfaces' address have their first 3 numbers in common: "138.27.1". This common addressing part is called the prefix of the subnetwork. Notice that each number represents 8 bits (see Section 2.2), so the prefix "138.27.1" is actually 24 bits long. Hence the denomination of the subnetwork "138.27.1.0/24", which indicates that all the interfaces on this subnetwork have their address beginning with the prefix "138.27.1". Conversely, the mention "138.27.1.0/24" also means that any node with an IP address matching the prefix "138.27.1" is supposed to be on this subnetwork, and "138.27.1.0/24" is therefore understood as the address of this subnetwork.

It is important to note how this property ties to the IP address of a node both (i) its identity and (ii) information about its location. This organization is essential in IP networks and many protocols rely on such an addressing scheme. A node with an interface on a subnetwork must be aware of the subnetwork's prefix, and its interface must be configured with an IP address that matches the subnetwork's prefix. This can be done manually, or automatically when the interface is connected, with a protocol such as DHCP (Dynamic Host Configuration Protocol [44]).

Now, if a node, say host A in Figure 2.5, has to reach another node in a different subnetwork, say host G, something special must be done because host A and host G are not connected to the same wire, and therefore cannot directly communicate. This "something special" is called routing: the task of finding a way to reach a remote node through intermediate nodes. For this task, *routing protocols* are used in the Internet. In

this case, if host A wants to send a message to host G, a routing protocol would enable the packet sent by host A to be transfered via Router 1, and then Router 2, on to host G.

Routing protocols rely on the IP addressing organization we have described above. For example, in Figure 2.5, host A can determine that it needs routing to reach host G by checking that host G's address does not match the prefix of the subnetwork of host A. Indeed, "138.27.3.10" does not match the prefix "138.27.1", therefore host A can conclude that host G is not in its subnetwork and that routing is needed in order to reach host G. On the other hand, host A can determine that it can communicate directly with host B because host A can verify that host B's address matches the prefix of its subnetwork "138.27.1", and in this case no routing is necessary. In summary, when a host sends a packet to another host, it either sends it directly to the destination, or to an intermediate node for forwarding closer to the destination. The choice of direct or indirect delivery depends on the destination address and the prefix of the subnetwork of the sender. If the host cannot deliver it directly, the packet is sent to a router on the subnetwork, that runs a routing protocol (contrary to hosts in general). The knowledge provided by this routing protocol will enable the router to forward the packet closer to destination.

More generally, in a network, a routing protocol ensures that information can be sent between distant devices connected to this network. For instance, when connecting to the Internet, a user may want to communicate with a friend somewhere in the country, or access a foreign website. The role of routing protocols is to enable users to communicate and exchange information between them and remote ressources.

2.6 Autonomous Systems

From an organizational point of view, the Internet is split into *domains*, also called *autonomous systems (or AS)*. Each domain typically gathers several subnetworks and routers that are run under the same technical and administrative control.

Autonomous systems are organizations of varying size. For example, large ISPs (Internet Service Providers) such as MCI or Sprint each run a large autonomous sytem, typically a geographically wide topology of big routers and ultra-fast links called a *backbone*. There are also smaller domains, such as ISPs that focus on residential dial-up services (for example AOL), or other types of organizations such as university campuses.

As connectivity is the most important value in the Internet, these different organizations must ensure that

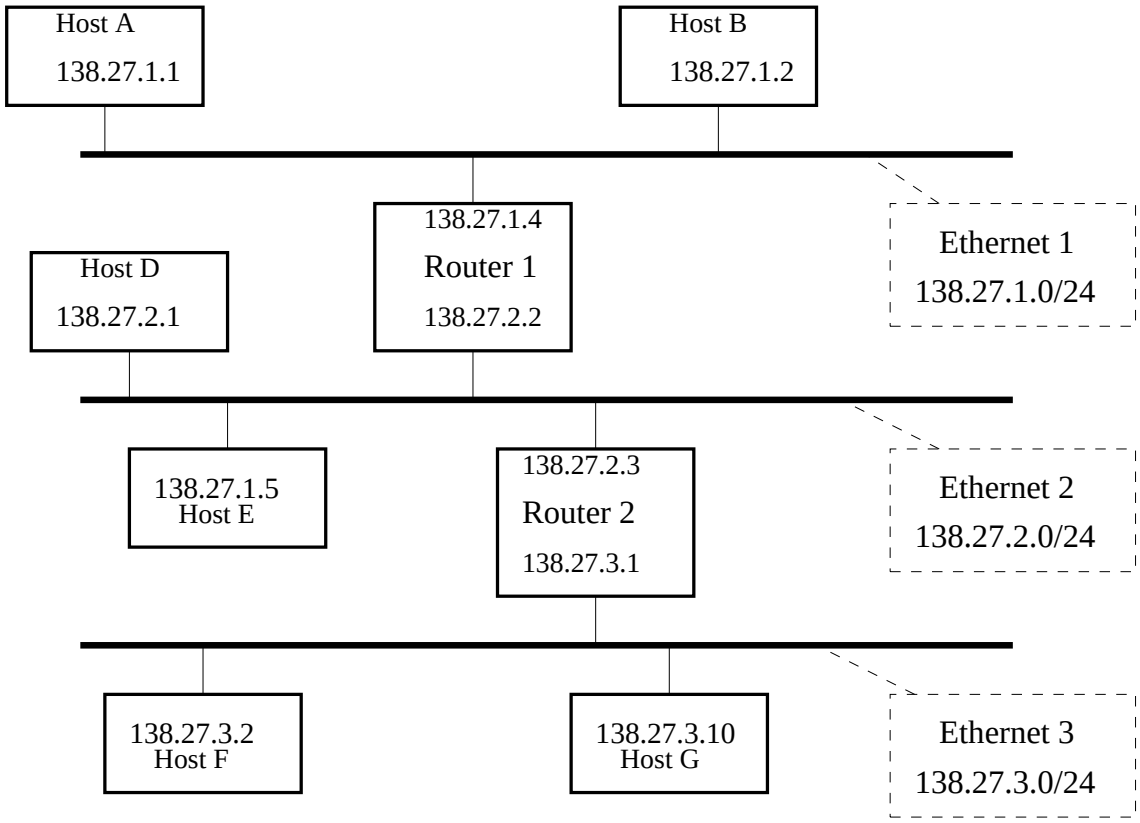


Figure 2.5: 3 subnetworks connected by 2 routers.

they have a path to reach one another at all time. Therefore, these domains are connected to each other. This is especially true for large ISPs, that connect directly to one another in many places – for backup in case of failure. Other smaller domains may pay a larger ISP to provide connectivity for them.

This complex hierarchical interconnection of networks and domains influences the way routing is done in the Internet. Indeed, one set of routing protocols is used for routing within a domain: such a protocol is called an *IGP (Internal Gateway Protocol)*. Another set of routing protocols is used for routing outside of a domain, between different autonomous systems: such a protocol is called an *EGP (External Gateway Protocol)*.

Basically, IGP routing protocols are run completely internally to each domain, and make sure that any node can reach any other node *in the same domain*. Several IGPs may run in the same AS, although most of the time only one is used. On the other hand, EGP routing protocols are the glue that ties the various domains together, to make sure that a user of one domain can reach resources *in other domains*, wherever in the Inter-

net. The next chapter will overview a typical IGP and a typical EGP.

The distinction between IGPs and EGPs is not specific to the routing techniques employed by these protocols. Indeed, these two classes of protocols simply have *different purposes*. On the other hand, the last section of this chapter will distinguish classes of routing protocols with respect to the *different techniques* they employ.

2.7 Routing Techniques

Routing is the process of finding a path from a node to every destination in the Internet. The question is: how to do that in a way that can *scale* to millions of possible destinations, and thus *dynamically*, as some links or routers may experience temporary failures, or new links may be added *etc*.

In this section we describe the techniques employed by Internet routing protocols to achieve their goal in a scalable and dynamical way.

2.7.1 Hop-by-Hop Distribution

Routers in the Internet run *routing protocols*, and these routing protocols establish *routing tables* in each router connected to the Internet. A routing table contains typically two columns: (i) the address of a destination node or subnetwork, and (ii) the address of the network element that is the next hop on the path to this destination. When a packet arrives at a router, the router consults the routing table to decide the next hop for the packet, *i.e.* in which direction is the next step forward for the packet to get to its destination. Thus, routers do not have to store detailed paths/destination information but rather simple directional pointers, and the complete information about an end-to-end path is distributed accross routers, hop-by-hop, along the way.

For example in Figure 2.6, router A is connected to router B and router C. The shortest path from A to D is through B. But on the other hand, from A to E or to H, the shortest path is through C. Figure 2.7, shows the summary of this information for every destination, from router A's point of view. That is: router A's routing table.

The role of a routing protocol is to provide each router with a routing table, such as whatever destination is specified, the router can consult its routing table and be able to perform forwarding, *i.e.* decide in which direction should a specific packet be transfered.

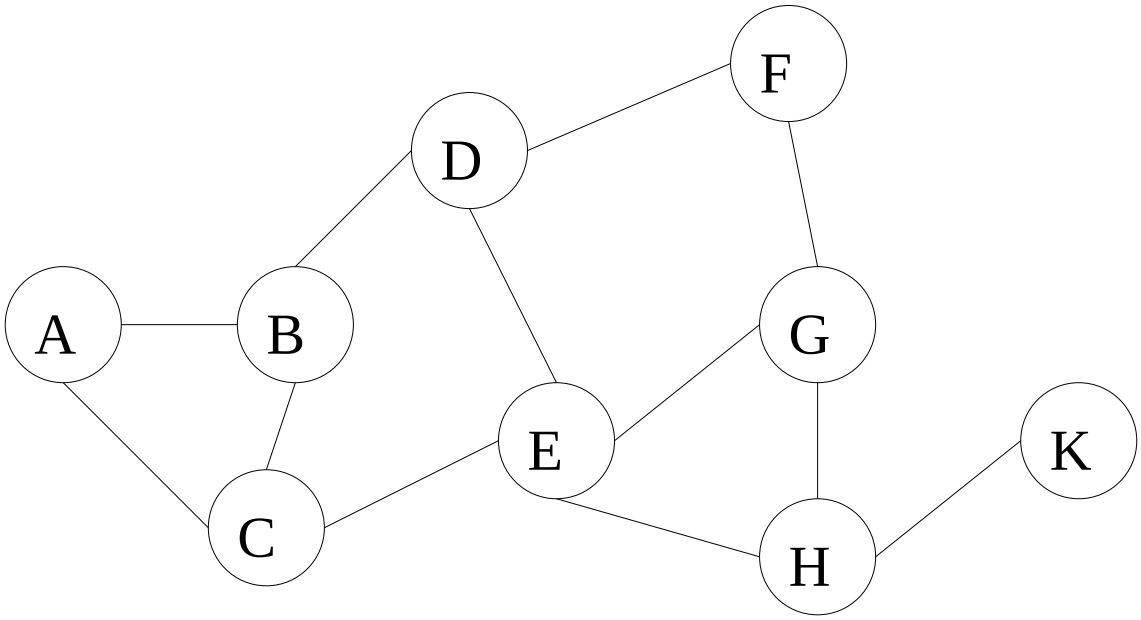


Figure 2.6: Example network.

Routing Table in Router A	
Destination	Next Hop
A	–
B	B
C	C
D	B
E	C
F	B
G	C
H	C
K	C

Figure 2.7: Example routing table.

In order to provide each router with a routing table, a routing protocol must organize the communication of global topological information to each router. Indeed, in Figure 2.6, router A may know of router B because it is a direct neighbor, but router A has no means to know of router D and how to reach it unless some

"topology information" mechanism is put in place. When we look at Figure 2.6, we see the whole topology at once, and therefore we are able to construct paths. It is the purpose of a routing protocol to provide a mechanism that gives a picture of the global topology to each router.

Routing protocols mainly differ in the kind of topology information they communicate to each router. The next section gives a brief overview of the two main classes of routing techniques: *Distance Vector* on one hand, and *Link State* on the other hand. However, whatever the technique used, the following requirements remain:

1. Minimizing routing table space taken in each router.
2. Minimizing the amount of information that needs to be exchanged between routers in order to build routing tables.
3. Minimizing routing errors, even when the network is unstable, and guaranteeing loop-free routes.
4. Using optimal paths, shortest paths.

The use of *optimal* paths supposes the use of a *metric*, with respect to which the paths are optimal. A very natural metric is the hop-count (*i.e.* the number of links used on a path). Another metric can associate different costs or *weights* on different links according to whatever criteria that needs to be optimized. Optimal paths then minimize the sum of the weights of the path. Figure 2.8 and Figure 2.9 show how the routing table is changed when links have different weights. Notice the difference with the routing table in Figure 2.8 (which is equivalent to the case where all the links in the network have the same weight).

The next sections describe the two main routing algorithms that are used in the Internet. Both algorithms assume that a node knows its neighbors as well as the cost of the link to its neighbors. Starting from there, each technique can be summarized as follows.

2.7.2 Distance Vector Routing

Distance vector routing is based on a very simple algorithm, called the *Bellman-Ford* algorithm, which has each router periodically telling its neighbors its "distance" to every other router in the network. In other

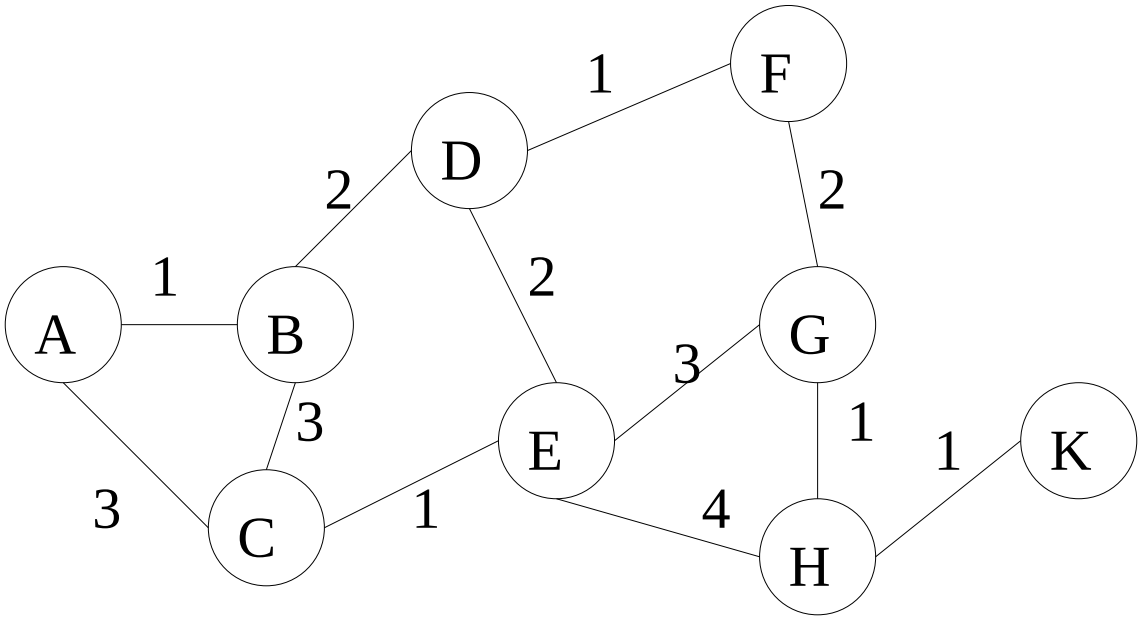


Figure 2.8: Example network with weighted links.

Routing Table in Router A	
Destination	Next Hop
A	–
B	B
C	C
D	B
E	C
F	B
G	B
H	B
K	B

Figure 2.9: Example routing table with weighted links.

words: each router periodically communicates its whole routing table (*i.e.* distance “vector”) to its neighbors. For instance, consider the simple topology in Figure 2.10. The initial routing state is shown in Figure 2.11, where each router knows its neighbors or networks it is connected to.

After the first round of routing information signalling, C learns about A, through B's advertisement. Router C can now include a route to A in its routing table. On the other hand, B learns about the network that A can reach, through A's advertisement, and B can now include a route to this network in its routing table. The routing tables after the first round of signalling are shown in Figure 2.12.

After the second round of information signalling, C learns in turn about A's network connection, through B's updated advertisement. C can now include a route to this network in its routing table. The routing tables after the second round of signalling are shown in Figure 2.13. At this point, all the nodes in the topology have full connectivity, *i.e.* a path to all possible destination – which was the goal that was aimed at initially.

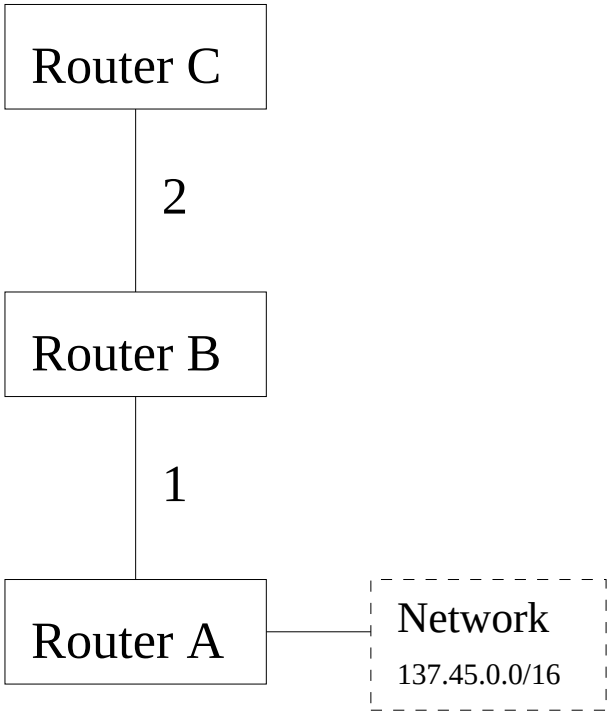


Figure 2.10: Example topology for Distance Vector.

Router A's Routing Table

Destination	Cost	Next–Hop
B	1	B
137.145.0.0/16	0	–

Router B's Routing Table

Destination	Cost	Next–Hop
A	1	A
C	2	C

Router C's Routing Table

Destination	Cost	Next–Hop
B	2	B

Figure 2.11: Initial routing tables in each router.

Router A's Routing Table

Destination	Cost	Next–Hop
B	1	B
137.145.0.0/16	0	–
C	3	B

Router B's Routing Table

Destination	Cost	Next–Hop
A	1	A
C	2	C
137.145.0.0/16	1	A

Router C's Routing Table

Destination	Cost	Next–Hop
B	2	B
A	3	B

Figure 2.12: Routing tables after each router has completed its first distance advertisement.

Router A's Routing Table

Destination	Cost	Next−Hop
B	1	B
137.145.0.0/16	0	−
C	3	B

Router B's Routing Table

Destination	Cost	Next−Hop
A	1	A
C	2	C
137.145.0.0/16	1	A

Router C's Routing Table

Destination	Cost	Next−Hop
B	2	B
A	3	B
137.145.0.0/16	3	B

Figure 2.13: Routing tables after each router has completed its second distance advertisement.

While the advantage of the distance vector algorithm is its simplicity, it suffers from a couple of important drawbacks: (i) the size of each router's advertisement basically grows linearly with the size of the network, which affects the ability of this algorithm to scale for a large topology, and (ii) the distance vector algorithm does not handle certain failure situations very well.

One simple case where the algorithm runs into problems is the following: suppose that in the topology shown in Figure 2.10 the link between router A and router B goes down. Router B then immediately knows that A and its network are now unreachable. But on the other hand, Router C does not know that immediately, and therefore C continues to advertise that it has a route to A and its network. If C's ignorant advertisement reaches B before B's updated advertisement reaches C, a situation is created where B will think that it can in fact reach A through C. At the same time, C will learn from B's advertisement that A is unreachable. But during the next round of advertisements, B and C's roles are reversed: B will advertise a route to A and C will then believe that it can in fact reach A through B. At the same time, B will then be convinced that A is unreachable. This abnormal ping-pong situation can continue a long time before the routers realize that A is indeed unreachable. This behavior is called the *count-to-infinity* problem.

Some fixes have been developed to patch the distance vector algorithm so that it behaves better when facing the count-to-infinity problem. However, using a radically different algorithm may be a better solution in some cases where the topology is large and subject to more frequent changes. The next section describes such an algorithm: the *link state* algorithm.

2.7.3 Link State Routing

In contrast with the distance vector algorithm, the link state algorithm is based on the idea that each router periodically tells every other router in the network its distance to its neighbors. This neighborhood information is communicated through *LSPs (Link State Packets)*, each of which essentially containing (i) the identity of the node that originated the LSP, (ii) the list of neighbor routers and networks connected to the node, and (iii) the cost of these links. Each router in the network periodically generates an updated LSP describing its neighborhood at the moment.

An algorithm called *flooding* then ensures that each router's LSP is delivered to every other router. A simple example of flooding algorithm is the following: when a router receives a new LSP, it stores the packet and then forwards it to all its neighbors, except to the one(s) from which it received the packet. This way, starting from its source, the packet is delivered hop-by-hop to each and every router. This algorithm is called *classical flooding*.

The set of up-to-date LSPs of all the routers forms a database that can be used to construct a map of the entire topology. This database of LSPs, called the *Link State Database* is in fact present in each router, after all the routers have generated and flooded their own LSP. Thus, each router can then independently construct its own map of the whole topology, and therefore compute optimal routes to every possible destination in the network.

For example, if we consider the example topology shown in Figure 2.14, the LSPs then generated by routers A, B, C and D are given in Figure 2.15. The link state data base in this case consists in these 4 LSPs, and after all these are flooded by their respective originator, each router has a copy of every LSP. Therefore the same link state database is present in each router, and from the information contained in the link state database, each router can derive (i) the identity of all the nodes in the network, (ii) the connections between the nodes, and finally (iii) the optimal routes to every node in the network.

A router can compute optimal routes through the use of *Dijkstra's shortest path algorithm* over its link state database. Dijkstra's algorithm computes the shortest paths from a *root* (*i.e.* the router where the algorithm

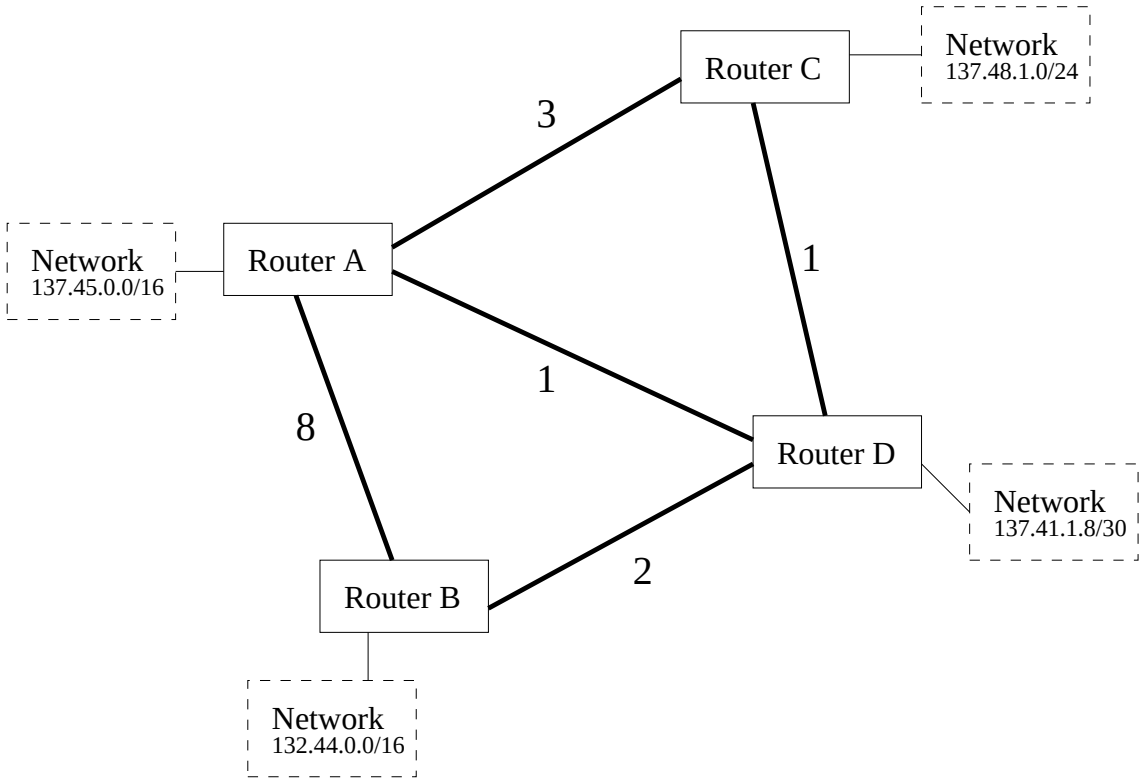


Figure 2.14: Example topology with link state routing.

is being run) to every other node in the network, through a process that can be summarized as follows. The idea is to maintain a set of nodes, S, for which the shortest paths have already been found – initially, this set contains the root and its direct neighbors. The algorithm then explores every way in which a given node n outside of S can be reached by a one-hop path, from a node that is already in S. The shortest of these is chosen as the path to n and n can then be added to S. The algorithm continues in such a fashion, adding new nodes to S until the shortest paths to all the nodes in the network are obtained.

The exact algorithm is described in Appendix B. Figure 2.16 shows an example topology and the corresponding shortest path tree computed from Router 1.

Even though conventional wisdom tends to consider link state protocols as more stable than distance vector protocols, routing loops can also form with link state routing depending on the topology, because of discrepancies between link state databases in different routers. This may happen if the network is very dynamic, with links constantly coming up and down. However, one clear advantage of link state routing over distance vector routing is that each node running a link state protocol possesses information about the entire network

LSP for Router A	
Neighbor	Cost
137.45.0.0/16	–
B	8
C	3
D	1

LSP for Router B	
Neighbor	Cost
132.44.0.0/16	–
A	8
D	2

LSP for Router C	
Neighbor	Cost
137.48.1.0/24	–
A	3
D	1

LSP for Router D	
Neighbor	Cost
137.41.1.8/30	–
A	1
B	2
C	1

Figure 2.15: The LSPs generated by each router.

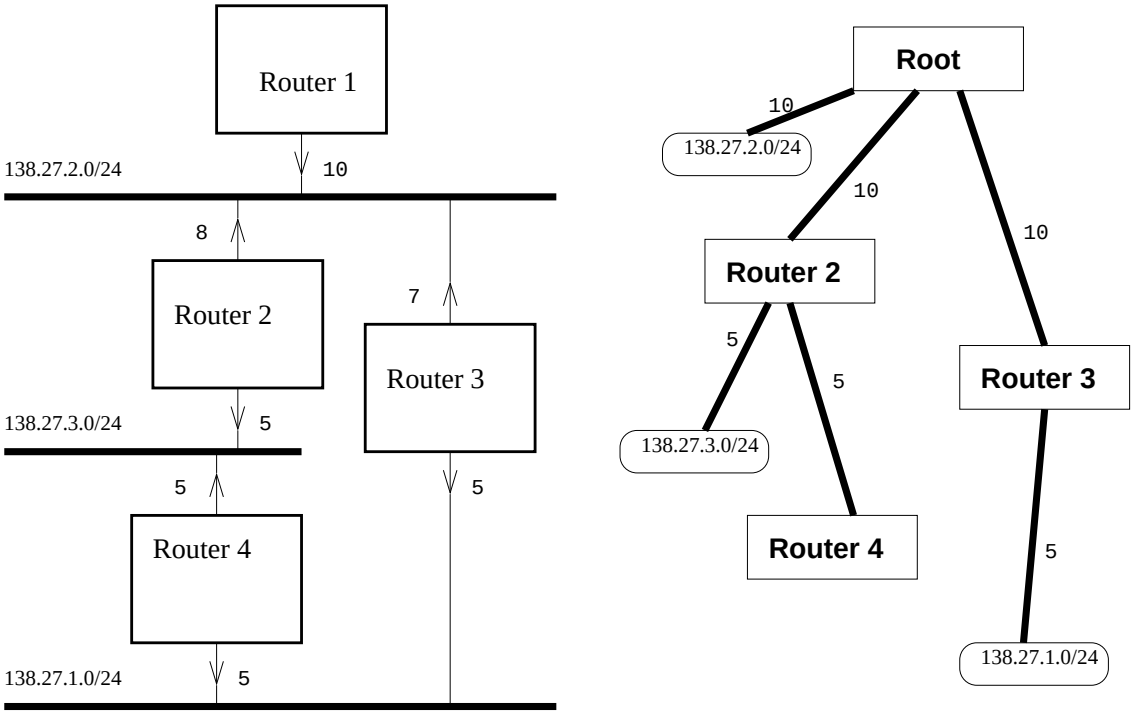


Figure 2.16: Example of shortest path tree computed with Router1 as root.

topology. This knowledge can indeed be very useful, even for purposes other than routing, as we will see throughout this document. Apart from this point, both distance vector and link state routing have their merits in terms of performance, robustness *etc*. Therefore both are actively used today on the Internet. The next chapter details OSPF, a typical link state protocol, and also overviews BGP, a typical distance vector protocol.

Chapter 3

Internet Routing Cornerstones: OSPF and BGP

In this chapter, we introduce the two routing protocols that are essential to today's Internet functionning. OSPF as the most common IGP, and on the other hand BGP, as the most common EGP. Moreover, also with respect to the categories defined in Chapter 2: OSPF is the archetype link state routing protocol, while BGP is based on the distance vector algorithm.

3.1 BGP

BGP (Border Gateway Protocol) is the predominant EGP used in the Internet. The version of the protocol currently in use is version 4 (BGP4), documented in [62]. While BGP is essentially a distance-vector algorithm, it features several additional mechanisms on top of this base in order to adapt and perform its task: routing between Autonomous Systems, at the top level of Internet's hierarchical organization, and carry full Internet routing. In this document we do not focus on EGP routing, therefore, only a very brief overview of BGP is given here. For a more complete introduction to BGP, refer to [61], or to the protocol's specification [62].

3.1.1 Peer-to-Peer Sessions

The first particularity of BGP is that it breaks down networking into one-to-one relationships: the base of BGP functionning happens between exactly two nodes: two neighbor routers running BGP, that are then called BGP *peers*, or *speakers*. BGP uses TCP to set up a reliable transport "session" between BGP peers. Therefore, BGP starts by establishing as many BGP sessions as there are links between the routers running BGP. Each session involves only two routers – but a router may be participating in many sessions at the same time, as many as the links it has with other BGP peers.

This approach is adapted to the fact that at the domain level, organizations such as large ISPs want to negotiate and control their connectivity on a one-to-one basis independently with peer ISPs, or with customers (such as smaller domains or ISPs) to whom they sell connectivity.

At the start of a BGP session, two BGP peers exchange complete copies of their routing tables, which can be quite large. However, from then on, only changes in these routing tables are exchanged, instead of again whole routing tables. In general, an information communicated by a neighbor is considered valid until it is explicitly invalidated by this neighbor, or until the BGP session with this neighbor is itself lost. Long running BGP sessions are therefore more efficient than shorter ones. This is adapted to the types of links used to connect Autonomous Systems, which are usually top capacity, long-lived links, which experience failures extremely rarely. This difference put aside, the traditional distance vector mechanisms are used to communicate routing information to each router in the network.

3.1.2 BGP Messages, Paths and Attributes

Messages passed between peers include (1) **OPEN messages**, to open the BGP session, (2) **UPDATE messages**, to inform the neighbor about new routes that are active, or about old routes that are no longer active, (3) **NOTIFICATION messages** to report possible unusual conditions before closing a BGP session, or (4) **KEEPALIVE messages** to inform the neighbor that the connection is still valid.

BGP's basic unit of routing information is the BGP *path*, a route to a certain set of IP prefixes. Paths are tagged with various *attributes*, which carry a wide range of information. The most important attributes are *AS-PATH* and *NEXT-HOP*. Basically, the AS-PATH attribute is the list of Autonomous Systems that a route

goes through to reach its destination (each AS is identified by a number). Loops are detected and avoided by checking for one's own AS number in AS-PATH's received from neighbors.

On the other hand, the NEXT-HOP attribute is the IP address of the first router to be reached in the next Autonomous System on the path. This may actually be several hops away if the next AS is connected on the "other side" of the currently traversed AS. The IGP used inside the latter is then used to route through the AS to reach the BGP NEXT-HOP. If necessary, the IGP is also used for relaying a BGP routing update received from a neighboring AS, to all the BGP speakers in the AS.

Several other path attributes are used by BGP. These include (i) various kinds of metrics specifying degrees of preference for the route, (ii) descriptions of the way an advertised prefix entered to routing table at the source AS, or (iii) other kinds of information added over time to bring new features in the protocol. This extensibility has allowed BGP to accomodate the Internet's growth and changing demands.

3.1.3 Policy Routing

It is the responsibility of the BGP implementation to select among competing paths using a nearly completely undefined algorithm. The specification [62] states only that the computation should be based on "preconfigured policy information", the exact nature of this policy information and the computation involved being a "local matter". Therefore BGP and its policies reflect the individual peer-to-peer agreements at the ISP level. A simple policy may impose that certain paths traverse certain trusted AS, while others must avoid some other untrusted AS. This is easily achieved with the manupulation of the AS-PATH attribute. However, more complex policies may also be used.

This document does not focus on EGP routing or on BGP, and therefore BGP will not be detailed further here. For more information on BGP, refer to [61] [62]. In the following chapters, we will focus on IGP routing, and we will start by describing in the next section the most commonly used IGP in the Internet today: OSPF.

3.2 OSPF

OSPF (Open Shortest Path First) is the most common IGP in today's Internet. The version of the protocol currently in use is version 2 (OSPFv2), documented in [34], which is compatible with IPv4 specifications. Another version of the protocol (OSPFv3) that is compatible with IPv6, is currently being developed. However, the predominant version is still OSPFv2, and therefore, in the following, OSPF will generally mean OSPFv2. Moreover, most of the principles that will be described here for OSPFv2 apply to OSPFv3 as well.

OSPF was developed due to a need to replace RIP (Routing Information Protocol [75]), another routing protocol based on the distance vector algorithm. RIP used to be the most common IGP before OSPF was established, but at some point was found to be limited and not adapted to "modern" Internet.

OSPF's design was on the other hand based on a link state algorithm. As such, a router running OSPF periodically generates and floods LSPs (see Section 2.7.3) called here *LSA (Link State Advertisements)*. Each LSA, describes the links that a router has with its neighbor routers or networks. Each router is therefore delivered updated LSAs of every router in the network, and is able to form a link state database that it can use to compute its routing table, with the Dijkstra algorithm (see Section 2.7.3).

However, in addition to the link state algorithm, OSPF uses several other mechanisms supplementing this base. These features enable OSPF to adapt to Internet's growing use of various layer 2 technologies, to scale to larger topologies, and to accelerate convergence.

3.2.1 Reliable Flooding, Sequence Numbers and Aging

In order to further protect the network against the formation of loops because of discrepancies between the link state databases present in different routers, OSPF provides special mechanisms that aim at ensuring that databases are consistent throughout the network.

Each LSA therefore includes a *sequence number* that is supposed to identify this message if coupled with the address of the router that originated the LSA. However, this identification is not unique, since (i) sequence numbers cannot actually grow to infinity, and (ii) a router that has crashed and that comes back online may not be able to "remember" which sequence numbers it was currently using before it crashed. Special and complex mechanisms are thus used to cope with such problems, called *wrap-arounds*. These sequence numbers are used to determine which information is valid: when a node has received two LSAs originated by

the same router, it can select the LSA with the newest sequence number as the up-to-date information, and discard the other LSA as out-dated.

Moreover, each LSA includes an *Age* field, that specifies how long the information contained in the LSA should be considered valid. The age of an LSA cannot exceed a certain value called *MaxAge* after which the information is considered invalid. This provides a mechanism to eliminate out-dated information from the link state databases of every router in the network.

Finally, sequence numbers are also used to provide *reliability* in flooding. A node that forwards an LSA to a neighbor expects an *acknowledgement* from that neighbor which confirms that the neighbor has indeed correctly received the LSA. Acknowledgements are special OSPF messages that basically list a set of {Originator Address, LSA Sequence Number} tuples that a node has recently received. A node that receives an LSA is supposed to send an acknowledgement containing a corresponding tuple, to the neighbor which sent the LSA, and such within a certain timeframe. Passed this timeout, the neighbor will retransmit the LSA, to make sure that the node receives the LSA correctly. Some special procedures are also in place, aiming at suppressing redundant acknowledgements, using other messages as implicit acknowledgements, or determining the number of retransmissions attempts before giving up in face of non-acknowledgement. These procedures incur some additionnal overhead, but on the other hand flooding is ensured to be as reliable as possible. This means that every node receives a flooded LSA, as each forwarder makes sure that the nodes down the stream did receive correctly the LSA (via the acknowledgement/retransmission procedures). This feature aims at guaranteeing that link state databases are indeed the same in every router in the network, therefore improving the protection of routing against unwanted loops.

3.2.2 Overhead Optimizations with Interface Types

The Internet spans over many different mediums using various layer 2 technologies, and yielding somewhat different properties. OSPF therefore categorizes the different mediums over which it may run, aiming at taking advantage of the specificities of each category. These categories are called *interface types*, as OSPF features some specific mechanisms to be run on a per-interface basis, depending on the type of medium the interface connects.

For example, some mediums have an interesting property called *broadcast*. This characteristic means that a message sent on a broadcast medium is heard by all the nodes that are connected to this medium. This is the

case with Ethernet for instance: if a node successfully sends a message on an Ethernet link, all the nodes on this network will automatically hear the message. This property can be used to optimize OSPF functionning on this type of link, when many routers are connected by the same broadcast medium, which is often the case.

OSPF therefore defines an interface type called *broadcast interface*, with special mechanisms to be run on this type of connection. Indeed, several OSPF routers connected by this type of medium will self-organize into a virtual hub-and-spokes topology. One router is elected as the center of this virtual topology, and will coordinate OSPF functionning on this medium. This router is called the *Designated Router (DR)*. Fig. 3.1 displays an example of such emulation.

By emulating this virtual star-shaped topology where each router has only one link, connecting it to the

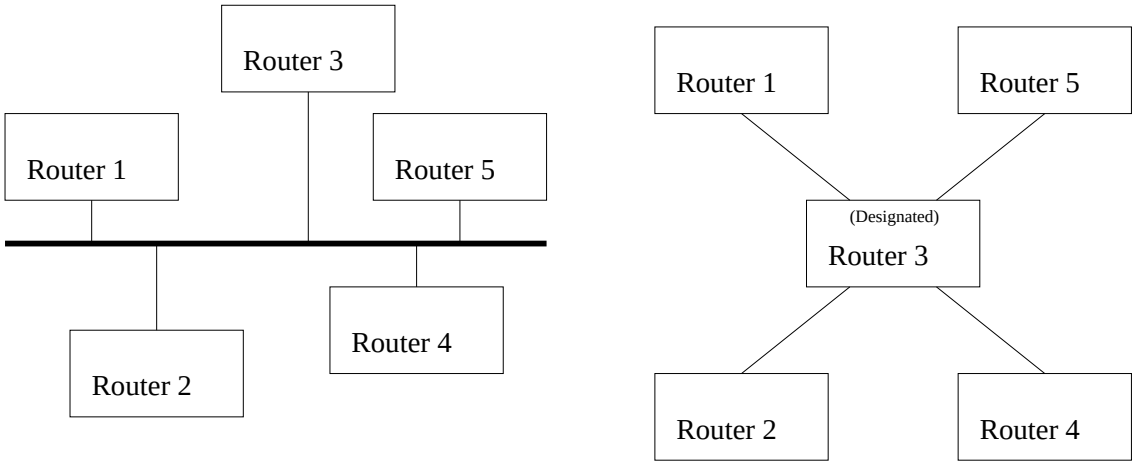


Figure 3.1: Example of Designated Router emulated topology. The routers are physically all connected by the same broadcast medium (on the left). The emulated topology is star-shaped, centered on the Designated Router, Router 3 in this example (on the right).

designated router (instead of the actual topology: each router is in fact indeed connected to every other router on the medium), the amount of needed link state information and signalling can be reduced, while full connectivity is still preserved. This is the case because the emulated star-shaped topology is itself optimal, at the expense of being centralized – and therefore more fragile.

In order to attenuate the incurred fragility, another router on the broadcast medium is also elected, as *Backup Designated Router (BDR)*. The role of the BDR is to always be ready to immediately take over the central job of the designated router, should the latter fail or crash.

Several other interfaces types, and respective specific mechanisms are defined in OSPF for different categories of connection: Point-to-Point, Point-to-MultiPoint, Non-Broadcast Multi-Access (NBMA), or Virtual Link. For more information on these, refer to [34] [35].

3.2.3 Convergence Acceleration with the Database Exchange Mechanism

An important part of the performance of a routing protocol is the time it takes to *converge*, *i.e.* the time it takes to bring the network from its initial dysfunctional state to its fully functional state. OSPF features a mechanism to accelerate this convergence in some cases: the *Database Exchange*.

OSPF database exchanges are intended to synchronize the link-state databases of routers throughout the network. This mechanism runs on a peer-to-peer basis, messages being exchanged between two neighbors, with one node (the master) polling, and other node (the slave), which responds. Both polls and responses have the form of "database description" messages, listing a set of {Originator Address, LSA Sequence Number} tuples, accurately describing parts of the link-state database of the node that originated the message. These messages enable neighbors to compare their link-state databases. If any of the two nodes involved in a database exchange detects it has out-of-date or missing information, it requests these pieces of information from the other node, which immediately updates its link-state database.

This database exchange mechanism can quickly jump-start a new router that just came online, and also offers a better guarantee that link state databases are indeed the same throughout the OSPF network, which is essential for a link state routing protocol to function correctly. However, it comes to the expense of more traffic control. Therefore, some optimizations are also in place, specifying who should synchronize with whom using this mechanism. For instance, in the case of a broadcast link with a designated router, each router on the medium synchronizes only with the designated router, and not with the other routers.

3.2.4 Scaling with Hierarchical Routing with Areas

In order to scale to large topologies, an efficient approach is to introduce an abstract hierarchy and different levels of knowledge. This approach is taken in the Internet in general, as we have seen with its organization into independent Autonomous Systems and EGP *vs* IGP routing (see Section 2.6). OSPF uses a similar technique to scale for large domains, called *Area Splitting*.

With the link state algorithm, any change, even minimal, has to be propagated throughout the whole net-

work via the flooding of one (or more) LSA, and Dijkstra has to be re-processed in every router, over the updated link state database. However, if the topology is sufficiently large, a small change on one "side" of the network does not change anything to the routing tables of the nodes on the other side of the network. In this case, it is interesting to minimize the overhead due to such link state updates. OSPF therefore splits a large domain into different parts, or *areas* (see Figure 3.3).

Area Splitting Terminology

An area is defined on a per-interface basis. A router can therefore belong to multiple areas if it has interfaces in more than one area – such a router is then called an *Area Border Router (ABR)*. If on the other hand all the interfaces of a router belong to the same area, it is called an *Internal Router (IR)*. A router that is a gateway between OSPF and other routing protocols (for instance BGP) is called an *Autonomous System Border Router (ASBR)*, as shown in Figure 3.2.

On the other hand, routes that are within the scope of a single area are called *intra-area routes*, whereas routes which span over several areas are called *inter-area routes*. Some routes may also be imported from informations originated by another routing protocol: an EGP such as BGP, or an IGP such as RIP, if several IGPs are used in the AS. These routes are called *external routes*. In case of the availability of multiple routes for a same destination, OSPF is configured to prefer intra-area routes over inter-area routes, and inter-area routes over external routes.

Hierarchical Routing with Areas

Areas gather around a central area, called *area 0* (or *backbone*) and different areas communicate *via* the backbone – using again a hub-and-spokes approach. LSA flooding stops at area boundaries, and boundary routers are responsible for controlling information leaking between areas. Each area injects its "summarized" link state information in the backbone via its boundary routers, and the backbone controls the injection of this information into the other areas (see Figure 3.3). This way, if a change happens in one of the areas, it affects only the routers in this area, and if the change is minimal, it doesn't change anything to the summarized information that is injected in the backbone, and therefore it does not affect any router outside the area. Basically, the way areas inject routing information in the backbone is the following:

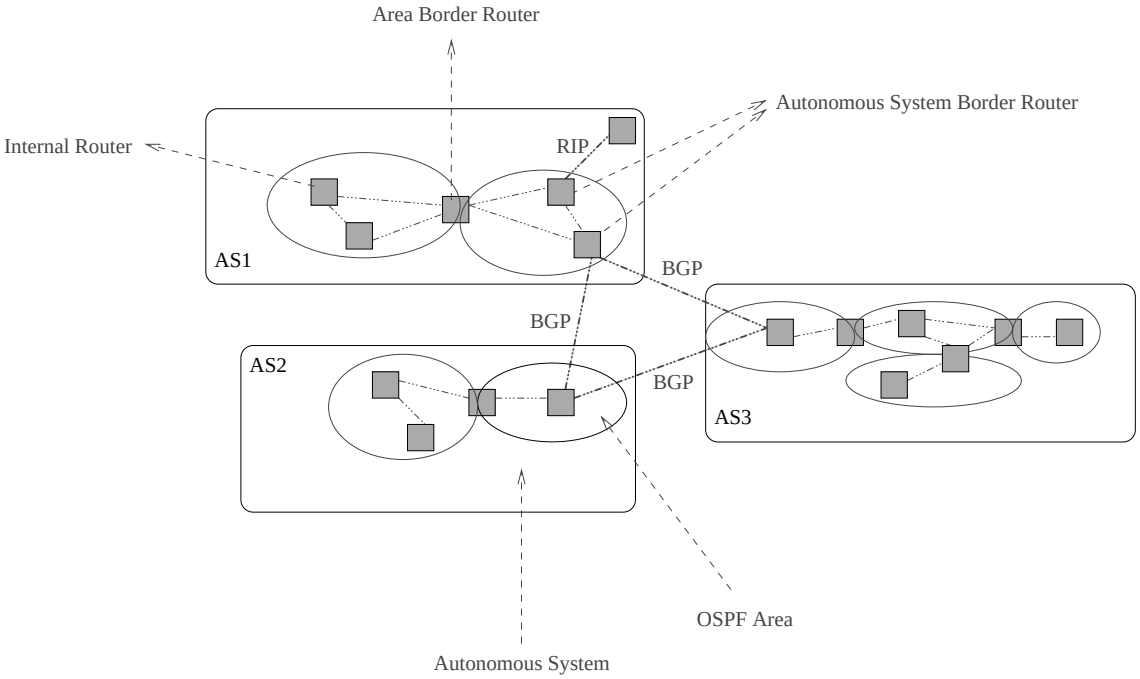


Figure 3.2: Autonomous System splitting and router denomination.

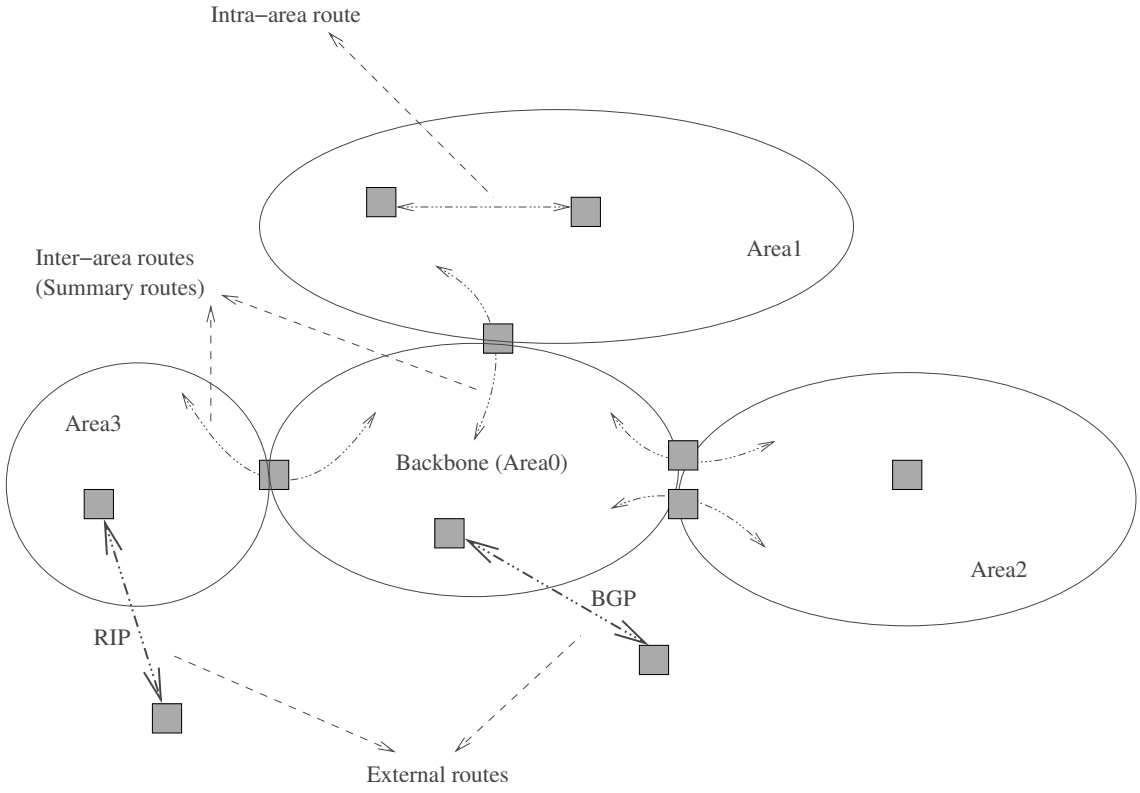


Figure 3.3: Area splitting and route denomination.

1. Each router in the area computes its own intra-area routes, with the usual link state algorithm.

2. Each ABR in the area aggregates information concerning routes from itself to all the destinations in the area. For instance, if the subnet 121.121.121.0/24 is in the area, instead of advertising every single route to every single interface on this subnet, the ABR will aggregate this information in a single advertisement: a route to the subnet 121.121.121.0/24. Similarily, if several subnets can be aggregated under a more general prefix, further aggregation is possible. Such an advertisement is called a route *summary*.

3. Each summary is advertised with a special kind of LSA: a *summary LSA*. A summary LSA advertises a route from the ABR to a set of destinations in the area, with a cost that is set to the maximum cost from the ABR to any of these destinations. For example, let interfaces 121.121.121.54 and 121.121.121.37 be the interfaces on the summarized subnet 121.121.121.0/24, and the costs from the ABR to 121.121.121.54 and 121.121.121.37 be 100 and 200 respectively. Then the ABR would generate a summary LSA that advertises the aggregate 121.121.121.0/24 with weight 200.

4. Each summary LSA is reliably flooded in the backbone and thus, each ABR in the backbone can maintain a "summary database" containing all the summary LSAs from all the ABRs in the backbone – in addition to the usual link state database.

5. Each ABR advertises, in its own area, routes to prefixes from other areas. In case of multiple routes to these "foreign" prefixes it chooses to advertises the shortest one.

OSPF specifications do not impose the way aggregation should be done, so different levels of summarization may be used. The advantages of summarization are those of hierarchical routing: better scalability, as the routing information is smaller and less prone to change. However, this comes to the expense of a loss in the precision of this information, which can lead to using sub-optimal routes in some cases.

3.2.5 OSPF Applicability

Because of its numerous specific mechanisms and optimizations, OSPF is able to adapt to most of the communication mediums used on the Internet today and to scale to large topologies. However, these mechanisms are rather complex, which makes this protocol less simple than some other protocols. Nevertheless, OSPF is widely deployed and used throughout the Internet, as it is the most commonly used IGP.

For more complete information about OSPF, refer to [35] or to the actual specifications [34].

Part II

Routing Challenges with Mobility in the Internet

Chapter 4

Edge Mobility

In parallel with the explosion of the Internet, another phenomenon has fundamentally been changing the aspect of networks since a number of years: the generalization of wireless communication with mobile devices, such as cellular phones (see Figure 4.1 and Figure 4.2), or portable computers with WLAN equipment, with already tens of millions of WiFi "hot-spot" users world-wide. Thus, there is nowadays a natural demand to fully integrate both phenomena, and to extend the reach of the Internet to mobile wireless nodes, hosts, and routers.

Date	Estimated Number of Mobile Phone Users
2000	more than 300 million
2002	more than 600 million
2004	more than 1 billion

Figure 4.1: Recent growth in the number of mobile phone users worldwide (Source: Gartner).

However, the first part of this document introduced to routing problematics in the Internet based on the assumption that nodes in the network are static: nodes, if online, are indeed expected to connect to the network from a specific, pre-determined geogaphic location.

If this assumption is waived, *i.e.* some nodes can now move about "at will" and connect to the Internet

Date	Estimated Number of Mobile Phone Sales
2001	399 million units
2002	423 million units
2003	519 million units
2004	674 million units

Figure 4.2: Recent growth in the number of mobile phone sales worldwide (Source: Gartner).

from different locations at different times depending on their mobility, a new challenge is set with the task of routing. Indeed, as seen in Part 1, IP routing assumes that the IP address of a node is tied to its location. Therefore, a node that changes location must also change its IP address. However, a node's IP address is also its identity. Thus, the point of view of the other nodes in the Internet is that someone that stops using its usual IP address has in fact disappeared and is now unreachable. So how can a mobile node thus change its location and still keep its identity? The following chapters will discuss this new challenge for different kinds of node mobility: on one hand edge mobility, and on the other hand, ad hoc mobility.

Edge mobility has been explored for a number of years and several solutions have gained both experience and popular consensus – some of these are now integrated in Internet's functionning. On the other hand, ad hoc mobility is a rather new domain, and solutions in order to integrate this new type of mobility in the Internet are still being discussed.

This chapter introduces edge mobility and some solutions that were developped in this domain, while the remainder of this document will focus on ad hoc mobility.

4.1 Introduction to Edge Mobility

In a network, edge mobility describes the freedom of some nodes to (i) be mobile along the edge of the fixed part of this network, and when necessary (ii) change from time to time their *point of attachment – i.e.* the geographical location of their connection to the network.

Typical examples of networks allowing edge mobility include cellular phone networks. Mobile phones connect to the network via the nearest available base-station. Such base stations are disseminated all over the

place, so that if a user moves too far from the base-station it currently attaches with, it will come to be in reach of another base-station – which will then become its current point of attachment to the phone network. In this case, the edges of the fixed part of the network are the base-stations, and the mobile nodes are the cell phones and their users.

There is however a clear limitation to the movement of the mobile nodes. Indeed, mobile nodes must at all times stay in direct reach of elements of the fixed edge. Further than the fixed edge's range, a mobile node is disconnected from the network. Hence the denomination of this category of mobility: *edge mobility*. For example, with a cell phone, if a user moves away from the coverage zone (*i.e.* away from its current base-station but not closer to any other base-station), the user will be disconnected and any phone conversation will be interrupted – even if he was communicating with another user who is a few meters away from him, but who is still in the range of the base-station. This situation is illustrated in Fig. 4.3.

Some solutions have been developed to enable such edge mobility in the Internet. The following sections overview some of these solutions.

4.2 Mobile IP

The requirements that are at the base of Mobile IP are that (i) a node must be able to communicate with other nodes after changing its link-layer point of attachment to the Internet, (ii) a node must be able to use its home address to communicate, despite the current link layer point of attachment to the Internet, and (iii) a node must be able to communicate with other nodes that may not implement Mobile IP.

One solution fulfilling these requirements and enabling nodes to move along the edge of the Internet, is to mimic the way the post office offers mail forwarding services for people who have temporarily moved away from their home. If a customer requests such a service, its home post office just needs to be specified the temporary address of the customer, and will (for a while) forward all incoming mail intended for the customer to the temporary care-of address specified by the customer. This is basically the approach taken by Mobile IP to enable host edge mobility[39].

With Mobile IP, as a host moves along the edge (see Figure 4.4), it acquires a succession of temporary care-of addresses that it communicates to a server located in its usual home network, called its *home agent*. The home agent is then able to forward IP packets intended for the mobile node to its temporary location.

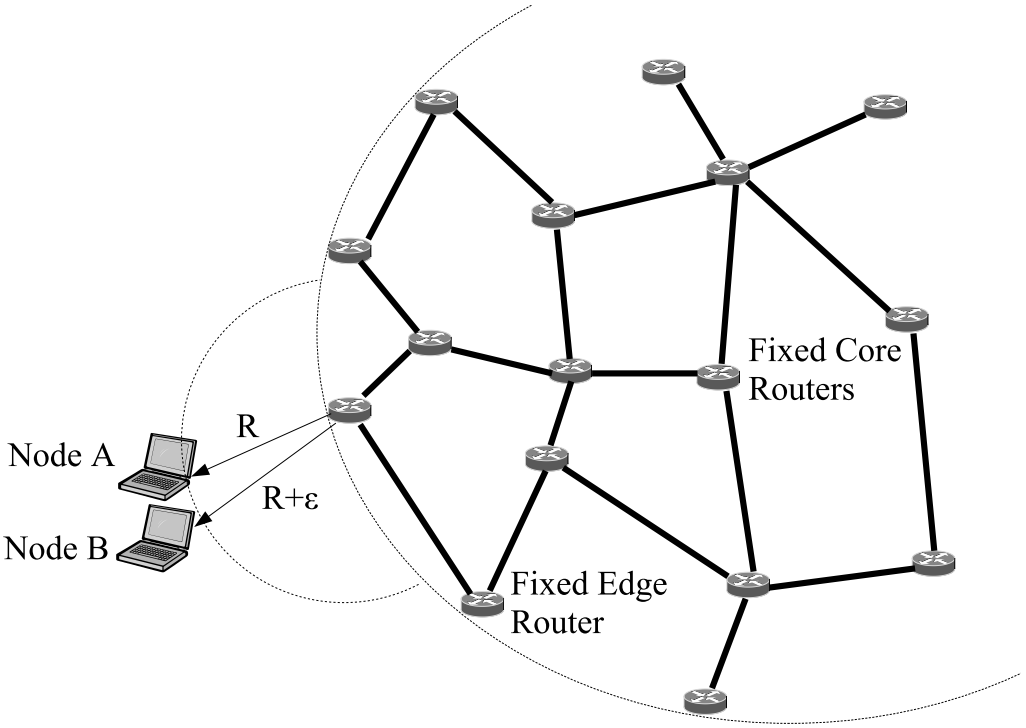


Figure 4.3: Edge mobility limitation. Node A and node B cannot communicate even though they are geographically very close to each other, as node B is slightly out of range of the fixed edge, while node A is still in range.

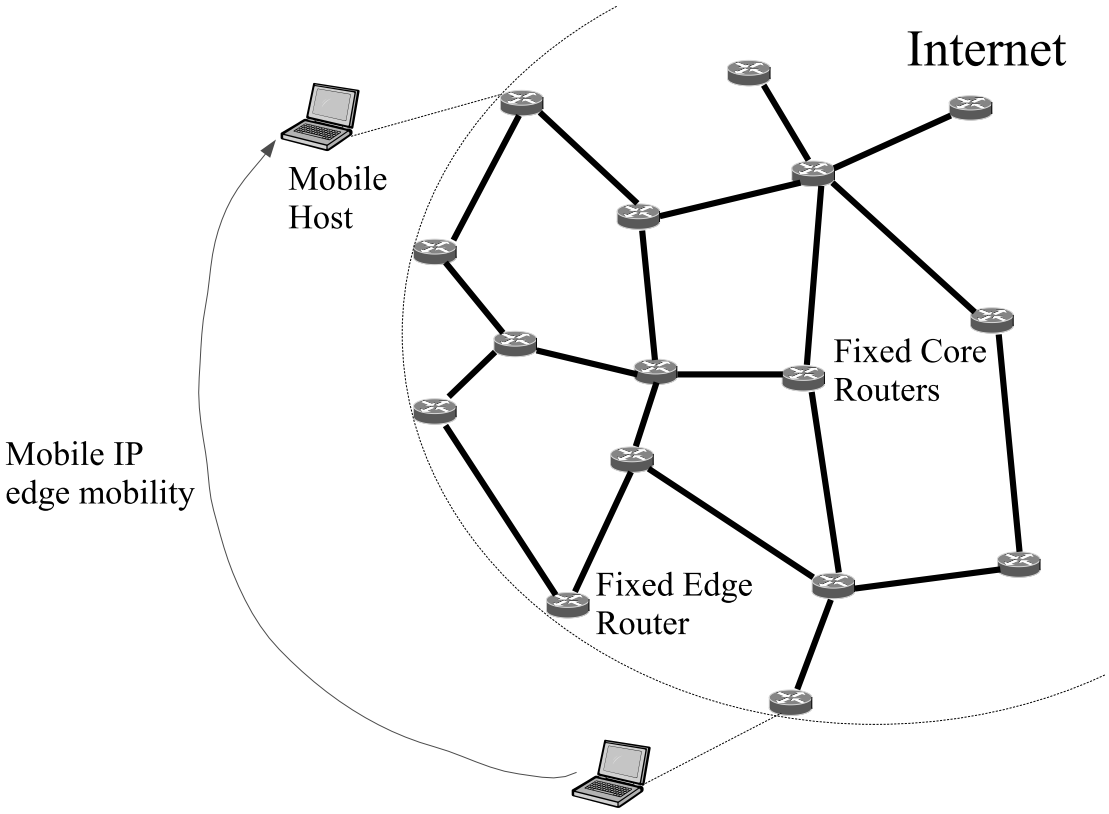


Figure 4.4: Mobile IP edge mobility: a host is mobile along the fixed edge.

More precisely, the general operation of Mobile IP can be summarized as the following series of steps:

1. **Agent Communication –** A mobile node identifies a Mobile IP server on its local network by engaging in a discovery process. This server is generally a router which implements Mobile IP, and which is then called an Mobile IP agent. A mobile node listens for AGENT-ADVERTISEMENT messages sent out by agents. From these messages, a mobile node can determine where it is located. If it does not currently hear such messages, a mobile node may ask for one by sending out an AGENT-SOLICITATION message on the network.

2. **Mobile Node Location Determination –** A mobile node determines whether it is on its home network, or a foreign network, based on the information included in AGENT-ADVERTISEMENT messages. If a mobile node is in its home network, it does not do anything special. On the other hand, if the mobile node is in a foreign network, it performs the following additionnal steps.

3. **Care-Of Address Acquisition –** The mobile node obtains a temporary address called a care-of address. This address can either come from an AGENT-ADVERTISEMENT message from an agent (then called the foreign agent), or through some means external to Mobile IP. This address will be temporarily used to supplement the mobile node's original home address. The home address always keeps the identity of the mobile node, while successive care-of addresses keep track of its location.

4. **Agent Registration –** The mobile node informs the home agent on its home network of its present location on the foreign network and enables packet forwarding, by sending a REGISTRATION-REQUEST to the home agent. The home agent confirms with a REGISTRATION-REPLY message to the mobile node. This exchange may be done either directly between the mobile node and the home agent, or indirectly through the foreign agent.

5. **Packet Tunneling –** The home agent intercepts IP packets intended for the mobile node as they are routed to its home network, and forward them to the current location of the mobile node. This is done by encapsulating these packets, *i.e.* basically wrapping them in new IP packets with the care-of address as destination and sending them to the node's whereabouts. When the wrapped packets arrive to the mobile node they are stripped of their encapsulation and delivered as if the mobile node was indeed in its home network. Fig. 4.5 shows an example of such forwarding.

6. **Periodic Registration Renewal –** Packet encapsulation and forwarding continues until the current registration expires, unless the mobile node renews its registration for the same location or for a new location. When the mobile node returns to its home network, it deregisters to cancel datagram forwarding and resumes normal operation.

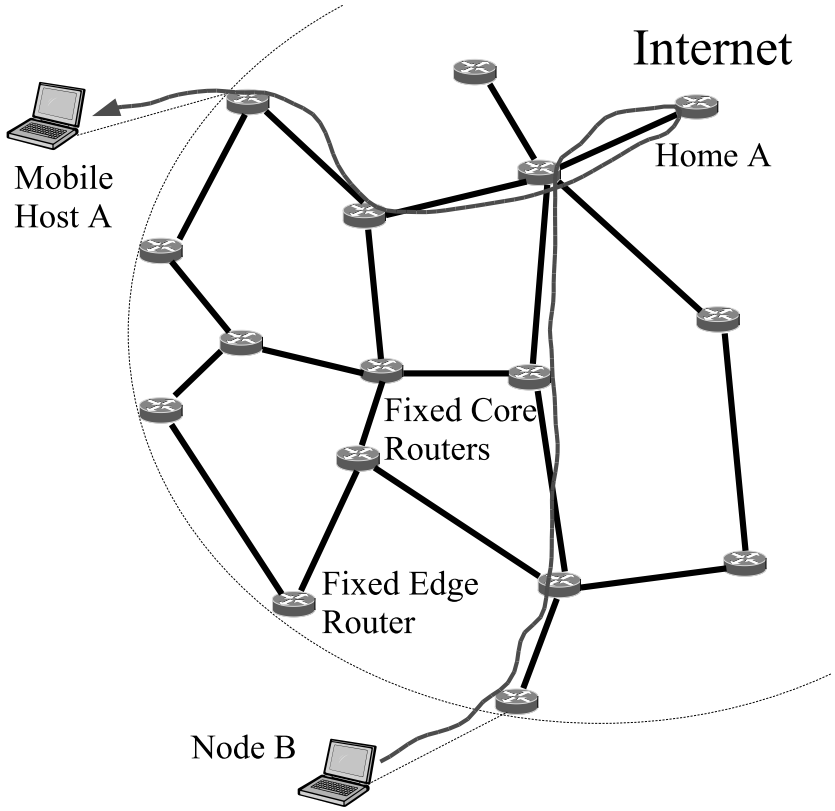


Figure 4.5: Mobile IP tunneling. Node B sends a packet to mobile node A. The packet is routed to the Home Agent of node A which then tunnels it to the current location of node A. The path through the network is highlighted.

An optional feature called reverse tunneling may also be used in some cases, such as when the foreign network does not allow outgoing IP packets with a foreign source IP address. When enabled, rather than sending packets directly to the destination, the mobile node tunnels all transmissions back to the home agent, which will strip them and send them on the Internet.

4.2.1 Performance Issues with Mobile IP

Since IP packets are always sent to a mobile node at its home address, each packet sent to the mobile node must first go back to its home network before being forwarded to the mobile node's current location. The level of inefficiency that results depends on the location of the sender. If the sender is near the home network of the mobile node, the path used by Mobile IP to reach the mobile node is merely just slightly suboptimal. However, if the sender is located elsewhere, the path used by Mobile IP may become substantially less optimal. The worst case actually occurs if the sender and mobile are on the same foreign network, in which case each transmission must make a round-trip to the mobiles home network and then back again.

Moreover, the encapsulation scheme used by tunneling also implies an additional overhead which may be substantial in some cases, especially if reverse tunneling is used. Section 6.4 will further detail these inefficiencies.

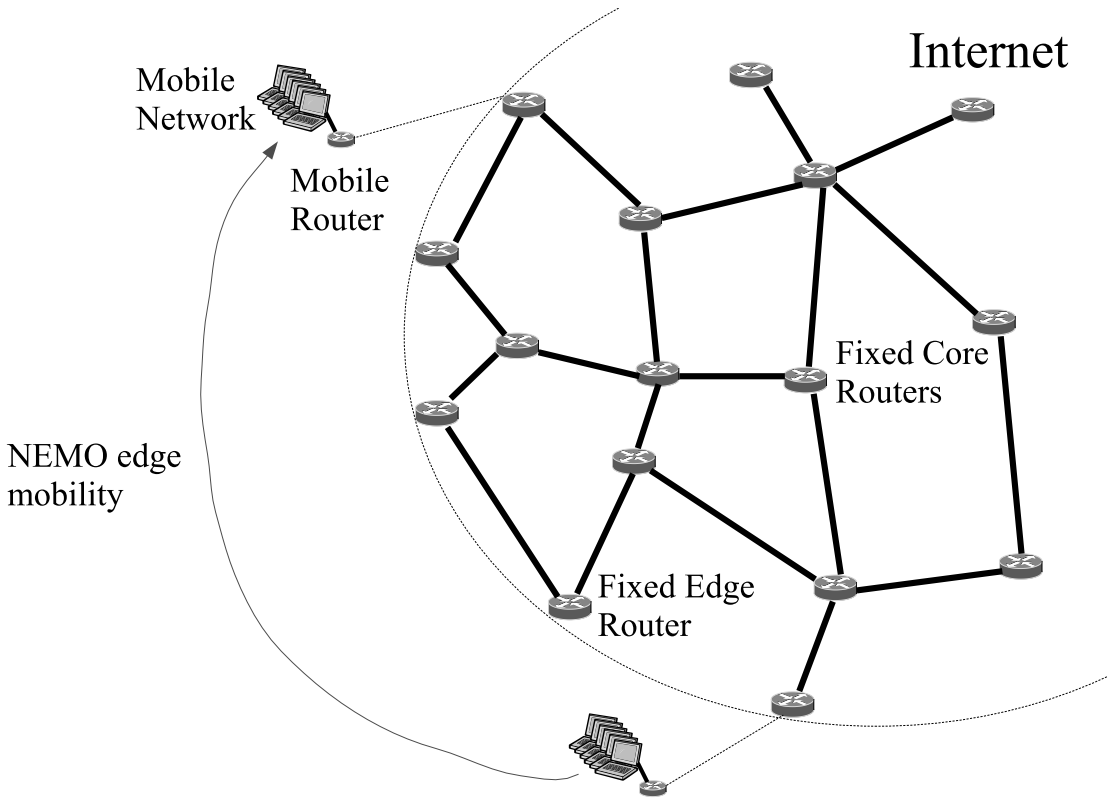


Figure 4.6: NEMO edge mobility: a mobile router, with its attached subnetwork and hosts, is mobile along the fixed edge.

4.3 NEMO

NEtwork MObility (NEMO [40]) is a protocol that enables a router to be mobile with its attached subnetwork, along the edge of the fixed Internet core. The mobile router basically runs Mobile IP on behalf of all of the attached hosts in its subnetwork, in order to continue to provide Internet reachability to its subnetwork as a whole, as the router and its subnetwork are mobile along the fixed edge (as shown in Figure 4.6).

When the mobile network moves away from its home link, and attaches to a new access router (*i.e.* the mobile network's new point of attachment to the Internet), the mobile router acquires a Care-of Address. As soon as the mobile router acquires its Care-of Address, it immediately sends a BINDING-UPDATE message (similar to REGISTRATION-REQUEST in Mobile IP) to its Home Agent. When the Home Agent receives this message, it creates a cache entry binding the mobile router's home address to its Care-of Address at the current point of attachment.

The Mobile Router can then provide connectivity to the Mobile Network by being the default gateway. It indicates this to the Home Agent by setting a flag in the BINDING-UPDATE message. This message may include information about the Mobile Network prefix(es), so that the Home Agent can forward to the Mobile Router packets meant for nodes in the Mobile Network attached to the mobile router. The Home Agent can then set up forwarding for all prefixes owned by the Mobile Router.

The Home Agent acknowledges the BINDING-UPDATE message with a BINDING- ACKNOWLEDGEMENT message sent to the Mobile Router (similar to REGISTRATION-REPLY in Mobile IP). Once the binding process

finishes, a bi-directional tunnel is established between the Home Agent and the Mobile Router. The tunnel end points are the Mobile Router's Care-of Address and the Home Agent's address. If a packet is sent from the Mobile Network, the Mobile Router reverse-tunnels the packet to the Home Agent through this tunnel (using IP-in-IP encapsulation). The Home Agent decapsulates this packet and forwards it to the destination.

When a data packet is sent to a node in the Mobile Network, the packet is routed to the Home Agent that currently has the binding for the Mobile Router and its attached prefix(es). The Home Agent can then tunnel the packet to the Mobile Router which will decapsulate and deliver it to the destination in the Mobile Network.

4.3.1 Performance Issues with NEMO

The performance issues with NEMO are basically the same as with Mobile IP (see Section 4.2.1), since NEMO is based on Mobile IP mechanisms. The performance issues with suboptimal routing and encapsulation overhead will be further addressed in Section 6.4.

Chapter 5

Ad Hoc Mobility

In this chapter, we introduce a new type of mobility going beyond the simple edge mobility addressed in Chapter 4: ad hoc mobility. For example, in the Internet, this feature enables a node to have an *indirect* point of attachment through other mobile nodes that can themselves reach the fixed edge.

Coming back to the example given in the introduction of Chapter 4: with a cell phone, if a user *B* moves away from the coverage zone, *B* will be able to stay connected to the network if it is in reach of another user *A* who is still in the coverage zone. Ad hoc mobility mechanisms ensure that *A* will *relay* the traffic between *B* and a base-station. This operation is shown in Fig. 5.1.

These mechanisms therefore enable *B* to communicate directly with *A*, in a *peer-to-peer* fashion that bypasses the use of any base-station. Further, if *B* wants to communicate with a user *C* that is not in its radio range, but in the radio range of *A*, the same mechanisms enable *B* to communicate with user *C via* user *A* as relay (see Fig. 5.1). Thus, a wireless network of mobile users that are in each others's radio range is automatically formed, and users inside such "clouds" can communicate between themselves, in a peer-to-peer, multi-hop fashion – even without connectivity to a fixed infrastructure.

The freedom and the peer-to-peer philosophy at the base of ad hoc networking are very attractive and may count many applications in the near future, including Internet access extension, peer-to-peer mobile telephony, mobile peer-to-peer networks, sensor networks, stand-alone emergency networks, location based services, *etc*. However, ad hoc mobility comes with a number of new challenges. The following sections overview these issues.

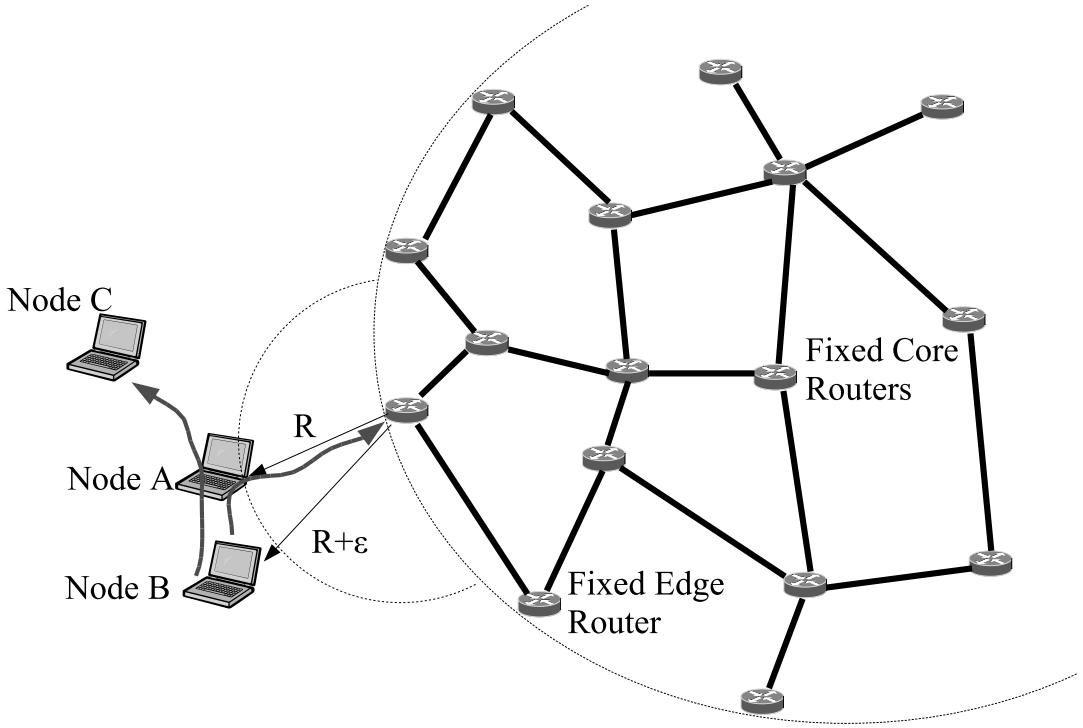


Figure 5.1: Ad hoc mobility. Node B is not disconnected even though it is slightly out of range of the fixed edge, as node A is still in range and will relay the traffic between node B and the fixed network. Node B can also communicate with node C via node A, bypassing the fixed infrastructure.

5.1 Introduction to Mobile Ad Hoc Networking

Mobile ad hoc networks (also known as MANETs [27]) are networks dynamically formed and maintained by a number of nodes that self-organize their wireless connectivity, without using any prior infrastructure or central control. As the topology changes due to the mobility of some nodes or to any other reason, the nodes dynamically reconfigure themselves and continue to provide the maximum network connectivity, which may or may not encompass Internet access (*i.e.* a MANET can also be a stand-alone network). Basically, the MANET philosophy is, in mobile wireless environments, similar to what the original Internet paradigm was in static wired environments. That is to say: any node can join the network, as nodes there are more or less equals, each one of them simply contributing as much as it can to extend the network's reach.

Similarily to the origins of the Internet itself, MANETs have emerged from a military context, in the 1960s – at the time, they were called packet radio networks. But with the boom of mobile devices and wireless communications, mobile ad hoc networking has seen its scope extended to commercial environments as well. MANETs essentially aim at achieving robust and efficient mobile wireless networking by incorporating routing functionality into mobile nodes. Ad hoc mobility has been the subject of numerous research and standardization efforts during the last few years, in academia as well as in the industry. It is foreseen as an important Internet component in the near future.

However, the challenges of ad hoc networking are somewhat different from those of wired networking, and thus, many classical Internet solutions successfully used on wired networks fail in a mobile ad hoc environment. Basically, these differences come from two aspects: (i) the use of broadcast wireless interfaces instead of traditional wired or point-to-point wireless interfaces, and (ii) the possible native mobility of a majority (if not all) of the nodes in the network, instead of assuming essentially static nodes.

In the following, we will give a more detailed overview of the new challenges encountered with ad hoc networking.

5.2 Challenges with Wireless Links using Broadcast Radio Interfaces

The use of broadcast wireless interfaces challenges one of the basis of the Internet architecture: the division of the network into a well defined hierarchy of *subnets*. Indeed, Internet's foundation relies on there being a tie between a physical link shared by *hosts* and managed by a *router*, on one hand, and a particular range of

IP addresses on the other hand. All the devices on a link (say, devices with an interface on the same Ethernet LAN for example) are supposed to share the same range of IP addresses, then called a subnet.

However, the use of broadcast radio interfaces (the most popular wireless solution so far, such as IEEE 802.11 [38]) makes the notion of shared link become very vague. The most common link properties fail in most cases, as a MANET is very often not contained in a single radio range. For instance, a single transmission is not able to reach all the nodes in the subnet (hosts or routers), and a shared physical segment cannot be accurately identified, as a MANET spans over several radio hops. In particular, this multi-hop aspect distinguishes MANETs from the traditional use of wireless links to access points infrastructures.

MANETs' wireless multi-hop nature also invalidates the traditional strict division of Internet nodes between (a) routers, that actively participate in network building and maintenance, and (b) hosts, that just use the network. Depending on the topology, which changes often due to mobility, any node in the MANET may be required to act as forwarder to complete a multi-hop broadcast, for instance. In this respect, nodes in a MANET are neither routers nor hosts, but rather more or less both. And therefore, nodes are all equals *a priori*.

5.3 Challenges with Mobility beyond Simple Edge Mobility

Another basis of the Internet infrastructure is the tie between a node's identity and its topological location – or in other words, IP addressing. The Internet was essentially designed for static nodes, and therefore it made sense to condense the location and the identity of a node into a single piece of information: its IP address. When combined with the organization of the network into a precise hierarchy of subnets and their well known associated range of IP addresses, this gives a very simple and efficient approach to networking, that is also scalable to large, heterogeneous networks – the basis of the Internet.

Node mobility is an issue in such an organization. Indeed, if a node changes location, it must also change its ID, which is a problem for most applications. Some solutions have been developed to solve this problem in case of limited edge mobility, and for a limited fraction of mobile nodes in the network. Approaches like MobileIP [39] and NEMO [40] are such solutions (see Chapter 4).

However, in a mobile ad hoc environment, these techniques fail as Internet connectivity may be indirect. Some nodes in the MANET may indeed be several hops away from the nodes providing Internet access, and the basic mechanisms behind these solutions do not handle multi-hops. Actually, in this case, the initial step will fail most of the time: the mobile node will not be configured as it should, with a correct care-of address.

Enhancements such as Hierarchical Mobile-IP [41] or the Nested NEMO mechanism described in [40] (see Chapter 4) may be able to cope to some extent with a mostly static multi-hop topology. But none of these solutions can handle (i) a fast changing topology, (ii) an arbitrary number of nodes, and (iii) an arbitrary MANET diameter.

Moreover, philosophically, if a bigger proportion of the nodes in the Internet are mobile, the concept behind Mobile IP is indeed questionnable in terms of scalability, as it may then produce prohibitive amounts of overhead with on one hand IP-in-IP encapsulation, and on the other hand extremely sub-optimal routing.

5.4 Mobile Ad Hoc Networks: a Harsher Environment

In general, MANETs induce a harsher environment compared to what it used to be so far, with wire-line networks for instance. Mobile ad hoc networking imposes scarcer wireless bandwidth, while at the same time featuring greater topology change rates due to mobility, and lower transmission quality. MANET nodes experience more complex interference issues due to the use of wireless broadcast links and its incurred hidden node problems. Many nodes have inherent energy constraints, running on battery for instance. Security is looser due to the use of wireless links and to the nodes' mobility.

These facts point at a difficult task: having to fit much more signalling into much less available bandwidth, with nodes that have limited processing capabilities. Moreover, full-fledged schemes for routing and address configuration are needed to handle the mobile multi-hop mix between fixed Internet nodes and MANET nodes. The following chapters will describe some of these schemes.

Part III

Internet Routing and Mobility Solutions for Ad Hoc Networks

Chapter 6

Ad Hoc Routing

A mobile ad-hoc network (also called a MANET) is a set of autonomous mobile nodes, communicating via a wireless medium over which they form a dynamic graph of wireless links. When the network size grows to the point where some pair of nodes no longer have a direct link between them, ensuring connectivity becomes the task of *routing*.

Routing in mobile ad hoc networks is a rather difficult task, due to the mobility of the nodes on one hand, and to the nature of the links interconnecting the nodes on the other hand (see Chapter 5). In particular, most of the routing solutions designed for traditional wired networking are not adapted to mobile ad hoc networking, as MANETs' different characteristics call for new trade-offs and optimizations. Indeed, compared to usual wired networks, ad hoc networks feature (1) a higher topology change rate, (2) less available bandwidth, (3) interference issues between nodes, (4) possible limitations in processing and battery power, (5) and a decentralized nature that brings new scalability, management, and security challenges.

Therefore, new routing protocols have been developed specifically for ad hoc networking, designed to be more efficient in this harsher environment. This chapter overviews some of these solutions.

6.1 Introduction to Ad Hoc Routing

There are currently two main approaches to ad hoc networking (i) the *proactive* approach, and (ii) the *reactive* approach. Each approach adapts to the ad hoc environment by reducing the overhead needed for routing using different techniques.

The proactive approach aims at refining the classical approach to routing: nodes optimize the overhead due to routing, but still establish routing tables with paths to every destination in the network, as in 2.7.1. Typically, when a route is requested for any given destination, the route has already been established prior to this request, and messages can immediately be sent to the destination – similarly to usual Internet routing with OSPF or BGP.

On the other hand, reactive approaches depart from this traditional philosophy, and only establish paths on-demand, as requested by actual user needs. Therefore, these approaches only maintain incomplete routing tables, containing only the paths currently used by user traffic. Typically, a route to reach a destination is not established until a user actually requests to send something to this destination. The route is constructed only at this point, as requested. Thus, if the destination is not on a currently used route, messages may not be sent immediately to the destination – they are delayed due to the route's acquisition time.

Both approaches have their merits, depending on the topology, the mobility of the nodes, and the user traffic patterns. The reactive approach is generally considered to achieve lighter overhead and processing in the case of networks where only few routes carry user traffic, and if nodes are mostly static. However, its departing from usual Internet routing standards may be a difficulty whereas the full integration of an ad hoc network in the Internet is concerned.

On the other hand, a natural application of reactive protocols is with sensor networks, where lightess is a must for typically extremely power-limited nodes. Reactive protocols can achieve better lightness than proactive protocols in sensor network, as sensor nodes usually (i) experience limited relative motion between them, (ii) use few routes, typically just one route, to the gateway, and (iii) are not sensitive to delays in data transfer.

In this document, we focus on ad hoc networking with full integration of mobile nodes in the Internet and user traffic patterns requesting many different routes. Thus we will not base ourselves on a reactive approach but rather on a proactive approach. The next section will overview a typical proactive ad hoc routing protocol: OLSR (Optimized Link State Routing [26]). For more details on reactive approaches to ad hoc networking, refer to the specifications of protocols such as AODV (Ad Hoc On-Demand Distance Vector Routing [31]), or DSR (Dynamic Source Routing [33]).

6.2 OLSR: Link State Routing Optimized for MANETs

In this section we essentially outline OLSR (Optimized Link State Routing [26]), an example of proactive ad hoc routing protocol. OLSR is based on a proactive link state algorithm, and therefore employs the periodic exchange of control messages in order to accomplish topology discovery and maintenance. This exchange results in a topology map being present in each node in the network, from which a routing table can be constructed (see Section 2.7.3 for more details on link state routing).

Basically, OLSR employs two types of control messages: HELLO messages and TC messages. HELLO messages have local scope and are exchanged periodically between neighbor nodes only, their role is essentially to track the status of links between neighbors. On the other hand, TC messages (Topology Control messages, OLSR's LSPs, see Section 2.7.3) have larger scope and are emitted periodically to diffuse link state information throughout the entire network.

6.2.1 MPR Techniques

In addition to the classical link state algorithm described in Section 2.7.3, OLSR features mechanisms that optimize the size of LSPs, and that optimize the number of LSP transmissions network-wide. These mechanisms are based on a technique called *multipoint relaying (MPR)* [30].

This technique drastically reduces the cost of performing a flooding operation, through having each node independently select a minimal set of "relay neighbors" called MPRs, responsible for relaying the broadcast packets transmitted by the node – while other neighbors do not forward the broadcast. As shown in Fig. 6.1, from the local point of view of a node flooding a packet – *i.e.* the center node in the figure – this corresponds to only the minimal number of neighbors (the black nodes) relaying the broadcast, instead of basically all the neighbors, which reduces the number of incurred transmissions while still delivering the broadcast to every node. Moreover, the MPR technique is used to reduce the size of LSPs (called TC messages in OLSR specifications), or even suppress the LSP generation of some nodes while still providing enough information for the link state algorithm to function properly. The MPR technique will be detailed further in Section 7.1.

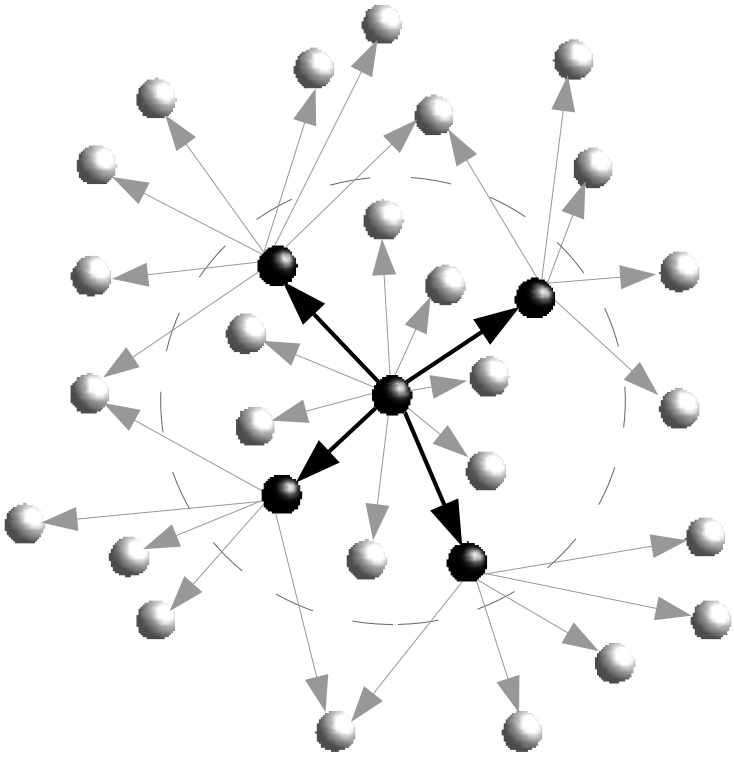


Figure 6.1: Multipoint Relays of a node. A node (center) floods a message that is forwarded only by the neighbors it has selected as its MPRs (the black nodes). The range of the neighborhood of the node is depicted by the circle.

6.2.2 Unified Formats

Similarily to OSPF specifications, OLSR control traffic is transmitted in an unified packet format. This allows different messages to be piggybacked together, therefore further optimizing the number of transmissions overall. The OLSR packet format is shown in Fig. 6.2. As seen in this figure, a packet wraps a collection of messages, each with individual headers, which allows the individual treatment (including flooding behavior) of each message. This unified format also allows extensions to easily take advantage of the MPR flooding mechanism. Examples of such extensions are given throughout the next chapters, as ad hoc networking with OLSR techniques is in the following the subject of several developments and analysis.
The fields of OLSR packet headers and messages headers can be decribed as follows:

Packet Length – *16 bits long*. This field specifies the length (in bytes) of the packet.

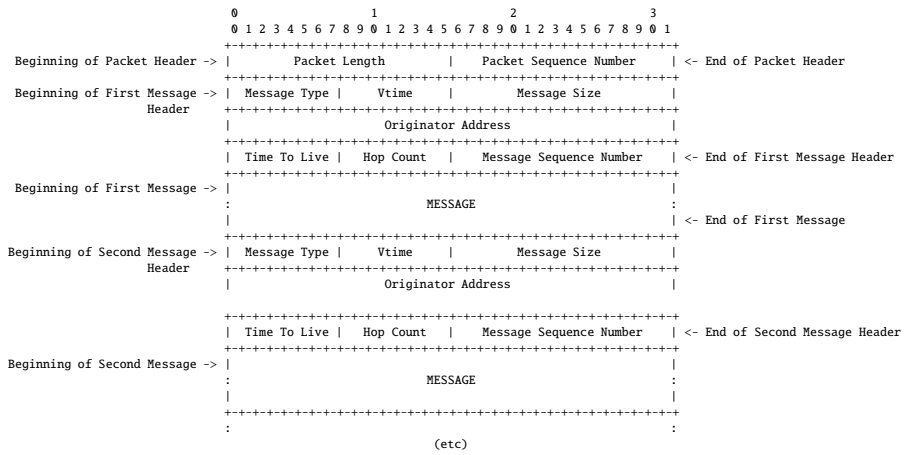


Figure 6.2: Generic OLSR packet format. Each packet encapsulates several control messages into one transmission.

Packet Sequence Number – *16 bits long*. This field uniquely identifies the packet. Each time a node emits a new packet, it uses a different, incremented packet sequence number.

Message Type – *8 bits long*. This field indicates which type of message is to be found in the "MESSAGE" part. For example HELLO, or TC.

Vtime – *8 bits long*. This field indicates how long after of the message must a node consider the information contained in this message as valid, unless a more recent update to the information is received.

Message Size – *16 bits long*. This field specifies the size of the message (in bytes).

Originator Address – *16 bits long*. This field contains the identity (IP address) of the node which has originally generated this message.

Time To Live – *8 bits long*. This field indicates the maximum number of "hops" a message will be forwarded.

Hop Count – *8 bits long*. This field indicates the number of hops a message has attained. Each node that forwards the message must increment this field before forwarding the message.

Message Sequence Number – *16 bits long*. This field uniquely identifies the message. Each time a node generates a new message, it uses a different, incremented message sequence number.

MESSAGE – *variable length*. This field contains a message. A packet typically wraps several messages of several types: TC messages, HELLO messages *etc*.

6.2.3 Additional Features

Additionally to the base described above (TC and HELLO messages and respective mechanisms), OLSR employs two other types of messages:

1. MID messages (Multiple Interface Declaration), with which a node with multiple interfaces can declare its interfaces configuration to the other nodes in the network.

2. HNA messages (Hosts and Network Association), with which a node can advertise routes to networks or hosts that are outside the OLSR network.

For further details on OLSR specifications, exact message formats *etc*. refer to [26].

6.3 wOSPF: Router Ad Hoc Mobility in the Internet

In this section we decribe a new extension to the routing protocol OSPF, designed to extend networks of fixed Internet routers with ad hoc networks of mobile Internet routers. This extension was first presented in [11].

6.3.1 Introduction to OSPF on MANETs

There is a need for a generic IP routing solution, spanning over different types of routers and network technology. Until now, OSPF has been the predominant protocol addressing this need throughout the Internet (see Section 3.2). However, with the emergence of wireless ad hoc networking, OSPF's generic ability is challenged, as it does not work well enough "as is" on MANETs [10] [2].

Nevertheless, OSPF was taylored to route in a heterogeneous environment, and should be able to incorporate an additional extension accomodating the specific needs of ad hoc networking. Furthermore, it was noted how OLSR, which was taylored for MANETs, is in fact in its essential functioning very close to that of basic OSPF: both protocols are of the same link state, proactive nature.

Therefore, some solutions have been developed in order to make OSPF operate efficiently on wireless ad hoc networks of routers, spanning a mobile extension around the fixed core of the Internet – special cases of MANETs aiming at specific mobility schemes and node characteristic. One of the proposed solutions is wOSPF (wireless OSPF, initially presented in [11]) where a new type of OSPF interface is specifically defined for interfaces on mobile ad hoc networks.

The following summarizes why OSPF does not work well enough on ad hoc networks, before an overview of the solution proposed by wOSPF is given.

6.3.2 OSPF's Issues on Ad Hoc Networks

There are three main problems with OSPF operation on mobile ad hoc networks.

(i) OSPF features a mechanism aiming at optimizing the retransmissions on interfaces featuring broadcast capabilities: the Designated Router mechanism (see Chapter 3.2). Optimizing retransmissions is crucial in an ad-hoc network, both in terms of overhead and in terms of radio collisions (interferences). However, not only is the Designated Router approach not efficient in an ad-hoc environment, but OSPF may even fail

to function because this mechanism does not work in face of the "hidden node" problem (*i.e.* a common wireless topology where some node hears a neighbor that other neighbors don't).

(ii) OSPF's positive acknowledgement and database exchange mechanisms are not adapted to node mobility. Furthermore, the significant amount of overhead and transmissions they incur threatens to degrade performance due to limited bandwidth, rather than actually achieving their purpose of garanteeing better link state database synchronization.

(iii) OSPF's hierarchical routing with areas creates additional problems when nodes are mobile. Indeed, when a mobile node moves from an OSPF area to another, it needs to be reconfigured in order to integrate properly in the OSPF infrastructure and not disrupt its hierarchical routing schemes, such as summarization (see Section 3.2).

Aside of these three main issues, some additional thought should be given to optimizations such as, for instance, a mechanism to differentiate the metric used on wireless links from the metrics used on wired links, or an efficient way to regulate and treat differently LSAs describing long-lasting wired routing information on one hand, and LSAs describing short-lived wireless mobile routing information on the other hand, *etc*.

The following section overviews how wOSPF addresses adaptation to ad hoc networking – a description that was first presented in [11].

6.3.3 Overview of the OSPF Wireless Interface Type

Ad hoc networks do not effectively map into one of the usual OSPF interface types (*i.e.* Point-to-Point, Broadcast, NBMA, or Point-to-MultiPoint, see Section 3.2). Indeed, NBMA, Point-to-Point, and Point-to-MultiPoint are defined as non-broadcast networks, and operation of non-broadcast interfaces on broadcast-capable networks leads to high overhead. On the other hand, the OSPF broadcast network type assumes direct connectivity between all routers on the subnet, while in mobile wireless networks, connectivity may only be partial.

An approach to adapting OSPF for MANETs can then be to use a slightly modified Point-to-MultiPoint interface type, but the induced exponential growth in router adjacencies as the number of nodes increases is not adapted to large topologies.

Another approach can be to allow routing at layer 2, which would emulate direct connectivity between all routers in an ad hoc network. This would then allow a traditional broadcast-based OSPF to operate over this emulation at layer 3. However, such an operation can lead to extra overhead unless neighbor discovery operations are coordinated across layers.

wOSPF takes another approach, by creating a brand new OSPF interface type – the *wireless* interface – for networks that have the properties of a MANET: (i) the medium is of a broadcast nature, (ii) link layer messages can be multicast or unicast, and (iii) a direct link between two given nodes may or may not be available, depending on the time – sometimes the pair can communicate over one hop, and sometimes their connection requires more than one hop communication. The new wireless interface type has the following special characteristics, inspired from OLSR ad hoc routing techniques:

1. The interface uses its multicast capability for an unreliable flooding of LSAs, *i.e.* without acknowledgements – similar to the way TCs are flooded in OLSR.
2. The interface does not elect a designated router on the wireless medium, but supports MPR selection of neighbor nodes on the wireless medium (similar to the technique employed by OLSR).
3. The interface does not attempt database exchange/synchronization with neighbors on the wireless medium. (all neighbor states beyond 2-Way are not reached, eliminating the formation of full adjacencies).

Two additional message types were defined to enable the new functionalities: (i) Link State Flood (LSF) messages for unreliable flooding, and (ii) wireless HELLO messages, that enable MPR selection advertising in place of usual designated router advertisement. The wireless interfaces implement the MPR selection algorithm and rules for flooding LSAs, which are gathered in LSFs in the same fashion OLSR TCs are being piggybacked in the same packet to be broadcasted.

6.3.4 Shortcomings of the wOSPF Approach

As seen in the previous section, wOSPF basically incorporates OLSR into OSPF formats. Therefore wOSPF proposes an extension of OSPF that is in theory as well adapted to ad hoc networks as OLSR itself, which is

a success.

However, in doing so, it does not address the problem of nodes moving accross different OSPF areas. Therefore, the mobility of nodes is in fact limited, as no configuration scheme is provided for mobile nodes.

Furthermore, wOSPF proposes only a partial adaptation of OSPF functionalities on wireless ad hoc networks: database exchange and reliable synchronization mechanisms are not active on these interfaces. This lack may be a problem when an OSPF network gathers both wired and wireless ad hoc parts – which is the goal with extending OSPF with wOSPF. A more detailed analysis of this problem is given in Section 7.3.

6.4 MANEMO: Network Ad Hoc Mobility in the Internet

In this section, we propose a new extension to NEMO specifications using ad hoc routing in order to (i) enable full ad hoc mobility for mobile routers and (ii) alleviate some NEMO issues with sub-optimal routing and encapsulation overhead. This solution was first presented in [7].

6.4.1 Nested NEMO Suboptimal Routing and Encapsulations Issues

Through a mechanism based on Mobile IP, NEMO allows a mobile router to continue to provision Internet connectivity to hosts in its attached subnetwork, as the router (and the subnet) is mobile along the edge of the Internet (see Section 4.3).

In fact, NEMO specifications further allow a mobile router to connect *indirectly* to the Internet, by attaching to another mobile router that has itself Internet access. This configuration is called *nested* NEMO, and is depicted in Figure 6.3. An arbitrary level of nesting can thus be envisionned, where a mobile subnetwork accesses the Internet through a number of nested NEMO hops before reaching the actual access router (the point where the nested NEMO network indirectly attaches to the Internet) such as depicted in Figure 6.4.

However, using the current specification, a nested NEMO topology may induce (i) unacceptable amounts of tunneling and (ii) extremely sub-optimal routing. The following sections analyze this problem and describe an optimization for NEMO using ad hoc routing techniques in order to reduce this issue. In a nested topology as shown in Figure 6.5, if hosts from different NEMO networks A and B wish to communicate, tunneling will occur through the home agents of A and B (see Chapter 4), the path is indicated by the arrows along the links in the network. In a more deeply nested NEMO network, such as illustrated in Figure 6.6, the path taken by the tunneled traffic in order to reach a node in an adjacent NEMO network can become substantially longer. It is clear, that in the topologies shown in Figure 6.5 and Figure 6.6, shorter paths exist – but are not used.

In addition to the long and suboptimal paths, more overhead stems from tunneling: essentially, whenever an IP packet transverses a home agent, encapsulation happens. Returning to figure 6.6, an IP packet from a node in network A, destined to a node in network C, will first be sent to the home agent of network A (see Chapter 4). There is a binding stating that mobile network A is attached to mobile network C – and hence, the IP packet is encapsulated with another IP-header and sent to the home agent for mobile network C. At the home agent for network C (then D, then F, and finally B) the same happens – and when the packet finally arrives back at the nested NEMO network, it thus carries the overhead of 5 encapsulations. Indeed, none of

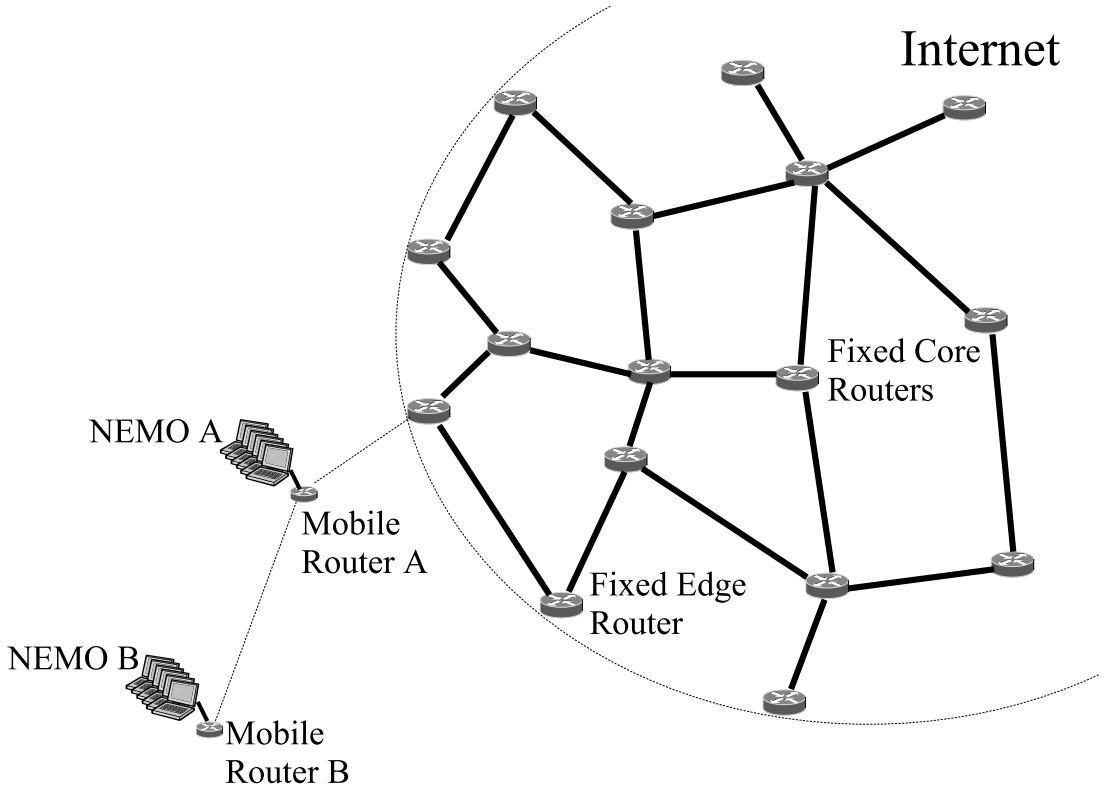


Figure 6.3: A simple nested NEMO network: one NEMO network attaches to another NEMO network in order to access the Internet.

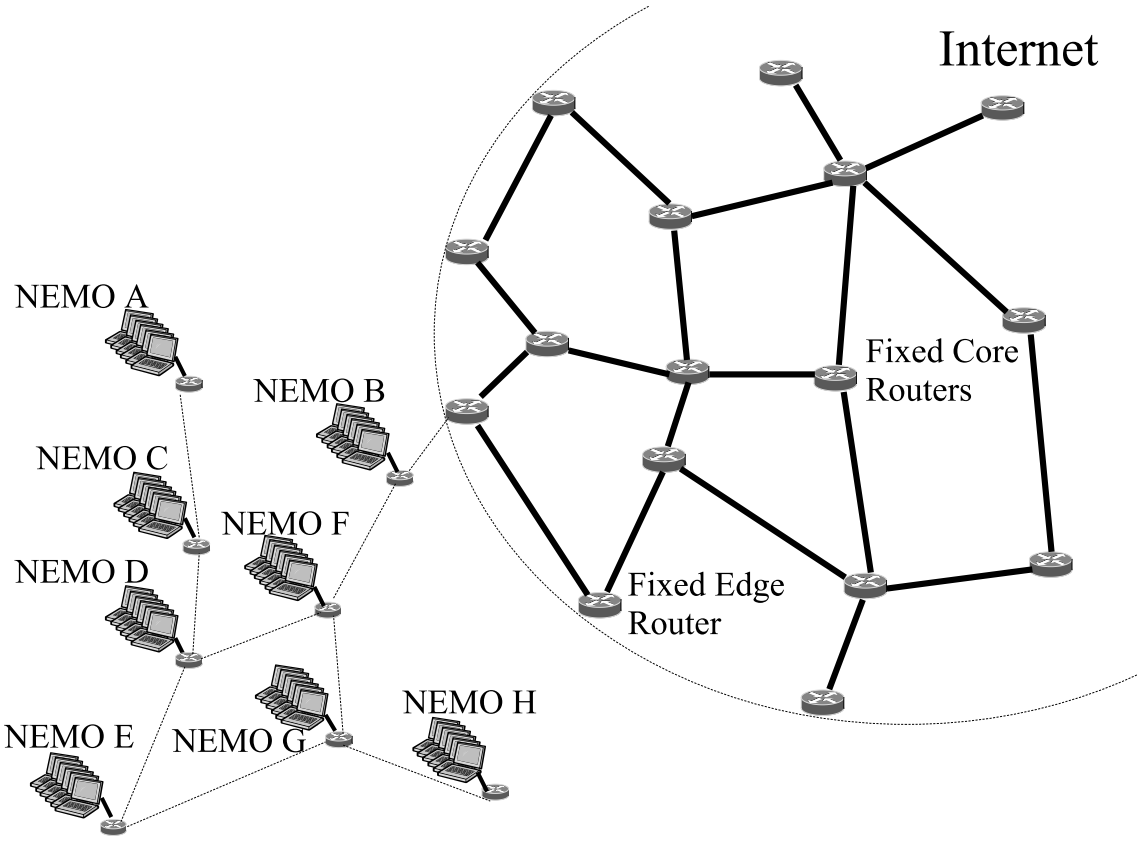


Figure 6.4: Deeply nested NEMO network.

the mobile routers in the nested NEMO network maintain topological information about the nested NEMO network, and thus are not able to correctly forward the packet without this encapsulation header.

Thus, in situations where nodes in nested NEMO networks communicate, this communication is subject to the overhead from suboptimal paths due to tunneling combined with the overhead from nested encapsulations. This comes from the following issues:

- The node which originates data traffic does not know where the destination node is located and therefore assumes that the node is at its “home network”, relying on subsequent tunneling to reach the destination’s current location.
- No router knows the full path to the destination.
- No router knows the topology of the nested NEMO network(s), thus relying on the encapsulation information in order to provide forwarding.

Route optimization is the task of reducing the encapsulation overhead and providing shorter (if not optimal) paths for data traffic. Extensions to NEMO have therefore been envisionned in order to address this problem. One of the solutions proposes to use mobile ad hoc routing between mobile routers. This solution will be described in the following.

6.4.2 NEMO Route Optimization with Mobile Ad Hoc Networking

The best known algorithms to provide paths in a network are routing protocols. In the case of arbitrary sized nested NEMO networks, the Mobile Routers naturally form an ad hoc network that can efficiently use a MANET routing protocol such as OLSR, which was engineered to accomplish this task: optimally provide routing in a mobile ad hoc environment. With OLSR, Mobile Routers can simply discover and maintain optimal routes to the Access Router, but also between Mobile Network Nodes themselves. This implies that communication between nodes within nested NEMO network can be routed through optimal paths, thereby avoiding layers of over-encapsulation and sub-optimal routing over the Internet, through the Home Agents, and back into the same nested NEMO network. With reference to Figure 6.4, this implies that the nodes in mobile network A and mobile network B can communicate directly via the link between their Mobile Routers, rather than through the long path (indicated in Figure 6.4) through the Internet.

Mobile Routers supporting OLSR exchange information in order to discover and maintain the network they

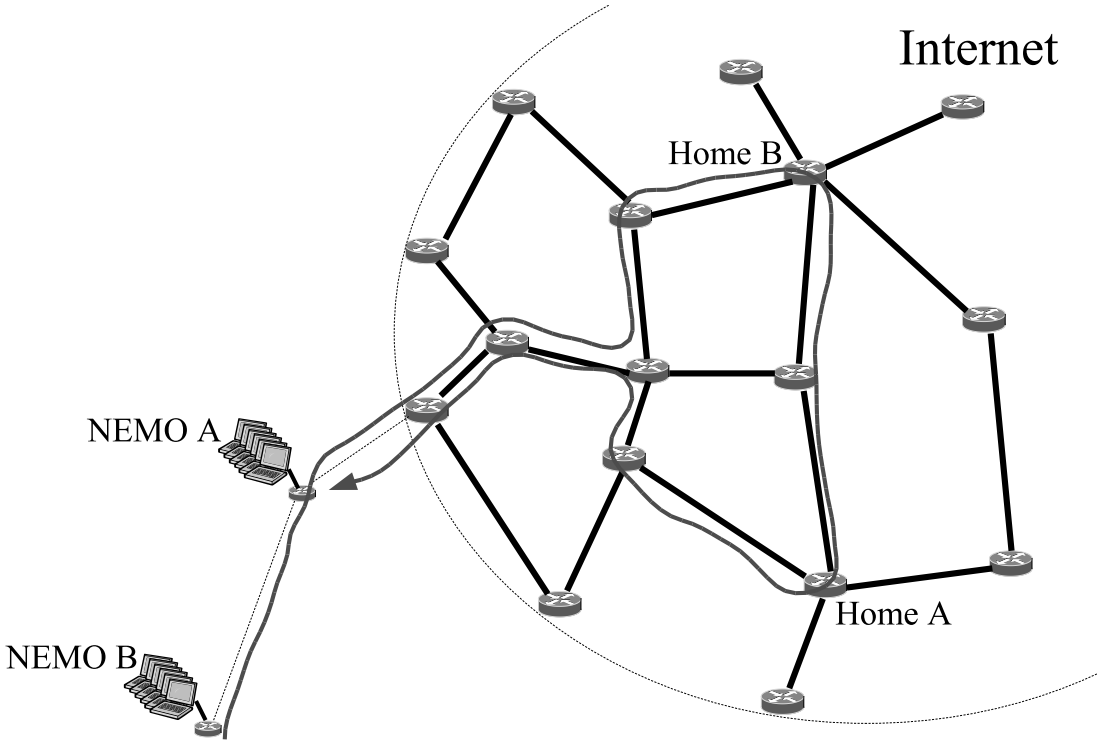


Figure 6.5: A simple nested NEMO network: one NEMO network attaches to another NEMO network in order to access the Internet. A host in NEMO B wants to communicate with a host in NEMO A. Instead of going directly from B to A, the path goes through the Internet and the home agents of A and B.

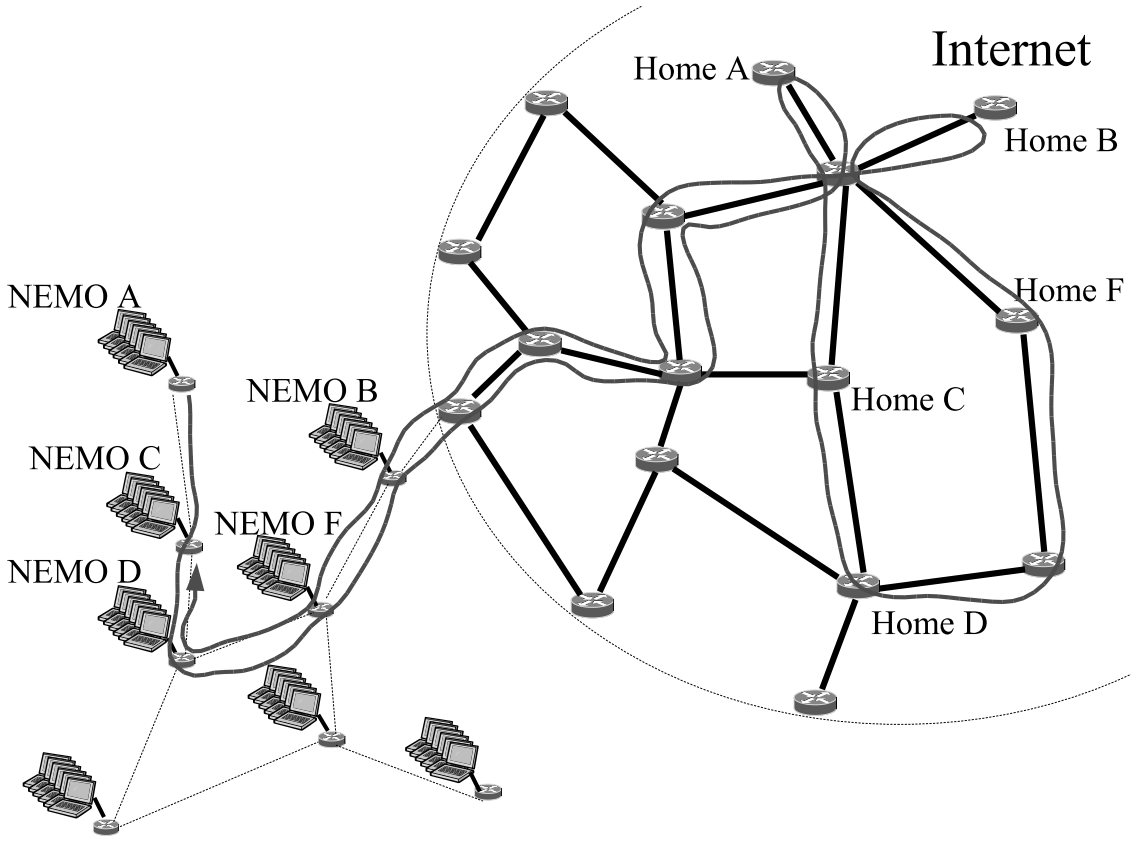


Figure 6.6: Deeply nested NEMO network, and the induced extremely suboptimal routing: through the Internet and the home agents of A, C, D, F and B instead of directly from A to C.

form at the edge of the Internet (behind the Access Router) through TC and HNA messages, using the light-weight signaling features of OLSR: by periodically exchanging (i) the network prefix(es) of the Mobile Network Nodes they aggregate (using HNA messages) and (ii) summarize topology information (using TC messages), the Mobile Routers in a nested NEMO can provide fully optimized routing in the ad hoc network they naturally form.

Thus, a Mobile Router running OLSR will include links to selected (via the MPR selection mechanism) adjacent Mobile Routers running OLSR in its TC messages. Mobile Network Nodes, which are not Mobile Routers, will be advertised through HNA messages.

Mobile Network Nodes inside the same nested NEMO network can thereby communicate directly through the routing provided by OLSR and the paths formed by the links between the Mobile Routers. Note that the Access Router doesn't necessarily need to be OLSR capable in order to benefit from the routing inside the OLSR NEMO network. Coming from the Internet, once a packet reaches an OLSR capable node, say a top level Mobile Router (if the Access Router is indeed not OLSR capable), fully optimized routing is available to allow the packet to be routed to the destination. Reaching out for the Internet, a packet is naturally routed from its NEMO network to the Access Router over an optimal (in terms of number of hops) path of Mobile Routers. The packet then reaches a top level Mobile Router, which will perform simple NEMO Mobile IP processing in order to forward it to the Internet through the Access Router. The top level Mobile Routers (*i.e.* the Mobile Routers that attach directly to an Access Router) advertise a default route in order to route packets going out of the nested NEMO to the Internet. In case the Access Router is OLSR capable, OLSR will naturally and dynamically transfer the role of the top level Mobile Routers described above, to the Access router itself.

6.4.3 NEMO Tunneling Optimization using MANET Routing

When a Mobile Node in a nested NEMO network communicates with a node in the Internet, outside the nested NEMO network, the level of nested mobility dictates the number of Home Agents (and therefore the amount of encapsulation) the packets have to go through. This implies unnecessary encapsulation and suboptimal routing, and there should be a way to limit the level of tunneling to only one encapsulation "IP in IP", while at the same time minimizing the traffic relayed by Home Agents.

Existing solutions to route optimization problems in NEMO (see [71]) therefore aim at, basically, minimizing the required amount of tunneling in various nested mobility cases. An acceptable level of tunnel optimization is attained if whatever the depth of nested NEMO networking, the amount of tunneling stays the same (as if there is no nested mobility, but just simple mobility). That is to say: there is at most one level of encapsulation, and at most one Home Agent involved per distinct nested NEMO network. Ideally, the ultimate optimization would be to bypass all Home Agents.

By combining a solution derived from HMIP like in [74], [72], or [73] with the solution presented in this section providing ad hoc routing within a nested NEMO network, the above acceptable level of optimization is achievable. Essentially, the encapsulation performed by the Home Agents in the Internet serves to ensure that the Access Router (or top level Mobile Router) is able to "route" a packet to the destination, based on the information contained in the encapsulation headers.

However with the Mobile Routers in the nested NEMO network forming a mobile ad hoc network, the information from the encapsulation is no longer required to ensure correct routing within the nested NEMO network. This has some interesting implications:

1. The Home Agents can carry out their usual role as forwarders (including encapsulation of outgoing messages), while safely discarding any existing encapsulation (relative to mobile networking) on incoming messages. The encapsulation on outgoing messages, essentially, is only required in order to ensure that packets are forwarded to the next "hop" along the path of Home Agents towards the nested NEMO network Access Router. Effectively, this implies that a packet never carries more than one encapsulation header – compared to one per Home Agent in basic NEMO.

2. If a Home Agent along the path towards the nested NEMO network Access Router can identify the Access Router, to which the nested NEMO network is attached, any encapsulation (relative to mobile networking) in an incoming packet to the nested NEMO network can be discarded. The packet is encapsulated and transmitted directly to the Access Router, thereby bypassing the Home Agents and carrying only one encapsulation header as described above.

3. Indeed, a signalling mechanism can be developed, whereby a Mobile Router can inform its Home Agent about its Access Router. This signalling mechanism is identical to the signaling in basic NEMO, with the important difference that rather than signalling a binding with another Mobile Router (in the nested case), a binding is signalled with the Access Router between the Internet and the nested NEMO

network. Referring to Figure 6.4, this implies that the Mobile Router for mobile network A would not signal a binding with Mobile Router C – but rather with Mobile Router B, thereby accomplishing the desired route optimization.

Having an ad hoc network in a nested NEMO network thereby reduces the route optimization problem to a simple application of the binding signalling mechanism found in basic NEMO.

Chapter 7

Optimizing Control Traffic in Mobile Ad Hoc Networks

Higher radio link capacity implies shorter radio ranges in ground communications, and thus, the routing protocol used between mobile nodes is a key network feature. However, the use of a routing protocol implies some control signalling, *i.e.* some necessary overhead which may in fact handicap the network's functionning if not carefully regulated.

Scarce bandwidth and interferences in mobile ad hoc networks yield the need for efficient overhead reduction techniques. Some techniques have already been developed for traditional networks, but these are usually not sufficient or not adapted to ad hoc networks because they were designed for an environment that does not suffer from the same limitations, as bandwidth and interferences are no problems on traditional wired networks.

Overhead reduction in an ad hoc environment is two-fold: (i) reducing the amount of information that needs to be transmitted because of the lack of abundant bandwidth, and (ii) reducing the number of transmissions needed to deliver this information because of interferences between nodes that potentially jam communications, and therefore diminish even more the amount of bandwidth that is actually available.

However, if routing information is incomplete, or out-of-date, routing protocols may not function correctly. In particular, the node's mobility in an ad hoc environment yields a need for more routing information, more often, to keep the information up-to-date.

These conflicting requirements introduce a trade-off problem that overhead optimization aims at solving in the most advantageous way. The optimizations must ensure that control signalling does not exhaust the available capacity, while it is still sufficient to provide a functional network, even if the topology changes, from sparse to dense, and/or from static to mobile.

This chapter analyzes and compares different techniques that aim at optimizing different parts of the overhead due to routing: flooding optimization techniques, partial topology techniques and synchronization optimization techniques.

7.1 Flooding Optimization Techniques

In this section we analyze and compare optimized flooding algorithms that aim at improving the traditional broadcast technique in use on wired networks. Indeed, the flooding algorithm is an essential part of most routing protocols, and the traditional mechanism implies too much overhead to work efficiently in an ad hoc environment where the bandwidth is limited and interference between users is a big issue when a lot of retransmissions occur. It is shown that flooding optimization can achieve a very substantial reduction in the number of transmissions needed for a flooding operation, down by 90% in some cases. This analysis was first presented in [5].

7.1.1 Introduction to Flooding

A large number of routing protocols require network-wide signalling (see for instance [34] [70] [26] [31] [33]). This is typically performed through a mechanism known as flooding, *i.e.* performing the task of distributing a piece of information to every node in the network. Flooding is a key component for most routing protocols. For example, proactive routing protocols (like OSPF [34], IS-IS [70] or OLSR [26]) rely heavily on flooding with the need for each router to periodically distribute its link state information to the whole network. On the other hand, reactive routing protocols (like AODV [31] or DSR [33]) use flooding each time they need to set up a path: each source needs to disseminate its route requests. And in fact, flooding makes the biggest part of the overhead induced by these routing protocols.

Let us define classical flooding as the following nave algorithm: from the source, each node, when receiving a piece of information for the first time, redistributes it to all its neighbours: this way the entire network ends up having received this information, which is the goal being aimed at. This simple mechanism is used in classic protocols like OSPF or IS-IS (see [34] [70]). When the network is dense, this approach leads to too

much overhead. Indeed, not only are most retransmissions actually unnecessary, but also: due to the amount of retransmissions that is required, even a single classical flooding operation can cause the network to break down in an ad hoc environment, as bandwidth is scarce and nodes experience interferences. Thus, in order to employ global signalling in ad hoc routing protocols, an optimized flooding operation is required – notice that such a mechanism might also be useful in usual wired network environments.

To reduce the overhead due to global signalling, the task of relaying a flooded packet is given to a subset of nodes in the network. The smallest this relay set, the less flooding costs, and the more bandwidth is available for user data communications. However, a flooding operation aims at delivering some data to all the nodes in the network. Therefore one should not reduce the relay set too much, at the expense of dropping the delivery rate.

Finding the smallest relay set is an NP-hard problem in general, but heuristics can be used in order to approach optimality. Several techniques exist, using different heuristics, mainly (i) multi-point relays (MPR), (ii) connected dominating set (CDS) and (iii) gateway flooding. We will first describe simply each flooding mechanism before comparing their abilities via mathematical modelling. Then we will present and compare results we obtained via simulation.

7.1.2 The MPR Technique

Multipoint relay (MPR) flooding is a broadcast mechanism extracted from the ad hoc routing protocol OLSR [26]. The principle of MPR flooding is that each node independently selects a *relay* set, *i.e.* a subset of its neighbours. Only these selected relays of a node will retransmit broadcast packets transmitted by this node.

Obviously, the smallest this relay set is, the more efficient the optimization will be. An important point is however that the series of nodes relaying a given flooded message may *vary* depending on where the source of this message is located. This is due to the fact that each node *independently* selects its relay set, and therefore different nodes may choose different relays.

MPR selection can be done as follows: let A be a given node in the graph. Let the neighbourhood of A be the set of nodes which have a bidirectional link to A. And let the two-hop neighbourhood of A be the set of nodes which do not have a bidirectional link to A but that have a bidirectional link to the neighbourhood of A. To perform MPR selection, local topology information up to a distance of 2 hops is required in each node.

This information can be acquired through simple HELLO message exchange, as specified in [26].

The multipoint relay set of *A*, that is called MPR(*A*), is then a subset of the neighbourhood of *A* which satisfies the following condition: every node in the two-hop neighbourhood of *A* must have a bidirectional link toward MPR(*A*). As we already stated, the smaller the multipoint relay set (*i.e.* MPR set), the more the broadcast mechanism is optimized.

With MPRs selected, MPR flooding works as follows:

> *A node retransmits a broadcast packet only if it receives its first copy from a neighbour that has chosen the node as multipoint relay.*

7.1.3 The Gateway Mechanism

Gateway node flooding is a broadcast technique extracted from the ad hoc routing protocol DDR [28]. The protocol uses, as foundation for its broadcast mechanism, a forest of logical trees, interconnected via a set of gateway nodes.

The protocol initially forms trees in the following way: each node selects as parent its *preferred neighbour*. A node's preferred neighbour is the neighbour which has the maximum *degree* (number of neighbours). A node which is a local maximum degree-wise (all its neighbours have lower degree) is then the *root* of its tree. Inside a tree a node is either a *leaf* or an *internal node*. A leaf is a node which is parent of none of its neighbours. On the other hand, an internal node is a node which is parent of at least one of its neighbours.

The network is then viewed as a *forest*, *i.e.* a collection of logical trees (see Section 8.1 and Figure 8.1 for an example tree topology). Each tree is identified by a random identifier, which is flooded from root to leaves via the logical links. A tree is connected to other trees via its *gateway nodes*. A gateway node is a node which has at least one neighbour that is part of a different tree.

Therefore, the broadcast mechanism can be summarized as follows:

> *A node retransmits a broadcast packet only if it is a gateway node or if it is an internal node in the tree it belongs to.*

7.1.4 The CDS Technique

Connected Dominating Set (CDS) flooding is a special case of MPR flooding, where the relay sets of nodes in the network coincide, *i.e.* the series of nodes relaying a flooded message is always the same, wherever the source of this message is located. In addition to the general MPR rules, CDS flooding assumes that all the neighbours of a given node *A* will take identical relay decisions concerning *A*. In other words, all the neighbours of *A* will agree either on (i) counting on *A* as relay, or on (ii) not counting on *A* as relay. This special property is not assumed in the general MPR case, where some neighbours may select *A* as relay while some other neighbours may not (see Section 7.1.2).

This broadcast mechanism can be summarized as follows:

A node forwards broadcast packets only if it is in the relay set.

This property is interesting if such a flooding traffic concentration is considered advantageous. This characteristic is sometimes called *source-independent* flooding. On the other hand, in the general MPR case, traffic is on the contrary naturally distributed over different relay paths, depending on the source of the floods. This property is sometimes called *source-dependent* flooding.

Several heuristics exist in order to build a CDS. Some techniques are based on the fact that nodes *select themselves* as being a relay (or not), and do not require nodes to communicate their relay selection. This may be considered advantageous in that no additional signalling is required. However, it implies some complexity and possible drops in delivery rates in cases where nodes are mobile, as analyzed in [15].

Nevertheless, CDS and MPR flooding show similar behaviours. In the following, we will therefore anaylze the general case of MPR flooding, on one hand, and gateway flooding on the other hand.

7.1.5 Performance Evaluation via Mathematical Modelling

The parameter we consider is the number of retransmissions of a single packet via each flooding technique, *i.e.* the *cost* of performing one flooding operation – or in routing protocol terms, the cost of performing one global signalling operation. The model under which we investigate the performance of these flooding mechanisms is the unit disk model, *i.e.* nodes are randomly dispatched uniformly on a domain and the network graph is then the network obtained by connecting nodes which are at a distance smaller than or equal to the

radio range. This model is a classic in the field of performance analysis of wireless networks, although not fully realistic since it omits interferences with obstacles and between simultaneous transmitters.

Let us define the *neighbourhood* of a given node A as the set of the nodes contained in the ball of volume 1 with node A as the center. Furthermore, a node is a *neighbour* of node A if and only if it is contained in node A's neighbourhood. The network graph is then the network obtained by connecting neighbour nodes.

Let us consider that the nodes are dispatched in the domain following a Poisson distribution of mean ν. Then one of the fundamental properties of the Poisson distribution [80] yields that the average number of nodes contained in a ball of volume 1 is ν, and that the average number of neighbours of a random node is also ν. We investigate dense networks, *i.e.* mathematically when $\nu \to \infty$. In the present analysis we will investigate dense networks, *i.e.* mathematically when $\nu \to \infty$, and we will restrict our model to the linear map, addressing networks with geographic locations that mainly stretch on a single geometric dimension (*e.g.* a road).

Note however that the simulation results presented in Section 7.1.6 show figures for both the linear map and the planar map, where the network can stretch in two dimensions (*e.g.* a city) therefore covering most of the real use-case scenarii. In the following, we will analyze each flooding technique and estimate the number of retranmissions they yield, or in other words: the proportion of nodes that retransmit packets. Obviously, the smallest this proportion is, the better the optimization.

Gateway Flooding Analysis

In this section, we will analyze gateway flooding, and estimate the proportion of nodes that retransmit packets when this technique is used. As seen in Section 7.1.3, gateway nodes retransmit flooded packets. Therefore, we will now estimate the proportion of gateway nodes in the domain, or in other words: the probability for a random node in the domain to be a gateway node.

Gateway flooding introduces the so-called *preferred neighbour* of a given node, according to a certain *selection criterion*. The selection criterion used to choose the preferred neighbour can be the degree, *i.e.* the preferred neighbour is the neighbour that has itself the largest number of neighbours. When several neigh-

bours attain this maximum, the node selects the one with largest ID. In this case the selection criterion is said to be (degree,ID). Another way to choose the preferred neighbour is to just use the ID selection criterion: the preferred neighbour is then simply the neighbour with highest ID. A node selecting itself as its preferred neighbour corresponds to it being a local maximum with respect to the selection criterion. Such a local maximum is therefore the root of its region tree.

In the following gathers some mathematical results about the density of trees and the proportion of gateway nodes via the analysis of local maxima distribution.

Density of Trees – In this section we evaluate the tree density, which corresponds to the distribution of local maxima for the selection criterion. As mentioned earlier, there are two distinct criteria: ID and degree. Note that the local maxima distributions vary depending on the criterion.

Theorem 1 *The probability that a node is a local maximum for the ID selection criterion is* $\frac{1}{\nu} + O(e^{-\nu})$ *in the case of the unit disk graph model in dimension 1.*

Proof – Without loss of generality we can assume that the IDs of the nodes are uniformly distributed in the interval $(0, 1)$. The probability that a node with ID equal to x is a local maximum is then the probability that no node has an ID falling between x and 1, which is equal to $e^{-(1-x)\nu}$. Therefore the unconditional probability that a node is local maximum is $\int_0^1 e^{-(1-x)\nu} dx = \frac{1-e^{-\nu}}{\nu}$. This is equivalent to to $1/\nu$ when $\nu \to \infty$. Thus, we can say that in the case of the ID selection criterion, the density of trees is close to 1 per neighbourhood area, or in other words, that the average interval covered by a tree is 1.

□

Theorem 2 *The probability that a node is a local maximum for the degree selection criterion is equivalent to* $\frac{2}{\pi\nu}$ *when* ν *increases.*

The proof of this theorem is distributed in the several lemmas and theorems that follow.

Lemma 1 *The events of a node being a local maximum for the right side of its neighbourhood, or for the left side of its neighbourhood, are independent events.*

Proof – Let $N(x)$ be the number of neighbours of a node at location x on the segment map. Let $I([a, b])$ be the number of nodes contained by interval $[a, b]$. Therefore $N(x) = I([x - 1/2, x + 1/2])$. Let $\Delta(x) = N(x) - N(0)$. If $x \in [0, 1/2]$, then we have $\Delta(x) = I([1/2, 1/2 + x]) - I([-1/2, -1/2 + x])$. If $x \in [-1/2, 0]$ then $\Delta(x) = I([-1/2 + x, -1/2]) - I([1/2 + x, 1/2])$. Since the intervals don't overlap, the events $\Delta(x) \leq 0$ for all $x > 0$,

and $\Delta(y) \leq 0$ for all $y < 0$, are independent.

□

Theorem 3 *The probability $P(\nu)$ that a node is a local maximum for the right side of its neighbourhood is equivalent to $\sqrt{\frac{2}{\pi\nu}}$ when ν increases.*

Proof – By symmetry, the probability of a node being a local maximum is equal to that of being a local minimum, which corresponds to having $\Delta(x) \geq 0$ for all $x \in [0, 1/2]$. Having $I([1/2, 1/2 + x]) - I([-1/2, -1/2 + x]) \geq 0$ for all $x \in [0, 1/2]$ is equivalent to an M/M/1 system with service rate and arrival rate equal to ν, starting with one customer, and that does not empty its queue during a time interval of $1/2$. This is also equivalent to an M/M/1 system with service rate and arrival rate equal to 1, starting with one customer, and that does not empty its queue during a time interval of $\nu/2$.

Let $f(\omega)$ be the Laplace transform of the distribution of the time T needed to empty this queue: $f(\omega) = E[e^{-\omega T}]$. Let θ be the time needed for the exit of the first customer. Classic queuing theory [79] then yields that:

$$T = \theta + N_\theta \times T \tag{7.1}$$

where N_θ is a Poisson random variable of mean θ, and $N \times T$ symbolizes the addition of N independent copies of T (N independent variables T_i (for $i = 1$ to N) identically distributed as T). Therefore:

$$\begin{aligned}
f(\omega) &= E[e^{-\omega(\theta + \sum_{i=1}^{N_\theta} T_i)}] && (7.2)\\
&= \int_0^\infty e^{-t} E[e^{-\omega(t + \sum_{i=1}^{N_t} T_i)}] dt && (7.3)\\
&= \int_0^\infty e^{-t} \sum_{k=0}^{\infty} \frac{e^{-t} t^k}{k!} E[e^{-\omega(t + \sum_{i=1}^{k} T_i)}] dt && (7.4)\\
&= \int_0^\infty e^{-2t - wt} \sum_{k=0}^{\infty} \frac{t^k}{k!} f(\omega)^k dt && (7.5)\\
&= \int_0^\infty e^{-t(2 + w - f(\omega))} dt && (7.6)\\
&= \frac{1}{2 + \omega - f(\omega)}. && (7.7)
\end{aligned}$$

From this equation we get $f(\omega) = 1 - \frac{2}{1 + \sqrt{1 + \frac{4}{\omega}}}$.

Lemma 2 *Quantity $P(\nu) \sim \sqrt{\frac{2}{\pi\nu}}$.*

Proof – Since $P(v) = P(T > \frac{v}{2})$, using the reverse Laplace transform, we have:

$$P(v) = \frac{1}{2i\pi} \int_{-i\infty}^{+i\infty} \frac{(1 - f(\omega))}{\omega} e^{\omega v/2} d\omega. \tag{7.8}$$

Using the fact that $1 - f(\omega) \sim \sqrt{\omega} + O(\omega)$ when $\omega \to 0$ we have from Flajolet and Odlyzko [69]:

$$\frac{1}{2i\pi} \int_{-i\infty}^{+i\infty} \frac{(1 - f(\omega))}{\omega} e^{\omega y} d\omega \sim \frac{\Gamma(1/2)}{\pi} y^{-1/2} . \tag{7.9}$$

With $y = \frac{v}{2}$ and $\Gamma(1/2) = \sqrt{\pi}$ we complete the proof for Theorem 3.

□

By symmetry, the probability for a node to be a local maximum for the left side of its neighbourhood is the same as for the right side. Obviously, the probability for a node to be a local maximum is its probability of being a local maximum for the right side and for the left side. Therefore, using the independence of these two events, we have also proven Theorem 2.

□

These results show that in the case of the (degree,ID) selection criterion, the average density of trees is close to $\frac{2}{\pi}$ per neighbourhood area, which is somewhat less than with the ID selection criterion.

Density of Gateway Nodes – In this section we evaluate the density of gateway nodes in a forest of trees formed with the ID selection criterion. As seen in Section 4.1.1, this criterion yields more trees and likely, more gateways nodes than the (degree, ID) selection criterion used in gateway flooding. Nevertheless, we believe that this approach gives a good idea of what kind of density of gateway nodes we can expect with the (degree,ID) selection criterion, and we validate this approach with the simulations in section 5.

Once again, without loss of generality we will assume that the identifiers of the nodes are randomly uniformly distributed between 0 and 1.

Theorem 4 *When the preferred neighbour is the one with highest ID, the probability that a randomly picked node is a gateway node is greater than* $\frac{2}{3} + o(\frac{1}{v})$*, when the network follows the unit disk model in dimension 1.*

Proof – If a node is not the root for its tree, then its preferred node is either in the left part of its neighbourhood or in the right part. Let us call a node with preferred neighbour on its left, a leftist node. Conversely, we will call a node with preferred neighbour on its right, a rightist node. A centrist node, which is a node that is both leftist and rightist, is then the root of its tree. Note that when a node is leftist, then the root of its tree is in the left part of the network, but not necessarily in its neighbourhood.

A sufficient condition for a leftist node to be a gateway node is to have a rightist node in the right part of its neighbourhood. Indeed, if all the right neighbours belonged to the same tree, then they would all be leftist.

Let us consider a random node A at a position y on the network map. We split the interval $[y-1, y+1]$ into four parts of equal size: $I_1 = [y-1, y-\frac{1}{2}]$, $I_2 = [y-\frac{1}{2}, y]$, $I_3 = [y, y+\frac{1}{2}]$, $I_4 = [y+\frac{1}{2}, y+1]$. Let x_k be the greatest identifier in the interval I_k (see Figure7.1). Ignoring the cases when there are no nodes in some I_k

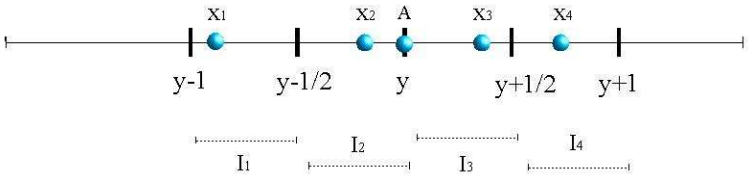


Figure 7.1: Intervals around node A.

(this occurs with probability $o(\frac{1}{v})$) we consider the order of the sequence (x_1, x_2, x_3, x_4). If $x_2 > x_3$ then the node A is leftist. If $x_3 < x_4$ then the rightmost node which still has a position smaller than $y + \frac{1}{2}$ is rightist, unless it is not neighbour of the node which has identifier x_4 (this occurs with probability $o(\frac{1}{v})$). Therefore all orders such that either $x_2 > x_3 < x_4$ (right case) or $x_1 > x_2 < x_3$ (left case) imply with probability $1 - o(\frac{1}{v})$ that the node A is a gateway node.

Let s_k be the order of x_k in sequence (x_1, x_2, x_3, x_4). If x_k is the greatest number in the sequence (x_1, x_2, x_3, x_4) then $s_k = 1$, if it is the second largest number then $s_k = 2$, etc. We call T the order tuple (s_1, s_2, s_3, s_4). If $x_1 > x_2 > x_3 > x_4$ then $T = (1, 2, 3, 4)$. The tuples that correspond to the right case are:

(1,2,4,3)	(1,3,4,2)
(2,1,4,3)	(2,3,4,1)
(3,1,4,2)	(3,2,4,1)
(4,1,3,2)	(4,2,3,1)

The left case is symmetric, therefore there are 16 order tuples that lead node A to be a gateway node with probability $1 - o(\frac{1}{\nu})$. Given that there are $4! = 24$ order tuples and that they are all equiprobable, the node A is a gateway node with probability greater than $\frac{2}{3} + o(\frac{1}{\nu})$, when ν is large.

□

Gateway Flooding Summary – Our analysis has thus revealed that the density of trees is $\frac{2}{\pi}$. We have then deduced an estimation for the density of gateway nodes as being $\frac{2}{3}$. This implies that for a given flooding operation with this mechanism, more than $\frac{2}{3}$ of the nodes are engaged, when ν is large.

MPR Flooding Analysis

Our goal is now to find the proportion of nodes that retransmit a flooded packet, when the MPR technique is used, or in other words: the probability for a random node to be used as a multi-point relay during a flooding.

Theorem 5 *The probability for a random node to retransmit during an MPR flooding is equivalent to* $\frac{2}{\nu}$*, when* ν *is large and the network follows the unit disk model in dimension 1.*

Proof – In the linear map case, the number of MPR per any given node is exactly 2: one at each end of its neighbourhood segment, right and left. When a flooding occurs, a packet is retransmitted via MPR on the right side and on the left side of the segment: retransmissions jump from one MPR to another MPR, with hops of length close to the radio range.

For instance, the MPR on the left of a node A will be the furthest node on the left of node A that is still in reach of node A. Therefore, this MPR will be on average at a distance $\frac{1}{2} - \frac{1}{\nu}$ on the left of node A. Thus, on the left side of node A, the average number of nodes that will retransmit is $\frac{2}{\nu-2}$. By symmetry, it is the same on the right side of node A, and so throughout the domain.

□

MPR Flooding Summary – Our analysis has revealed that the density of MPRs is $\frac{2}{\nu-1}$, where ν is the node density. This implies that for a given flooding operation with this mechanism, a fraction $\frac{2}{\nu-1}$ of the nodes are engaged. Therefore, the denser the network is, the smaller the fraction of nodes is engaged.

7.1.6 Performance Evaluation via Simulation

To complement our theoretical analysis, this section will present a simulation study of the discussed flooding techniques. We evaluate the density of retransmitters involved in a broadcast in each case: gateway flooding and MPR flooding. This evaluation was carried out on a simple custom simulator (implementation from scratch in C) and we tested the performance of the algorithms. In other words, this evaluation does not take into account the actual network layer and the actual exchange of protocol messages between nodes, but rather, is simply based on the unit disk model described in Section 7.1.5. We have simulated both the linear map (as studied mathematically in Section 7.1.5) and the planar map cases (*i.e.* nodes randomly distributed over two dimensions). The linear map simulations should validate the results developed in Section 7.1.5. The planar map simulations will allow us to evaluate if it is possible to extend the results from a one- to a two-dimensional domain. The figures we obtained come from averages over several hundreds of random node distributions.

Gateway Simulations

The simulations for the road map confirm the predictions of the mathematical models of Section 7.1.5. The figures Figure 7.2 and 7.3 each feature two graphs labelled (i) *total* and (ii) *gateway only* (dashed). On one hand (i) shows the total number of nodes that retransmit with this flooding technique, *i.e.* the sum of the number of *internal nodes* and the number of *gateway nodes* (see Section 7.1.3). On the other hand (ii) shows the number of *gateway nodes* only. The results show that when the density is such that nodes have more than a dozen neighbours, half of the nodes retransmit during a broadcast with this technique, *i.e.* counting retransmissions *inside* the trees (the internal nodes) and retransmissions *between* trees (the gateway nodes). This is shown in Figure 7.2. It is also shown that the biggest chunk of retransmitters are gateway nodes, from far (dashed plot).

Though our analytical results do not extend beyond one dimension, we can anticipate that there will be even more retransmitters in two dimensions. The simulations for the planar map are consistent with this anticipation as they show that when the density is such that nodes have a dozen neighbours, $\frac{2}{3}$ of the nodes participate in the broadcast retransmissions. When the density is higher, up to $\frac{3}{4}$ of the nodes turn out to be retransmitters. Once again, as with only one dimension, the vast majority of these retransmitters are gateway nodes, as shown in Figure 7.3.

It is obvious that retransmissions *within a tree* are optimized, following the specifications for gateway flooding. The simulations confirm this point, with very reasonable numbers for internal retransmitters. On the other hand, as we have already stated, the number of gateway retransmissions *between trees* is very substantial. This comes from the fact that no optimization is proposed by this technique to eliminate redundant retransmissions from gateway nodes.

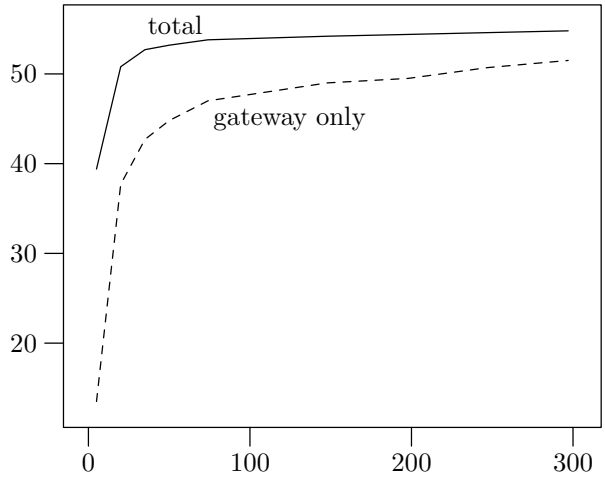


Figure 7.2: Percentage of retransmitters in a one-dimension domain, in function of the node density, with Gateway flooding.

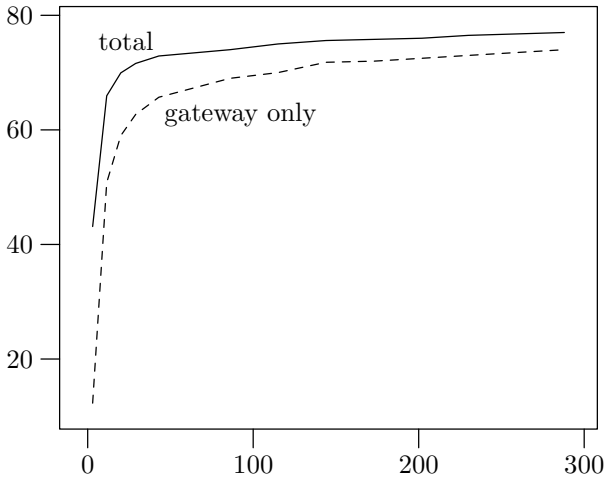


Figure 7.3: Percentage of retransmitters in a two-dimensions domain, in function of the node density, with Gateway flooding.

MPR Simulations

The simulations for the road map confirm the mathematical model of Section 7.1.5. The percentage of retransmitters decreases when the density increases in a way that is roughly proportional to $2/\nu$. This is

shown

in Figure 7.4.

Once again, our analytical results do not extend beyond one dimension, but still we anticipate that the behaviour for one dimension will hold for two dimensions. Indeed, the simulations for the planar map show that the highest percentage of retransmitters is about 45%, which coincides with a density of a dozen neighbours. And further, for higher densities, the one dimension behaviour seems to hold with again a percentage of retransmitters that drastically decreases when the density increases, and such in a hyperbolic fashion. This is shown in Figure 7.5.

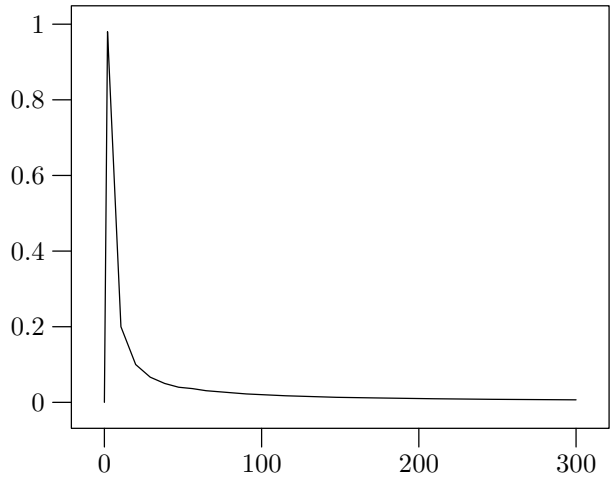


Figure 7.4: Percentage of retransmitters in a one-dimension domain, in function of the node density, with MPR flooding.

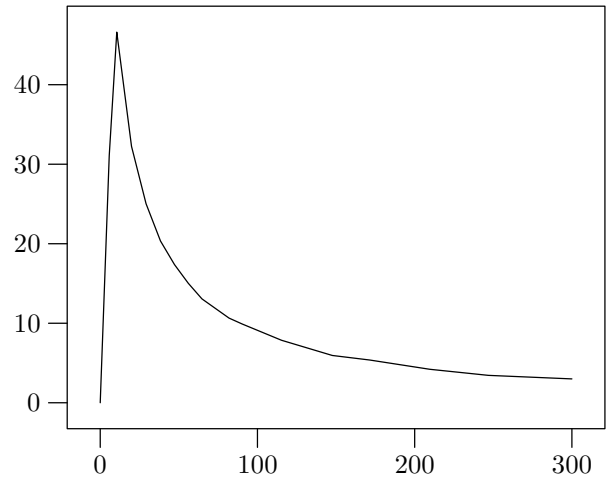


Figure 7.5: Percentage of retransmitters in a two-dimensions domain, in function of the node density, with MPR flooding.

7.2 Partial Topology Optimization Techniques

In this section, we analyze and compare different topology reduction schemes that aim at optimizing the size of the routing information exchanged by nodes. New ways to introduce such schemes in an extension to OSPF are proposed (these were first presented in [16]).

7.2.1 Introduction to Partial Topology

An orthogonal approach to overhead optimization is to aim at reducing the amount of information that nodes need to exchange in order to form and maintain the network. However, there must be enough information exchanged for the network to be fully functional. In particular, the routing information delivered to each router must be sufficient in order for them to be able to construct efficient paths to every destination in the network.

Some techniques have been developed in order to optimize the amount of routing information that needs to be exchanged between nodes. These techniques basically rely on the idea that nodes do not actually need to know the whole topology (*i.e.* all the links between all the nodes). Rather, nodes only need to know about a subset of these links, a *partial topology*, that is essential to construct the paths that will actually be used throughout the network.

Two main approaches to partial topology exist. On the one hand *link state optimization*, and on the other hand the *on-demand optimization*. An overview of these different approaches is given in the following.

7.2.2 On-Demand Topology Information

In the cases where only a few routes usually carry user traffic in the network, it may be possible to reduce the topological information to that which relates to the links that form these currently *active* routes. This is the approach taken by the reactive protocols seen in Section 6.1, also called the *on-demand* approach.

With an on-demand approach, nodes acquire and maintain only the part of the topology that is needed to maintain the routes that have been requested by actual user data traffic. Basically, when a node is requested to find a path to an unknown destination, this node floods a special ROUTE-REQUEST message in the network. This message eventually reaches either (i) the destination itself, or (ii) nodes that know a path to the destination. The destination (or a node that knows how to reach the destination) may then reply with a ROUTE-REPLY message that is sent directly back to the node that requested the route, informing the node

on how to reach the destination.

This approach may radically reduce the amount of routing information needed to be exchanged by nodes. Indeed, the partial information that is needed is only the information relative to the links forming routes that where requested by actual user data. If there are few routes used by traffic in the network, and if nodes are mostly static, this reduction is indeed drastic, while still successfully providing the needed paths.

However, if there are many different routes being used, more ROUTE-REQUEST floodings will be necessary to find these routes. Moreover, if the nodes are mobile, therefore often breaking an established path, even more flooding operations will be necessary. In such cases, what was intended to be a reduction may turn into an augmentation of the incurred overhead, and may jeopardize the successful provision of the needed paths in the network.

Some additional mechanisms can be used in order to attenuate this unwanted behaviour, as in AODV [31], for instance. Orthogonaly, the use of an optimized flooding scheme along with this partial topology approach will indeed help minimize the cost of the necessary flooding operations.

7.2.3 Link State Optimization

A less radical partial topology approach is the *link state optimization* technique. This approach aims at minimizing the size of each individual LSP generated by nodes running a link state routing protocol. It is indeed possible for nodes to exclude from their LSP information about links detected as not being essential to shortest paths computation. Some nodes may even entirely suppress their LSP generation if they detect that neighbours are already generating the necessary information for them.

An example of such an approach is the partial topology scheme employed in OLSR. Based on the MPR mechanism (see Section 7.1), a node generates an LSP containing only the information about links to the neighbours that have selected it as their multi-point relay. This reduces the size of each generated LSP, and suppresses some nodes' LSP generation (the nodes which have not been selected as MPR by any neighbour). One advantage of this method is that it does not depart from routing with complete routing tables: each node still knows about all the destinations in the network, and there is no "acquisition time" to find a new route,

contrary to the on-demand method.

This approach may also drastically reduce the amount of routing information needed to be exchanged by nodes in the network, especially if the network is dense. By suppressing some nodes' LSP generation, a reduction in the number of needed transmissions overall is achieved. If numerous different routes carry user traffic throughout the network, and nodes are mobile, this approach may provide a better optimization than the on-demand approach, especially when coupled with optimized flooding as in OLSR. However, if only a few routes are used and nodes are mostly static (typically for a sensor network), the on-demand approach may provide a better overhead reduction.

7.2.4 Partial Topology Optimizations for Overlapping Relays OSPF

Using reduced topology within link state routing is an efficient way to decrease routing overhead while still providing sufficient route quality. There are various ways to achieve topology reduction with link state protocols, based on different ways to form a backbone in the network – this backbone usually originates from the flooding optimization scheme in use, such as MPR or CDS.

In case of mobile ad hoc networks, flooding using MPR backbones is preferable as it is more robust in face of topology changes, compared to flooding using CDS backbones. This section therefore describes several methods to enable the use of reduced topology in wireless OSPF for MANETs, when MPR-based flooding optimizations are used. The topology reduction methods that are here proposed for an MPR-based approach perform at least as well as the similar schemes that were recently proposed for CDS-based approaches. Indeed, as described in the following sections, the same CDS-based topology reduction schemes can be used over MPR backbones, while furthermore MPR-based topology reduction schemes are also possible with MPR backbones – these feature optimal properties that are not obtained with the CDS-based approaches that were proposed so far.

OSPF and MANET Background

One of the most fundamental conlusion of the research accomplished so far in the field of mobile ad hoc networking, is that the flooding overhead must be reduced, one way or another. The flooding optimization algorithms established by the research community are based on reducing the number of nodes actively participating in the forwarding of a given flood. Though there are many such different algorithms, they can nevertheless be classified in two main categories: (i) the algorithms that bring members of the set of forwarders to *select themselves* as part of this set, and (ii) the algorithms that bring members of the set of forwarders to *be selected* by their neighbours. CDS algorithms [76] are examples of the (i) approach, while the MPR algorithm is the archetype (ii) approach. However, going further than this classification, it is to be noted that the (i) approach is a special case of the (ii) approach (see Section 7.1.4).

Another fundamental conlusion mobile ad hoc research is that the use of partial topology, if done correctly, is also an efficient way to help reduce the amount of bandwidth used by a routing protocol. An example of such a mechanism is the partial topology strategy developed with OLSR [26], enabling the use of reduced topology while still guaranteeing shortest paths and not impairing nework connectivity, or stability. Fig. 7.6 shows the substantial decrease in overhead with the use of partial topology schemes (bottom curves*) compared to the full topology overhead (top curve).

Recent efforts in the IETF [63] [67] [66] propose designs for an extension of the OSPF routing protocol for MANETs. Several proposals are being evaluated, including Overlapping Relays OSPF [85], MDR OSPF [84] and wOSPF, aiming to converge to a common extended OSPF standard.

The proposals for wireless OSPF on MANETs each feature an optimized flooding mechanism (based on CDS for MDR OSPF, while based on MPR for Overlapping Relays OSPF and wOSPF). MPR-based flooding is preferable as it is more robust than CDS-based flooding in face of topology changes and mobility [15]. However, partial topology schemes have been described in MDR OSPF for a CDS-based solution, while similar schemes are yet to be described for an MPR-based wireless OSPF. In the following we will therefore present approaches to partial topology for Overlapping Relays OSPF.

The following section will present methods to use reduced topology based on MPR backbones achieving at least as good results as the ones obtained using the CDS schemes described in MDR OSPF. However,

*The CDS was built with the Wu-Li rules [76].

based on MPR or CDS backbones, these schemes will produce slightly sub-optimal route quality: they introduce some route stretching, *i.e.* routes may include some unnecessary hops. Therefore, another section will next present other partial topology methods based on MPR backbones, achieving optimal route quality: no route stretching, the shortest paths are used.

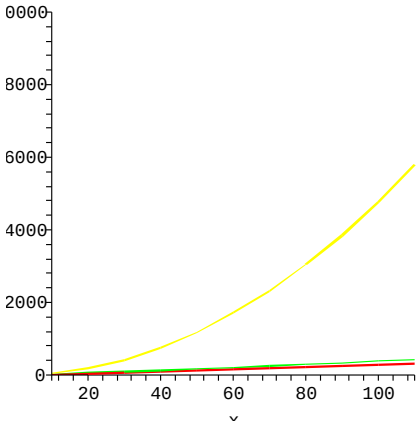


Figure 7.6: Full topology overhead (top) compared with partial topolgy overhead. Reduction to MPR selection links (bottom) and reduction to links to CDS nodes (middle). The overhead is measured in number of IP addresses (4 bytes per IP address), in function of the number of nodes in the network.

Approach with Route Stretch

In this section we will outline an approach to the use of reduced topology based on MPR backbones that achieves at least as good results as the ones obtained using the CDS schemes described in MDR OSPF.

An MPR-based approach, such as Overlapping Relays OSPF or wOSPF, can *decouple* its flooding mechanism from traditional OSPF Designated Routers mechanisms. Designated Routers are nodes that are given a special role of topology centralization and topology reduction in an OSPF network (see Chapter 3.2). In that respect, an MPR-based approach can separate flooding optimization on one hand, and topology optimization schemes on the other hand.

A way to achieve this is to consider the MPR-CDS [54] laying naturally on top of the MPR selections. This CDS can be used, along with the same topology reduction techniques proposed in MDR OSPF. This approach then provides the same partial topology, and the same route quality properties (which means some

amount of route stretching) as obtained with MDR OSPF. The advantage with the present solution is that the robustness of MPR flooding is kept, while still obtaining the same gains in topology reduction.

In fact, any other kind of CDS can be used on top of an MPR-based approach. The cost of using a different CDS is the additional complexity due to the computation of this specific CDS. Furthermore, any other kind of topology reduction technique may be used along with the chosen CDS, as long as it does not impair network stability, or connectivity.

The decrease in overhead with this method is shown with Fig. 7.6, comparing the middle curve to the full topology overhead (the top curve). In fact, the decrease in overhead may be slightly bigger, depending on the chosen topology reduction scheme. However, wether they are used over an MPR approach or a CDS approach, the schemes proposed in MDR OSPF, will produce slightly sub-optimal routes. These schemes introduce some route stretching, that is to say: routes may include some unnecessary hops (*i.e.* transmissions). Indeed, some links may not be advertized network-wide while they are in fact needed to construct optimal routes, as CDS topology reduction typically reduces the advertized topology to links towards the CDS backbone. Fig. 7.7 displays a simple example of route stretching, where the link between B and C is not known network-wide. Therefore, if node F wants to reach C, it will route through node E (in 4 hops), while the optimal route is through node A (in 3 hops).

Route stretching is a concern in an environment where the number of transmissions are to be as limited as possible to reduce the bandwidth consumption, interference issues, number of collisions, or power consumption of the nodes in the MANET. In fact, route stretching really introduces some additional routing overhead, as it reduces the available bandwidth that can be used for data traffic. In other words, a 10% route stretch factor (*i.e.* routes being 10% longer) means a 10% decrease in available bandwidth for data traffic.

In the next section, we will therefore outline other approaches to partial topology based on MPR backbones, that do not introduce route stretching, and provide optimal routes.

Approach without Route Strech

In this section we will describe another partial topology method for MPR-based wireless OSPF solutions. Contrary to approaches described in MDR OSPF and in the previous section, this method does not degrade route quality.

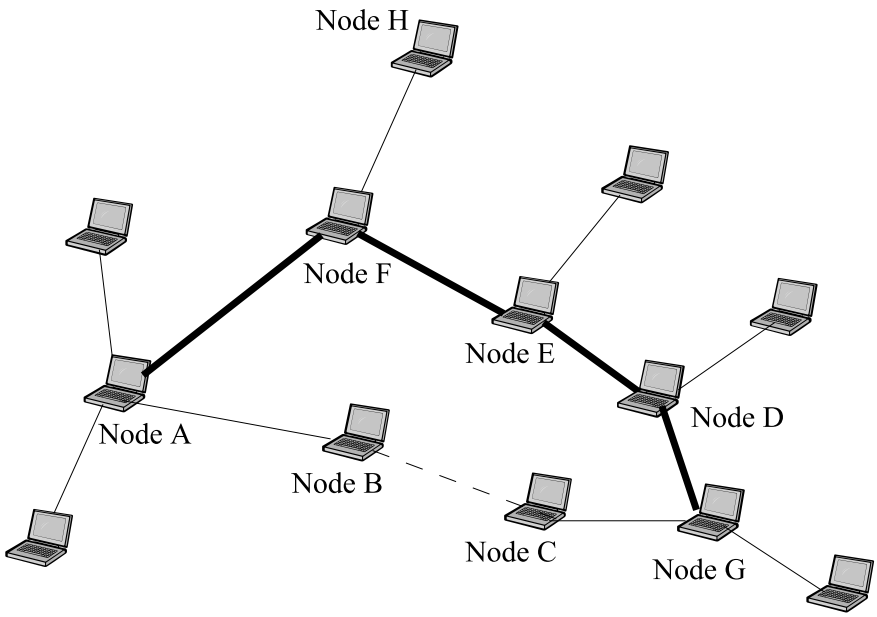


Figure 7.7: CDS topology reduction. The backbone is optimally formed by nodes A, F, E, D and G. The reduction scheme will typically slim down the advertized topology to links towards the CDS backbone. The link between nodes B and C is not advertized network-wide, and therefore not known to node F.

Based on the approach developed in OLSR, and also taken by wOSPF, an MPR-based solution can provide partial topology without incuring any route sub-optimality, *i.e.* as if the full topology was indeed used.

Route optimality is achieved with efficient reduction of the flooded topology information to only the state of links between MPRs and their selectors, while still using the full topology information available locally. Analysis shows that this approach provides shortest paths (see [78]), as if the full topology was indeed advertized and flooded, while still drastically reducing the flooding overhead. In particular, contrary to CDS-based topology reduction approach, cases such as shown in Fig. 7.7 will not cause suboptimal routing with MPR topology reduction since node B will select A and C as MPRs, and the link between B and C will thus be known network-wide.

More precisely, in a wireless OSPF framework, wOSPF specifies that adjacencies are not to be formed between routers and that routing is done over any bi-directionnal link, as in OSLR. Such an approach relies

on periodic transmission of LSAs with short enough intervals so that it is more beneficial to just transmit updated information periodically, rather than to verify that the old information got through (through traditional adjacency OSPF mechanisms such as Acknowlegements or Database Exchange see Chapter 3.2).

The decrease in overhead with this method is shown with Fig. 7.6, comparing the bottom curve to the full topology overhead (the top curve).

However, if this approach to link state routing is desireable in most MANET environments, in some cases it is not totally appropriate. These cases include scenarii targeted by the wireless OSPF design: mixing wired nodes and mobile nodes, or more generally, cases featuring more stable topologies, where updating topological information too often is wasteful. In these cases, the period with which different nodes update link information may vary greatly (from a few seconds, up to an hour), and this lack of homogeneity breaks the assumption that any update will come soon enough anyways.

In these cases, it is desireable to keep adjacencies and acknowledgements. Therefore an intermediate approach bringing optimal paths, but using partial topology, is to form adjacencies only between MPRs and their selectors. This yields more overhead than the approach described in Section 7.2.4 but the reward comes from being able to keep away from any route stretching and only use optimal, shortest paths.

7.3 Optimizing Reliability for Link State Information

In this section, we introduce the notion of database exchange and reliable synchronization in the context of mobile ad hoc networks. Inspired by the mechanisms from the routing protocol OSPF, it is shown that in their present form, these mechanisms are not suitable for the mobile ad hoc domain. We therefore introduce a new link state synchronization mechanism that is optimized for the specific environment of wireless ad hoc networks. This mechanism is proposed as an extension to wOSPF, and was first presented in [4].

7.3.1 Introduction to Reliable Link State Synchronization

In order to ensure that the link state algorithm functions correctly, the link state databases in every node must be as synchronized as possible. In order to protect databases from discrepancies, OSPF features mechanisms such as reliable flooding and database exchange (see Section 3.2). On the other hand, OLSR rather relies on frequent unreliable flooding of topology information, that is more adapted to dynamic topologies and ad hoc environments. The idea behind the periodic unreliable flooding of topology information is that since the topology of the network is thought to be changing rather frequently, link state packets (LSP) are transmitted periodically and frequently to reflect these changes. Consequently, loss of a single LSP is relatively unimportant since the information contained within the message will be repeated or updated shortly.

This approach may work well if LSP periods are roughly homogeneous and short enough. However, in a heterogeneous network with both traditional wired parts and wireless ad hoc parts, wired nodes will have rather long LSP generation period (down to one per hour), whereas wireless nodes will typically generate LSPs much more frequently (often more than one per minute). In this case, of course, the short period argument fails, at least for the LSPs with a long period, and there is a need to device mechanisms ensuring “OSPF-like” treatment for long period LSPs.

7.3.2 Managing Wired, Ad Hoc Heterogeneity

Managing wired/ad hoc heterogeneity is the goal of a protocol such as wOSPF, an extension to OSPF that aims at adapting OSPF for ad hoc networks (refer to Section 3.2 for information on OSPF, and Section 6.3 for information on wOSPF). However, OSPF’s reliable flooding with acknowledgements and database exchange mechanism are not appropriate for ad hoc networks.

OSPF's reliable flooding employs positive acknowledgements (ACK) on delivery, with retransmissions. In other words, an ACK is a "retransmission-repressing" message. In mostly static point-to-point-like network topologies (as for instance fixed wired networks), ACKs and retransmissions occur over a single link. An ACK transmitted by the recipient of an LSP will be received by a node which is directly able to interpret the ACK message, as the recipient of an ACK will be the node which sent the LSP to which the ACK corresponds.

In wireless ad hoc networks, the network topology may be changing frequently, due to node mobility. Interfaces are typically wireless of broadcast nature, and any transmission may thus interfere with all the neighbours of the node originating the transmission. An ACK, which can be only interpreted by the node which relayed corresponding LSA, will thus be interfering with all the nodes in the neighbourhood. If, due to node mobility or fading radio links, a node does not receive an expected ACK, unnecessary (re)transmissions will occur, consuming precious bandwidth. In other words, reliable topology information diffusion through ACK's imposes the assumption that the network conditions are such that an ACK that is sent can be received by the intended node. This does not hold for a wireless ad hoc network, where the network may be substantially more dynamic (nodes may move out of range *etc.*).

OSPF's database exchanges are intended to synchronize the link-state database between routers. In OSPF, database description packets are exchanged between two nodes through one node (the master) polling an other node (the slave). Both polls and responses have the form of database description packets containing a set of LSA headers, describing (a partial set of) the respective link-state databases of each of the two nodes. These database description packets are used by the nodes to compare their link-state databases. If any of the two nodes involved in the exchange detects it has out-of-date or missing information, it issues link-state request packets to request the pieces of information from the other node, which would update its link-state database.

In wireless ad hoc networks, wireless broadcast interfaces and a higher degree of node mobility are typically assumed. Therefore, inconsistencies between the link-state databases of the nodes in the network should occur more frequently, calling for more frequent database synchronizations. Moreover, the broadcast nature of the network interfaces implies that the bandwidth in a region is shared among the nodes in that region and thus less bandwidth is available between any pair of nodes to conduct the database exchange. At the same time, lists of complete LSA headers can amount to a lot more overhead than necessary.

In this section we therefore describe a mechanism, adapted for the low-bandwidth high-dynamics condi-

tions of wireless ad hoc networks, for conducting efficient database synchronization and reliable flooding in wOSPF. This mechanism aims at providing these functionalities while optimizing both the amount of information that needs to be exchanged and the number of transmissions that have to take place for their operation.

Database Signatures

The basic idea is to employ an exchange of compact ”signatures” (hashing of the link state database) between neighbour nodes, in order to detect differences in the nodes' link state databases. When a discrepancy is detected, the bits of information required to synchronise the link state databases of the involved nodes are then identified and exchanged. The purpose of the exchange is to provide the nodes with a consistent view of the network topology – the task is doing so in an efficient way.

The proposed approach is somewhat inspired by IS-IS [70], in which packets which list the most recent sequence number of one or more LSPs (called Sequence Numbers packets) are used to ensure that neighbouring nodes agree on the most recent link state information. In other words, rather than transmitting complete LSA headers as done in OSPF, a more compact

representation for database description messages is employed. Sequence Numbers packets also accomplish a function similar to conventional acknowledgement packets.

The method described here differs from the mechanism employed in IS-IS by the use of age. For example, it may be considered a waste of resources to check for database consistency for LSAs issued from within a very dynamic part of a wireless ad-hoc network (*e.g.* RFID tags on products in a plant): LSAs from nodes within this domain should be transmitted frequently and periodically, thus information describing these nodes is frequently updated and “with a small age”. LSAs from a less mobile part of the wireless ad-hoc network (*e.g.* sensors on semi-permanent installations in the plant) might be updated less frequently. Thus consistency of these corresponding entries in the link-state databases should be ensured.

The following subsections outline how database signatures are generated, exchanged, interpreted and used for correcting discrepancies.

7.3.3 Definition of Link State Database Signatures

We define a signature message as a tuple of the following form:

Signature Message = (Age Interval, Key, Prefix Signature).

A signature features a set of prefix signatures:

Prefix Signature = (Prefix, Sign(Prefix)).

Each *Sign(Prefix)* results from hashing functions computed on the piece of the link state database matching the specified prefix, and represents this part of the database in the signature message.

More specifically, each *Sign(Prefix)* has the following structure:

$$\begin{aligned} \textit{Sign(Prefix)} \quad = \quad & \textit{(Primary Partial Signature, Secondary Partial Signature,} \\ & \textit{Timed Partial Signature, \#LSA, Timed \#LSA).} \end{aligned}$$

A primary partial signature (PPS) for a prefix is computed as a sum over all LSAs in a nodes link-state database, where the prefix matches the advertising router of the LSA:

$$PPS = \sum_{prefixes} \textit{(Hash(LSA-identifier))},$$

with $\sum_{prefixes}$ denoting the sum over prefixes matching the advertising router of the LSA. The secondary partial signature (SPS) for a prefix is similarly computed as a sum over all LSAs in a nodes link-state database, where the prefix matches the advertising router of the LSA, but also uses a random key as follows:

$$SPS = \sum_{prefixes} \textit{(Hash(LSA-identifier))} \cdot key,$$

with $\sum_{prefixes}$ denoting the sum over prefixes matching the advertising router of the LSA. The timed partial signature (or TPS) for a prefix and an age interval is computed over LSAs in a nodes link-state database where:

- the prefix matches the advertising router of the LSA,
- the age falls within the age interval of the advertisement,

and has the following expression:

$$TPS = \sum_{prefixes,time} \textit{(Hash(LSA-identifier))},$$

with $\sum_{prefixes,time}$ denoting the sum over prefixes matching the advertising router of the LSA and where the age falls within the age interval of the advertisement. The LSA identifier is the string, obtained through concatenating the following LSA header fields:

- LS type
- LS ID
- Advertising router
- LSA sequence number

7.3.4 Signature Exchanges

Signatures are exchanged between nodes in two forms: informational signatures, broadcast periodically to all neighbour nodes, and database exchange signatures, employed when a node requests a database exchange with one of its neighbours.

Informational Signature Exchange – Each node periodically broadcasts informational (info) signatures, as well as receives signatures from its neighbour nodes. This exchange allows nodes to detect any discrepancies between their respective link-state databases. The following sections detail how info signatures are generated and employed to detect link-state database discrepancies.

Database Signature Exchange – Database exchange (dbx) signatures are directed towards a single neighbour only. The purpose of emitting a dbx signature is to initiate an exchange of database information with a specific neighbour node.

When a node detects a discrepancy between its own link-state database and the link-state database of one of its neighbours, a database exchange is needed. The node, detecting the discrepancy, generates a dbx signature, requesting a database exchange to take place. In OSPF terms, the node requesting the database exchange is the ”master” while the node selected for receiving the dbx signature is the “slave” of that exchange. The dbx signature is transmitted with the destination address of one node among the discrepant neighbours. The node builds a dbx message signature, based on the information acquired from the info signature exchange.

Signature Message Generation

This section details how info and dbx signature messages are generated.

Info Signature Generation – An info signature message describes the complete link state database of the node that sends it. Absence of information in a signature indicates absence of information in the sending nodes link state database – in other words, if no information is given within an informational signature about a specific prefix, it is to be understood that the sending node has received no LSAs corresponding to that prefix.

The set of prefix signatures in an informative signature message can be generated with the following splitting algorithm, where the length L of the info signature (the number of prefix signatures in the message) can be chosen at will.

We define the weight of a given prefix as the function:

Weight(prefix) = # of LSAs whose originator matches the prefix.

And similarly, the timed weight as the function:

Timed Weight(prefix) = *# of LSAs whose originator matches the prefix and whose age falls inside the age interval*.

Then, starting with the set of prefix signatures equal to *(0,signature(0))*, recursively do the following.

As long as: |*set of prefix signatures*| < *L*

1. Find in the set of prefix signatures the prefix with largest timed weight, let it be called *mprefix*.
2. Replace the single *(mprefix, signature(mprefix))* by the pair *(mprefix0,signature(mprefix0)),(mprefix1,signature(mprefix1))*.
3. If one of the expanded prefix of *mprefix* has weight equal to 0, then remove the corresponding tuple.

The proposed format of info signature messages are shown in Appendix D.

Dbx Signature Generation – Dbx signatures serve to trigger an exchange of discrepant LSAs with one neighbour, known to have more up-to-date link-state information – the ideal is to pick the neighbour which has the "most complete" link-state database and which at the same time is going to remain a neighbour for a sufficient period of time. In wOSPF, database exchanges are to be conducted in preference with nodes selected as MPR.

The set of prefix signatures in a database exchange signature message can be generated with the following algorithm, where the length L of the dbx signature (the number of prefix signatures in the message) can be chosen at will.

1. Start with the same set of prefix signatures as one of the received info signature where the discrepancies were noticed.

2. Remove from that set all the prefix signatures such that signature(prefix) is not discrepant (with the LSA database). Use the same age interval and key used in the received info signature. Then use the recursive algorithm described for info signatures, skipping step 3.

Indeed, contrary to info signature messages, the prefixes with zero weight are not removed here, since the signature is not complete, i.e. the signature might not describe the whole database. Therefore a prefix with empty weight may be an indication of missing LSAs.

The proposed format of dbx signature messages are shown in Appendix D.

Checking Signatures

Upon receiving a signature message from a neighbour, a node can check its local LSA database and determine if it differs with the neighbour's database. For this purpose, it computes its own prefix signatures locally using the same prefixes, time interval and key specified in the received signature message. A prefix signature differs with the local prefix signature when any of the following conditions occurs:

1. both the number of LSAs and the timed number of LSAs differ;

2. both the timed partial signatures and the (primary partial signature, secondary partial signature) tuples differ.

The use of a secondary signature based on a random key is a way to cope with the infrequent, but still possible, situations when the primary signatures agree although the databases differ. In this case, it can be assumed that using a random key makes the probability of both primary and secondary signatures agreeing while databases are different very small.

7.3.5 Database Exchange

When a node receives a dbx signature with its own ID in the destination field, the node has been identified as the slave for a database exchange. The task is, then, to ensure that information is exchanged to remove the discrepancies between the link-state databases of the master and the slave.

Thus, the slave must identify which LSA messages it must retransmit, in order to bring the information in the master up-to-date. The slave must then proceed to rebroadcast those LSA messages.

More precisely, the slave rebroadcasts the LSA messages which match the following criteria:

- the age belongs to the age interval indicated in the dbx signature, AND
- the prefix corresponds to a signed prefix in the dbx signature, where the signature generated by the master differs from the signature as calculated within the slave for the same segment of the link-state database.

When a node is triggered to perform a database exchange, it generates a new LSF (see Section 6.3) with TTL equal to 1 (one hop only) and fills it with the update LSAs. These LSAs must indicate the age featured at the moment in the database from which they are taken.

Optionally, the host can use a new type of LSF denoted an LSF-D, which is retransmitted making use of MPRs, contrary to the one-hop LSF described above. An LSF-D is transmitted with TTL equal to infinity. Upon receiving such a packet, successive nodes remove from the LSF-D the LSAs already present in their

database before retransmitting the LSF-D. If the LSF-D is empty after such a processing, a node will simply not retransmit the LSF-D. The use of LSF-D packets is more efficient for fast wide-area database updates in case of merging of two independent wireless networks.

7.3.6 Performance Evaluation of the Database Signature Exchange Mechanism

In this section we estimate the performance of the database signature exchange protocol. In this analysis we consider the "cost" of the protocol's operation when two neighbour's databases differ on a single record. While this is a special case, it gives a good idea of what kind of performance gains we can get.

We denote by n the number of records in the database (typically n can range from a few tens to a thousand) and by Q the number of signatures contained in a signature message (typically $Q = 10$). To simplify we assume that a signature and a record exchange yield the same cost. Let this cost be the unit.

Synchronizing a Single Mismatch

Let D_n be the average cost of database retrieval when the mismatch occurs on a random record among n. In this sections, in order to estimate D_n, we will consider that the number of records in the database is P, a poisson variable of mean n.

Theorem 6 *The average recovery cost of a single mismatch is bounded by:*

$$D_n \leq 2Q + \sum_{i=0}^{\infty} 1 - \left(1 - \sum_{k=i}^{\infty} e^{\frac{-n}{Q^2}} \frac{n^k}{k!Q^{2k}}\right)^{Q^2},$$

where Q is the number of signatures contained in a signature message.

Proof – The signature mechanism yields an exchange consisting in (i) an info signature, responded to by (ii) a dbx signature, which is in turn responded to by (iii) the LSF(s) containing the part of the database that is identified to be discrepant.

The info signature generation implies the partitionning of the database in Q parts (based on IP prefixes). Let us consider that the records (IP addresses) are uniformly distributed, and therefore, a record has an equal

chance to fall into each of the Q parts. Then the responding dbx signature generation implies further partitionning of one of the Q parts into Q subparts. Similarly, the records in the part have an equal chance to fall into each of the subparts.

This operation is therefore equivalent to the equiprobable dispatching of the database into Q^2 parts, which gives birth to Q^2 independent poisson variables P_i (for $i = 1$ to Q^2) of mean $\frac{n}{Q^2}$ (since the number of records in the database is a poisson variable of mean n). Let P_k be the part of the database which contains the mismatch.

Then since an info signature and a dbx signature each yield a cost Q, and the update a cost P_k, we have that:

$$D_n = 2Q + P_k.$$

Let P_{max} be the maximum of the variables P_i (from $i = 1$ to Q^2). We then have the following bound:

$$D_n \leq 2Q + P_{max}.$$

Let us denote $f(a)$ the probability that $P_1 \geq a$. Since the P_i variables are independent and identically distributed, we can say that the probability that $P_{max} \geq a$ is equal to the value $1 - (1 - f(a))^{Q^2}$.

Therefore we can compute the mean of P_{max} as $\sum_{i=0}^{\infty}(1 - (1 - f(i))^{Q^2})$, which terminates the proof of the theorem.

□

Fig. 7.8 shows the bound for the cost of a single mismatch, for $Q = 10$ and n varying from 100 to 1000. This cost is in fact divided in at least 3 different packet transmissions: one for the info signature, one for the responding dbx signature, and at least one for the database update. Indeed, there may be more than one transmission for the update, depending on the size of the update and on the maximum size of a single transmission allowed by the network.

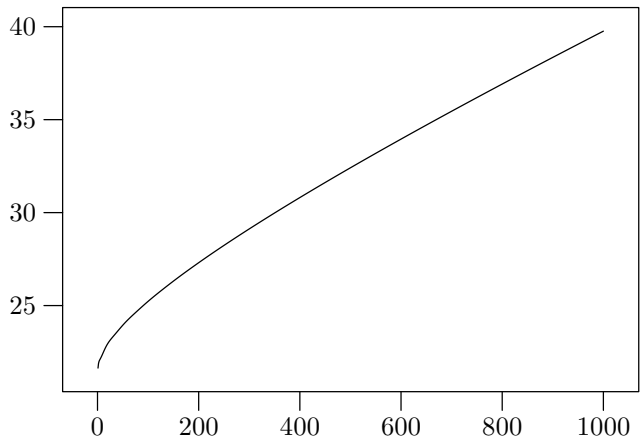


Figure 7.8: Bound for the signature retrieval cost with a single record mismatch and $Q = 10$. The cost is measured in number of exchanged IP addresses, and it is shown here in function of the size of the database, also measured in number of IP addresses.

Synchronizing a Single Mismatch with the Narrowing Enhancement

Let us denote b the maximal size of a single transmission (typically $b = Q$). We can then also evaluate the signature mechanism modified in that it continues exchanging signature packets, "narrowing down" until the discrepant part of the database is identified with a size smaller than b (and only then updates the database with an LSF).

This enhanced mechanism should yield a lower cost when n is large, *i.e.* the cost should $O(\log n)$ instead of $O(n)$, since we are basically browsing down the tree of successive Q-partitions of the database, instead of simply stopping after two partitions. However it should imply a slower convergence, since it implies more signature computation and exchanges between slave and master. Nevertheless, for $n \leq 1000$ and $b = Q$ around 10, the simple mechanism and the enhanced mechanism should yield approximately the same cost, since with these values, $\frac{n}{Q^2}$ is close to b (or smaller). Therefore, on average, the enhanced method will anyways also use only two successive Q-partitions, and will consider that to be narrow enough.

In the following, we study the case where $n > b$ (the whole database cannot be transmitted in a single packet), and we take the convention $D_n = n$, for $n < b$. Contrary to our first analysis, the size of the database is no longer considered to be a random variable: it is determined to be n.

Theorem 7 *The average recovery cost of a single mismatch with the narrowing enhancement is:*

$$\begin{aligned} D_n &= \frac{1}{\log Q}(Q+1-\frac{1}{Q})\log n \\ &+(Q+1-\frac{1}{Q}H_{b-1}+\frac{Q-1}{Q^2}b) \\ &+P(\log n)+O(\frac{1}{n}), \end{aligned}$$

where $H_k = \sum_{i=1}^{k} \frac{1}{i}$ *denotes the harmonic sum, Q the number of aggregated signatures contained in a signature message, and* $P(x)$ *is a periodic function of period* $\log Q$ *with very small amplitude.*

Proof – Simple algebra on random partitions yields, with the multinomial development:

$$D_n = Q + \sum_{n_1+\cdots+n_Q=n} Q^{-n}\binom{n}{n_1\cdots n_Q}(\frac{n_1}{n}D_{n_1}+\cdots+\frac{n_Q}{n}D_{n_Q}).$$

Then, if we denote $S_n = nD_n$, we have:

$$S_n = nQ + \sum_{n_1+\cdots+n_Q=n} Q^{-n}\binom{n}{n_1\cdots n_Q}(S_{n_1}+\cdots+S_{n_Q}).$$

Denoting $S(z) = \sum_n S_n \frac{z^n}{n!}e^{-z}$ the so-called Poisson generating function of S_n, we get the functional equation:

$$\begin{aligned} S(z) &= QS(\frac{z}{Q})+Qz-((Q+1-\frac{1}{Q})ze_b(z) \\ &+\frac{(Q-1)}{Q^2}z^2e_{b-1}(z))e^{-z}, \end{aligned}$$

with the convention that $e_k(z) = \sum_{i\le k} \frac{z^i}{i!}$. Let $D(z) = S(z)/z$. We therefore have the functional equation:

$$\begin{aligned} D(z) &= D(\frac{z}{Q})+Q-((Q+1-\frac{1}{Q})e_b(z) \\ &+\frac{(Q-1)}{Q^2}ze_{b-1}(z))e^{-z}. \end{aligned}$$

Using the Mellin transformation $D^*(s) = \int_0^\infty D(x)x^{s-1}dx$, defined for $\Re(s) \in]-1,0[$, and using the fact that the Mellin transformation of $D(\frac{z}{Q})$ is $Q^sD^*(s)$, we get to the closed form solution:

$$D^*(s) = \frac{\Gamma_b(s)(Q+1-1/Q) + \Gamma_{b-1}(s+1)(Q-1)/Q}{1-Q^s},$$

with the convention that $\Gamma_k(s)$ is the Mellin transformation of $e_b(z)e^{-z}$. We note by the way that, $\Gamma_k(s) = \sum_{i \leq k} \frac{\Gamma(s+i)}{i!}$.

The reverse Mellin transformation then yields:

$$D(x) = \frac{1}{2\mathbf{i}\pi} \int_{c-\mathbf{i}\infty}^{c+\mathbf{i}\infty} D^*(s) \exp(s \log x) ds,$$

for any c such that $\Re(c) \in]-1, 0[$. When the integration path moves to the right, it encounters a succession of singularities on the vertical axis $\Re(s) = 0$. There is a double pole on $s = 0$ and there are single poles on $s_k = \frac{2\mathbf{i}k\pi}{\log Q}$ for k being integer. Therefore, by virtue of singularity analysis, we have for any m:

$$\begin{aligned} D(x) &= -\mu_0 \log x - \lambda_0 \\ &\quad - \sum_k \lambda_k \exp(-2\mathbf{i}k\pi \frac{\log x}{\log Q}) \\ &\quad + O(\frac{1}{x^m}), \end{aligned}$$

where λ_0 and μ_0 are, respectively, the first and second order residues of $D^*(s)$ at $s = 0$, and λ_k is the first order residue at $s = s_k$. Classic calculus can be used to determine the residues. Notice that $P(x) = -\sum_k \lambda_k \exp(-2\mathbf{i}k\pi \frac{\log x}{\log Q})$, which is periodic of period $\log Q$. Also note that the estimate is true for every m, since there are no more singularities in the right half plan.

We can then use the depoissonization theorem to assess that $S_n = nD_n = nD(n) + O(1)$ when $n \to \infty$, which terminates the proof of the theorem.

□

Figure 7.9 shows the asymptotic behaviour of retrieval cost with the narrowing enhancement, with $Q = b = 16$ for n varying from 100 to 1,000. It is compared with the cost of the full database of OSPF (*i.e.* without any signature mechanism).

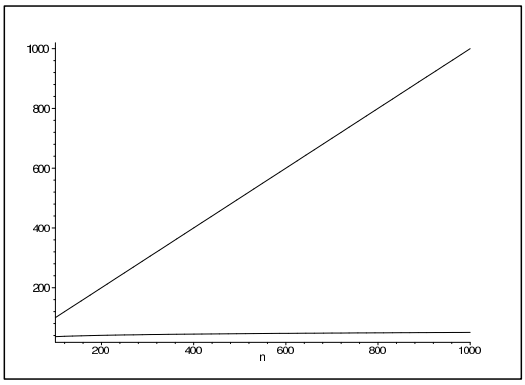


Figure 7.9: Signature retrieval cost (bottom) compared with full database retrieval cost (top) with a single record mismatch, $Q = 16$ and $b = 16$. The cost is measured in number of exchanged IP addresses, and it is shown here in function of the size of the database, also measured in number of IP addresses.

7.3.7 Applicability of the Database Signature Exchange Mechanism

This section outlines the applicability of the specified mechanisms in a set of common scenarii. One application has been discussed previously, ensuring that information from LSA messages originating from attached wired networks with potentially long intervals between LSA message generation, is maintained in all nodes in the wireless ad-hoc network. The scenarii outlined in this section go beyond that situation, and consider how the database signature exchange may apply even in pure wireless scenarii.

Emerging Node

When a new node emerges in an existing network, the initialisation time for that node is the time until it has acquired link-state information, allowing it to participate fully in the network. Ordinarily, this time is determined solely by the frequency of control traffic transmissions. In order to reduce the initialisation time, the database exchange mechanisms can be employed as soon as the node has established a relationship with one neighbour node already initialised. This emerging node will select a neighbour as slave and transmit a dbx signature of the form ([age min, age max] , (*, signature(*))), with "*" implying an empty prefix. The slave will respond by, effectively, offering its entire link-state database to the master. In particular in situations where some LSAs are not transmitted frequently (outside LSAs would be an example of such), this mechanism may drastically reduce the initialisation time of new nodes in the network.

Merging Wireless Clouds

Two disjoint sets of nodes, employing wOSPF as their routing protocol, may at some point merge or join – *i.e.* a direct (radio) link is established between the two sets. Prior to the merger, the respective clouds are ”stable”, periodically transmitting consistent info signatures within their respective networks. At the point of merger, at least two nodes, one from each network, will be able to establish a direct link and exchange control traffic. The combined network is now in an unstable state, with great discrepancies between the link-state databases of the nodes in the formerly two networks. Employing signature and database exchanges through the LSF-D mechanism, the convergence time until a new stable state is achieved can be kept at a minimum.

Reliable Flooding

If a node wants a specific LSA to be reliably transmitted to its neighbour, the db signature mechanism can be employed outside of general periodic signature consistency check. The node transmitting the LSA message broadcasts an info signature, containing the full LSA-originator ID as signed prefix and a very narrow age interval, centered on the age of the LSA which is to be reliably transmitted. A neighbour which does not have the LSA in its database will therefore automatically trigger a database exchange concerning this LSA and send a dbx signature containing the LSA-originator ID signed with an empty signature. The receiving of such a dbx signature will trigger the first node to retransmit the LSA right away with a new LSF to ensure that the LSA does get through.

Chapter 8

Scalability Solutions for Massive Ad Hoc Topologies

Experience has shown that the considered ad hoc routing solutions (OLSR, AODV [31], DSR [33], TBRPF [32], or wOSPF) cannot sustain an arbitrary number of nodes in the network. Although they already feature optimizations such as the mechanisms described in Chapter 7, these routing protocols fail to provide nodes with a connected network when nodes are too numerous.

With proactive protocols (OLSR, TBRPF, or wOSPF), the amount of traffic due to the routing protocol's functionning grows steadily with the number of nodes in the network – at least in principle (see Section 6.1). Therefore, since the wireless links have a limited bandwidth, there is an obvious limit in the number of nodes that can be supported. Over this limit, the available bandwidth is completely occupied by the control traffic, and there is "no space" left for users to communicate their data over the network – which is absurd. If the number of nodes grows further than this limit, the available bandwith is not even enough for control traffic needs, and the routing protocol will fail to construct the network properly.

With reactive protocols (AODV, DSR), the amount of traffic due to the routing protocol's functionning is not directly tied to the number of nodes in the network (see Section 6.1). However, if the nodes are not static and if there are more than a few active routes used throughout the MANET (which is in general the case in a mobile ad hoc network), reactive protocols also rely on a growing amount of control traffic. Therefore the same wireless bandwidth limitation produces the same effect as with proactive protocols: a limitation in the maximum number of nodes that can be supported.

For instance, experience with OLSR as specified in [26] shows some serious problems over 100 nodes in the network (with 802.11b as the underlying wireless technology). Depending on the traffic patterns and the mobility of the nodes, AODV as specified in [31] shows even worse scaling properties on the same network. A horizon limited to 100 nodes for ad hoc routing can surely be improved, but how far can we go? This chapter presents an analysis on this subject, and mechanisms enhancing the scalability of ad hoc routing – that can be used in addition to the techniques described in Chapter 7. In the following, we will focus on link state routing with OLSR.

The approaches that will be described offer scalability in "orthogonal" and complementary ways. Trees propose a "vertical" scaling with the introduction of clusters and hierarchical networking (connecting to Chapter 7, where trees were already studied in a dense environment). On the other hand, Fish-Eye enhancements aim at a "horizontal" scaling, for networks that feature large diameters (in number of hops).

8.1 Hierarchical Routing with Trees

A classical way for an organization to scale is to introduce hierarchy. In this section we therefore describe a new dynamic clustering mechanism for the ad hoc routing protocol OLSR. A new hierarchical routing scheme using this clustering is presented, enabling ad hoc routing to scale for larger topologies. This approach was first presented in [12].

8.1.1 Introduction to Hierarchical Networking

While the main ad hoc routing solutions generally provide only flat routing, the Internet has always been hierarchical in nature. Hierarchy was introduced as a tool to cope with scalability problems, concerning both routing and managing administratively. This approach is orthogonal to the Fish Eye approach described in Section 8.2. Indeed, having several levels of hierarchy limits the growth of the routing information needed in the biggest routers in the Internet. Hierarchy enables this growth to be only *logarithmic* with respect to the size of the network, instead of *linear*. And on the other hand, when an organization grows in size, hierarchy and clustering have obvious advantages in terms of management in general. Issues due to scalability have not been entirely resolved with the main solutions that were proposed (see [26] [31] [32] [33] [11] [27]). However, mobile ad hoc routing is in dire need to address these issues, as it suffers from what is also its advantage: native mobility disturbing the Internet architecture, and decentralized wireless access incurring a lack of bandwidth limiting its flat growth.

The following therefore presents a mechanism providing dynamic clustering and hierarchical routing in an OLSR network. The provided clustering can be used for different purposes: (i) to enable hierarchical routing, or (ii) to create relatively natural regions for some administrative purposes such as address (auto)configuration, security, or any other purpose needing a dynamic partitionning of the network.

8.1.2 OLSR Tree Formation and Maintenance

The base is to pragmatically and yet optimally identify the root of trees, in other words the heads of the clusters. This must be done in a dynamic fashion, as well as the tree formation that is induced by these choices. This is done as described in Section 7.1.

Taking advantage of local maximum connectivity, *i.e.* nodes that feature the most neighbors are designated cluster heads. This mechanism initially forms trees in the following way: each node selects as parent its *preferred neighbor*. A node's preferred neighbor is the neighbor which has the maximum *degree* (number of neighbors). A node which is a local maximum degree-wise (all its neighbours have lower degree) is then the *root* of its tree. Ties are broken with the classical highest ID criteria.

The network is then viewed as a *forest*, *i.e.* a collection of logical trees, as described in Section 7.1, where this mechanism is used for flooding following the branches of the trees. In the following, we on the other hand use the clustering produced by the trees (shown in Fig. 8.1).

In order to enable OLSR nodes to form and maintain trees, OLSR nodes periodically exchange so-called Branch messages (in addition to usual OLSR messages). Typically a Branch message will be piggy-backed with a Hello message and have the same 1 hop scope. This approach is most scalable, since light, local and non-centralized. With a Branch message a node specifies information such as its identity (the *Node ID* field), the tree it belongs to (the *Tree ID* field) and its parent in the tree (the *Parent ID* field). The format of these messages is shown in Fig. 8.2. The *Depth* field indicates the distance of the node to the root. The format also reserves room for eventual extensions with the *Reserved* field, unused and zeroed out for now. Note that the IDs of the nodes are generally the IP addresses of the nodes.

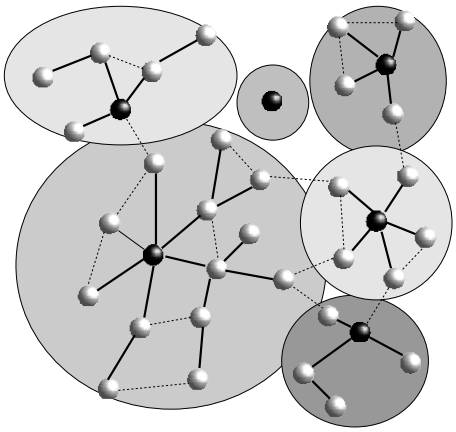


Figure 8.1: Tree clustering. Roots are shown as black nodes, and branches of the trees are shown as plain links between nodes. Links that are not branches are dashed. One tree is reduced to its root, as it is disconnected from any other node.

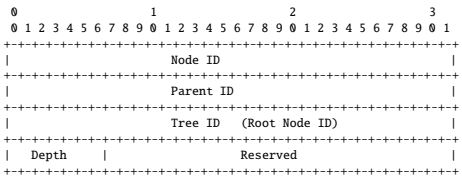


Figure 8.2: OLSR Branch message format.

8.1.3 Tree Options

Several options may be provisionned in order to tune the tree mechanisms. They are discussed in the following.

Tree Depth Control

Roots can choose to limit the size of their tree by imposing a maximum supported depth. The idea is that a root may have to perform some extra work, as being responsible for the communication outside the tree for example. The amount of work grows with the number of nodes in the tree. A root can therefore choose to limit the extra work by imposing some limitation as to how to join its tree, based on its ressources.

This is done by the root setting the maximum depth it supports in the *Max Depth* field in the Branch messages it emits (see Fig. 8.2). Nodes in the tree can then be aware of this limitation and enforce it. These in turn advertise this maximum depth in their Branch message and also precise which depth they are at with the *Depth* field (the root is at depth 0). A node wanting to join the tree can then check what is the depth limitation for this tree, and therefore if it can join the tree or not. If it cannot join the tree due to depth limitation, the

node will consider joining the “next best” available tree, via the neighbour which features the “next best” (*degree*, *ID*) criteria (and which is not over depth limit in its tree). If none of the available trees are joinable, the node will then consider itself as root, in its own tree.

Note that the tree depth control option can be disabled. If the root sets the *Max Depth* field to a special value (all the bits set to 1), there is no depth limitation for its tree.

Tree Mode Threshold

Ideally, the tree mode should appear only when the topology requires it, i.e. when the MANET grows big enough. There should be a threshold above which the trees start to develop and a way to transition smoothly into the tree mode, *i.e.* a state where all the nodes in the MANET are tree-aware, sending and receiving Branch messages. This way, an application using clustering can then start being ensured that the tree structures are in place in the entire network – this may be very important to have, depending on the application. The reverse should also be made possible: below this threshold, trees should start to disappear and there should be a way to smoothly transition out of the tree mode.

This threshold can be of various nature: the size of the link state database, the frequency of TC receival or any complex equation determining if it would be beneficial to transition into or out of this hierarchical mode.

Transition Into Tree Mode – When a node decides that the threshold is reached, it checks if it is in a position to be root of its tree. If it is, it starts sending Branch messages as such. A node that receives a Branch message checks

if its threshold is indeed reached and if it is, it may decide to join the tree it belongs to according to the afore-mentioned rules, and start sending Branch messages too. This way, trees grow, starting from the root. Note that a root emiting Branch messages also marks the TCs it emits with setting the R bit (see Fig. 8.3), this signals to other nodes in the network and outside the tree, that the node is a root. During the transition into tree mode, some nodes may be already in tree mode while some other nodes are not. In order to signal the transition status, nodes that are in tree mode mark their TC messages as coming from the forest. This is done with root nodes setting the R bit in their TCs and other nodes setting the T bit in their TCs (see Fig. 8.3).

Once there are no more unmarked TCs being flooded in the MANET, the MANET is ready to shift to tree

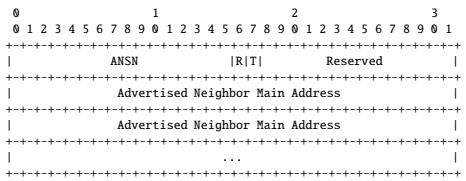


Figure 8.3: OLSR TC packet format with tree options R and T.

mode: all the nodes have shifted to tree mode and the tree structures are in place. Therefore the transition can happen, as smooth as possible.

If after some amount of time there are still unmarked TCs being flooded in the MANET, this either means that (i) the network is not too big after all, but rather stable at the limit of being so, or (ii) some nodes are tree-mode incapable and therefore tree mode is impossible in this MANET. In that case nodes may decide to abandon the transition into tree mode and stop sending branch messages (and marking TCs).

Transition Out of Tree Mode – When a root determines that the threshold is reached, it may decide to transition back into regular mode. In that case, it will start marking its TCs with both T and R bits set. Setting both R and T bits indicate that this tree wants to revert back to not using the tree structure any more. When another root receives a TC both marked with R and T bits set, it may check if its threshold is reached or not and may also start to mark its own TCs with both R and T bits set.

If a state is reached where all the TCs marked with the R bit set also have the T bit set, the MANET is ready to transition back, out of tree mode, as smoothly as possible.

If after some amount of time there are still some TCs being diffused in the MANET with the R bit set but without the T bit set, this means that the network is not ready to revert. In that case roots may decide to abandon the transition out of tree mode and stop marking their TCs with T bit set.

8.1.4 Hierarchical Routing with OLSR Trees

One application of the tree structuring described above can be the introduction of hierarchical routing in OLSR, using the dynamic clustering defined by the trees. The following sections briefly describe a way to achieve that when the tree structures are in place. Note that, as mentionned in the introduction, there may be other applications that may benefit from using this clustering, and even, other ways to use OLSR trees for

hierarchical routing.

Routing within Tree Scope

Within a tree, OLSR operates as if there was no tree, except for the following points:

1. Messages coming from a neighbor that is not in the same tree are generally not considered and not forwarded.
2. The root of a tree has the special additional role of being responsible for the communication of the tree with the rest of the MANET.
3. A node in contact with another tree must inform its own tree and especially its root.

In the following, we will describe how the restriction to tree scope is done, and how the root performs its special role. Note that routing within a tree is identical to routing with regular OLSR, and that the only difference stands in routing outside the tree.

Flooding within Tree Scope – MPR selection is unaltered by the use of trees: MPRs are selected as if there were no trees. The MPR mechanism is local and therefore very scalable. What is less scalable is the diffusion by all the nodes in the network (no hierarchy) of all the link state information (i.e. TC messages).

Addressing this, the tree mode enables the flooding of TC messages by any node in a tree to be restricted to that tree. In other words: TC messages originated and flooded inside a tree remain inside this tree *i.e.* they are not forwarded and not considered outside the tree. This is done via usual MPR flooding, with an additional rule: a node will not forward a message coming from a neighbor from another tree, except if

1. It is selected as MPR by this neighbor, AND
2. It is the first time it receives this message, AND
3. It has another neighbor that is in the same tree as this neighbor.

This rule ensures the MPR flooding will be complete inside the tree. In order to make sure that the MPR flooding completeness is not broken since MPR selection does not take into account tree segregation, border nodes just oustside the tree may relay messages between two different neighbors from the same tree (different from the border node's tree).

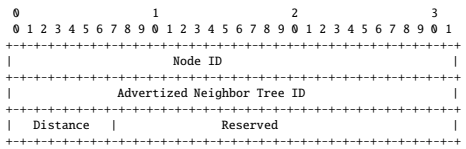


Figure 8.4: OLSR Leaf packet format.

Leaf Nodes – A node in contact with another tree (a node that has one or more neighbors that are not in the tree) must inform its tree and especially its root node. For each other tree this node reaches to, it can inform its tree with a so-called Leaf message specifying the roots of the other trees and its estimation of the distance between the roots (i.e. the sum of its depth in its tree and the depth of its neighbor in its own tree). The node will periodically flood this Leaf message throughout the tree, unless it has already received another Leaf message advertising the same tree with a shorter distance estimation (and this information is still fresh enough). This way, the root and the other nodes in the tree are informed of the paths leading to any neighbor tree, and these are the shortest available paths through the trees, from root to root.

Leaf messages are typically piggybacked with TC messages inside a tree and share the same scope, *i.e.* tree-scope. Their format is shown in Fig. 8.4. They include information such as the identity of the advertising node (the *Node ID* field), the identity of the advertised tree (the *Advertised Neighbor Tree ID* field), or the estimated distance between the root of the tree and the root of the advertised tree (the *Distance* field).

Communication with Other Trees

OLSR routing and MPR flooding being restricted to a tree, something special must be done in order to distribute routing information MANET-wide, from tree to tree. This is the additional task of the root of a tree. In order to address this task, the root basically operates OLSR at a higher level: over the super-topology formed by the roots of trees throughout the MANET. At this level, each tree, embodied by its root, behaves as if it were a single OLSR node: a super node. Similarly to regular OLSR, these super nodes (i.e. the roots) periodically send Super-Hellos, and Super-TCs. These super-messages are the only messages to be forwarded outside a tree. This is described in the following.

Super Messages – Super messages are identical to regular messages except that they feature an additional IP address in their header that indicates the next super-hop (the next root to reach). The essential difference with regular OLSR messages stands in the fact that super-messages are routed and use OLSR-established

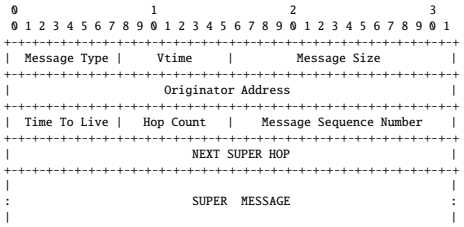


Figure 8.5: OLSR Super packet format.

paths inside each tree, instead of being simply flooded. With hierarchical routing in place, these messages are the only messages that are forwarded outside tree scope, therefore featuring MANET scope. The format is shown in Fig. 8.5. All the fields are as specified in [26], except that the *Message Type* field is set to a special value indicating a super message, and the fact that the header of the message (actually the beginning of the super-message) is completed with an additional IP address specifying the next super-hop.

Super Hello Messages – The root periodically sends a Super-Hello message to all the other roots it knows of via Leaf messages. Super-Hellos are unicasted and use the shortest root-to-toot paths advertised by the Leaf messages and OLSR routing/forwarding inside each tree. This way, as in OLSR, roots are informed of their super-neighborhood and can perform super-MPR selection. Super Hellos only have one super-hop scope (they are not forwarded further than the neighbor roots).

Super-Hellos are similar in functionality and format to regular Hellos messages, except that they also feature the next super-hop in their header (as mentionned above). Nodes use this IP address to route the message from root to root.

Super TC Messages – In addition to Super-Hellos, the root periodically sends a Super-TC message that is super-flooded (concurrent unicasts using Super-MPR and the shortest root-to-root OLSR paths) to all the roots in the network. Note that Super-TC messages therefore have a scope that is bigger than one super-hop since they are forwarded way beyond neighbor roots: throughout the whole MANET. This way, roots are informed of the whole super-topology formed by the roots.

Super-TC messages are similar in functionality and format to regular TC messages, except that they also feature the next super-hop in their header (as mentionned above). Subsequent roots update this field in order to achieve super-MPR flooding over the super-topology. The format is specified in the last section.

Super HNA Messages – Super-HNA messages are also periodically super-flooded by each root to all the other roots in the MANET. With the generation of a Super-HNA message, a root summarizes the link state information its cluster encompasses. This way, roots are aware of the link state information of the other trees.

Super-HNA messages are similar in functionality and format to regular HNA messages, except that they also feature the next super-hop in their header (as mentionned above). They are generally piggy-backed with the generated Super-TCs. Note that it can actually be envisionned to collapse Super-TCs and Super-HNAs in only one message type that would accomplish both functionalities. It was not presented here for purposes of simplicity in explaining OLSR over the super-topology.

Routing Beyond Tree Scope – Being in possession of MANET-wide information with Super-HNA and Super-TC messages, a root node will then be able to route beyond tree scope. It will therefore advertise the default route inside its tree and traffic with outside the tree will transit via the root.

8.2 Fish Eye Enhancement

In this section, we evaluate and compare the scalability of classical and optimized link state routing, with the help of a simple interference model. We show that the nature of the routing algorithm in use impacts on the maximum manageable neighbourhood size, via the control traffic it induces. Having a greater neighbourhood size actually reduces the overhead network-wide, by reducing the number of needed retransmissions on paths through the network.

We also show how two typical link state routing protocols (OLSR and OSPF) fail to scale to large ad hoc topologies in their present form. Indeed, there is a limit to the number of nodes in the ad hoc network above which there is no significant connected component, due to incompressible topology update control traffic. However, both protocols feature practical scalability issues even well within this theoretical limit, and OSPF performs quite poorly compared to OLSR (which is not a real surprise, since OSPF was not designed for ad hoc environments, contrary to OLSR).

Therefore, we describe and evaluate a solution coupling optimized link state and Fish Eye techniques that enables ad hoc routing to scale for large networks. This analysis was first presented in [9].

8.2.1 Introduction to Link State Routing coupled with Fish Eye

In an ad hoc network, part of the traffic generated by each node is the control traffic due to the routing protocol in use. If this protocol is based on link state, control traffic consists in topology information generated and relayed by each node in the network, and the amount of this information tends to increase linearly with the size of the network.

However, each node has a limited available wireless bandwidth: for example Wi-Fi usually offers less than 10 Mbps. This bandwidth is thus not only shared with its direct neighbours, but also shared with all the other nodes in the network (since each node is supposed to receive link state information about all the other nodes). This fact obviously puts an upper bound on the maximal size of the network, above which the amount of control traffic generated by topology updates purely and simply prevents the network from being formed and connected. Moreover, well below this theoretical limit, the growing amount of interferences and (re)transmissions between neighbours tends to gradually shrink the “useful” neighbourhood radius in such a way that links between nodes become unusable.

In fact, this scalability issue is common to every "flat" routing protocol, where all nodes have the same role and are put on the same level of information importance, *i.e.* every node is supposed to know the same amount of information about its direct neighbours as about any node in the network, however remote it may be.

One way to work around this problem is to establish a hierarchical protocol that takes advantage of the scaling properties of node clustering (Section 8.1 described such an approach). This technique greatly reduces the transit of topology information between clusters. However, complex algorithms and signalling are required in order to adequately distinguish and form different clusters.

An different solution was proposed by Gerla *et al.* in [52], who introduced the concept of Fish Eye Routing. Contrary to the hierarchical approach, the Fish Eye technique is does not require any additional complexity or signalling. Essentially, it consists in reporting remote nodes information less frequently than nearby nodes information: the further away a node is, the less frequently information about it will be reported. The idea is that, in order to route data to a remote destination, what a node really needs is just a "general direction" in which the data is to be sent, while totally accurate routing information is superfluous at that point. And as the data approaches

the destination, the available routing information becomes increasingly more accurate, finally enabling it to be delivered correctly.

The interesting characteristic of the Fish Eye technique is that network-wide, the weight of the control traffic generated by a node decreases as a function of the distance from this node. Thus, if the employed Fish Eye technique uses an appropriate function of the distance from the node to decrease the frequency of topology updates, we can get the control traffic (generated or relayed by each node) to converge to a finite upper bound, even when the network size grows infinitely.

The following analysis will therefore study the coupling of link state routing and Fish-Eye techniques, and is organized as such: the next section introduces some basic elements in order to model ad hoc networks: slotted time, propagation model, fading model, uniform density of transmitters dispatched on a domain *etc.* We will apply this model to study link state routing in the context of ad hoc networking, with a particular focus on OSPF and OLSR.

We will study the properties of both protocols, and show how they do not scale to large networks, and how

OSPF performs badly compared to OLSR. We will then study the properties of these two protocols enhanced with a Fish Eye technique, and we will show that on the other hand, these enhanced protocols can scale to infinitely large networks (enhanced OLSR still showing a better performance than enhanced OSPF). We will compare these new properties with previous general results on ad hoc network capacity.

Note that introducing Fish Eye features in OLSR is immediate, by playing on TTL and V-Time parameters in topology update packets (TC packets), as described in [53]. Introducing Fish Eye features in the OSPF framework is a little less straightforward since LSAs do not feature TTL inside their format. Nevertheless, playing on Age fields should essentially do the same job.

8.2.2 Modeling Ad Hoc Networks

In this section we will describe how we model the different aspects of ad hoc networks.

Propagation Model

We consider the following model: a domain of arbitrary area $\mathcal{A}$ (we will ignore border effects), where N nodes are uniformly distributed with a density ν. We consider time to be slotted and the nodes to be synchronized so that transmissions occur at the beginning of slots and according to an ALOHA-like protocol (*i.e* nodes select at random their transmission slots). Let λ be the density of transmitters per slot and per area unit (*i.e.* the traffic density).

With this model, we thus consider the number of transmitters per slot as varying with time and changing from slot to slot randomly, following a Bernoulli distribution "filtering" a Poisson distribution. Therefore we consider the number of transmitters at beginning of slots as following a Poisson distribution (in pratice, this model turns out to be very accurate).

Let X be a node at a random position. We will again ignore border effects and assume that all nodes transmit at the same power. The reception signal at distance r is then $P(r) = r^{-\alpha}$ with $\alpha > 2$. Typically $\alpha = 2.5$. Notice that the expression of quantity $P(r)$ does not involve any fading factor. Fading is an alteration of the signal which is due to factors other than the distance (obstacles, co-interferences with echos, and so on). Fading is generally modeled via the introduction of a non-zero factor that varies randomly with time and node location.

We will address the fading issue more thoroughly in the following.

Let W be the signal intensity received by node X at a random slot. The quantity W is then a random variable since the number and location of transmitters are random and vary with the slot. Let $w(x)$ be its density function. If we consider $\mathcal{A}$ to be infinite, we can use [55] where it is shown that the Laplace transformation of $w(x)$, $\tilde{w}(\theta) = \int w(x)e^{-x\theta}dx$ satisfies the identity (still with no fading):

$$\tilde{w}(\theta) = \exp(2\pi\lambda \int_0^\infty (e^{-\theta r^{-\alpha}} - 1)rdr) \, . \tag{8.1}$$

Then, using standard algebra we get:

$$\tilde{w}(\theta) = \exp(-\lambda\pi\Gamma(1 - \frac{2}{\alpha})\theta^{2/\alpha}) \, . \tag{8.2}$$

Note that if instead of an area, the node location map was a line (for instance a sequence of mobiles nodes on a road) we would then have:

$$\tilde{w}(\theta) = \exp(-\lambda\Gamma(1 - \frac{1}{\alpha})\theta^{1/\alpha}) \, . \tag{8.3}$$

And similarly, if the location map was a volume (for instance a network formed by aircrafts), we would instead have:

$$\tilde{w}(\theta) = \exp(-\frac{4}{3}\lambda\pi\Gamma(1 - \frac{3}{\alpha})\theta^{3/\alpha}) \, . \tag{8.4}$$

In the following, we will restrict ourselves to the case where nodes are located on a domain with only two dimensions.

Neighbour Model

A node is considered to be a *neighbour* of another node if the probability of successfully receiving hellos from each other is greater than a certain threshold p_0. The *neighbourhood* of a node is then defined as the set of all its neighbours. For example we can take $p_0 = 1/3$. This can be achieved by keeping track of the hello receival success rate per neighbour, as it is done in the "advanced neighbour sensing" of OLSR [26].

We will assume that a packet can be successfully decoded if its signal-over-noise ratio is greater than a given threshold K. Typically $K = 10$. Therefore a node will correctly receive a packet from another node at distance r with probability $P(W < r^{-\alpha}/K)$. Since hello packets are never retransmitted, the hello success rate from a node at distance r is exactly $P(W < r^{-\alpha}/K)$. Therefore nodes at distance r are neighbours as

long as $P(W < r^{-\alpha}/K) > p_0$. This is equivalent to $r < r(\lambda)$, where $r(\lambda)$ is the critical radius such that $\int_0^{r(\lambda)^{-\alpha}/K} w(x)dx = p_0$. In fact quantity λ is a parameter which is easy to handle since by simple algebra it comes that $r(\lambda) = \lambda^{-1/2}r(1)$ (see appendix C). The surface covered by the radius $r(\lambda)$ is then the neighbourhood area $\sigma(\lambda) = \frac{\sigma(1)}{\lambda}$.

We will now compute $\sigma(1)$. We remind that factor λ is now omitted ($\lambda = 1$). For simplification purposes, we set $C = \pi\Gamma(1 - \frac{2}{\alpha})$ and $\gamma = \frac{2}{\alpha}$. By application of the reverse Laplace transformation we get:

$$P(W < x) = \frac{1}{2i\pi} \int_{-i\infty}^{+i\infty} \frac{\tilde{w}(\theta)}{\theta} e^{\theta x} d\theta. \tag{8.5}$$

Expanding $\tilde{w}(\theta) = \sum_n \frac{(-C)^n}{n!}\theta^{n\gamma}$, it comes:

$$P(W < x) = \frac{1}{2i\pi} \sum_n \frac{(-C)^n}{n!} \int_{-i\infty}^{+i\infty} \theta^{n\gamma-1} e^{\theta x} d\theta. \tag{8.6}$$

Then by bending the integration path towards the negative axis we get:

$$\begin{aligned} \frac{1}{2i\pi} \int_{-i\infty}^{+i\infty} \theta^{n\gamma-1} e^{\theta x} d\theta &= \frac{\sin(\pi n\gamma)}{\pi} \int_0^{\infty} \theta^{n\gamma-1} e^{-\theta x} d\theta \\ &= \frac{\sin(\pi n\gamma)}{\pi} \Gamma(n\gamma) x^{-n\gamma}. \end{aligned}$$

Figure 8.6 shows the plot of $P(W < x)$ versus x for $\alpha = 2.5$

and $\lambda = 1$. Let x_0 denote the value such that $P(W < x_0) = p_0$, therefore $r(1) = (x_0 K)^{-1/\alpha}$.

Notice that if $p_0 = \frac{1}{3}$, then $x_0 \approx 20$, $r(1) = (x_0 K)^{-1/\alpha} \approx 0.12$. And then that $\sigma(1) = \pi r(1)^2 \approx 0.045$.

Optimizing the Neighbourhood

In this section we estimate the value of p_0 that optimizes the neighborhood in order to minimize the number of overall retransmissions of packets in the network (by retransmission we mean the retransmission due to multihoping as well as the retransmissions due to packet collisions). We assume that each slot is used by unicast packets (re)transmitted *à la* ALOHA until they are correctly received by the next node.

We can indeed optimize the neighbourhood by excluding from it “bad” nodes that feature a too low probabil-

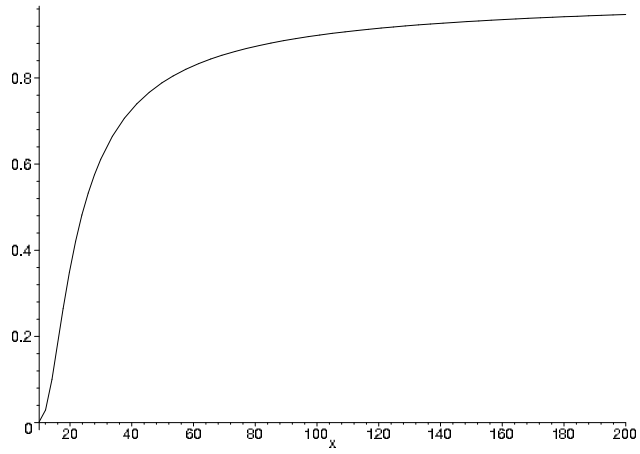


Figure 8.6: Quantity $P(W < x)$ versus x for $\alpha = 2.5$, no fading.

ity of successful one hop packet transmission. They might be too far or behind an obstacle: in any case the link is not reliable enough and the number of retransmissions needed for a correct reception is not worth the hop distance. In other words, we want the best possible *ratio* of hop distance over number of retransmissions.

To this end we can tune the parameter p_0. The optimal value does not depend on λ as we see below. If the probability of successful transmission is p_0 then the average number of retransmission for one hop is $\frac{1}{p_0}$. And thus we have to optimize the quantity $p_0 r(\lambda)$, or in other words (since $r(\lambda) = \lambda^{-1/2} r(1)$) we simply have to optimize $r(1)P(W < r(1)^{-\alpha}/K)$. All computations done (see Fig. 8.7, with $\alpha = 2.5$) we get the optimum $r(1) \approx 0.089$ and we see that the optimum $p_0 \approx 0.75$. So on average, if a node logically excludes from its neighbourhood any node from which it successfully receives less than 75% of the hellos actually sent by this node, we ensure a simple optimization of the overall number of retransmissions on a network-wide level.

Fading Model

The propagation of radio waves in presence of random obstacles experiences random fading. Usually, modelization of fading consists in the introduction of a random factor F modeling signal attenuation at distance r: $r^{-\alpha}$. For example $\log F$ is uniform on $[-v, v]$. In this case we have a new expression of $\tilde{w}(\theta)$:

$$\tilde{w}(\theta) = \exp(-\pi\lambda\Gamma(1 - \frac{2}{\alpha})\phi(-\frac{2}{\alpha})\theta^{2/\alpha}). \tag{8.7}$$

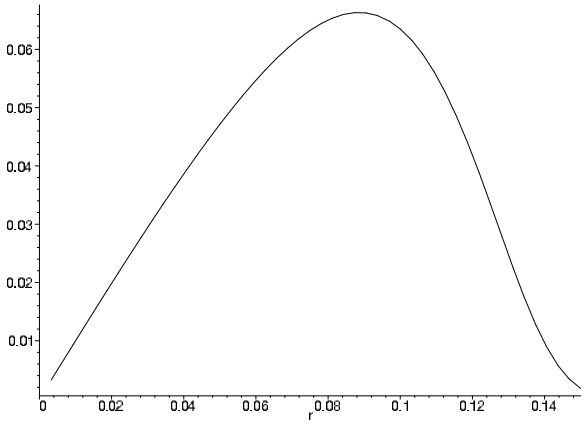


Figure 8.7: Quantity $p_0 r$ versus r for $\alpha = 2.5$, no fading.

with $\phi(s) = E(F^{-s})$, the Dirichlet transformation of the fading. When fading is uniform on $[-v, v]$ we have $\phi(s) = \frac{\sinh(sv)}{sv}$. For any given real number x we also have $P(W < xF)$ equaling

$$\sum_n \frac{(-CF(-\gamma))^n \sin(\pi n\gamma)}{\pi n!} \Gamma(n\gamma)\phi(n\gamma)x^{-n\gamma}, \tag{8.8}$$

which helps the computation of $\sigma(1)$ with fading.

8.2.3 OSPF and OLSR Scalability

The model described in the previous section states that when the nodes are distributed over an infinite plane, the average traffic generated inside the neighbourhood radius is equal to $\lambda\sigma(\lambda) = \sigma(1)$, a constant that we determined, and thus, the average number of neighbours per node can be expressed as $M = \sigma(\lambda)\nu = \sigma(1)\frac{\nu}{\lambda}$.

The neighbourhood size therefore depends on the traffic control generated by each node: the bigger is the amount of control traffic, the smaller is the neighbourhood size. Thus, performance may vary with the use of different protocols, yielding different control traffic patterns. In this section we study more precisely the properties of OSPF on one hand, and OLSR on the other hand.

General Control Traffic Model

The aim here is to derive the traffic density generated by the protocol control packets. Generally, there are two sources of control traffic: neighbour sensing on one hand, and topology discovery on the other hand.

Neighbour sensing is the same for all link state protocols: it consists in each node periodically transmit-

ting a hello message containing the list of neighbours heard by the node. By comparing their lists the nodes can determine the set of neighbours with which they have symmetric links. Let h be the rate at which nodes refresh their neighbour information base and let B be the maximum number of node identifiers that a slot can contain. For a network with the capacity of Wifi (1-10 Mbps) we have $B = 100$ and 1,000 slots per second. For instance, an OLSR node generates hellos every 2 sec, *i.e.* $h = 1/2000$. If the neighbour list exceeds B then the node generates several hellos per update period and distributes the neighbour list among these several hellos. The node must generate $\lceil \frac{M}{B} \rceil$ hellos per hello period. Therefore the hellos lead to a traffic density of $h\nu\lceil \frac{M}{B} \rceil$. Omitting fractional part, we get:

$$\lambda = h\nu \frac{M}{B} . \tag{8.9}$$

if the hellos is the only source of control traffic. Since $M = \sigma(1)\frac{\nu}{\lambda}$ we get:

$$\frac{\sigma(1)}{M} = h\frac{M}{B} . \tag{8.10}$$

In fact this is only an upper bound because the network size might be smaller than $\sigma(1)$. Therefore, taking into account only the hello control traffic, the maximum manageable neighbourhood size is $M_{max} = \sqrt{B\sigma(1)/h} \approx 71$. This applies to both OLSR and OSPF as well as to any other protocol that uses such Hellos.

Topology discovery varies with each protocol. With OSPF, each node periodically broadcasts its list of adjacent links in an LSA (Link State Advertisement) message, and nodes re-broadcast in turn the LSA towards their neighbours. In OLSR, on the other hand, the nodes periodically broadcast TC (Topology Control) messages containing only a subset of their adjacent links - the MPR (MultiPoint Relay) selector links. Moreover, only a subset of the neighbours (the MPR nodes) re-broacast the TC messages. However we will assume that in both protocols the topology discovery update period is the same, in order to compare two protocols with the same agility to adapt their topology to mobility. For instance, OLSR's TC rate per node is $\tau = 1/5000$.

OSPF Specific Model

In this section we will work on modeling the overhead induced by OSPF. The idea is to express λ only in function of the protocol overhead. We consider no other traffic than the signalling protocol. In OSPF a node periodically:

1. transmits Hellos with rate h. A Hello contains the list of all neighbour identifiers (if the list is too long,

it will take several packets on several slots),

2. transmits LSAs with rate τ. An LSA contains the list of all adjacent links,

3. retransmits received LSAs with a large jitter, to all neighbours separately (one copy per neighbour).

Therefore the traffic density satisfies the following identity:

$$\lambda = h\nu\lceil\frac{M}{B}\rceil + \tau\nu NM\lceil\frac{M}{B}\rceil \,. \tag{8.11}$$

In the following, we drop the ceil factor.

$$\lambda = h\nu\frac{M}{B} + \tau\nu N\frac{M^2}{B} \,. \tag{8.12}$$

Using $M = \sigma(1)\frac{\nu}{\lambda}$ we have the identity:

$$\frac{\sigma(1)B}{M} = hM + \tau NM^2 \,. \tag{8.13}$$

This outlines a direct relation between the total size of the network N, and the average neighbourhood size M. Notice that when N increases, M decreases. This corresponds to the fact that as more and more nodes are concentrated in a single radio range, interferences and collisions make more and more links perform too badly to be considered valid. Therefore more and more nodes that are theoretically directly reachable (because physically within radio range) are not considered neighbours, and hence, M decreases. The minimum for M is 1, below which the network does not have a significant connected component (see [51]).

Furthermore, the limit $M = 1$ yields a maximum network size of:

$$N_{\max} = (\sigma(1)B - h)\frac{1}{\tau} \,. \tag{8.14}$$

Which gives $N_{\max} = 25,000$.

On the other hand, when the network size decreases, it reaches a level where $N = M$. Below this level the network is only one hop (full meshed), and the control traffic does not saturate the neighbourhood. This corresponds to the maximum manageable neighbourhood size. From (8.13) we get that the maximum man-

ageable neighbourhood size for OSPF is $N = 11$. Having an average neighbourhood size as big as possible is important in that it reduces the average number of hops needed to go from a given source to a given destination. This way the amount of retransmissions network-wide (hence the overhead) is reduced.

OSPF-B

In this section we propose an adaptation of OSPF which aims at reducing the overhead. OSPF-B slightly differs from OSPF with the fact that the nodes broadcast the LSA only once, instead of duplicated in several copies to each neighbour.

In this case the equation (8.13) should be rewritten in

$$\frac{\sigma(1)B}{M} = hM + \tau NM. \tag{8.15}$$

It then comes that in the case of OSPF-B we get a maximum manageable neighbourhood size of $N = 22$.

OLSR Specific Model

In this section we will work on modeling the overhead induced by OLSR. With this protocol, a node periodically:

1. transmits TCs with rate τ. A TC contains the list of neighbours having selected the node as MPR (its MPR selectors),
2. retransmits received TCs only once (and with large jitter), and such only when the node has been selected as MPR by the neighbour from which it first received the TC.

Let M_r be the average number of MPRs selected by a node with neighbourhood size M. Since the network is modeled as a disk unit graph, it comes from [56] that $M_r \leq (9\pi^2 M)^{1/3}$. Simulations show that $M_r \sim \beta M^{1/3}$ when $M \to \infty$ with $\beta \approx 5$ (see Figure 8.8). Simulations were performed up to $M = 6,000,000$.

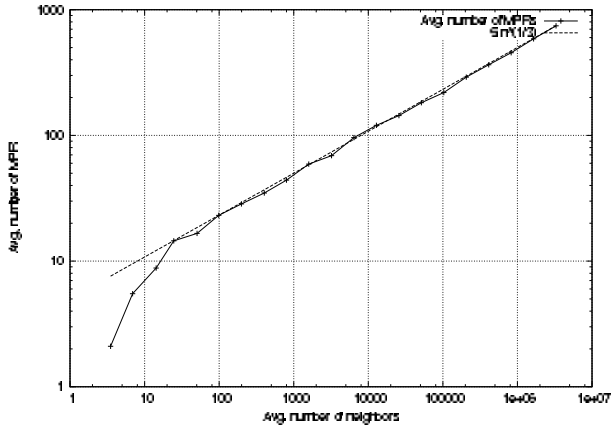


Figure 8.8: Average MPR set of a node versus neighbourhood size.

In [54] it is proven that an MPR flooding costs on average $M_r N/M$ retransmissions. Therefore we get the following traffic density identity:

$$\frac{\sigma(1)B}{M} = hM + \tau(M_r)^2 \frac{N}{M}. \tag{8.16}$$

It then comes that the maximum manageable neighbourhood size for OLSR is $M_{max} = 35$. Also note that this identity and the connectivity limit of $M = 1$ (which in turn implies that $M_r = 1$) gives the same maximum network size for OLSR as with OSPF, that is $N_{\max} = 25,000$.

F-OLSR

In this section we introduce a slight modification of OLSR called F-OLSR, for Full Optimized Link State Routing. In F-OLSR the TCs contain the list of all the adjacent links, and not just MPRs. Therefore every node has the knowledge of the complete link state of the network instead of its restriction to MPR links. The TCs are still forwarded via MPR nodes. The identity for F-OLSR is then:

$$\frac{\sigma(1)B}{M} = hM + \tau M_r N. \tag{8.17}$$

It then comes that the maximum manageable neighbourhood size for F-OLSR is at $N = 27$.

Comparisons between the Protocols

In Fig. 8.9 we show the respective neighbourhood size versus network size for the two versions of OSPF. With Fig. 8.10 we show the respective neighbourhood size versus network size for the two versions of OLSR.

And finally, Fig. 8.11 compares the network diameter as a function of the network size (number of nodes) in the case of OLSR and OSPF. The number of hops is estimated as the square root of the ratio network size over neighbourhood size.

Basically, what we can conclude from this analysis is that optimized link state (OLSR) shows here much better performance than classical link state (OSPF). However, as the network size increases, both types of approaches feature slowly decreasing (towards 0) neighbourhood size. This fails to scale to large networks: if the network size grows too big,

it will break down by not being able to create significant connectivity.

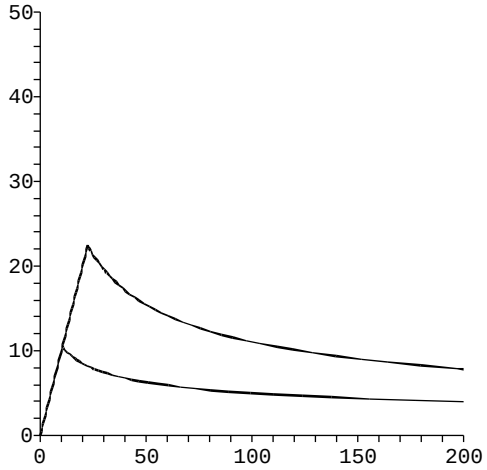


Figure 8.9: Neighbourhood size versus the network size, $\alpha = 2.5$, no fading, respectively for OSPF (bottom) and OSPF-B (top).

8.2.4 Scaling Properties of OSPF and OLSR Enhanced with Fish Eye Strategy

With OSPF and OLSR as well as with any other flat routing protocol, the neighbourhood size tends to slowly decrease towards zero as the network size increases. Therefore they do not scale to arbitrarily large networks. This is due to the fact that the topology information that each node in the network has to (re)transmit tends to increase linearly with the size of the network. This in turn yields an upper bound on the maximal size of the network, which we have computed to be of about $25,000$ nodes for OSPF as well as for OLSR.

However, when N is well below this limit of $N_{\max} = 25,000$ the two routing protocols have their neighbourhood size almost constant as N increases and thus the number of hops increases in $J\sqrt{N}$. The constant J depends on the nature of the routing protocol and can vary greatly. We analyzed the impact of the routing

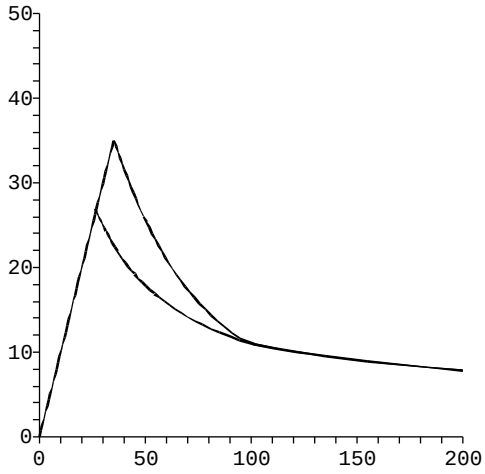


Figure 8.10: Neighbourhood size versus the network size, $\alpha = 2.5$, no fading, respectively for F-OLSR (bottom) and OLSR (top).

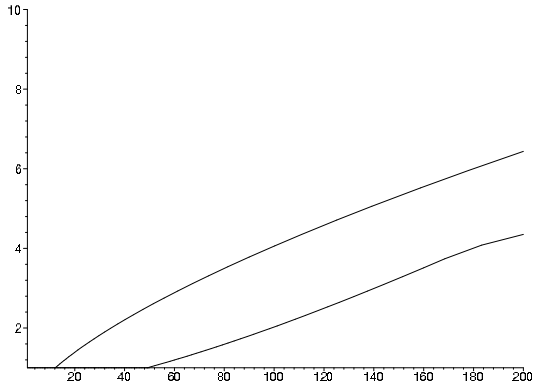


Figure 8.11: Hop number estimated diameter of the network versus network size, $\alpha = 2.5$, no fading, respectively for OLSR (bottom) and OSPF (top).

protocol on the value of this constant: we have shown that it changes quite a bit between pure link state (OSPF) and optimized link state (OLSR). In particular we have shown that as the network

size increases, the maximum manageable neighbour size is respectively of 11 nodes with OSPF, while it is of 35 with OLSR. Note that as the maximum manageable neighbour size decreases, the average number of hops (and hence retransmissions) between any random source and destination increases, therefore augmenting the overall traffic overhead.

However, both OLSR and OSPF just need minor modifications in order to scale for arbitrarily large topologies. In this section we describe the “Fish Eye” strategy [52] that can easily be inserted inside both OSPF and OLSR frameworks. With this strategy the overall incompressible overhead induced by periodical topology

updating tends to be constant instead of linearly increasing with the network size. Of course this doesn't come without a cost, *i.e.* less accurate information about the link status of remote nodes. However, this cost is not expensive: it does not degrade the delivery reliability and it does not introduce additional overhead in form of longer paths (see [53]).

The principle of Fish Eye strategy is that TC (or LSA) information from remote nodes are less frequently received, and the more remote, the less frequent. For example, inside the OLSR framework, nodes send TC packets with variable TTL count and VTime. The TTL limit is the maximum number of hops a packet can be relayed before being discarded and the VTime is the maximum time for which the information carried by this packet is considered valid. A node transmitting a packet with low TTL value ensures that the packet will be forwarded only inside the vicinity of this node, and not further. Conversely, a large TTL value (the maximum value is 255) ensures that the packet will be forwarded in the entire network.

Each node uses a decreasing function $f(D) \leq 1$ to determine the fraction of the TCs (or LSAs) which are generated with a TTL larger than D (D is an integer indicating the number of hops away that the TC may reach). When no Fish Eye strategy is employed, $f(D) = 1$ for any value of D. We can assume that $\sum_{D=1}^{\infty} Df(D) < \infty$. This is indeed always the case, since $f(D) = 0$ for all $D \geq 255$. Of course, information that is received less frequently should not age as rapidly as frequently received information. This can be achieved by adequately tuning the Age field in the LSAs (for OSPF) or the VTime field in the TC packets (for OLSR).

Let us consider a node at the center of a circular network consisting in N nodes uniformly dispatched on a disk. M is the average number of neighbours of the central node. In this case, the central node has $3M$ two hop neighbours, and $(D^2 - (D-1)^2)M$ D-hop neighbours, for $D \leq \lfloor \sqrt{N/M} \rfloor$ (it comes that $2\sqrt{N/M}$ is the diameter of the network).

Fish Eye Enhanced OSPF

Let us now consider the OSPF protocol enhanced with Fish Eye strategy. The frequency of LSAs received by the central node from D-hop neighbours is $f(D)\tau$. Therefore the frequency at which the central node relays LSAs is $\tau M \sum_{D=1}^{\lfloor \sqrt{N/M} \rfloor} (D^2 - (D-1)^2) f(D)$.

We will call

$\phi(x) = \sum_{D=1}^{\lfloor \sqrt{x} \rfloor} (D^2 - (D-1)^2) f(D)$. It then comes that the control traffic of the central node equals to

$h\frac{M}{B} + \tau\phi(\frac{N}{M})\frac{M^3}{B}$ and we get the following general identity:

$$\frac{\sigma(1)B}{M} = hM + \tau\phi(\frac{N}{M})M^3 \ . \tag{8.18}$$

When the networks grows and $N \to \infty$ with $\phi(\infty) = 4$ we get an average neighbourhood size converging towards $M \to 7.5$.

Fish Eye Enhanced OLSR

In the case of OLSR the identity 8.18 becomes:

$$\frac{\sigma(1)B}{M} = hM + \tau\phi(\frac{N}{M})(M_r)^2 \ . \tag{8.19}$$

When $N \to \infty$ with $\phi(\infty) = 4$ we get an average neighbourhood size converging towards $M \to 18$. That is: three times better than Fish Eye enhanced OSPF.

Figure 8.12 shows an example for function ϕ: $\phi(x) = \frac{4x}{3+x}$. Figure 8.13 shows the neighbour size evolution with respect to this function ϕ and compares it to basic OLSR.

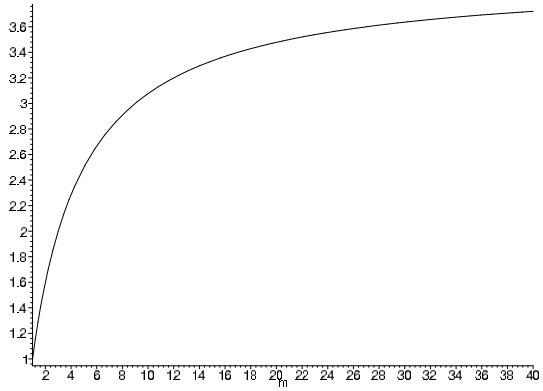


Figure 8.12: Example of function ϕ used for Fish eye strategy.

Useful Capacity

In this section we estimate the useful capacity with the OLSR protocol. We denote ρ the average quantity of data traffic generated by each node, with each node choosing a random destination in the network, and transmitting data to this destination. On average, the number of hops of a data path is $\ell\sqrt{N/M}$, where ℓ

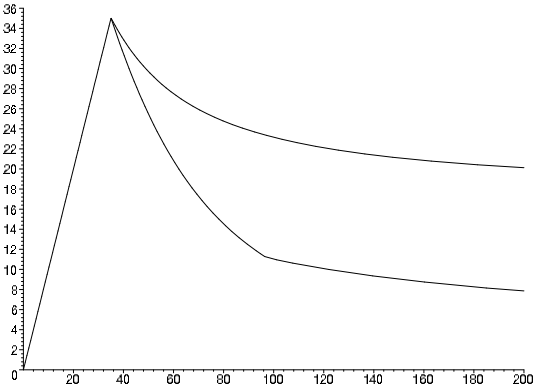


Figure 8.13: Neighbourhood size versus the network size, $\alpha = 2.5$, no fading, respectively for OLSR (bottom) and OLSR with Fish eye (top).

denotes a linear factor that depends on the actual shape of the network area $\mathcal{A}$. Therefore each packet must be retransmitted $\frac{\ell\sqrt{N/M}}{p_0}$ times, which leads to an average traffic density (including control traffic and retransmissions) of:

$$\lambda = h\frac{M}{B} + \tau(M_r)^2\frac{N}{MB} + \frac{\rho\ell}{p_0}\sqrt{N/M}.$$

Therefore, using the identity $\lambda = \frac{\sigma(1)}{M}$ we get an expression of ρ as a function of N and M. Clearly, for a given fixed N, ρ is maximized when M is minimized, the minimal value being $M = 1$. This yields figure 8.14, which displays the overall maximum capacity $N\rho$ versus network size for basic OLSR and for Fisheye OLSR (we took $\frac{\ell}{p_0} = 1$). Notice that basic OLSR with default tuning collapses at $N > 12000$, while Fisheye OLSR features an overall capacity that keeps growing in $\sqrt{N}$.

8.2.5 Comparison with Previous Results on Ad Hoc Network Capacity

In a famous paper, Gupta and Kumar [51] have shown that when the size N of the network increases (with randomly placed nodes), the optimal neighbourhood size is $O(\log N)$, which leads to the optimal network capacity per node being $O(\frac{1}{\sqrt{N\log N}})$.

Further, they have shown that if we drop the requirement for the network to be connected, and just require the

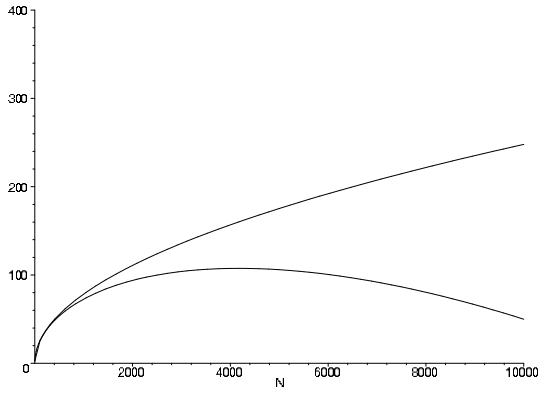


Figure 8.14: Maximum overall capacity versus the network size, $\alpha = 2.5$, no fading, respectively for OLSR (bottom) and OLSR with Fish eye (top).

existence of a giant component, we can actually drop the $\log N$ factor in these formulas. They have shown that in this case the optimal capacity per node is then $O(\frac{1}{\sqrt{N}})$ (note that Gupta and Kumar have also shown that when the nodes are optimally placed, the giant component is actually the whole network). Thus, Gupta and Kumar's results state that the optimal network capacity is $O(\sqrt{N})$.

Their result is therefore noticeably consistent with the network capacity we have found above for FishEye enhanced OLSR, for the same traffic model. However, if our propagation model is similar to their so-called physical model (taking into account interferences between nodes), it is somewhat different in that we further analyze the actual control traffic with the use of real routing protocols.

Chapter 9

Integration Solutions for Ad Hoc Networks in the Internet

In the previous chapters, the focus was put on mechanisms providing efficient routing in ad hoc networks. However, aside of pure routing issues, ad hoc mobility brings a number of other challenges. Indeed, when connected to the Internet, an ad hoc network introduces new problems yet to be addressed, as the Internet architecture and its protocols were not initially designed to cope with such node "freedom".

Important examples of such issues include network security and node autoconfiguration. This chapter will therefore overview these issues and propose some solutions.

9.1 The Security Problem in Ad Hoc Networks

Ad hoc networking principles enable any node to participate in the network. This approach is the basis of the specifications of all the main ad hoc routing solutions ([26] [32] [31] [33]). However, the assumption made with this basis is that all the nodes are well-behaved and are therefore welcome. If that assumption fails, *i.e.* if the network may count malicious nodes, the integrity of the network is not ensured.

Network integrity is indeed an important issue in a mobile ad hoc environment, contrary to traditional networks. A static or wired node that behaves abnormally can quickly be *identified*, located and "unplugged" if needed, avoiding serious network damage. However, this is not possible with a malicious MANET node, that is not identified and located so easily, and that cannot be excluded from the network in such a simple way.

Of course, ad hoc networking also needs to address the usual security issue of maintaining confidentiality and integrity of the data being communicated between endpoints in the network (for instance between a mail server and a mail client). This task – ensuring end-to-end security of data communications in MANETs – is equivalent to that of securing end-to-end security in traditional wire-line networks, an issue which is already commonly addressed.

Therefore, a specific issue concerning MANET routing protocol security is network integrity, *i.e.* ensuring that the network functions as well as possible, even in spite of some malicious nodes.

There are different approaches to securing a MANET routing protocol in general. One such approach is to ensure that only "trusted" nodes are admitted into the network and, subsequently, that these are the only nodes relied on for forwarding traffic. This approach makes the assumption that a trusted node will not misbehave.

Orthogonal to this trust discrimination is the ability to detect and deal with the situation where a node has become compromised. More precisely, this approach is about detecting that a node is misbehaving and then deciding to classify it as "non-trusted", for exclusion from the network. This, however, opens an additional vulnerability: a node can be malicious in that it "denounces" other (non-malicious) nodes and manages to get these excluded from the nework. Indeed, an important fact about security mechanisms is that they must be carefully scrutinized in order to prevent that they themselves introduce new vulnerabilities.

Each node in an ad hoc network has mainly two responsibilities: (i) to correctly generate routing control traffic, according to the protocol specification, and (ii) to correctly forward routing control traffic on behalf of other nodes in the network, as required by the protocol specification. Thus incorrect behavior of a node can result from either a node generating incorrect control messages or from incorrect relaying of control traffic from other nodes.

Correctly generating and forwarding control traffic can be considered as a criterion for having a correctly functionning routing – *i.e.* the routing protocol is able to consistently provide to each node, the required correct information about the network topology. This assumption obviously implies that all the nodes in the network correctly implement the routing protocol - specifically that each node correctly processes and emits control traffic.

In the following, a non-exhaustive list of categories of node misbehaviour will be given, as example of vulnerabilities that should be addressed in order to ensure ad hoc network integrity.

9.1.1 Incorrect Traffic Generation

In general, a node may misbehave in three different ways: (i) through generating control traffic "pretending" to be another node, *i.e.* Identity Spoofing, or (ii) through advertising incorrect information in the control messages, *i.e.* Link Spoofing, or (iii) through generating enough artificial amounts of traffic to overload the network, *i.e.* Traffic Overloading.

Identity Spoofing – For instance, with a proactive routing protocol, a misbehaving node X may send HELLO messages pretending to have the identity of another node A. This may result in the network containing conflicting routes to node A, and can create routing loops.

Link Spoofing – Another example, with a reactive protocol, a misbehaving node X may send ROUTE-REPLY messages, pretending it has a route to a requested destination, whereas in fact it has not. This may result in no route being found for some destinations, and in continuous ROUTE-REQUEST broadcasting which can lead to traffic overloading.

Traffic Overloading – A misbehaving node may simply generate artificially massive amounts of traffic, which will starve legitimate traffic (control traffic or data traffic) on parts of a network, potentially leaving the network without the ability to maintain connectivity.

9.1.2 Incorrect Traffic Relaying

Nodes in a mobile ad hoc network relay two types of traffic: routing protocol control traffic and data traffic. A node may misbehave through failing to forward either type of traffic correctly.

Incorrect Control Traffic Relaying – If the routing protocol's control messages are not properly relayed, connectivity loss may happen. For example, with a reactive protocol, if a node does not forward ROUTE-REPLY or ROUTE-REQUEST messages as it should, routes to some destinations may never be found, and in some topologies, the network may endure partition.

Replay Attack – If at least two nodes coordinate their attack on network integrity, one node can record traffic in its proximity and tunnel it to the other node, which replays the recorded traffic, creating a virtual link in the topology that may be used by some routes. If data traffic routed over this virtual link is then not also recorded and tunnelled the same way, some data may be lost. Moreover, if the replayed control traffic is modified using link or identity spoofing techniques, some important topology information may be destroyed all over the network.

Incorrect Data Traffic Relaying – Even a node correctly generating, processing and forwarding control traffic as required, may act in a malicious way through not forwarding data traffic. The node thereby breaks connectivity in the network (data traffic cannot get through). However this connectivity loss may not be detected by the routing protocol, as control traffic is correctly relayed.

9.1.3 Secure Integration of Ad Hoc Networking in the Internet

The field of ad hoc security is key in order to fully integrate ad hoc networking in the Internet. Indeed, one of the main reasons why ad hoc networking is not yet widely deployed in the Internet is a concern regarding the security breach ad hoc nodes may bring. Substantial efforts are therefore needed in this field, which reveals to be quite vast and complex.

This document does not introduce any solution to the security issues that were described in the previous section. The above was however included despite this fact, for the sake of completeness in overviewing the main ad hoc networking fields throughout this document.

Pursuing this goal, the next section introduces another important ad hoc networking field, autoconfiguration, and proposes new solutions in order to solve issues related to this field.

9.2 Address Autoconfiguration in Ad Hoc Networks

In a mobile ad hoc network, a routing protocol is employed in order to enable communication between nodes. The task of such a protocol is to discover the network topology and track its changes over time, as links and nodes appear and disappear. A routing protocol enables each node to acquire a recent image of enough parts of the network topology so that it can construct routes to reach desired destinations.

In order to function correctly, however, routing protocols require that each node in the network is *uniquely identified*. In the Internet, as we have already seen, this unique name is an *IP address*. Therefore, there is a need for a mechanism dynamically assigning unique addresses to nodes in a MANET.

Contrary to traditional networks, the roles of hosts (*i.e.* nodes that use the network) and routers (*i.e.* nodes that form and maintain the network) are not clearly separate in a mobile ad hoc environment. Indeed, each node may act in both capacities simultaneously. Another particularity of MANETs is that no assumption can be made regarding preexisting infrastructures. However, classical autoconfiguration mechanisms, such as DHCP [44], ZeroConf [45] or similar mechanisms, assume the presence of a *server*, which can coordinate and assign addresses to hosts in the local network. Other traditional mechanisms, such as IPv6 Stateless Autoconfiguration [49], assume that direct communication is available between each interface in the local network. However, the multi-hop nature of MANETs does not allow the assumption that direct communication between an arbitrary pair of nodes is always possible.

Therefore, classical autoconfiguration techniques do not work properly in a MANET environment, as the problem of autoconfiguration in a mobile ad hoc network is more complex than on traditional networks. Indeed, a solution cannot be based on a central server, while at the same time, it should address issues such as (i) ensure uniqueness of addresses in independent MANETs which later merge, (ii) select non-overlapping address-spaces, (iii) perform duplicate address detection and conflict resolution, as well as (iv) cope with issues related to dealing with specific application requirements – such as operating over ongoing data streams without incurring data loss. A solution should also not yield too much additional control traffic or processing, as ad hoc bandwidth is scarce, and MANET nodes' power is in general limited.

Solving the autoconfiguration problem in ad hoc networks as a whole with a simple, light-weight mechanism is a hard task. However, solutions to specific use-cases can be more easily developed. In this chapter, we will describe autoconfiguration mechanisms applicable when OLSR is the routing protocol in use in a

MANET that extends the edge of the Internet. In other words, ad hoc nodes aim at maintaining connectivity between each other as well as to the Internet.

The approach described in the following is *active*: it relies on some additional signalling. Some other approaches, such as the one proposed in [48] are on the other hand passive: they do not require any additional signalling, a property that is desireable in a bandwidth-challenged environment such as a mobile ad hoc network. However, in the special case considered here, where MANETs are connected to the Internet, an active approach may be more interesting as it can use enhancements of traditional mechanisms such as DHCP when available (as described in the next section), whereas a purely passive approach cannot.

9.2.1 A Light-weight Autoconfiguration Mechanism for OLSR

In the following we introduce a new ad hoc autoconfiguration mechanism, enabling a mobile node to acquire a unique and routable IP address when the ad hoc network it is part of changes Internet access point. The mechanism is described as an extention to the ad hoc routing protocol OLSR, and was first presented in [8]. In the following, the terminology shown below will be used:

- A *new node* is a node which is not yet assigned an address, and that is therefore not part of the network.
- An *OLSR node* is a node which was assigned an address, and which is part of the network.
- A *configurating node* is an OLSR node which is currently assisting a new node in acquiring an address.

With the above definitions, the autoconfiguration extension can be summarized as follows: OLSR nodes behave as specified in the OLSR RFC [26]. Additionally, OLSR nodes periodically emit so-called ADDR_BEACON messages to signal to possible new nodes that they may act as configurating nodes (refer to Section 9.2.2 for details).

A new node does not emit HELLO or TC messages, but listens for ADDR_BEACON messages. From among the OLSR nodes emitting ADDR_BEACON messages, a new node selects a configurating node, and issues a request for address configuration through a so-called ADDR_CONFIG message.

The configurating node then aims at providing the new node *first* with a temporary local address, and *then* with a permanent global address. The following sections describe these mechanisms in details.

9.2.2 Local Beaconing

Each OLSR node, must ensure that it has the ability to provide temporary addresses to new nodes. It is important that, within a region, these temporary addresses are unique, *i.e.* prevent two new nodes within the same neighborhood of being assigned the same temporary address. In order to avoid this kind of event, a predefined address space is allocated to use as "temporary addresses". The task is then to ensure that this address space is divided (without overlap) between the OLSR nodes in a region of the network:

1. Each OLSR node will independently select a continuous address sequence from the address space allocated for "temporary addresses";
2. Each OLSR node will signal this selected sequence with periodic ADDR_BEACON messages. These messages are transmitted to neighbor nodes only, *i.e.* they are not forwarded;
3. Each OLSR node will record the address sequences selected by all its neighbors.

Different independent selection schemes can be used to choose address sequences. As shown in Figure 9.1, ADDR_BEACON messages also list the temporary addresses currently assigned by the originator node. Therefore, if an ADDR_BEACON message lists some temporary addresses as being in use, it is an indication that the originator of the message is at the moment an active configurating node.

Upon receiving an ADDR_BEACON message, a node can detect if there is a conflict in address sequence selection, *i.e.* if the address sequence it has chosen overlaps with the address sequence advertised by the received ADDR_BEACON message. In this case, arbitration must happen:

- If the conflict does not involve an active configurating node, the node with the lowest ID (IP address) yields, and selects a new address sequence.
- If the conflict involves only one active configurating node, the other node yields and selects a new address sequence. The aim here is to allow ongoing configuration sessions to complete.
- If the conflict is between two active configurating nodes, but not about addresses currently assigned to new node(s) being configured, both nodes give up their address sequence but still keep all the addresses they have currently assigned.

- If the conflict is between two active configurating nodes, and about one (or more) addresses currently assigned to new node(s) being configured, the configurating node with lowest ID must yield and select a new address sequence.

The ADDR_BEACON message has the format specified in figure 9.1. The OLSR RFC [26] specifies the values of Message Size, Originator Address, Message Sequence Number and Vtime. In case a node has to give up its address sequence, subsequent ADDR_BEACON messages will feature "Address Sequence Start" and "Address Sequence Stop" fields both set to zero (empty address sequence), until a new address sequence is selected. Currently assigned addresses continue to be advertised if no conflict in assignment is detected, until the autoconfiguration process they are used for completes. In case of a conflict in address assignment, the configurating node that yields will stop advertizing the address as currently assigned.

Each OLSR node periodically sends ADDR_BEACON messages, listing both its address sequence and the temporary addresses it has currently assigned. ADDR_BEACON messages are piggybacked with HELLO messages, in the same OLSR packet.

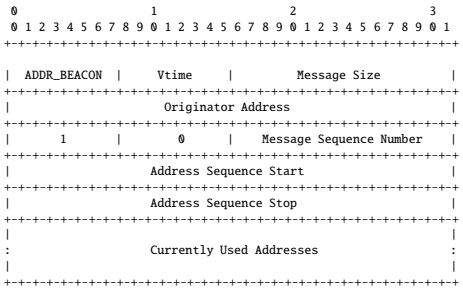


Figure 9.1: ADDR_BEACON message format.

9.2.3 Temporary Address Assignment

The first task of a new node is to associate itself with an OLSR node. Thus, the new node listens for ADDR_BEACON messages and selects one "configurating node" with non-empty address sequence. It then generates and transmits an ADDR_CONFIG message, in order to request address configuration from the selected configurating node. Absent an IP-address, the MAC address of the new node must be included in the message, in order to uniquely identify the new node.

Upon receiving an ADDR_CONFIG message, the configurating node assigns a local address to the new node, and signals this assignment through another ADDR_CONFIG message. Additionally, the configurating node marks the assigned address as “used” in its ADDR_BEACON messages.

Upon receiving a local address through an ADDR_CONFIG message, the new node will start sending HELLO messages, including only the configurating node as neighbor. This allows the new and configurating node to track each other, while it does not cause the new node to be advertised to the network: advertising a node with a non-unique address leads to possible data loss, loops etc. This procedure allows both nodes to “reset”, should the link disappear before a global address is assigned to the new node.

If a new node does not receive an ADDR_CONFIG reply, it may either retransmit the ADDR_CONFIG to the same configurating node – or give up and select an alternative configuration node. Absent the HELLO message exchange described above, the configuration node may retransmit its ADDR_CONFIG reply – or give up, in which case any assigned temporary address will be reclaimed.

In case of address assignement conflict where the configurating node must give up the temporary address assigned to the new node, it stops sending HELLO and ADDR_BEACON listing this address. If a new node stops receiving HELLO and ADDR_BEACON messages from the configurating node with its temporary address being listed, it gives up and proceeds to select another configurating node. It may select any node that sends ADDR_BEACON messages with a non-empty address sequence (it may even re-select the same configurating node if the latter has already obtained a new address sequence).

An ADDR_CONFIG message has the format specified in Figure 9.2. The OLSR RFC [26] specifies the values of

Message Size, Originator Address, Message Sequence Number and Vtime. If the “Assigned Local Address”, “Assigned Global Address” and “Originator Address” fields are all set to zero, the ADDR_CONFIG message is a request to the “Configurating Node” to perform local address assignment.

If the “Assigned Local Address” is non-zero (i.e. contains an actual address) and “Originator Address” is non-zero, but the “Assigned Global Address” field is set to zero, the ADDR_CONFIG message is an assignment of a temporary local address. I.e. this is the reply generated by a configurating node.

The “Assigned Global Address” field is discussed in Section 9.2.4.

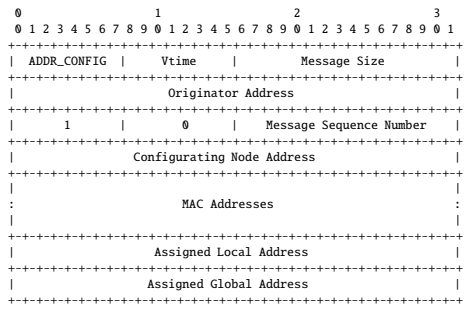


Figure 9.2: ADDR_CONFIG message format.

9.2.4 Permanent Address Assignment

When the HELLO message exchange is started between the new node and the configurating node, temporary local address assignment is completed and the goal is now for the new node to acquire a permanent global address. The configurating node is in charge of acting on behalf of the new node, with respect to acquiring a global address. Since the configurating node is already part of the OLSR network, several different mechanisms can be employed. One such mechanism for acquiring a global address would be for the configurating node to act as a modified DHCP proxy [47] and transmit a request to an existing DHCP server in the network.

Another option would be to consult the node's topology table. This table (in a stable state) contains all the global addresses of the nodes in the network. The configurating node can therefore pick an unused global address and assign it to the new node.

To prevent duplicate address assignment, the configurating node includes the tentative global address in a few TCs. If a node receives a TC containing its own address (or an address which the node has claimed for the configuration of a new node) and if the originator of the message is not the node itself nor an MPR of the node, a duplicate address assignment is detected. A node that detects an address duplication can then signal it to the originator of the "offending" TC, with the purpose of resolving the conflict. More generally, even outside of configurating phases, OLSR nodes should perform duplicate address detection routines, in order to prevent nodes from staying in any "mis-configured" state. Some such mechanisms are introduced in Section 9.3.

Once the configurating node has acquired a globally unique address, this address is assigned to the new

node through an ADDR_CONFIG message, containing the “Assigned Local Address” and “Originator Address” as before, but with the selected global address set in the “Assigned Global Address” field. The new node can from then on participate fully in the OLSR network.

The configurating node will continue to transmit this ADDR_CONFIG message periodically until either it stops receiving HELLO messages from the new node with the local address, or until an ADDR_CONFIG message from the new node is received, listing the new nodes global address in both the originator field and the “Assigned Global Address” field, and with the “Assigned Local Address” and “MAC address” set as earlier on.

9.3 Duplicate Address Detection

In this section we give an overview of the issues with duplicate address detection in ad hoc networks. We take the example of OLSR and using the proactive nature of the routing protocol, we show how monitoring the routing control messages allows the detection of many cases of duplicate addresses in the network, in a simple and overhead-free fashion. We also show how this simple passive approach has, however, some shortcomings, both in terms of false alarms and in terms of undetected duplications under some specific network conditions. This analysis was first presented in [13].

9.3.1 Introduction to Duplicate Address Detection

Addresses in a network are assigned in ways which try to ensure uniqueness in identification: no two interfaces within the same network should have the same address. This assumption is used by routing protocols to ensure that routes are accurate, and loop-free. In traditional wire-line networks uniqueness in identification is ensured by statefull centralized servers such as DHCP [44] or through stateless autoconfiguration such as in [49]. The former assumes that a centralized server is always present and reachable in the network. The latter assumes that all interfaces in the network share a broadcast or multicast link, over which they are reachable and able to participate in a distributed address assignment algorithm.

In a mobile ad hoc network, these assumptions cannot be made, as explained in the Section 9.2.1. Therefore, alternative mechanisms such as the one described in Section 9.2.1, provide different ways for nodes within an OLSR network to acquire unique addresses. This approach just assumes that the nodes form a connected network.

However nodes in MANETs are mobile, and therefore network topology may change over time. These topological changes might then cause disrupting events such as the following:

- *Network partitioning*: a set of nodes suddenly looses connectivity to the rest of the nodes.
- *Network merger*: formerly independent or partitioned networks are suddenly (re)connected.

In the situation where two networks merge, there is no guarantee that interfaces across the two networks have unique addresses. Therefore, a mechanism is required to verify that interface address uniqueness is preserved in the face of network mergers. Such a mechanism supplements mechanisms ensuring initial uniqueness of

configuration (such as the one described in Section 9.2.1).

Nodes should be permanently aware and able to detect if an address, currently assigned to one of its own interfaces, is concurrently used by another interface in the network. This task is called duplicate address detection (DAD) and two kinds of approaches exist for this: the passive approach, and the active approach. The *passive approach* is based on each node's monitoring and analysing of the traffic going by, in order to search for any abnormal packet that may indicate that an address is duplicated. The *active approach* is based on sending specific "probes" in the network to verify address uniqueness.

The advantage of passive approaches is that no additional information exchange is required between the nodes in order to perform duplicate address detection, while active approaches generate additional traffic overhead in order to function. However, active approaches may detect some abnormal situations quicker than passive approaches, since the latter basically waits for abnormal signs to go by, while active approaches go about "poking" the network.

Section9.3.2 describes a passive approach to duplicate address detection based on OLSR. Benefiting from its proactive, link state nature, monitoring OLSR's control traffic can be used to detect duplicate addresses efficiently. All that is required is for the nodes participating in the network to be running OLSR, and for the duplicate address detection mechanism to have access to the internal state of the OLSR routing daemon.

9.3.2 Performing Duplicate Address Detection in an OLSR Network

This section depicts different mechanisms that were first presented in [13], through which an OLSR node can detect if an address, currently assigned to one of its interfaces, is concurrently being used by another interface on another node somewhere in the OLSR network. The mechanisms presented here do not impose any additional information exchange between nodes beyond what is already performed by usual OLSR.

These duplicate address detection mechanisms are based on inspecting received OLSR control messages, as well as the receiving node's state, to determine if an address on the receiving node is duplicated elsewhere in the network. More precisely, a node inspects received OLSR messages to detect if (i) the message appears to have been sent from an interface of the receiving node, or (ii) the message contains information about interfaces of the receiving node. In either of these cases, the information contained in the received OLSR message is compared to the actual state recorded in the receiving node, allowing the latter to detect a poten-

tial duplicate of one of its addresses.

With this in mind, the following sections will describe the inspection of the three main OLSR message types: HELLO, TC and MID messages. In the figures shown in the following, nodes are identified by addresses ”A”, ”B”, ”C”, ... If an address is duplicated, two nodes will appear in the figure with the same address. The inspection is described based on the node indicated by a double-circle.

HELLO: Mismatching neighborhood

If a node receives a HELLO message on one of its interfaces, where the HELLO message appears to come from the node itself, a potential address duplication may have occurred: since HELLO messages are never forwarded, an OLSR node should not receive a copy of a HELLO message with any of its own interface address as originator address(es)*.

Should a node receive a HELLO message with one of its own interface addresses listed as originator, there’s a likely collision: two adjacent nodes have interfaces configured with the same address, as illustrated in Figure 9.3. From the point of view of the leftmost node ”A” (indicated by a double-circle), this can be confirmed by inspecting the neighborhood being advertised in the HELLO message: the HELLO message will include nodes B and C as neighbors, whereas neither are neighbors of the node receiving the HELLO. Thus, it can be detected by the leftmost node ”A” in Figure 9.3 that one of its interface addresses is also being used elsewhere in the network.

HELLO: MPR Selection Abnormality

A second intuitive diagnostic on HELLO messages is to consider MPR selection: an MPR node must be selected from among neighbors with which a symmetric link exist. Thus, if the leftmost node ”A” on Figure 9.4, which has a recorded asymmetric link with node ”B”, receives a HELLO from node B declaring it as MPR, then a conflict exists as indicated: a second node ”A’”, adjacent to ”B”, has the same address as ”A”.

This could, however, be a false conclusion. At the establishment of the link between ”A” and ”B” node ”A” receives a HELLO from ”B”, bringing node ”A” to see the link to ”B” as ASYM. In the next HELLO from node ”A”, node ”B” will see its own address listed and conclude that the link is symmetric. Node ”B” may, then, select ”A” as MPR and include this selection in the next HELLO message. In this way, node ”A”

*This ignores the situation where a node has two radio interfaces running OLSR on the same channel.

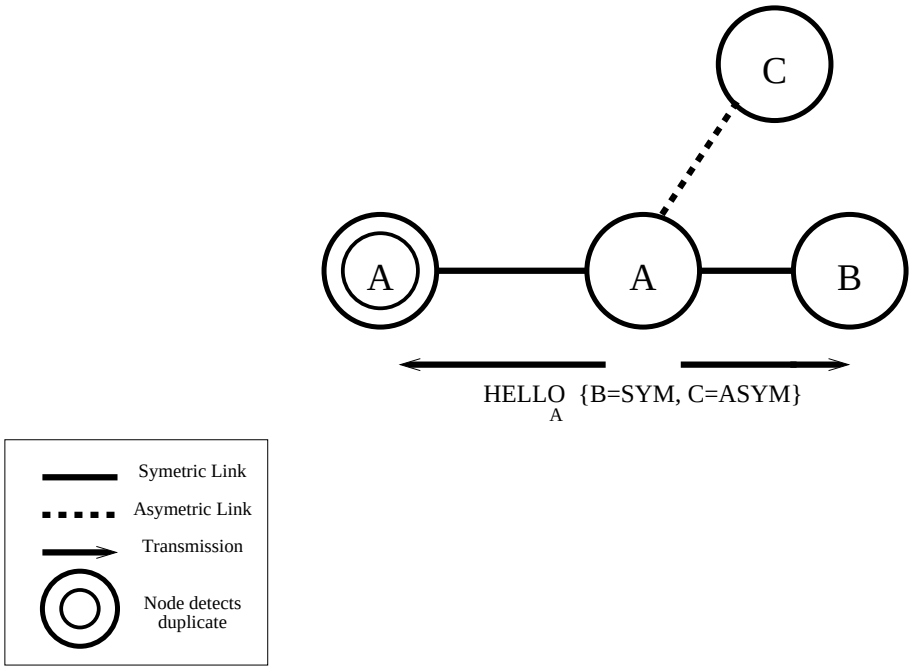


Figure 9.3: Node A detects an address duplication as it receives a HELLO message with its own address listed as originator address.

will receive an MPR selection from a node with which it has only an asymmetric link, without this being an indication of address conflicts in the network.

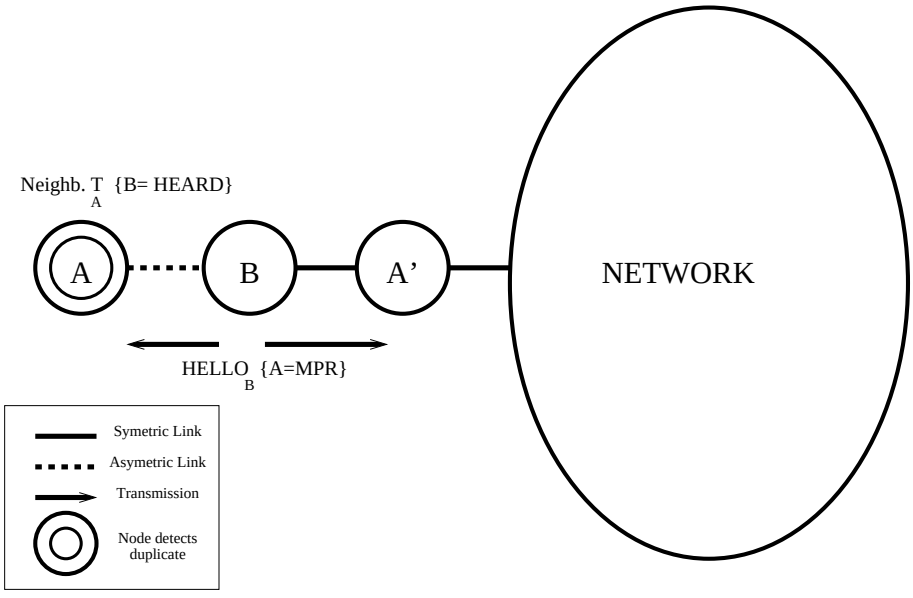


Figure 9.4: Node A detects an address duplication as it receives a HELLO message with its own address listed as MPR from a neighbor B with which it has no symmetric link.

TC: Sequence Number Mismatch

If a node, ”A”, receives a TC message with the address of one of its own interfaces listed as originator address and with a sequence number very different from the sequence number that node ”A” is currently using, this can be an indication that an interface of node ”A” is concurrently being assigned to another interface on another node somewhere in the OLSR network. This situation is illustrated in Figure 9.5.

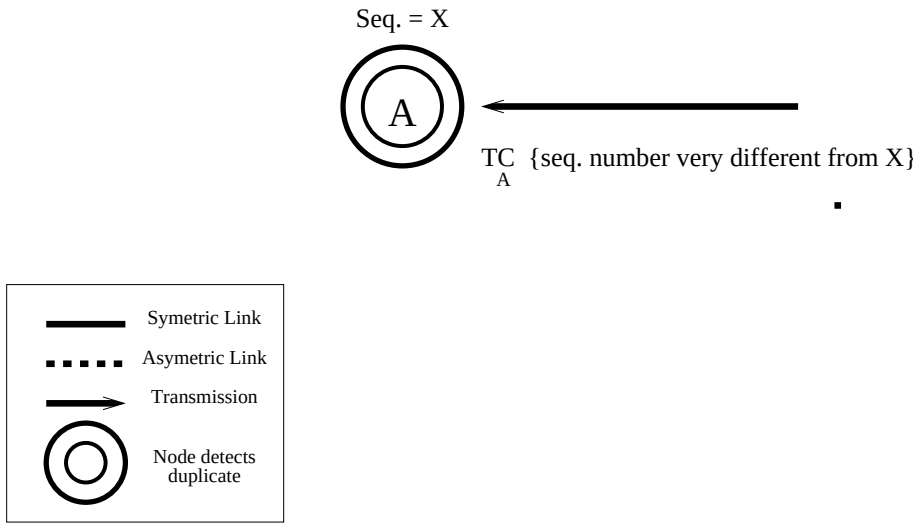


Figure 9.5: Node A detects an address duplication as it receives a TC message with its own address listed as originator address and with a sequence number that is very different from the current sequence numbers it uses.

TC: Link-State Mismatch

If a node, ”A”, receives a TC message, declaring the address of one of the interfaces of node ”A” as MPR selector, the originator of that TC message must be a direct neighbor of node ”A”. Considering, however, the situation illustrated in Figure 9.6: the rightmost node ”A” selects node ”C” as MPR, and thus node ”C” will advertise ”A” in its TC messages. From the point of view of the leftmost node ”A”, an address conflict will be detected thus: a TC will be received from node ”C”, advertising a link between node ”C” and node ”A”, yet in the leftmost node ”A” no such link exists.

MID: Interface Mismatch

With MID messages, an OLSR node with multiple interfaces declares its interfaces configuration to the other nodes in the network. If a node, ”A”, receives an MID message, in which the address of one of its own interfaces is listed, the remaining addresses listed in the MID must also belong to node ”A”. Alternatively, if a node, ”A”, receives an MID-message, containing one or more addresses, belonging to node ”A” but also

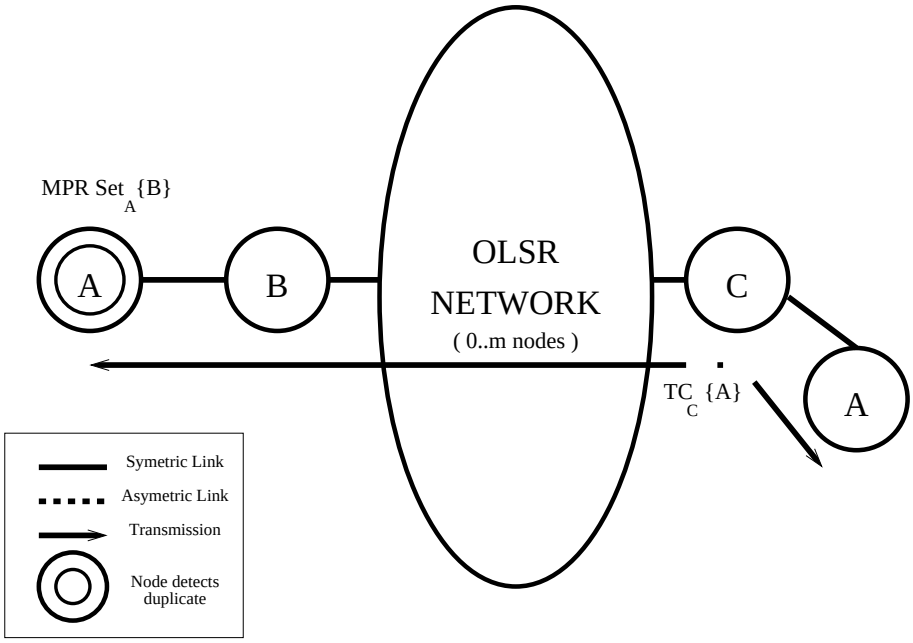


Figure 9.6: Node A detects an address duplication as it receives a TC message with its own address listed as originator address and with listed addresses that are not in its neighborhood, or MPR selection.

listing addresses which do not belong to node ”A”, then at least one address is assigned to more than one node. This is illustrated in Figure 9.7, in which one node has the addresses ”A”, ”A1” and ”A2, whereas the other node has the addresses ”A”, ”A3” and ”A4”.

9.3.3 Shortcomings of the Passive Approach

Passive mechanisms, such as those presented in this section, are based on the monitoring of the control messages of the routing protocol. These aim at detecting abnormal messages, that can hint to possible address collisions. However, this approach has a few shortcomings, both in terms of false alarms and in terms undetected duplications. In the rare case of a totally symmetric MANET, such as the one as depicted in Figure 9.8, routing message monitoring may not be sufficient to detect the duplicate addresses. In Figure 9.8, the duplicate nodes cannot detect the collision with each other since the routing messages produced by the left side of the network are identical to the routing messages produced by the right side of the network (because the topology is symmetric). Sequence number mismatch monitoring may help in this case, but it may also crash the network further, as such mismatches may invalidate the link state information with each TC transmission, alternatively from the right side and the left side of the network.

Another shortcoming is with the sequence number mechanism. This technique is not completely reliable

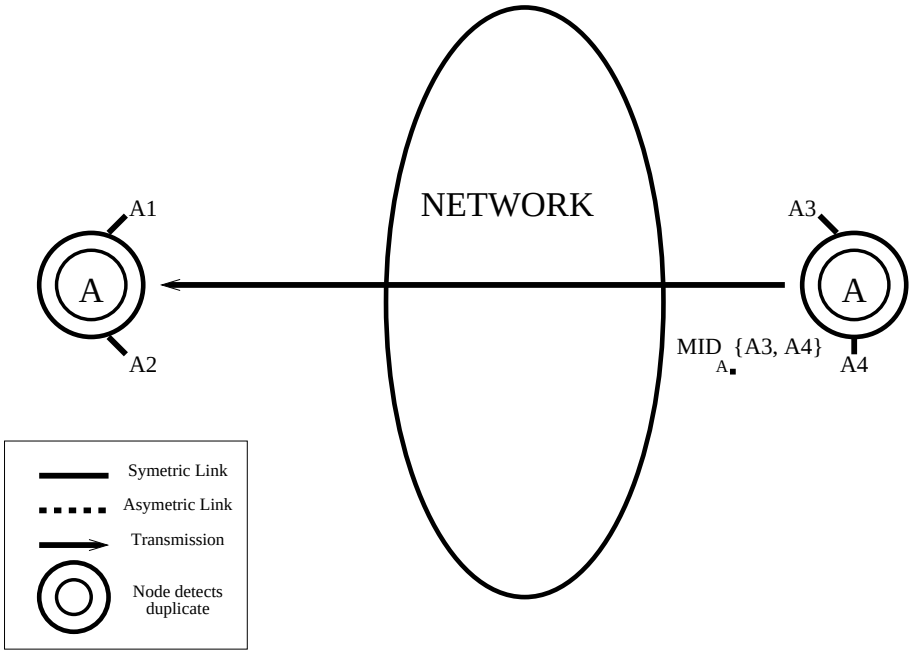


Figure 9.7: Node A detects an address duplication as it receives an MID message with its own address listed as originator, and with listed addresses that are different from its own.

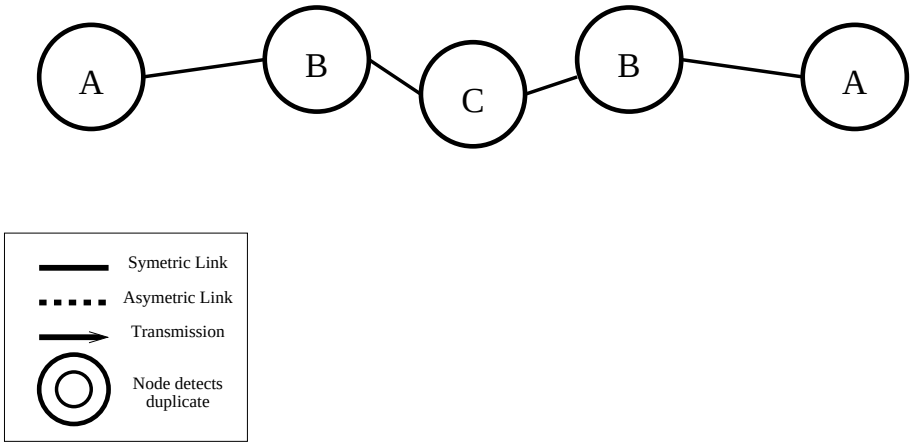


Figure 9.8: A completely symmetric OLSR network, where tracking of control traffic fails to detect address duplication

in order to detect duplicate addresses, as delayed delivery can cause an outdated control message to be possibly wrongly interpreted as a case of address duplication. This category of false alarm is more likely to be caused by TC or MID messages rather than HELLO messages, as they feature only one hop scope, suppressing delays due to forwarding.

Such cases challenge the passive approach to DAD. Therefore other techniques may be employed in ad-

dition to passive mechanisms in order to increase the reliability of the DAD. These techniques can be called active, or semi-passive, depending on how much additional overhead is produced by the mechanism.

Semi-passive techniques involve deeper analysis of the link state information traffic, such as tracking and processing the history of such traffic, in order to prevent errors. However, these techniques come with much more processing and memory needs, a fact that should be further evaluated.

Active techniques involve sending specific DAD information or messages, in addition to the routing control overhead. For instance, flooding a neighbor solicitation message is part of such a technique. These can be more efficient than passive waiting, but they nevertheless come with greater overhead, a fact that should also be further evaluated.

9.3.4 Resolving Duplicate Address Conflicts

The purpose of the mechanisms described so far is to detect when two or more interfaces in the network have been configured with the same address, *i.e.* that a duplicate address conflict exists in the network. The next-step is then to resolve the conflict, and to reconfigure nodes such that each interface participating in the OLSR network has a unique address, network-wide.

Functionally, resolving a duplicate address conflict is orthogonal to detecting a duplicate address conflict and, depending on the specificities of the network, different mechanisms may be employed. In this section, a few general approaches to resolving duplicate address conflict are briefly outlined. The objective, however, is to remove conflicting interfaces from the OLSR network, while incuring as little network disruption as possible.

Once a duplicate address conflict is detected, the simplest solution is for a node to simply disable the local interfaces which are conflicting. In this case, if these interfaces wish to re-enter the network, a new autoconfiguration cycle must be initiated. The advantage of this method is its simplicity and the fact that no lengthy election procedure must be completed before duplicate address conflicts are resolved. The disadvantage is that when a conflict arises, all conflicting interfaces will be disabled without consideration on traffic. One may also simply notice that when two interfaces are conflicting, it suffices to disable one of them, instead of both.

A more refined solution for conflict resolution is for node(s) that detect a conflict to ”negotiate” which interface should yield – possibly based on metrics such as active traffic flows for a given interface. This negotiation could take form in a broadcast of information (a ”CONFLICT” message), containing the necessary information for a recipient to decide if it should yield and disable a given interface, or not.

Chapter 10

Conclusions on Ad Hoc Networking

Ad hoc networking is seen as an important Internet component in the near future. Its autonomous and spontaneous deployment capabilities, coupled characteristic node mobility and its ability to work even without predetermined infrastructures, make this new type of wireless connectivity very desirable in the quest for ubiquitous networking. However, these characteristics do not come without a number of issues that are not addressed by usual network solutions – and by the Internet in particular.

In order to accomplish routing in mobile ad hoc networks, drastic optimizations are necessary to address the scarce bandwidth, the high topology change rates, the radio interferences between nodes *etc*. Thus, this thesis has shown that the coupling of optimized flooding, link state routing and Fish-Eye techniques is a suitable solution, able to adapt for ad hoc networks of arbitrary size. This thesis also proposed to introduce clustering and hierarchical routing orthogonally to these techniques, which could further increase the bandwidth available per node in large topologies. A further study on the subject may be of interest. More generally, the question is: how much can we further increase the bandwidth available to users in large ad hoc networks? Some theoretical bounds on network capacity are known for models where ad hoc nodes are static. Nevertheless, an approach using mobility to create additional capacity [86] may enable the network capacity to go beyond these bounds, for some types of traffic and mobility patterns.

On the other hand, the integration of mobile ad hoc nodes at the periphery of the Internet, seems a natural way to extend the Internet access to mobile users. However, several new mechanisms are needed to not disrupt the Internet's original architecture, which was not designed for nodes to be mobile. Therefore, this thesis has introduced new ways to integrate ad hoc mobility in the Internet, with MANEMO on one hand, and

wOSPF on the other hand. In addition, new solutions for ad hoc node IP autoconfiguration were proposed.

Ad hoc networking has been the subject of numerous different efforts in the recent years, both in the industry and in academia. Capitalizing on the accumulated experience, the standardization of ad hoc networking protocols is currently taking place in the IETF [63]. Several new proposals in this thesis actively participate in this standardization process, currently as Internet-Drafts (see Apendix A) in different working groups such as MANET [66], NEMO [68], or OSPF [67]. In order for the standardization process to complete, some additional issues specific to ad hoc networks remain to be thouroughly addressed. The most important example of such open problem is perhaps that of security in ad hoc networks.

Once a complete suite of ad hoc routing and mobility protocols will be standardized, the face of the Internet and networking in general can be anticipated to change substantially. Indeed, it is likely that in a few years, ad hoc connectivity will be by default expected to come with every device able to communicate over wireless, and will seem as natural as the "one-hop" wireless connectivity we have nowadays. The Internet is thus likely to naturally expand much faster from its fixed base to the wireless and mobile domains with the advent of ad hoc networking. One can envision the Internet to soon be extended by a "mobile core" evolving at the periphery of its fixed core, *e.g.* mobile routers in buses, taxis, planes, cars, boats, trains, *etc.*. These will serve millions of (mobile) ad hoc nodes and users directly integrated in the Internet and actually *making* the wireless Internet, accessing the fixed core only when necessary (contrary to systematically, nowadays). Ad hoc nodes may be of kinds as varied as smart phones, fridges, smart dust, car parts, sensors or any device worthy of communication. In that respect, ad hoc networking is the natural vector of Internet's original philosophy over the wireless medium, as well as an essential step in direction of ubiquitous networking.

Part IV

Appendixes

Appendix A

Internet Standards

The main body that produces materials that become Internet standards is the *IETF* (Internet Engineering Task Force [63]). The IETF is organized into several functional areas such as: *Applications*, *Internet*, *Operations & Management*, *Routing*, *Security*, *Transport*, or *User Services*. Each area typically has one or two *area directors*, and all the area directors form the *IESG* (Internet Engineering Steering Group), the assembly that ultimately decides wether and when anything should become a norm on the Internet – then called a *standard*.

Each area is composed of a number of *working groups* that specialize on a particular topic. For example, the routing area contains a working group called MANET [66] that specializes on ad hoc networking. Each working group typically has one or two chairs who are responsible for running all aspects of the working group according to the guidelines created by the IESG.

Anyobody can participate in the standardization process: the IETF is more or less a democracy with the bodies described above, based on "rough consensus and running code". One just needs to attend the meetings (three times a year, with non-prohibitive registration fees) and/or participate in the discussions on the mailing lists. People participating in the process typically come from varied industrial and academic organizations. However, one can just represent one's self too.

The documents produced by the IETF are of two types: *Internet Drafts (ID)* and Request For Comments (RFC). While a standard is being designed and developed, draft versions of the standards and other associated documents must be published. At this stage, the type of document used is an Internet draft. Internet drafts are named in a way that describes the working group, the author, and the subject of the draft. For ex-

ample *draft-ietf-manet-dsr-10.txt* is the 10th revision of the internet draft on Dynamic Source Routing for ad hoc networks, written by the IETF's MANET working group. However, the author can also be any individual, and not be tied to a specific working group.

An Internet draft may go through several subsequent versions, which are always made available publicly, downloadable for free on the Internet. An Internet draft is a temporary document describing work in progress that has only three possible outcome. Either (i) it is published as a permanent document, called an *RFC (Request For Comments)* (ii) it is updated within 6 months with a new, revised version, or (iii) it administratively expires after 6 months – but can still be found in archives on the Web.

RFCs are numbered sequentially, and once published, are never revised. For instance, RFC3626 is the publication of the Optimized Link State Routing protocol specifications, that was developed within the MANET working group and the Routing Area. There are however several *categories* of RFC: *Standards Track*, *Best Current Practice (BCP)*, *Informational*, *Experimental*, or *Historical*, depending on the nature of the subject. For a protocol specification that is candidate to standardization (*i.e.* going standards track), the corresponding RFC goes through successive stages: *Proposed Standard*, then *Draft Standard*, and finally, if ratified after gaining wide acceptance and operational experience ("rough consensus and running code"), the specification is adopted as a norm, a *Standard*. A standard stays a standard as long as it is not deprecated. If so, it is changed to the Historical status.

Appendix B

The Djikstra Algorithm

The exact Djikstra algorithm is the following. Let S be the set of nodes to which shortest paths have already been found. Let C be the set of nodes to which we are currently computing shortest paths. We initially start with S containing only the root r, and C containing nothing. The algorithm then repeats the following steps:

1. For the node s just added to S, add each of its neighbors n to C, such that: (i) if n is not in C, add it, mentionning the cost to reach it through s, as well as the identity of s, and (ii) if n is already in C and the path to n through s has a lower cost than the one that is mentionned so far, then remove the earlier instance of n and add a new instance of n annotated with the cost to reach it through s as well as the identity of s.
2. Pick the node p that has the smallest cost in C, and if it is not already in S, add it to S. Use its annotation to determine the router s to use to reach p, and therefore the optimal next hop from r to reach p.
3. If C is now empty, the algorithm stops. If not, go back to step 1.

When the algorithm stops, we have the optimal next hops from the root to every destination in the network, which gives the root's routing table.

Appendix C

Factor λ in $r(\lambda)$

By definition $\int_0^{r(\lambda)^{-\alpha}/K} w(x)dx = p_0$. Using the reverse Laplace transformation we have:

$$w(x) = \frac{1}{2i\pi}\int_{-i\infty}^{+i\infty} \tilde{w}(\theta)e^{\theta x}d\theta.$$

Inserting this expression in the first equation and commuting integral signs, since $\int_0^{r(\lambda)^{-\alpha}/K} e^{\theta x}dx = \frac{e^{\theta r(\lambda)^{-\alpha}/K}-1}{\theta}$, yields:

$$\frac{1}{2i\pi}\int_{-i\infty}^{+i\infty} \frac{e^{\theta r(\lambda)^{-\alpha}/K}-1}{\theta}\tilde{w}(\theta)d\theta = p_0.$$

The change of variable $\lambda^{\alpha/2}\theta = \theta'$ makes λ disappear from the $\tilde{w}(\theta)$ expression:

$$\frac{1}{2i\pi}\int_{-i\infty}^{+i\infty} \frac{e^{\theta'(r(\lambda)\sqrt{\lambda})^{-\alpha}/K}-1}{\theta'}\tilde{w}(\lambda^{\alpha/2}\theta')d\theta' = p_0.$$

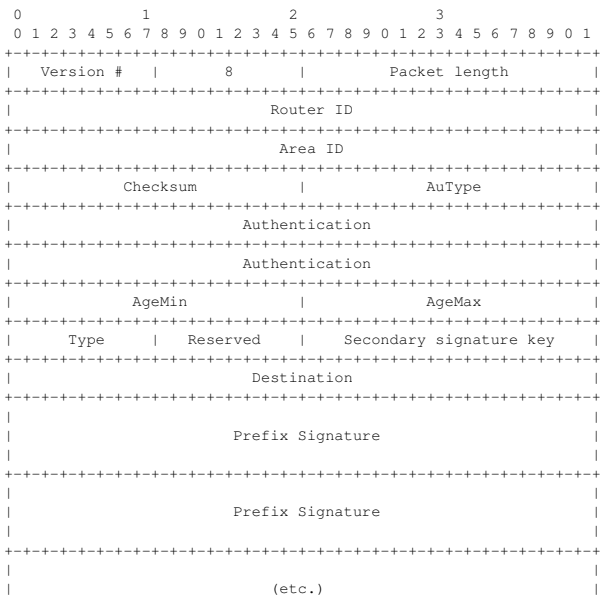
Since $\tilde{w}(\lambda^{\alpha/2}\theta')$ is independent from λ and $r(\lambda)$ appears multiplied by $\sqrt{\lambda}$, we get that $r(\lambda)$ is simply proportional to $1/\sqrt{\lambda}$: $r(\lambda) = r(1)/\sqrt{\lambda}$.

Appendix D

DBX Packet Formats

Info and dbx signatures share the same packet format, detailed in this section.

Signature Packet Format



Version #, Packet length, Router ID, Area ID, Checksum, AuType and Authentication fields are the OSPF control packet header as described in [34].

AgeMin, AgeMax

AgeMin and AgeMax define the age interval [AgeMin,AgeMax], used for computing the timed partial

signatures in the prefix signatures as described in section 7.3.4.

Type

Specifies if the signature is an info or a dbx signature, according to the following:

Value	Type
1	info (informative)
2	dbx (database exchange)

Reserved

Must be set to ”00000000” for compliance with this specification.

Secondary signature key

The key of the secondary signature is a random number of 32 bits. Used for computing the secondary partial signature as described in section 7.3.3.

Destination

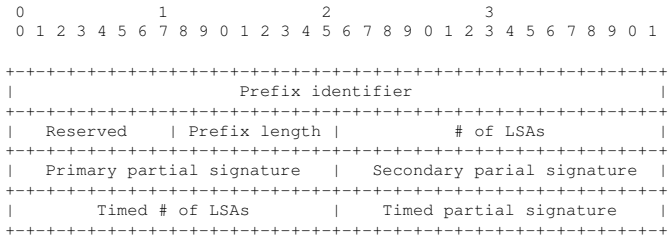
- If the signature is of type = 2, then this field contains the address of the slave, with which a database exchange is requested.

- If the signature is of type = 1, then this field must be zeroed.

Prefix signature

The set of prefixes signatures contains the sub-signatures for different parts of the link-state database. The layout of the prefix signatures is detailed in section 7.3.4.

Prefix Signature Format



Prefix identifier and Prefix length

Indicates the length of the prefix for the part of the link-state database, as well as the exact prefix.

# of LSAs

The number of LSAs in the emitting nodes link-state database, matching by the prefix identifier and prefix length.

Primary partial signature

The arithmetic sum of the hashing of each string made of the concatenation of sequence number and LSA-originator ID fields of the tuples (LSA-originator ID, LSAsequence-number, LSA-age) from the emitting nodes link-state database such that the LSA-originator ID and prefix ID has same prefix of length prefix-length.

Secondary partial signature

The arithmetic sum of the XOR between the secondary signature key and each of the hashing of each string made of the concatenation of sequence number and LSA-originator ID fields of the tuples (LSA-originator ID, LSAsequence-number, LSA-age) from the emitting nodes link-state database such that the LSA-originator ID and prefix ID has same prefix of length prefix-length.

Timed # of LSAs

The number of LSAs in the emitting nodes link-state database, matching by the prefix identifier and prefix length and satisfying the condition that the LSA age is between AgeMin and AgeMax.

Timed partial signature

The arithmetic sum of the hashing of each string made of the concatenation of sequence number and LSA-originator ID fields of the tuples (LSA-originator-ID,LSA sequence-number, LSA-age) from the emitting nodes link-state database such that:

- Prefix ID and LSA-originator ID has same prefix of length prefix-length
- LSA-age is between AgeMin and AgeMax.

Bibliography

Publications

[1] E. Baccelli, T. Clausen, P. Jacquet, “Ad-hoc and Internet Convergence: Adapting OSPF-style Database Exchanges for Ad-hoc Networks,” Proceedings of the Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs), Oct. 2004.

[2] C. Adjih, E. Baccelli, P. Jacquet, “Link State Routing in Wireless Ad Hoc Networks,” Proceedings of MILCOM 2003 - IEEE Military Communications Conference, vol. 22, no. 1, pp. 1274-1279, Oct. 2003.

[3] E. Baccelli, R. Rajan, “Monitoring OSPF Routing,” proceedings of IM 2001 - IFIP/IEEE International Symposium on Integrated Network Management, no. 1, pp. 825-838, May 2001.

[4] E. Baccelli, T. Clausen, P. Jacquet, “OSPF Database Exchange and Reliable Synchronization in Mobile Ad Hoc Networks,” Proceedings of the IASTED Conference on Wireless Networks and Emerging Technologies (WNET), July 2004.

[5] E. Baccelli, P. Jacquet, “Diffusion Mechanisms for Multimedia Broadcasting in Mobile Ad Hoc Networks,” Proceedings of the IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA), Aug. 2004.

[6] E. Baccelli, T. Clausen, P. Jacquet, “OSPF-style Database Exchange and Reliable Synchronization in the Optimized Link-State Routing Protocol,” Proceedings of the IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON), Oct. 2004.

[7] E. Baccelli, T. Clausen, R. Wakikawa, “Route Optimization in Nested Mobile Networks (NEMO) using OLSR,” Proceedings of the IASTED International Conference on Networks and Communication Systems (NCS), April 2005.

[8] E. Baccelli, T. Clausen, “A Simple Address Autoconfiguration Mechanism for OLSR,” Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), May 2005.

[9] C. Adjih, E. Baccelli, T. Clausen, P. Jacquet, R. Rodolakis, “Fish Eye OLSR Scaling Properties,” published in the IEEE sponsored Journal of Communication and Networks (JCN), Special Issue on Mobile Ad Hoc Wireless Networks, Dec. 2004.

[10] E. Baccelli, F. Baker, M. Chandra, T. Henderson, J. Macker, R. White, “Problem Statement for OSPF Extensions for Mobile Ad Hoc Routing,” Internet Engineering Task Force (IETF) Internet Draft draft-baker-manet-ospf-problem-statement-00.txt, Sept. 2003.

[11] J. Ahrenholz, E. Baccelli, T. Clausen, T. Henderson, P. Jacquet, P. Spagnolo, “OSPFv2 Wireless Interface Type,” Internet Engineering Task Force (IETF) Internet Draft draft-spagnolo-manet-ospf-wireless-interface-01.txt, May 2004.

[12] E. Baccelli, “OLSR Trees: A Simple Clustering Mechanism for OLSR,” Proceedings of the 4TH Mediterranean Workshop on Ad-Hoc Networks (MED-HOC-NET), June 2005.

[13] E. Baccelli, T. Clausen, J. Garnier, “Duplicate Address Detection in OLSR Networks,” Proceedings of WPMC, Sept. 2005.

[14] E. Baccelli, T. Clausen, “OLSRv2 Link Hysteresis,” Internet Engineering Task Force (IETF) Internet Draft draft-clausen-olsrv2-link-hysteresis-00.txt, July 2005.

[15] C. Adjih, E. Baccelli, T. Clausen, P. Jacquet, “On the Robustness and Stability of Connected Dominating Sets,” INRIA Research Report 5609, http://www.inria.fr/rrrt/rr-5609.html, June 2005.

[16] E. Baccelli, T. Clausen, P. Jacquet, “Partial Topology in an MPR-based Solution for Wireless OSPF on Mobile Ad Hoc Networks,” INRIA Research Report 5619, http://www.inria.fr/rrrt/rr-5619.html, July 2005.

[17] E. Baccelli, T. Clausen, J. Garnier “OLSR Passive Duplicate Address Detection,” Internet Engineering Task Force (IETF) Internet Draft draft-clausen-olsr-passive-dad-00.txt, July 2005.

[18] E. Baccelli, T. Clausen, P. Jacquet, “DB Exchange for OSPFv2 Wireless Interface Type,” Internet Engineering Task Force (IETF) Internet Draft draft-clausen-manet-ospf-dbx-00.txt, February 2004.

[19] C. Adjih, E. Baccelli, P. Jacquet, “Link State Routing in Wireless Ad-Hoc Networks,” INRIA Research Report 4874, http://www.inria.fr/rrrt/rr-4874.html, July 2003.

[20] E. Baccelli, P. Jacquet, “Flooding Techniques in Mobile Ad Hoc Networks,” INRIA Research Report 5002, http://www.inria.fr/rrrt/rr-5002.html, November 2003.

[21] E. Baccelli, T. Clausen, R. Wakikawa, “NEMO Route Optimisation Problem Statement,” Internet Engineering Task Force (IETF) Internet draft-clausen-nemo-ro-problem-statement.txt, October 2004.

[22] E. Baccelli, T. Clausen, “Simple MANET Address Autoconfiguration,” Internet Engineering Task Force (IETF) Internet draft-clausen-manet-address-autoconf-00.txt, February 2005.

[23] E. Baccelli, T. Clausen, “Securing OLSR Problem Statement” Internet Engineering Task Force (IETF) Internet draft-clausen-manet-securing-olsr-problem-statement-00.txt, February 2005.

[24] E. Baccelli, T. Clausen, P. Jacquet, “Signature and Database Exchange for Wireless OSPF Interfaces,” INRIA Research Report 5096, http://www.inria.fr/rrrt/rr-5096.html, January 2004.

[25] E. Baccelli, T. Clausen, P. Jacquet, “OSPF-style Database Exchange and Reliable Synchronization in the OLSR,” INRIA Research Report 5283, http://www.inria.fr/rrrt/rr-5283.html, July 2004.

References

[26] T. Clausen, P. Jacquet Ed. “RFC 3626: Optimized Link State Routing Protocol,” Internet Engineering Task Force (IETF) Request For Comments, http://ietf.org/rfc/rfc3626.txt, 2003.

[27] S. Corson, J. Macker, “RFC 2501: Mobile Ad hoc Networking (MANET), Routing Protocol Performance Issues and Evaluation Considerations,” Internet Engineering Task Force (IETF) Request For Comments, http://ietf.org/rfc/rfc2501.txt, 1999.

[28] Navod Nikaein, Houda Labiod, Christian Bonnet, “DDR - Distributed Dynamic Routing Algorithm for Mobile Ad hoc Networks,” Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC), Aug. 2000.

[29] T. Clausen, P. Jacquet, L. Viennot, “Comparative Study of Routing Protocols for Mobile Ad-hoc NETworks,” Proceedings of the IFIP Mediterranean Workshop on Ad-Hoc Networks (MED-HOC-NET), Sept. 2002.

[30] A. Qayyum, L. Viennot, A. Laouiti, “Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks,” INRIA Research Report RR-3898, March 2000.

[31] C. E. Perkins, E. M. Royer, S. R. Das, “RFC 3561: Ad Hoc On-Demand Distance Vector Routing,” Internet Engineering Task Force (IETF) Request For Comments (experimental), http://ietf.org/rfc/rfc3561.txt, July 2003.

[32] R. Ogier, F. Templin, M. Lewis, “RFC 3684: Topology Dissemination Based on Reverse-Path Forwarding,” Internet Engineering Task Force (IETF) Request For Comments (experimental), http://ietf.org/rfc/rfc3684.txt, February 2004.

[33] D. Johnson, D. Maltz, Y. Hu, “The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks,” Internet Engineering Task Force (IETF) Internet Draft draft-ietf-manet-dsr-10.txt, July 2004.

[34] J. Moy, “RFC 2328: OSPF version 2,” Internet Engineering Task Force (IETF) Request For Comments, http://ietf.org/rfc/rfc2328.txt, 1998.

[35] J. Moy, “OSPF, Anatomy of an Internet Routing Protocol,” Addison-Wesley, 1998.

[36] B. Fortz, M. Thorup, “Internet Traffic Engineering by Optimizing OSPF Weights,” Proceedings of INFOCOM, 2000.

[37] R. Coltun, D. Ferguson, J. Moy, ”OSPF for IPv6”, RFC 2740, 1999.

[38] IEEE 802.11 Standards Working Group, http://grouper.ieee.org/groups/802/11/index.html

[39] C. Perkins et al. “RFC 3344: IP Mobility Support for IPv4,” Internet Engineering Task Force (IETF) Request For Comments, http://www.ietf.org/rfc/rfc3344.txt, August 2002.

[40] V. Devarapalli et al. “RFC 3963: Network Mobility (NEMO) Basic Support Protocol,” Internet Engineering Task Force (IETF) Request For Comments, http://www.ietf.org/rfc/rfc3963.txt, January 2005.

[41] L. Bellier et al. “Hierarchical Mobile IPv6 mobility management (HMIPv6),” Internet Engineering Task Force (IETF) Internet Draft (expired), http://www.watersprings.org/pub/id/draft-ietf-mobileip-hmipv6-08.txt, June 2003.

[42] K. Mase, S. Kameyama, S. Yoshida, M. Goto, T. Hasegawa, “A Novel Routing Paradigm for Mobile Ad Hoc Networks: Multihop Hello Guided Routing (MHGR),” Proceedings of VTC, 2005.

[43] K. Mase, S. Narita, S. Yoshida, “Efficient IP Address Assignment in Mobile Ad Hoc Networks,” Proceedings of WPMC, 2003.

[44] R. Droms, “RFC 2131: Dynamic host configuration protocol,” Internet Engineering Task Force (IETF) Request For Comments, March 1997.

[45] A. Williams, “Requirements for Automatic Configuration of IP Hosts,” Internet Engineering Task Force (IETF) Internet Draft (expired), http://www.watersprings.org/pub/id/draft-ietf-zeroconf-reqts-12.txt, September 2002.

[46] K. Mase, T. Sato, K. Nakano, M. Sengoku, S. Shinoda, “Efficient Flooding Schemes in Mobile Ad Hoc Networks,” Proceedings of MDMC, 2001.

[47] M. Patrick, “RFC 3046: DHCP Relay Agent Information Option,” Internet Engineering Task Force (IETF) Request For Comments, http://www.ietf.org/rfc/rfc3046.txt, January 2001.

[48] K. Mase, C. Adjih, “No Overhead Autoconfiguration OLSR,” Internet Engineering Task Force (IETF), Internet-Draft (work in progress), May 2005.

[49] S. Thomson, T. Narten, “RFC 2462: Ipv6 Stateless Address Autoconfiguration,” Internet Engineering Task Force (IETF) Request For Comments, http://www.ietf.org/rfc/rfc2462.txt, December 1998.

[50] http://www.internetworldstats.com

[51] P. Gupta and P.R. Kumar. Capacity of Wireless Networks. Technical Report, University of Illinois, Urbana-Champaign, 1999. http://citeseer.nj.nec.com/gupta99capacity.html

[52] M. Gerla, X. Hong, G. Pei, “Fisheye State Routing Protocol (FSR) for Ad Hoc Networks,” Internet Engineering Task Force (IETF) Internet Draft (expired), http://www.watersprings.org/pub/id/draft-ietf-manet-fsr-03.txt, 2002.

[53] T. Clausen, “Combining Temporal and Spatial Partial Topology for MANET Routing - Merging OLSR and FSR,” WPMC Proceedings, 2003.

[54] C. Adjih, P. Jacquet, L. Viennot, “Computing Connected Dominating Sets with Multipoint Relays,”INRIA Research Report RR-4597, 2002.

[55] P. Jacquet, “Elément de Théorie Analytique de l’Information, Modélisation et Evaluation de Performances,” INRIA Research Report RR-3505, http://www.inria.fr/rrrt/rr-3505.html, 1998.

[56] P. Jacquet, A. Laouiti, P. Minet, L. Viennot, “Performance Evaluation of Multipoint Relaying in Mobile Ad Hoc Networks,” Networking 2002 Proceedings, 2002.

[57] M. Chandra et al. “Extensions to OSPF to Support Mobile Ad Hoc Networking,” Internet Engineering Task Force (IETF) Internet Draft, draft-chandra-ospf-manet-ext-03, 2005.

[58] S. Keshav, “An Engineering Approach to Computer Networking,” Addison-Wesley, 1998.

[59] M. Gerla, C.R. Lin, “Adaptive Clustering for Mobile Wireless Networks,” IEEE Journal on Selected Areas in Communications, 1997.

[60] T. Chen, C. Chiang, M. Gerla, A. Iwata, G. Pei, “Scalable Routing Strategies for Ad-hoc Wireless Networks,” IEEE Journal on Selected Areas in Communications, Special Issue on Ad Hoc Networks, 1999.

[61] J. Stewart “BGP4: Inter-Domain Routing in the Internet”, Addison-Wesley, 2000.

[62] Y. Rekhter, T. Li (Ed.) “A Border Gateway Protocol 4 (BGP-4),” RFC 1771, http://ietf.org/rfc/rfc1771.txt, 1995.

[63] The Internet Engineering Task Force, http://www.ietf.org.

[64] M. Gerla, G. Pei, “Mobility Management for Hierarchical Wireless Networks,” ACM/Kluwer Mobile Networks and Applications, 2000.

[65] M. Gerla, X. Hong, G. Pei, “LANMAR: Landmark Routing for Large Scale Wireless Ad Hoc Networks with Group Mobility,” Proceedings of IEEE/ACM MobiHOC, 2000.

[66] IETF MANET Working Group, http://www.ietf.org/html.charters/manet-charter.html.

[67] IETF OSPF Working Group, http://www.ietf.org/html.charters/ospf-charter.html.

[68] IETF NEMO Working Group, http://www.ietf.org/html.charters/nemo-charter.html.

[69] P. Flajolet, A. Odlyzko, “Singularity analysis of generating functions,” SIAM J. of Discrete Math. 3, 1990.

[70] D. Oran, “RFC 1142: OSI IS-IS Intra-domain Routing Protocol,” Internet Engineering Task Force (IETF) Request For Comments, http://ietf.org/rfc/rfc1142.txt, 1990.

[71] P. Thubert et al. “Taxonomy of Route Optimization Models in the NEMO Context,” Internet Engineering Task Force (IETF) Internet Draft (expired), http://www.watersprings.org/pub/id/draft-thubert-nemo-ro-taxonomy-02.txt, 2004.

[72] H. Kang et al. “Route Optimization for Mobile Network by Using Bi-directional Between Home Agent and Top Level Mobile Router,” Internet Engineering Task Force (IETF) Internet Draft (expired), http://www.watersprings.org/pub/id/draft-hkang-nemo-ro-tlmr-00.txt, 2004.

[73] C. Ng et al. “Securing Nested Tunnels Optimization with Access Router Option,” Internet Engineering Task Force (IETF) Internet Draft (expired), http://www.watersprings.org/pub/id/draft-ng-nemo-access-router-option-01.txt, 2004.

[74] H. Ohnishi et al. “HMIP based Route Optimization Method in a Mobile Network,” Internet Engineering Task Force (IETF) Internet Draft (expired), http://www.watersprings.org/pub/id/draft-ohnishi-nemo-ro-hmip-00.txt, 2003.

[75] C. Hedrick, “RFC 1058: The Routing Information Protocol (RIP),” Internet Engineering Task Force (IETF) Request For Comments, http://ietf.org/rfc/rfc1058.txt, 1988.

[76] J. Wu, H. Li, “On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks,” Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, 1999.

[77] J. Wu, F. Dai, “Performance Analysis of Broadcast Protocols in Ad Hoc Networks Based on Self Pruning,” Proceedings of WCNC, 2004.

[78] A. Qayyum, L. Viennot, A. Laouiti, “Multipoint Relaying for Flooding Broadcast Messages in Mobile Wireless Networks,” Proceedings of HICSS, 2002.

[79] L. Kleinrock “Queuing Systems, Vol. 1: Theory”, J. Wiley and Sons, 1975.

[80] W. Feller “An Introduction to Probability Theory and its Applications, Vol. 2”, J. Wiley and Sons, 1971.

[81] J. Wu, F. Dai, “Double Covered Broadcast (DCS): A Simple Reliable Broadcast Algorithm in MANETs,” Proceedings of INFOCOM, 2004.

[82] J. Wu, F. Dai, “On Constructing k-Connected k-Dominating Set in Wireless Networks,” Proceedings of IPDPS, 2005.

[83] J. Wu, F. Dai, “Efficient Broadcasting with Guaranteed Coverage in Mobile Ad Hoc Networks,” IEEE Transactions on Mobile Computing, 2005.

[84] R. Ogier, “MANET Extension of OSPF using CDS Flooding”, IETF Internet Draft draft-ogier-manet-ospf-extension-03, 2005.

[85] M. Chandra et al. “Extensions to OSPF to Support Mobile Ad Hoc Networking”, IETF Internet Draft draft-chandra-ospf-manet-ext-03, 2005.

[86] M. Grossglauser, D. Tse, “Mobility Increases the Capacity of Ad Hoc Wireless Networks,” Proceedings of INFOCOM, 2001.