ORDER N° : 8388

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PARIS-SUD XI ORSAY

discussed by

Saadi Boudjit

on September 25, 2006

to obtain the degree of

Docteur de l'Université Paris-Sud XI

Discipline: Computer Science

Host laboratory: INRIA Rocquencourt

Title

Autoconfiguration and Security Schemes for OLSR Protocol for Mobile Ad Hoc Networks

Thesis Director: Paul Mühlethaler

Jury

Reviewers:	Stéphane Ubéda	INSA Lyon
	Ken Chen	University of Paris 13
Examiners:	Guy Pujolle	University of Paris 6
	Philippe Jacquet	INRIA Rocquencourt
	Paul Mühlethaler	INRIA Rocquencourt
	Khaldoun Alagha	University of Paris-Sud 11
Guest:	Cédric Adjih	INRIA Rocquencourt

To the memory of my young sister

Abstract

Rapid advancements in wireless technology and the emergence of new wireless applications such as wireless Internet, have resulted in the vast proliferation of wireless communications networks which are growing in popularity due to the abundance of low-cost mobile devices, and the speed and convenience of their deployment. These networks can be used in many domains (domestic use, professional use, ...), however their configuration remains a task for a network administrator. Network configuration is the assignment of network parameters necessary for these mobile devices to integrate the network, as, for instance, an IP address, netmask, the IP address of the gateway, ... Hence, it is essential to provide autoconfiguration services destined for people who are not familiar with the complex task of network configuration within an IP-based network. Furthermore, in the case of mobile ad hoc networks (MANETs), nodes are highly dynamic and a central administration or configuration by the user is very difficult. For small scale MANETs, it may be possible to allocate free IP addresses manually. However, the procedure becomes impractical for a large-scale or open system where mobile nodes are free to join and leave.
On the other hand, securing communications is a relevant issue within an ad hoc network. In fact, the broadcast nature of wireless communication links makes them unique in their vulnerability to security attacks.
This thesis addresses these two problems in ad hoc networks. Some of the existing solutions are reviewed while a number of new solutions are introduced.
Notice that the security algorithm proposed in this thesis deals with the generation of a symmetric group key in an ad hoc network. Hence, it can be seen as an autoconfiguration protocol related to security.

Keywords: ad hoc, OLSR, IPv6, autoconfiguration, security.

Acknowledgements

I would like to express my sincere appreciation to my advisor, Dr. Paul MUHLETHALER, and to Dr. Anis LAOUITI for their kind help, guidance, support and encouragement throughout my study.

I am very grateful to Dr. Cédric ADJIH for providing technical advice, cooperating on both experiments and simulation and for his immensely helpful comments, criticisms and suggestions.

I want to thank Pr. Stéphane UBÉDA and Pr. Ken CHEN for accepting to review this document and for their helpful comments.

I also want to thank Pr. Guy PUJOLLE, Dr. Philippe JACQUET and Pr. Khaldoun ALAGHA for being members of my Ph.D. dissertation jury and evaluating my work.

I am grateful to all the friends and colleagues of HIPERCOM team and INRIA-Rocquencourt for creating a pleasant and enjoyable working environment.

The last word is for my family and friends who have always been encouraging me. This work is dedicated to them.

Contents

Chapter 1

Introduction to wireless networks

In contrast to wired networks where signals pass from one device to another through a finite set of existing physical cables, wireless data transmission occurs in the open air and is enabled within a given coverage area, allowing for the unrestricted movement of the "connected" device within a given coverage area. Within these coverage areas, information is transferred using electromagnetic waves, most commonly via radio and infrared signals.

Wireless networking technologies range from global voice and data networks, which allow users to establish wireless connections across long distances, to infrared light and radio frequency technologies that are optimized for short-range wireless connections. Devices commonly used for wireless networking include portable computers, desktop computers, hand-held computers, personal digital assistants (PDAs), cellular phones, pen-based computers, and pagers. Wireless technologies serve many practical purposes. For example, mobile users can use their cellular phone to access e-mail. Travelers with portable computers can connect to the Internet through base stations installed in airports, railway stations, and other public locations. At home, users can connect devices on their desktop to synchronize data and transfer files.

1.1 Wireless network standards

To lower costs, ensure interoperability, and promote the widespread adoption of wireless technologies, organizations such as the Institute of Electrical and Electronics Engineers (IEEE), Internet Engineering Task Force (IETF), European Telecommunications Standards Institute (ETSI), and the International Telecom-

munication Union (ITU) are participating in several major standardization efforts. For example, IEEE working groups are defining how information is transferred from one device to another (whether radio waves or infrared light is used, for example) and how and when a transmission medium should be used for communications.

Wireless network standards can be classified into different categories according to the channel access schemes they use.
The first type of standards fall in the category which uses a control based scheme for the channel access, for example TDMA (Time Division Multiple Access) or FDMA (Frequency Division Multiple Access). The second type of standards are those which use random access protocols like CSMA [29], or CSMA with collision avoidance [30][31]. We will briefly look to these two main types of standards in the following sections.

1.1.1 TDMA type networks

TDMA (or FDMA) type of networks uses a centralized control scheme for channel access. This scheme requires a central entity controlling the network which has access to all resources and which is distributing these resources fairly among network nodes according to their needs. In this scheme, the active nodes have some type of "resource reservation" either in time or in frequency, for a certain time. GSM (Global System for Mobile Communications) [13] is an example of the standards using TDMA scheme. It is an ETSI standard which is designed principally for voice traffic.

1.1.2 CSMA type networks

The second type of data network standards are those which use random access protocols. Wireless networks which use a random access scheme for accessing the common shared medium generally employ a CSMA (Carrier Sense Multiple Access) technique [29] or its variants such as CSMA with Collision Avoidance (CSMA/CA) [30][31]. In a random access protocol the control is completely distributed and there is no need to a central entity. All the nodes, sharing the same frequency band, contend for channel access and the winner of the contention process transmits its packets. At the end of the packet transmission, the contention process restarts among the nodes desiring to transmit data over the medium.

Two important, existing standards of CSMA type networks are:

. IEEE 802.11 standards. They are a network access technologies for providing connectivity between wireless stations and wired networking infrastructures. Some examples of these standards are given in Section 1.2.

. ETSI Standards. The main standard issued by the ETSI is HiperLAN (High Performance Radio LAN), which is a competitor of IEEE 802.11. It defines two kinds of networks:

  - HiperLAN type 1 standard [18] uses the 5.2 GHz band. It provides a maximum throughput of up to 23 Mbps. This radio network protocol performs dynamic routing at the MAC (Media Access Control) layer.
  - HiperLAN type 2 standard [19] uses the 5.47-5.725 GHz band which is reserved for it and offers a data rate up to 54 Mbps. It uses the base station topology as contrary to HIPERLAN type 1, which uses internal routing.

1.2 Wireless network types

As with wired networks, wireless networks can be classified into different types based on the distances over which data can be transmitted.

. **Wireless wide area networks (WWANs):** WWAN technologies enable users to establish wireless connections over remote public or private networks. These connections can be maintained over large geographical areas, such as cities or countries, through the use of multiple antenna sites or satellite systems maintained by wireless service providers. WWAN technologies include Global System for Mobile Communications (GSM) [13] [1], and Code Division Multiple Access (CDMA) [2].

. **Wireless metropolitan area networks (WMANs):** WMAN technologies enable users to establish wireless connections between multiple locations within a metropolitan area (for example, between multiple office buildings in a city or on a university campus), without the high cost of laying fiber or copper cabling and leasing lines. WMANs use either radio waves or infrared

[1] GSM is an open standard which is currently developed by the 3rd Generation Partnership Project (3GPP) [26], which is a collaboration agreement that was established in December 1998.

[2] A number of different terms are used to refer to CDMA implementations. The original standard was known as IS-95 [27], and after a couple of revisions, IS-95 was superseded by the IS-2000 [28] standard.

light to transmit data. Broadband wireless access networks, which provide users with high-speed access to the Internet, are in increasing demand. Although different technologies, such as the multichannel multipoint distribution service (MMDS) [14] and the local multipoint distribution services (LMDS) [15], are being used, the IEEE 802.16 working group for broadband wireless access standards is still developing specifications to standardize development of these technologies. Therefore, the IEEE 802.16e [16] (marketed as Mobile WIMAX: Worldwide Interoperability for Microwave Access) is an improvement on the modulation schemes stipulated in the original (fixed) WiMAX standard. It operates on frequencies from 10 to 66 GHz, and should ensure network coverage for several square Km.

. **Wireless local area networks (WLANs):** WLAN technologies enable users to establish wireless connections within a local area (for example, within a corporate or campus building, or in a public space, such as an airport). WLANs can be used in temporary offices or other spaces where the installation of extensive cabling would be prohibitive, or to supplement an existing LAN so that users can work at different locations within a building at different times. WLANs can operate in two different ways. In infrastructure WLANs, wireless stations (devices with radio network cards or external modems) connect to wireless access points that function as bridges between the stations and the existing network backbone. In peer-to-peer (ad hoc) WLANS, several users within a limited area, such as a conference room, can form a temporary network without using access points, if they do not require access to network resources.

  In 1997, IEEE approved the 802.11 standard [20] for WLANs, which specifies a data transfer rate of 1 to 2 megabits per second (Mbps). The 802.11 family currently includes six over-the-air modulation techniques that all use the same protocol. The most popular (and prolific) techniques are those defined by the b, a, and g amendments to the original standard. Under 802.11b [22], which was the first widely accepted wireless networking standard, data is transferred at a maximum rate of 11 Mbps over a 2.4 gigahertz (GHz) frequency band. It was followd by 802.11a [21] and 802.11g [23] standards. 802.11a standard specifies data transfer at a maximum rate of 54 Mbps over a 5 GHz frequency band, while 802.11g works in the 2.4 GHz band (like 802.11b) but operates at a maximum raw data rate of 54 Mbit/s, or about 24.7 Mbit/s net throughput like 802.11a.

  More details on 802.11 networks functioning can be found in [17].

. **Wireless personal area networks (WPANs):** WPAN technologies enable users to establish ad hoc, wireless communications for devices (such as

PDAs, cellular phones, or laptops) that are used within a personal operating space (POS). A POS is the space surrounding a person, up to a distance of 10 meters. Bluetooth [3] is an example of WPAN technologies. Bluetooth is a cable replacement technology that uses radio waves to transmit data to a distance of up to 10 meters. Technology development for Bluetooth is driven by the Bluetooth Special Interest Group (SIG), which published the Bluetooth version 1.0 specification in 1999 [24]. Alternatively, to connect devices at a very close range (1 meter or less), users can create infrared links.

To standardize the development of WPAN technologies, IEEE has established the 802.15 working group for WPANs. This working group is developing a WPAN standard [25], based on the Bluetooth version 1.0 specification. Key goals for this draft standard are low complexity, low power consumption, interoperability, and coexistence with 802.11 networks.

1.3 WLAN architectures

The 802.11 logical architecture contains several main components: station (STA), wireless access point (AP), independent basic service set (IBSS), basic service set (BSS), distribution system (DS), and extended service set (ESS). Some of the components of the 802.11 logical architecture map directly to hardware devices, such as STAs and wireless APs. The wireless AP functions as a bridge between the wireless STAs and the existing network backbone for network access.

A 802.11 wireless network can be structered to function in either IBSS mode or BSS mode.

1. **IBSS mode:** An IBSS is a wireless network, consisting of at least two STAs, used where no access to a DS is available. An IBSS is also sometimes referred to as a *peer to peer* or *ad hoc* wireless network. It allows nodes or stations to communicate directly (point-to-point) without the need for an AP.

2. **BSS mode:** A BSS is a wireless network, consisting of a single wireless AP supporting one or multiple wireless clients. A BSS is also sometimes referred to as an infrastructure wireless network. All STAs in a BSS communicate through the AP. The AP provides connectivity to the wired LAN and provides bridging functionality when one STA initiates communication to another STA or a node on the DS.

[3] http://www.bluetooth.org.

An ESS is a set of two or more wireless APs connected to the same wired network that defines a single logical network segment bounded by a router (also known as a subnet). The APs of multiple BSSs are interconnected by the DS (distribution system). This allows for mobility, because STAs can move from one BSS to another BSS. The distribution system (DS) is the logical component used to interconnect BSSs. The DS provides distribution services to allow for the roaming of STAs between BSSs.

3. **Ad hoc networks:** A wireless *ad hoc* network is a collection of two or more devices/nodes or terminals with wireless communications and networking capability that communicate with each other without the aid of any centralised administrator. In an ad hoc network each node functions as both a host and a router. Hence, packets to be exchanged between two nodes that are not in direct range with each other are forwarded by intermediate nodes. These messages are exchanged and relayed between nodes using a routing protocol [4]. More details on wireless ad hoc networks will be given in the following section.

1.4 Wireless Ad hoc Networks

A wireless ad hoc network is an autonomous network that consists of mobile nodes that communicate with each other over wireless links. This type of networks is suited for use in situations where a fixed infrastructure is not available, not trusted, too expensive or unreliable. A few examples include: a network of notebook computers or PDAs in a conference or campus setting, rescue operations, temporary headquarters, industry, ...

There are two major types of wireless ad hoc networks: Smart Sensor Networks and Mobile Ad Hoc Networks (MANET).

A Smart Sensor Network consists of a number of sensors spread across a geographical area. Each sensor has a wireless communication capability and sufficient intelligence for signal processing and networking of the data, however it has limited resources in terms of battery energy, bandwidth, memory, and computational power.

A Mobile Ad Hoc Network (MANET) is an independent collection of nodes or mobile users which are free to move about arbitrarily, and that communicate over wireless links. It is an independent system of mobile nodes that may operate in

[4] An ad hoc network must not be confused with a network in ad hoc mode. Nodes in a network with ad hoc mode do not relay packets.

isolation, or may have gateways to and interfere with a fixed network. Significant examples of MANETs include establishing communication for emergency or rescue operations, disaster relief efforts and military networks.
Due to the mobility of the nodes, the topology of the network may rapidly be changing, making it impossible to use conventional routing tables maintained at fixed points (routers). Instead, each node is required to determine the best route to a given destination node by itself. Hence, efficient routing protocols for mobile ad hoc networks are needed. The following section describes some of the existing routing protocols for mobile ad hoc networks.

1.5 Routing in mobile Ad Hoc networks

In the absence of a fixed infrastructure, nodes in a mobile ad hoc network have to cooperate in order to provide the necessary network functionality. Routing is one of the primary functions each node has to perform in order to enable connections between nodes that are not directly within each others send range. The development of efficient routing protocols is a non-trivial and challenging task because of the specific characteristics of a MANET environment:

- Due to node movements, the network topology may change randomly and rapidly at unpredicted times.
- The avalaible bandwidth is limited and can vary due to fading, noise, interference, ...
- Most mobile devices are battery powered, therefore energy consumption plays an important role.

Hence, there is a need for efficient routing protocols to allow the nodes to communicate.

Numerous ad hoc routing solutions have been proposed within the *MANET* working group in the *IETF*(The Internet Engineering Task Force). These ad hoc routing protocols can be categorised into three, namely, table-driven proactive protocols, on-demand-driven reactive/source initiated protocols and the hybrid protocols. However, only two proactive protocols (OLSR [55] and TBRPF [70]) and one reactive protocol (AODV [69]) have been promoted to experimental RFCs. The Dynamic Source Routing Protocol (DSR) [73] has also been choosed to be validated as experimental RFC, but right now it stills in the stage of an *IETF* draft.

1.5.1 Reactive routing protocols

On-demand-driven or the source initiated protocols create routes only when desired by source nodes. When a node requires a route to destination, it initiates route discovery process within the network. This process completes once one route is found or all possible route permutations are examined. Once a route is discovered and established, it is maintained by route maintenance procedure until either destination becomes inaccessible along every path from source or route is no longer desired. Following is an example of a routing protocol belonging to this category.

1.5.1.1 Ad-hoc On Demand Distance Vector (AODV) Protocol

This routing algorithm is based on the DSDV (Destination Sequenced Distance Vector) [71] algorithm which is one of the first routing protocols proposed for ad hoc networks. DSDV, it self, is based on the distance-vector protocol RIP (Routing Information Protocol) [72], which is an internet interior routing protocol used for wired networks. The major improvement of AODV is the minimising of the number of required broadcasts by creating routes on an on-demand basis, as opposed to maintaining a complete list of routes to all destinations as in DSDV. Hence, a path discovery is initiated only when a route to a destination is needed by a source node.
AODV protocol uses 3 types of messages:

- Route Request (RREQ) message
- Route Reply (RREP) message
- Route Error (RERR) message

When a source node desires to communicate with a destination node for which it does not already have a route, it broadcasts a route request (RREQ) packet across the network and the information packets are queued. Nodes receiving this packet update their information for the source node and set up backward pointers to the source node in their routing tables. That is the "*reverse path setup*" operation. These receiving nodes forward the RREQ packet only if an entry for the destination node is not found in their routing tables (Figure 1.1).

In addition to the source node's IP address, current sequence number, and RREQ ID[5], the RREQ message also contains the most recent sequence number

[5]A sequence number uniquely identifying the particular RREQ when taken in conjunction with the originating node's IP address.

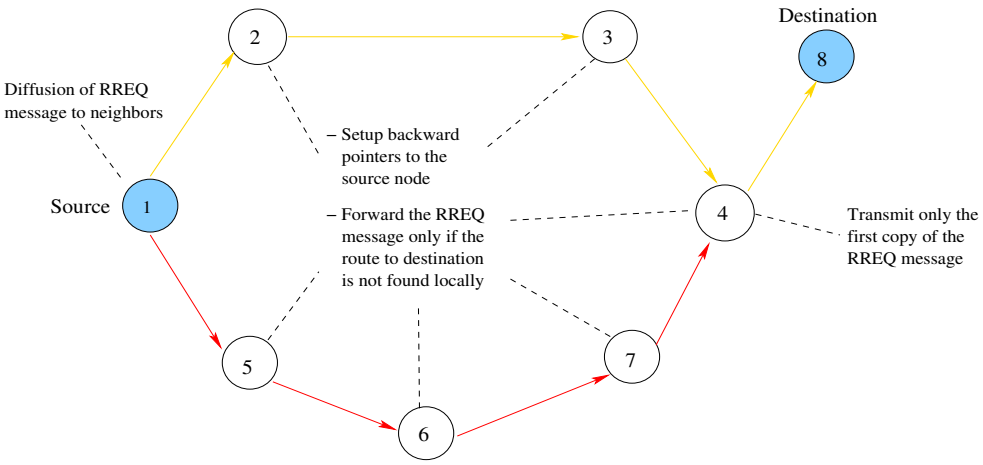


Figure 1.1: Propagation of a RREQ message

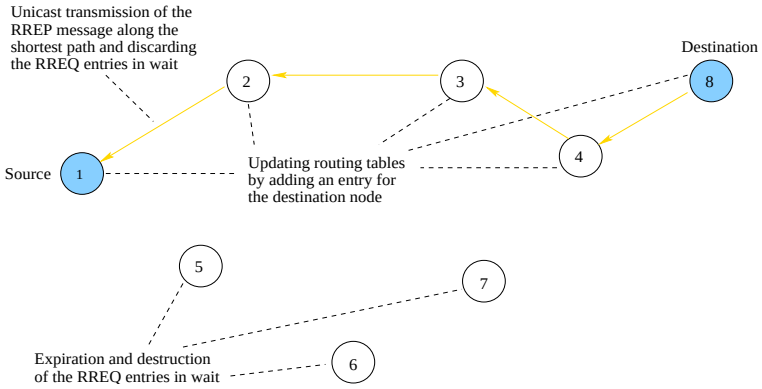


Figure 1.2: Propagation of a RREP message

for the destination of which the source node is aware. A node receiving the RREQ may send a RREP message if it is either the destination or if it has a route to the destination with the corresponding sequence number greater than or equal to that contained in the RREQ. If it is the case, the receiving node unicasts a RREP back to the source node along the "*reverse path*". Nodes keep track of the RREQ's source IP address and RREQ ID. If they receive a RREQ which they have already processed, they discard the RREQ and do not forward it. As the RREP message propagates back to the source node, intermediate nodes set up forward pointers to the destination. The entries for the RREQ's source node in the intermediate receiving nodes that didn't send a RREP packet to that source node after a certain period, will time out and hence be deleted. These intermediate nodes consider

that either the route to the destination doesn't exist, or that they are not on the shortest path between the source and the destination (Figure 1.2).

Once the source node receives the RREP, it may begin to forward data packets to the destination. If the source node later receives a RREP containing a greater sequence number or contains the same sequence number with a smaller hopcount, it may update its routing information for that destination and begin using the better route. As long as the route remains active, it will continue to be maintained. A route is considered active as long as there are data packets periodically travelling from the source to the destination along that path. Once the source stops sending data packets, the links will time out and eventually be deleted from the intermediate nodes' routing tables. If a link break occurs while the route is active, the node upstream of the break propagates a route error (RERR) message to the source node to inform it of the now unreachable destination(s). After receiving the RERR message, if the source node still desires the route, it can reinitiate route discovery.

1.5.2 Proactive routing protocols

In proactive routing protocols, nodes continuously search for routing information within a network, so that when a route is needed, the route is already known. The OLSR protocol developed in project *HIPERCOM* at *INRIA*(Institut National de Recherche en Informatique et Automatique) belongs to this category.

1.5.2.1 Optimized Link State Routing (OLSR) Protocol

This section describes the main features of the *OLSR* (Optimized Link State Routing) protocol [55]. *OLSR* is a proactive routing protocol for mobile ad hoc networks which inherits from the *link state* functioning of the Interior Gateway Protocol [6] *OSPF* (*O*pen *S*hortest *P*ath *F*irst) [56]. It has the advantage of having routes immediately available when needed due to its proactive nature. OLSR is an optimization of a pure link state routing protocol and it is based on the concept of *multipoint relays* (*MPRs*) [57] [58], to diffuse its control traffic in the network. *MPRs* are selected nodes which forward broadcast messages during the flooding process. The use of *MPRs* significantly reduces the control traffic overhead as compared to a classical flooding mechanism, where every node retransmits each message when it receives the first copy of the message. Indeed only *multipoint*

[6]This means that it distributes routing information between routers belonging to a single Autonomous System.

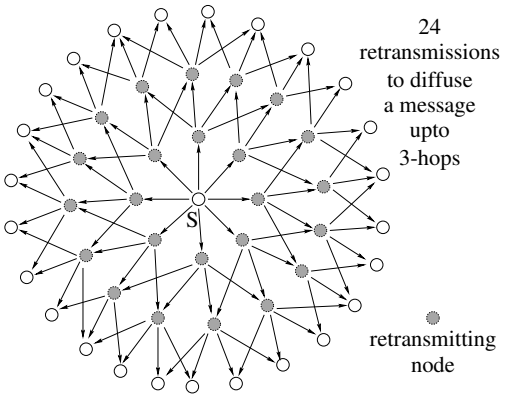


Figure 1.3: Diffusion of broadcast message using pure flooding

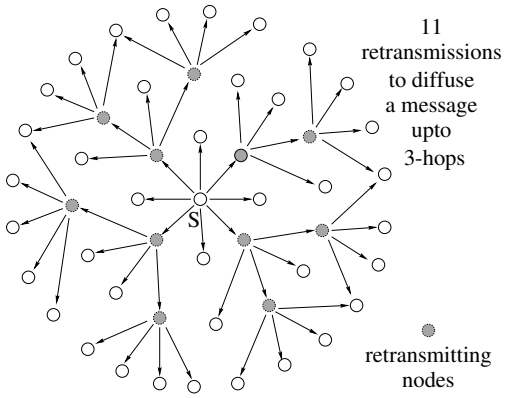


Figure 1.4: Diffusion of broadcast message using MPR flooding

relays forward control messages. Figure 1.4 shows an example where a broadcast message of node S is diffused in the network using the multipoint relays. In this case, it tooks only 11 retransmissions for the message to reach up to 3-hops, as compared to 49 retransmissions in Figure 1.3. In OLSR, link state information is generated only by nodes elected as MPRs. Thus, a second optimization is achieved by minimizing the number of control messages flooded in the network. As a third optimization, OLSR requires only partial link state information to be flooded in the network in order to provide shortest path routes. The minimal set of link state information required is, that all nodes, selected as MPRs, MUST declare the links to their MPR selectors. Hence, using *multipoint relays* reduces the size of the control messages: rather than declaring all links, a node declares only the set of links with its neighbors that are its "*multipoint relay selectors*".

Now we will present the OLSR packet format and then we will detail the two main functionalities of *OLSR*, Neighbor Discovery and Topology Dissemination. Finally we will present the heuristic used for the selection of MPRs since it directly affects our autoconfiguration algorithms proposed later for OLSR.

- **OLSR packet format**

  OLSR communicates using a unified packet format for all data related to the protocol. This provides an easy way of piggybacking different "types" of information into a single transmission, and thus for a given implementation to optimize towards utilizing the maximal frame-size, provided by the network. Therefore, each OLSR packet encapsulates one or more messages and the messages share a common header format. These packets are embedded in UDP datagrams for transmission over the network. Figure 1.5 shows the basic layout of any packet in OLSR (omitting IP and UDP headers).

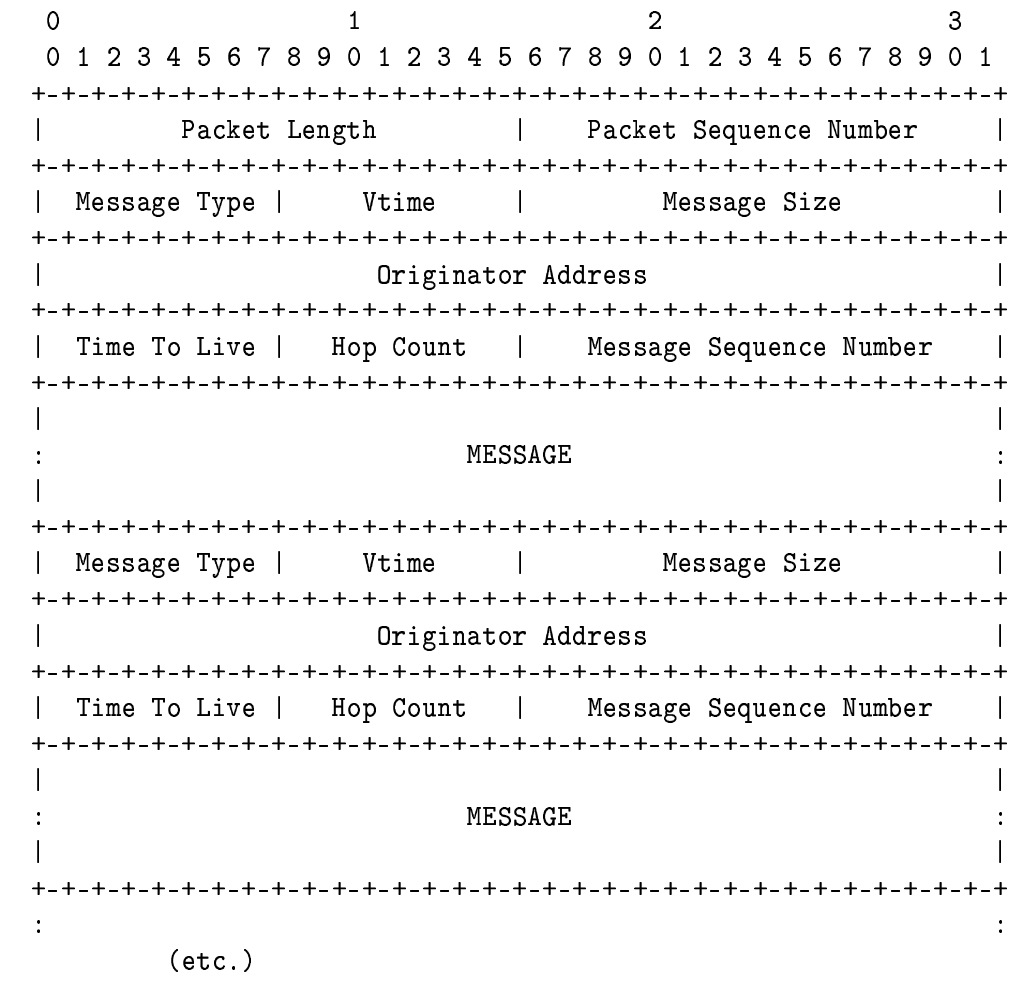


Figure 1.5: OLSR Packet Format

The *Message Type* field, in the Message Header, indicates which type of message is to be found in the *MESSAGE* part. The required message types for the OLSR functioning are:

- *HELLO* messages: performing the task of link sensing, neighbor detection and MPR signaling
- *TC* messages: performing the task of topology declaration (advertisement of link states)
- *MID* messages: performing the task of declaring the presence of multiple interfaces on a node

- *HNA* messages: nodes which may act as "gateways" to associated hosts and networks, periodically generate a *HNA (Host and Network Association)* message, containing pairs of (network address, netmask) corresponding to the connected hosts and networks. The networks and associated hosts are added to the routing table and they have the same next hop as the one to reach the appropriate "gateway".

- **Neighbor Discovery**

  Each node must detect the neighbor nodes with which it has a direct link. For this, each node periodically broadcasts *Hello* messages, containing the list of neighbors known to the node and their link status. The link status can be either *symmetric* (if communication is possible in both directions), *asymmetric* (if communication is only possible in one direction), *multipoint relay* (if the link is symmetric and the sender of the *Hello* message has selected this node as a *multipoint relay*), or *lost* (if the link has been lost). The *Hello* messages are received by all 1-hop neighbors, but are not forwarded. They are broadcasted once per refreshing period called the "*HELLO_INTERVAL*" (the default value is 2 seconds). Thus, *Hello* messages enable each node to discover its 1-hop neighbors, as well as its 2-hop neighbors. This neighborhood and 2-hop neighborhood information has an associated holding time, the - "*NEIGHBOR_HOLD_TIME*", after which it is no longer valid.
  On the basis of this information, each node s independently selects its own set of *multipoint relays* among its 1-hop neighbors in such a way all 2-hop neighbors of s have *symmetric* links with $MPR(s)$. This means that the *multipoint relays* cover (in terms of radio range) all 2-hop neighbors (Figure 1.4). The *MPR* set is computed whenever a change in the 1-hop or 2-hop neighborhood is detected. In addition, each node s maintains its "*MPR selector set*". This set contains the nodes which have selected s as a *multipoint relay*. Node s only forwards broadcast messages received from one of its *MPR selectors*.

- **Topology Dissemination**

  Each node of the network maintains topological information about the network obtained by means of *TC (Topology control)* messages. Each node m selected as a *multipoint relay*, broadcasts a *TC* message at least every "*TC_INTERVAL*" (the default value is 6 seconds). The *TC* message originated from node m declares the *MPR selectors* of m. If a change occurs in the *MPR selector* set, the next *TC* can be sent earlier. (e.g. after some pre-specified minimum interval). The *TC* messages are flooded to all nodes in the network and take advantage of *MPRs* to reduce the number of retransmissions. Thus, a node is reachable either directly or via its *MPRs*.

This topological information collected in each node has an associated holding time "*TOP_HOLD_TIME*", after which it is no longer valid.

The neighbor information and the topology information are refreshed periodically, and they enable each node to compute the routes to all known destinations. These routes are computed with Dijkstra's shortest path algorithm [78]. Hence, they are optimal as concerns the number of hops. Moreover, for any route, any intermediate node on this route is a *multipoint relay* of the next node. The routing table is computed whenever there is a change in neighborhood or topology information.

- **Heuristic for selection of MPRs**

  It was proven in [57] and [58] that the problem of computing a set of MPRs of size less than an integer k is NP-hard. However, an heuristic for a simple selection of MPRs was proposed in [57]. It constructs a MPR-set that enables a node to reach any node in its symmetrical strict 2-hop neighborhood.

  To select the multipoint relays for a node x, the following terminology will be used in describing the heuristic:

  - *MPR(x):* the multipoint relay set of node x which is running this algorithm;
  - *N(x):* the 1-hop neighbor set of node x containing only symmetric (bi-directional link) neighbors;
  - *N2(x):* the 2-hop neighbor set of node x containing only symmetric neighbors of nodes in *N(x)*. The 2-hop neighbor set *N2(x)* of node x does not contain any 1-hop neighbor of node x;
  - *D(x,y):* the degree of 1-hop neighbor node y (where y is a member of of *N(x)*), is defined as the number of bi-directional link 1-hop neighbors of node y EXCLUDING the node x and all the bi-directional link 1-hop neighbors of node x which exist also in *N(y)*, i.e.,
    $D(x,y) = N(y) - x - (N(x) \cap N(y))$

  The proposed heuristic to calculate *MPR(x)* is as follows:

  1. Start with an empty multipoint relay set *MPR(x)*;
  2. Calculate *D(x,y)*, $\forall$ nodes $y \in N(x)$;
  3. First, select those 1-hop neighbor nodes in *N(x)* as the multipoint relays which provide the "only path" to reach some nodes in *N2(x)*, and add these 1-hop neighbor nodes to the multipoint relay set *MPR(x)*;

4. While there still exists some nodes in *N2(x)* that are not covered by the multipoint relay set *MPR(x)*:
   (a) For each node in *N(x)* which is not in *MPR(x)*, calculate the number of nodes that are reachable through it among the nodes in *N2(x)* and which are not yet covered by *MPR(x)*;
   (b) Select that node of *N(x)* as a MPR which reaches the maximum number of uncovered nodes in *N2(x)*.
   (c) In case of a tie in the above step, select that node as MPR whose *D(x,y)* is greater;
5. To optimize, remove each node in *MPR(x)*, one at a time, and check if *MPR(x)* still covers all nodes in *N2(x)*.

In a multi-interface OLSR node, the heuristic must be applied per interface and the MPR set for the node is the union of the MPR sets found for each interface.

Notice here that any address duplications in the 2-hop neighborhood of node x can impair the correct functioning of the heuristic.

1.5.3 Hybrid routing protocols

Zone Routing Protocol (ZRP) [74] framework is a hybrid routing protocol suitable for mobile ad hoc networks with large network spans and diverse mobility patterns. ZRP divides its network into different zones which are the nodes' local neighborhoods. The size of a zone is not determined by geographical measurement, as one might expect, but is given by a radius of length ρ, where ρ is the number of hops to the perimeter of the zone. Hence, the routing zone comprises a few mobile ad hoc nodes within one, two or more hops away from where the central node is formed.
Each node has its own zone and may be within multiple overlapping zones. Each zone may be of a different size.

Figure 1.6 shows an example of routing zone with $\rho = 2$. Note that in this example node A has multiple routes to node F, including one that has a hopcount of $hc > \rho$. Since it also has a route with $hc \leq \rho$, F still belongs to A's zone. Node K is out of A's zone. The nodes on the perimeter of the zone (i.e. with a hopcount $hc = \rho$) are referred to as peripheral nodes, nodes with $hc < \rho$ are interior nodes.

ZRP combines the advantages from proactive routing (for example OLSR) to proactively maintain routes within its routing zones (Intrazone Routing Protocol(IARP) [76]), and the advantages from reactive routing (for example AODV)

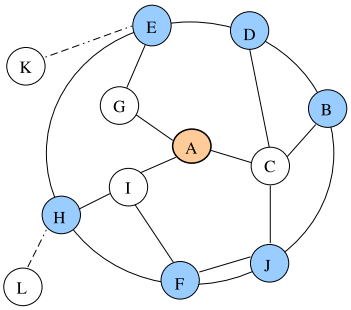


Figure 1.6: Routing Zone of node A with $\rho = 2$

for communication between these zones (Interzone Routing Protocol (IERP) [75]). The Bordercast Resolution Protocol (BRP) [77] is used in the ZRP to direct the route requests initiated by the global reactive IERP to the peripheral nodes. In fact, rather than broadcasting a route query from neighbor to neighbor, bordercasting allows the query to be directed outward, toward regions of the network (specifically, toward peripheral nodes) that have not yet been "covered" by the query. Note that a covered node is one that belongs to the routing zone of a node that has received a route query.

1.6 Other technical problems in MANET networks

Two of the main challenges in design of mobile ad hoc networks are their vulnerability to security attacks and to address duplications since, by definition, an ad hoc network is an open system and therefore, it is extremely vulnerable to malicious node presence and to certain types of attacks that can occur. Moreover, due to frequent network partitioning and merging, address duplications may happen.

A node in an IP-based network is configured with an IP address, a netmask and a default gateway (the node to which packets for destinations not having an explicit entry in the routing table are sent). These network parameters can be assigned manually to the node, however this procedure becomes impractical in a large scale MANET where hundreds of nodes constitute the network. Therefore, an automatic configuration of the nodes in a mobile ad hoc network is necessary. Dynamic configuration in a wired network is accomplished using the Dynamic Host Configuration Protocol (DHCP). However, this requires the presence of a centralized DHCP server which maintains the configuration information of all nodes in the network. Such a solution cannot be employed in MANETs due to the unavailability of any centralized DHCP server, and the mobility of nodes which

can lead to frequent and unpredictable topology changes(the node assuring the role of DHCP server may be out of the radio range of the other nodes). Moreover, due to frequent network partitioning and merging, a new node joining one partition can be assigned an IP address belonging to the other partition, leading to address duplication if these partitions merge again later. In addition, when a merge of two or more separately configured MANETs occurs, address duplications may happen. Hence, there is a need for an efficient distributed dynamic host configuration mechanism to configure nodes in a MANET and to handle the problem of address duplications.

The second core issue with mobile ad hoc networks is security, because all network services are configured on the fly. For security in fixed networks, the mechanism of maintaining a list of authenticated users is followed. The server checks the available list and authenticated password whenever a network access attempt is made. If the user information matches, access is granted. This client-server structure is not applicable with mobile ad hoc networks, as it operate without a fixed infrastructure. Furthermore, because of the mobility of nodes in mobile ad hoc networks, topology changes and link breakage happen quite frequently. Therefore, a dynamic security solutions are needed. In other words, any proposed solution should be able to cope with the ad hoc network environment.
Malicious or misbehaving nodes can create hostile attacks. These kinds of attacks can seriously damage basic security aspects, such as integrity, confidentiality, and privacy of the node. One way to secure further communication after nodes authenticate each other and establish a shared secret session key, is the use of symmetric key cryptography. However, the use of such schemes is a non-trivial problem in mobile ad hoc networks. Hence, efficient secret key management services for mobile ad hoc networks are needed.

This thesis will therefore address on one hand designing efficient autoconfiguration schemes for ad hoc networks running OLSR and for both IP protocols, IPv4 and IPv6, and on the other hand designing solutions to guarantee the integrity, confidentiality, and privacy of an ad hoc network.

1.7 Organization of the thesis and contributions

This document is organized in three parts. The first part (Chapters 2 and 3) describes the issue of adapting OLSR (Optimized Link State Routing Protocol) to IPv6, with a particular focus on a specific IPv6 functionality: *autoconfiguration*. The second part (Chapters 4 and 5) describes some solutions for OLSR autoconfiguration, mainly the improvements needed for the classical MPR flood-

ing mechanism to ensure the propagation of control messages to the whole network even in the presence of IP address duplications. And finally, the third part (Chapter 6) introduces a dynamic group key agreement solution for establishing and maintaining a session key in ad hoc networks for the purpose of cryptography.

Chapters contents and the contributions of this thesis are described in the following:

- Chapter 2 is dedicated to the design of IPv6 protocol. I start with presenting the IPv6 header format and its improved option mechanism over IPv4. Then I present the main features of IPv6 namely, IPv6 addressing schemes, IPv6 neighbor discovery protocol, IPv6 stateless address autoconfiguration, and finally, for coexistence to occur between IPv4 infrastructures and IPv6 infrastructures, I present the IPv6 transition technologies which allow the IPv4 and IPv6 nodes to coexist.

  Advanced readers may skip this chapter and start directly with Chapter 3.

- In Chapter 3, I describe the changes needed to adapt OLSR protocol to IPv6. Those changes include addressing issues, packet diffusion in OLSR IPv6 networks, and essentially IPv6 address autoconfiguration adapted to OLSR. That was published in [1].

- The first section in Chapter 4 is dedicated to address autoconfiguration in ad hoc networks. First, I highlight some scenarios where address duplications may occur. Then, I present some autoconfiguration protocols for ad hoc networks which have been proposed in the IETF or published in academic papers. In the sections after, I describe in detail the autoconfiguration mechanism we propose for OLSR protocol and I discuss the formal proof of correctness of this solution. The main contributions are the new rules added to the OLSR MPR flooding algorithm for the purpose of duplicate address detection. We propose two autoconfiguration schemes:

  (a) An autoconfiguration and duplicate address detection algorithm for a single interface OLSR network. That was published in [3] and well detailed with some simulation results in [6] and [4].

  (b) An autoconfiguration and duplicate address detection algorithm for a multi-interface OLSR network. That was published in [7].

- In Chapter 5, I present the implementation details of our autoconfiguration solution for OLSR and I evaluate its performance on the basis of the simulation results. To be published in [8].

. In Chapter 6, I present a Dynamic Group Key Agreement protocol we propose for ad hoc networks. This protocol allows to establish and maintain a session key in these dynamic networks. It is based on a cryptographic scheme that was already published by *Raghav et al* in [89]. In [9] we have added some mechanisms to adapt this scheme to the context of ad hoc networks.

At the end, I summurize this thesis and give some directions for the future work.

Part I

OLSR for IPv6 Networks

Internet Protocol Version 4 is the Internet protocol that is predominantly deployed and extensively used throughout the world, although there are some questions about its capability to serve the Internet community much longer. IPv4 was finished in the 1970s and has started to show its age. The main issue surrounding IPv4 is the lack of addresses, because many experts believe that we are nearly out of the four billion addresses available in IPv4. Although this seems like a very large number of addresses, multiple large blocks are given to government agencies and large organizations. IPv6 could be the solution to this problem, but it is still not fully developed and is not a standard yet. Hundreds of RFCs have been written and have detailed some major areas, including expanded addressing, simplified header format, flow labeling, authentication, and privacy. Also a number of books are now being published that cover in detail this emerging standard.

Ideally, IPv6 is used in a pure environment, that is, an environment where IPv6 is the exclusive Internet protocol used between computers. Currently, however, pure IPv6 transmissions are attainable only with routers and computers that support IPv6. As IPv6 supplants IPv4, pure IPv6 across the Internet will become more prevalent and will eventually replace IPv4. Until that occurs, some transition technologies may be used to bridge the technological gap between IPv4 and IPv6. In addition to describing the transition technologies between IPv4 and IPv6, the following chapter describes how IPv6 relates to other networking protocols, which functions IPv6 performs, and how IPv6 addresses are structured and assigned.

Chapter 2

The Design of IPv6

IPv6 has not officially become a standard, therefore it is very possible that the information I will present here will change as we move closer to IPv6 as a standard. The changes from IPv4 to IPv6 fall primarily into the following categories:

. **Expanded Addressing Capabilities :** IPv6 increases the IP address size from 32 bits to 128 bits, to support more levels of addressing hierarchy, a much greater number of addressable nodes, and simpler auto-configuration of addresses.

. **Header Format Simplification :** Some IPv4 header fields have been dropped or made optional, to reduce the common-case processing cost of packet handling and to limit the bandwidth cost of the IPv6 header.

. **Improved Support for Extensions and Options :** Changes in the way IP header options are encoded allows for more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future.

. **Flow Labeling Capability :** A new capability is added to enable the labeling of packets belonging to particular traffic "flows" for which the sender requests special handling, such as non-default quality of service or "real-time" service.

. **Authentication and Privacy Capabilities :** Extensions to support authentication, data integrity, and (optional) data confidentiality are specified for IPv6.

2.1 The IPv6 Header Format [33][34][35]

The IPv6 header is composed of a 64-bit header, followed by two 128-bit IPv6 addresses for source and destination, for a total length of 40 bytes. See Figure 2.1.

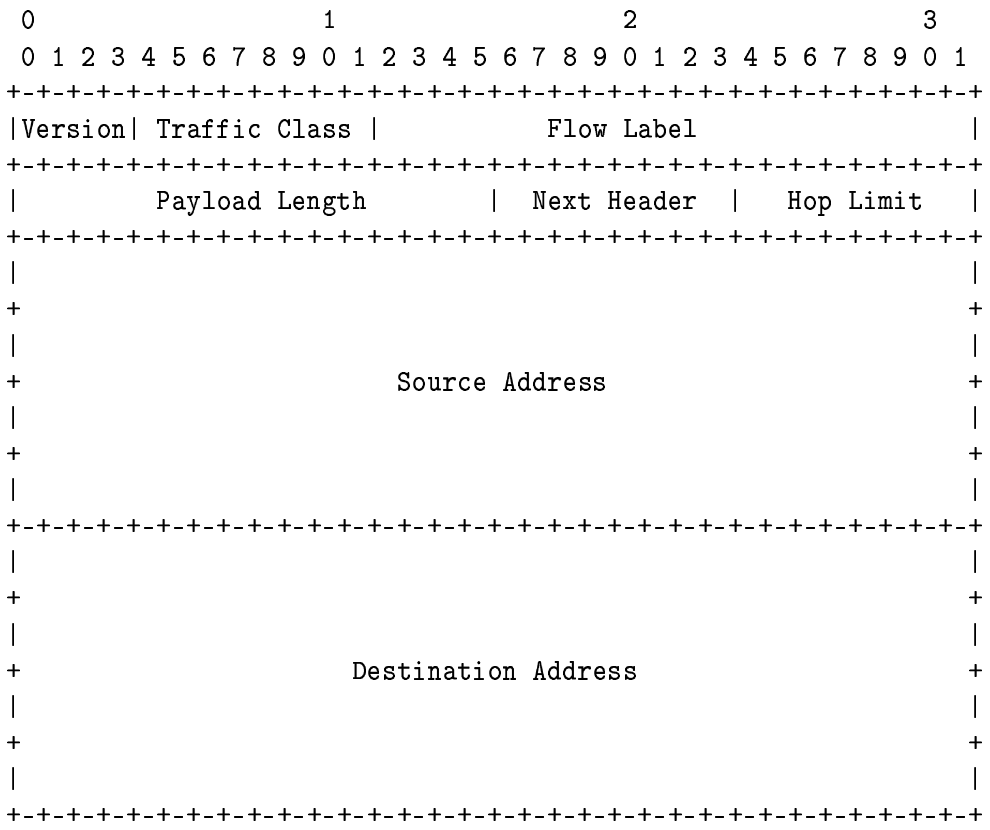


Figure 2.1: IPv6 Header Format

The initial 64 bits are composed of the following :

. **Version field (4 bits) :** Version number = 6.

. **Traffic Class (8 bits) :** The Traffic Class field in the IPv6 header is available for use by originating nodes and/or forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets.

. **Flow Label (20 bits) :** The Flow Label field in the IPv6 header may be used by a source to label sequences of packets for which it requests special

handling by the IPv6 routers, such as non-default quality of service or "real-time" service. Hosts or routers that do not support the functions of the Flow Label field are required to set the field to zero when originating a packet, pass the field on unchanged when forwarding a packet, and ignore the field when receiving a packet.

. **Payload Length (16 bits) :** Length of the IPv6 payload, i.e., the rest of the packet following this IPv6 header, in octets. Note that any extension headers (Section 2.1.2) present are considered part of the payload, i.e., included in the length count. This field is set to zero when large packets length cannot be encoded on 16 bits.

. **Next Header (8 bits) :** Identifies the type of header immediately following the IPv6 header. An IPv6 packet may carry zero, one, or more extension headers, each identified by the Next Header field of the preceding header (Section 2.1.2).

. **Hop Limit (8 bits) :** Decremented by 1 by each node that forwards the packet. The packet is discarded if Hop Limit is decremented to zero.

The IPv6 header is in fact much simpler than that of IPv4 header (Figure 2.2). The IPv6 header counts only six fields and two addresses, while the IPv4 header had 10 fixed header fields, two addresses, and some options [32].

2.1.1 From Options to Extension Headers

The IPv4 header had room for options, allowing special-case treatment of some packets(Figure 2.2). The original specification included codes for security options, source routing, and recording or timestamping.

IPv6 includes an improved option mechanism over IPv4. IPv6 options are placed in separate extension headers that are located between the IPv6 header and the transport-layer header in a packet. Each header is identified by a header type and carries the header type of the following header in the chain, or that of the payload in the case of the last extension (Figure 2.3). We observe that the IPv6 next header field may contain either the type of an extension header or the protocol type of the payload, for example, TCP or UDP.

Most IPv6 extension headers are not examined or processed by any router along a packet's delivery path until it arrives at its final destination. This facilitates a major improvement in router performance for packets containing options. In IPv4 the presence of any options requires the router to examine all options.

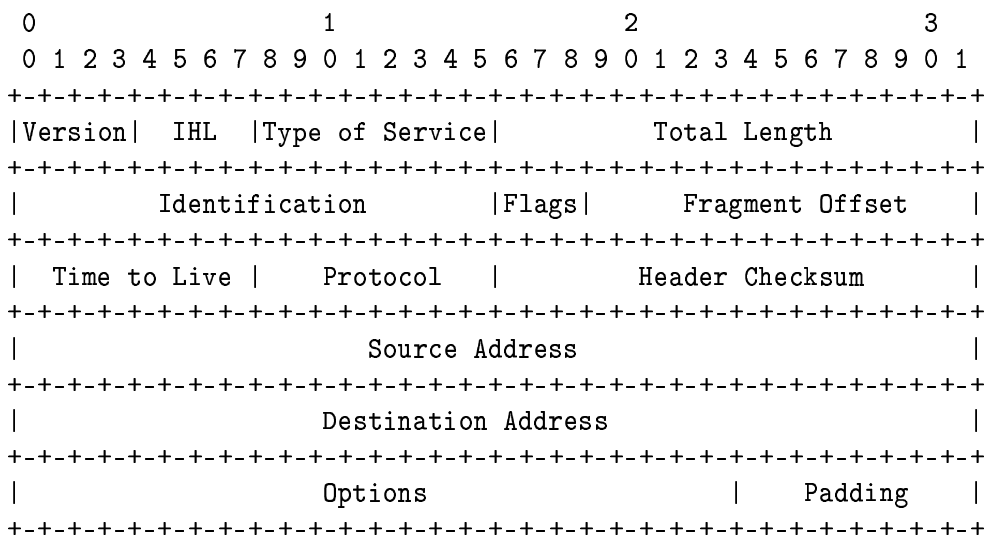


Figure 2.2: IPv4 Header Format

The other improvement is that unlike IPv4 options, IPv6 extension headers can be of arbitrary length and the total amount of options carried in a packet is not limited to 40 bytes. This feature plus the manner in which they are processed, permits IPv6 options to be used for functions which were not practical in IPv4. A good example of this is the IPv6 Authentication and Security Encapsulation options.

2.1.2 IPv6 Extension Headers

The IPv6 extension headers which are currently defined are :

. **Hop-by-hop options header :** The Hop-by-Hop Options header is used to carry optional information that must be examined by every node along a packet's delivery path. The Hop-by-Hop Options header is identified by a Next Header value of 0 in the IPv6 header. A null next header value in the IPv6 header implies the presence of the hop-by-hop option header that shall be processed even if the destination address is not one of the local node addresses. Four options are included in this category. The *Pad1 option* is used to insert a single octet of padding into the Options area of a header for 64-bit alignment, while the *PadN option* is used to insert two or more octets of padding. The *Jumbo Payload* option is used to indicate the length

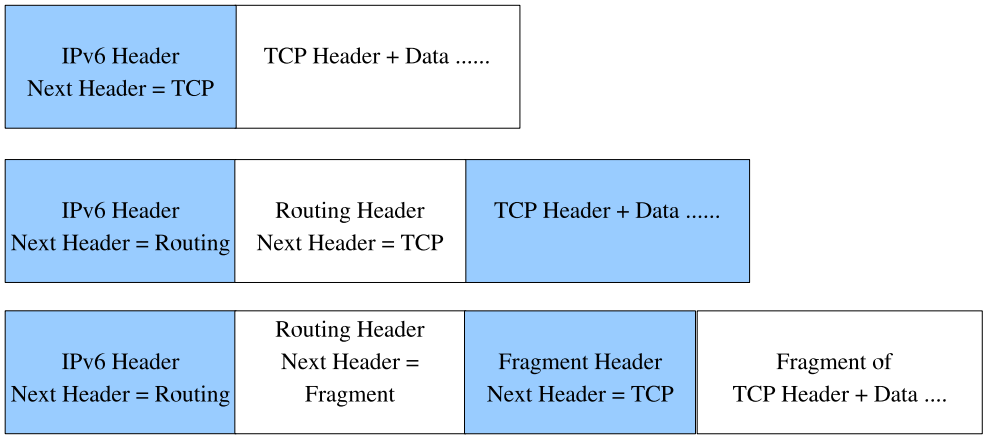


Figure 2.3: Examples of chains of headers

of the packet when the payload portion is longer than 65,535 octets (this option is employed when the IPv6 Payload Length field is set to zero). The *Router Alert* option is used to warn routers that the packet contains specific information that should be looked at closely.

. **Destination options header :** The Destination Options header is used to carry optional information that has to be examined only by a packet's destination node(s). The only destination options defined in the current specification are *Pad1* and *PadN*, as described above.

. **Routing header :** The routing header plays the same role as the source routing option of IPv4. This header essentially carries a list of intermediate addresses through which the packet shall be relayed, a source route that may be either strict[1] or loose[2]. In IPv6, routers will only look at the routing header if they recognize one of their own addresses in the destination field of the main header. If it is the case and if the received packet contains a source routing extension, the receiving router replace its address in the destination field with the address of the next router in the list of intermediate routers. Finally it forwards the packet. Intermediate routers that are not explicitly mentioned in the source route will forward the packet without any

[1]In **strict** source routing, the sender specifies the exact route the packet must take, and the next router in the list of intermediate routers must be a directly accessible neighbor of the receiving router. This is virtually never used.

[2]In **loose** source routing, the receiving router can use its routing table to join the next router in the list of intermediate routers.

additional processing. This should result in better performance.

. **Fragment header :** The Fragment header is used by an IPv6 source to send packets that are larger than the maximum transmission unit (MTU) on the path to the destination. Unlike IPv4, where fragmentation information is carried in every packet header, IPv6 only carries fragmentation/reassembly information in those packets that are fragmented. Fragmentation in IPv6 is performed only by the source and not by the routers along a packet's path. All IPv6 hosts and routers must support an MTU of 576 octets; it is recommended that path MTU discovery procedures [40] be invoked to discover, and take advantage of, those paths with a larger MTU.

. **Authentication header [43] :** IPv6 provides two mechanisms for addressing security problems. In the first mechanism it has an extension header called "Authentication Header (AH)", which provides authentication and integrity (without confidentiality) to IPv6 datagrams. The receiver of a packet can be sure who sent it, unlike in IPv4 in which no guarantee is present. The payload of the authenticated packet is sent unencrypted. While the extension is algorithm- independent and will support many different authentication techniques, the use of *keyed MD5* (derived from the Message Digest 5 algorithm [41]) is proposed to help ensure interoperability within the worldwide Internet. This can be used to eliminate a significant class of network attacks, including address spoofing. It will also protect users against the stealing of connections.

. **Encapsulating Security Payload header [42] :** The second mechanism to address security problems is the "Encapsulating Security Payload header (ESP)". This mechanism provides integrity and confidentiality to IPv6 datagrams. It is simpler than some similar security protocols such as *SP3D* and *ISO NLSP* but remains flexible and algorithm-independent. To achieve interoperability within the global Internet, the Cipher Block Chaining mode of the Data Encryption Standard (DES-CBS) is suggested by the specification to be used as the default algorithm for use with the IPv6 Encapsulating Security Payload header. As for *MD5* in the case of authentication, *DES-CBS* is only a default algorithm. Other algorithms can be selected when the security association is being established.

2.2 IPv6 Addressing Scheme Overview

IPv6 quadruples the number of network address bits from 32 bits (in IPv4) to 128 bits or approximately 3.4 x 10^{38} addressable nodes, which provides more than enough globally unique IP addresses for every network device on the planet. In many situations, IPv6 addresses are composed of two logical parts: a 64-bit network prefix, and a 64-bit host-addressing part, which is often automatically generated from the interface MAC address. IPv6 address consists of 8 groups of 16-bit hexadecimal values separated by colons (:) as shown in Figure 2.4. HHHH

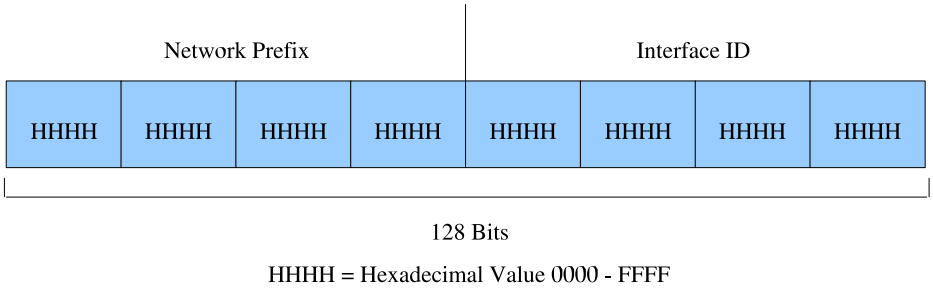


Figure 2.4: The format of IPv6 Addtress

is a 16-bit hexadecimal value, and H is a 4-bit hexadecimal value. The following is an example of an IPv6 address: 2001:0000:0000:0200:002D:D0FF:FE48:4672 Note that the sample IPv6 address includes hexadecimal fields of zeros. To make the address less cumbersome, we can do the following:

- We can omit the leading zeros; for example, 2001:0:0:200:2D:D0FF:FE48:4672.
- We can compress the successive groups of zeros at the beginning, middle, or end of an IPv6 address to two colons (::) once per address; for example, 2001::200:2D:D0FF:FE48:4672.

When specifying an IPv6 address in a command syntax:

- We can use the two colons (::) once in the address to represent the longest successive hexadecimal fields of zeros.
- The hexadecimal letters in the IPv6 addresses are not case-sensitive.

As shown in Figure 2.4, the IPv6 network prefix is composed of the left-most bits of the address. As with an IPv4 address, we can specify the IPv6 prefix using the <prefix>/<prefix-length> format, where the following applies:

- The <prefix> parameter is specified as 16-bit hexadecimal values separated by a colon.
- The <prefix-length> parameter is specified as a decimal value that indicates the left-most bits of the IPv6 address.

The following is an example of an IPv6 prefix: 2001:FF08:49EA:D088::/64

2.2.1 Types of IPv6 addresses

IPv6 supports three types of addresses:

1. **Unicast addresses:** a unicast address identifies a single interface within the scope of the type of unicast address. With the appropriate unicast routing topology, packets addressed to a unicast address are delivered to a single interface. To accommodate load-balancing systems, RFC 3513[84] [3] allows multiple interfaces to use the same address as long as they appear as a single interface to the IPv6 implementation on the host.
   Unicast IPv6 addresses fall into one of five types:

   Global Unicast addresses : Global unicast addresses are equivalent to public IPv4 addresses. They are globally routable and reachable on the IPv6 Internet. Unlike the current IPv4-based Internet, which is a mixture of both flat and hierarchical routing, the IPv6-based Internet has been designed from its foundation to support efficient, hierarchical addressing and routing. The scope (that is, the region of the IPv6 internetwork over which the address is unique) of a global unicast address is the entire IPv6 Internet. Figure 2.5 shows the structure of a global unicast address as defined in RFC 3587 [85].

   . The three high-order bits are set to 001. The address prefix for currently assigned global addresses is 2000::/3.

   . The Global Routing Prefix indicates the global routing prefix for a specific site of an organization. The combination of the three fixed bits and the 45-bit Global Routing Prefix creates a 48-bit site

[3] Obsoleted by RFC 4291[45].

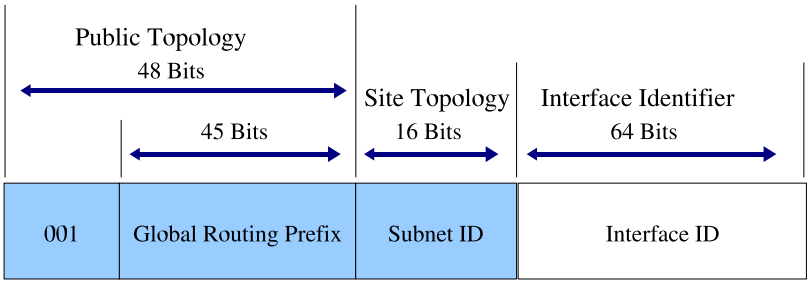


Figure 2.5: IPv6 global unicast addresses structure

prefix, which is assigned to an individual site of an organization. After this prefix is assigned, routers on the IPv6 Internet forward IPv6 traffic matching the 48-bit prefix to the routers of the site of the organization.

- The Subnet ID is used within the site of an organization to identify subnets. This field is 16 bits long. The site of the organization can use these 16 bits within its site to create 65,536 subnets or multiple levels of addressing hierarchy and an efficient routing infrastructure.
- The Interface ID indicates the interface on a specific subnet within the site. This field is 64 bits long.

The public topology is the collection of larger and smaller Internet Service Providers that provide access to the IPv6 Internet. The site topology is the collection of subnets within the site of an organization. The interface identifier identifies a specific interface on a subnet within the site of an organization.

Link-Local addresses : Nodes use link-local addresses when communicating with neighboring nodes on the same link. For example, on a single-link IPv6 network with no router, hosts use link-local addresses to communicate with other hosts on the link. The scope of a link-local address is the local link.
A link-local address is required for Neighbor Discovery processes and is always automatically configured, even in the absence of all other unicast addresseses.
The Figure 2.6 shows the structure of the link-local address. Link-local addresses always begin with FE80. With the 64-bit interface identifier, the prefix for link-local addresses is always FE80::/64. An IPv6 router

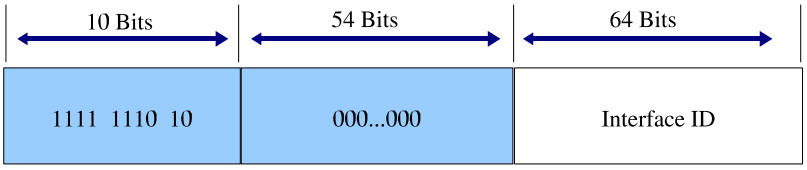


Figure 2.6: IPv6 link-local unicast addresses structure

never forwards link-local traffic beyond the link.

Site-Local addresses [4] **:** Site-local addresses are equivalent to the IPv4 private address space (10.0.0.0/8, 172.16.0.0/12, and 192.168.0.0/16). For example, private intranets that do not have a direct, routed connection to the IPv6 Internet can use site-local addresses without conflicting with global addresses. Site-local addresses are not reachable from other sites, and routers must not forward site-local traffic outside the site. Site-local addresses can be used in addition to global addresses. The scope of a site-local address is the site (the organization internetwork). Unlike link-local addresses, site-local addresses are not automatically configured and must be assigned through either stateless or stateful address configuration.
The Figure 2.7 shows the structure of the site-local address as defined

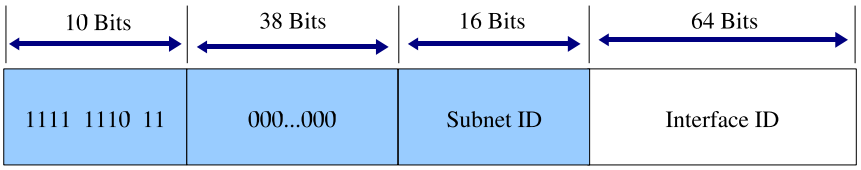


Figure 2.7: IPv6 site-local unicast addresses structure

in RFC 3513 [84]. The first 10 bits are always fixed for site-local ad-

[4]Site-local addresses are now deprecated as defined in RFC 3879 [51]. Existing implementations and deployments may continue to use this prefix.

dresses, beginning with FEC0::/10. After the 10 fixed bits is a 54-bit subnet identifier (Subnet ID field) that provides 54 bits with which you can create a hierarchical and summarizable routing infrastructure within the site. After the Subnet ID field is a 64-bit Interface ID field that identifies a specific interface on a subnet.

Special IPv6 addresses : The following are special IPv6 addresses

- The unspecified address (0:0:0:0:0:0:0:0 or ::) : it indicates the absence of an address. It is equivalent to the IPv4 unspecified address of 0.0.0.0. The unspecified address is typically used as a source address for packets attempting to verify the uniqueness of a tentative address. The unspecified address is never assigned to an interface or used as a destination address.
- The loopback address (0:0:0:0:0:0:0:1 or ::1) : it is used to identify a loopback interface, enabling a node to send packets to itself. It is equivalent to the IPv4 loopback address of 127.0.0.1. Packets addressed to the loopback address must never be sent on a link or forwarded by a router.

2. **Multicast addresses:** In IPv6, multicast traffic operates in the same way that it does in IPv4. Arbitrarily located IPv6 nodes can listen for multicast traffic on arbitrary IPv6 multicast addresses. IPv6 nodes can listen to multiple multicast addresses at the same time. Nodes can join or leave a multicast group at any time.
   A multicast address identifies multiple interfaces. With the appropriate multicast routing topology, packets addressed to a multicast address are delivered to all interfaces that are identified by the address. A multicast address is used for one-to-many communication, with delivery to multiple interfaces. An IPv6 address is easy to classify as multicast because it always begins with "FF.". Multicast addresses cannot be used as source addresses or as intermediate destinations in a Routing header. Multicast addresses include additional structure to identify their flags, scope, and multicast group. The Figure 2.8 shows the structure of the IPv6 multicast addresses. The fields in the multicast address are:

   - Flags : Indicates flags set on the multicast address. The size of this field is 4 bits. As of RFC 3513 [84], the only flag defined is the Transient (T) flag. The T flag uses the low-order bit of the Flags field. When set to 0, the T flag indicates that the multicast address is a well-known multicast address that has been permanently assigned by the Internet Assigned Numbers Authority (IANA). When set to 1, the T flag indicates that

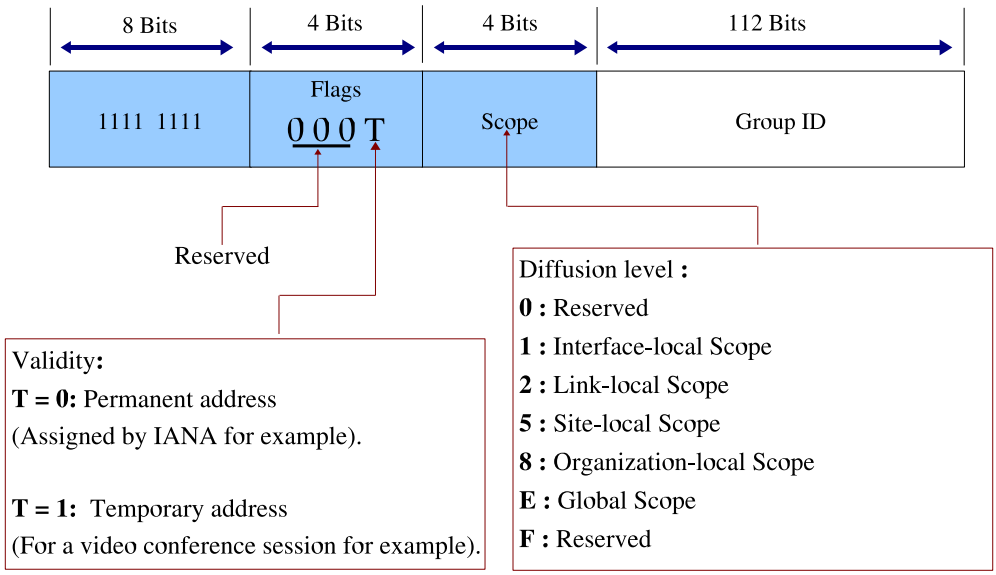


Figure 2.8: IPv6 multicast addresses structure

the multicast address is a transient multicast address that IANA has not permanently assigned.

- Scope : Indicates the scope of the IPv6 internetwork for which the multicast traffic is intended. The size of this field is 4 bits. In addition to using information provided by multicast routing protocols, routers use the multicast scope to determine whether they can forward multicast traffic. The most prevalent values for the Scope field are 1 (interface-local scope), 2 (link-local scope), and 5 (site-local scope).
- Group ID : Identifies the multicast group and is unique within the scope. The size of this field is 112 bits. Permanently assigned group IDs are independent of the scope. Transient group IDs are relevant only to a specific scope. Multicast addresses from FF01:: through FF0F:: are reserved, well-known addresses.

To identify all nodes for the interface-local and link-local scopes, the following addresses are defined:

- FF01::1 (interface-local scope all-nodes multicast address).
- FF02::1 (link-local scope all-nodes multicast address).

To identify all routers for the interface-local, link-local, and site-local scopes, the following addresses are defined:

- FF01::2 (interface-local scope all-routers multicast address).
- FF02::2 (link-local scope all-routers multicast address).
- FF05::2 (site-local scope all-routers multicast address).

With 112 bits for the group ID, it is possible to have 2112 group IDs. However, because of the way in which IPv6 multicast addresses are mapped to Ethernet multicast media access control (MAC) addresses, RFC 3513 recommends assigning the group ID from the low-order 32 bits of the IPv6 multicast address and setting the remaining original group ID bits to 0. By using only the low-order 32 bits, each group ID maps to a unique Ethernet multicast MAC address.

Solicited-node Multicast Address : The solicited-node address facilitates the efficient querying of network nodes during address resolution. In IPv4, the ARP Request frame is sent to the MAC-level broadcast, disturbing all nodes on the network segment, including those that are not running IPv4. IPv6 uses the Neighbor Solicitation message to perform address resolution. However, instead of using the local-link scope all-nodes multicast address as the Neighbor Solicitation message destination, which would disturb all IPv6 nodes on the local link, the solicited-node multicast address is used. The Figure 2.9

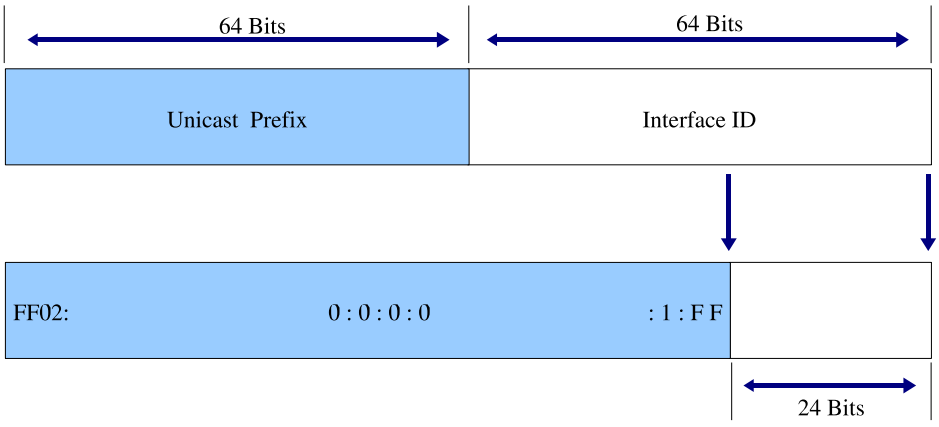


Figure 2.9: Solicited-Node multicast IPv6 Addresses structure

shows how the solicited-node multicast address comprises the prefix

FF02::1:FF00:0/104 and the last 24-bits of the IPv6 address that is being resolved.
For example, for the node with the link-local IPv6 address of FE80::2AA:FF:FE28:9C5A, the corresponding solicited-node address is FF02::1:FF28:9C5A. This node is listening for multicast traffic at the solicited-node address of FF02::1:FF28:9C5A and, for interfaces that correspond to a physical network adapter, has registered the corresponding multicast address with the network adapter. To resolve the address of
FE80::2AA:FF:FE28:9C5A to its link-layer address, a neighboring node sends a Neighbor Solicitation to the solicited-node address of
FF02::1:FF28:9C5A.
The result of using the solicited-node multicast address is that address resolutions, a common occurrence on a link, are not required to use a mechanism that disturbs all network nodes. By using the solicited-node address, address resolution disturbs very few nodes. In practice, due to the relationship between the Ethernet MAC address, the IPv6 interface ID, and the solicited-node address, the solicited-node address acts as a pseudo-unicast address for very efficient address resolution.

3. **Anycast addresses:** an anycast address identifies multiple interfaces. With the appropriate routing topology, packets addressed to an anycast address are delivered to a single interface, the nearest interface that is identified by the address. The nearest interface is defined as being closest in terms of routing distance. At present, anycast addresses are used only as destination addresses and are assigned only to routers. Anycast addresses are assigned out of the unicast address space, and the scope of an anycast address is the scope of the type of unicast address from which the anycast address is assigned.
   The Subnet-Router anycast address is predefined and required. It is created from the subnet prefix for a given interface. To construct the Subnet-Router anycast address, the bits in the subnet prefix are fixed at their appropriate values, and the remaining bits are set to 0. All router interfaces attached to a subnet are assigned the Subnet-Router anycast address for that subnet. The Subnet-Router anycast address is used to communicate with one of multiple routers attached to a remote subnet.

2.2.2 IPv6 Interface Identifiers

The last 64 bits of an IPv6 address are the interface identifier that is unique to the 64-bit prefix of the IPv6 address. IPv6 interface identifiers are determined as follows:

. Derived from the Extended Unique Identifier (EUI)-64 address.

. Generated randomly and changed over time to provide a level of anonymity.

. Assigned during stateful address autoconfiguration (for example, through DHCPv6 [49] [5]).

1. **EUI-64 Address-based Interface Identifiers :** RFC 3513 [84] states that all unicast addresses that use the prefixes 001 through 111 must also use a 64-bit interface identifier derived from the EUI-64 address. EUI-64 addresses have 64-bits and are defined by the Institute of Electrical and Electronic Engineers (IEEE). EUI-64 addresses are either assigned to network adapters or derived from IEEE 802 addresses.

   - **IEEE 802 addresses :** Traditional interface identifiers for network adapters use a 48-bit address called an IEEE 802 address. It consists of a 24-bit company ID (also called the manufacturer ID) and a 24-bit extension ID (also called the board ID). The combination of the company ID, which is uniquely assigned to each manufacturer of network adapters, and the board ID, which is uniquely assigned to each network adapter at the time of assembly, produces a globally unique 48-bit address. This 48-bit address is also called the physical, hardware, or media access control (MAC) address.
     The IEEE 802 address includes the following defined bits:

     . **Universal/Local(U/L) bit :** The next-to-the-low order bit in the first byte indicates whether the address is universally or locally administered. If the U/L bit is set to 0, the IEEE (through the designation of a unique company ID) has administered the address. If the U/L bit is set to 1, the address is locally administered, which means that the network administrator has overridden the manufactured address and specified a different address.

[5] Updated by RFC 4361 [50].

. **Individual/Group(I/G) bit :** The low-order bit of the first byte indicates whether the address is an individual address (unicast) or a group address (multicast). If the I/G bit is set to 0, the address is a unicast address. If the I/G bit is set to 1, the address is a multicast address.

For a typical IEEE 802 network adapter address, both the U/L and I/G bits are set to 0, indicating a universally administered, unicast MAC address.

- **Mapping IEEE 802 addresses to an IPv6 Interface Identifier :**
  . To create an EUI-64 address from an IEEE 802 address, the 16 bits of 11111111 11111110 (0xFFFE) are inserted into the IEEE 802 address between the company ID and the extension ID.
  . To obtain the 64-bit interface identifier for an IPv6 unicast address, the U/L bit in the EUI-64 address is complemented (If it is set to 1, it is changed to 0, and if it is set to 0, it is changed to 1).

  The Figure 2.10 shows this conversion process for a universally administered, unicast IEEE 802 address.

2.3 Neighbor Discovery

Neighbor Discovery Protocol uses ICMPv6 [53] messages to manage neighboring node interaction. Neighbor Discovery replaces ARP, ICMP Router Discovery, and ICMP Redirect and provides additional functionality.

2.3.1 Neighbor Discovery Protocol Messages

All functions of Neighbor Discovery are performed with the following messages:

. **Router Solicitation :** IPv6 hosts send Router Solicitation messages to discover IPv6 routers present on the link. To prompt IPv6 routers to respond immediately, hosts send multicast Router Solicitation messages rather than waiting for a periodic Router Advertisement message.

. **Router Advertisement :** IPv6 routers send Router Advertisement messages either periodically or in response to the receipt of Router Solicitation messages. Router Advertisement messages contain the information required by

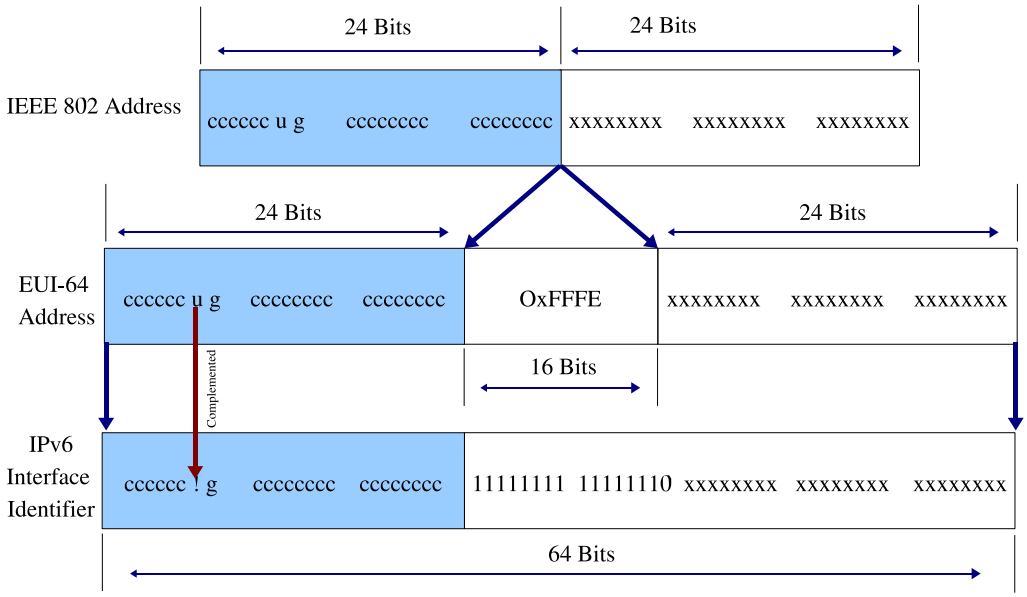


Figure 2.10: Conversion of a Universally Administered, Unicast IEEE 802 Address to an IPv6 Interface Identifier

hosts to determine what the link prefixes are, what the link MTU is, whether or not to use address autoconfiguration, and the duration for which addresses created through address autoconfiguration are both valid and preferred.

. **Neighbor Solicitation :** IPv6 hosts send Neighbor Solicitation messages to discover the link-layer addresses of on-link IPv6 nodes. Neighbor Solicitation messages include the link-layer address of the sender. Hosts send Neighbor Solicitation messages to multicast addresses to resolve addresses and to unicast addresses to verify the reachability of a neighboring node.

. **Neighbor Advertisement :** IPv6 nodes send Neighbor Advertisement messages in response to the receipt of Neighbor Solicitation messages. Nodes also send unsolicited Neighbor Advertisements to inform neighboring nodes of changes in link-layer addresses. Neighbor Advertisement messages contain information that nodes require to determine the type of Neighbor Advertisement message, the link-layer address of the sender, and the sender's role on the network.

. **Redirect message :** IPv6 routers send Redirect messages to inform an originating host of a better first-hop address to which traffic should be forwarded for a specific destination. Routers send Redirect messages for unicast traffic only and only to originating hosts. Only hosts process Redirect messages.

To ensure that all Neighbor Discovery messages were sent by a node on the local link, all Neighbor Discovery messages are sent with a hop limit of 255. When a Neighbor Discovery message is received, the Hop Limit field in the IPv6 header is checked. If the field is not set to 255, the message is silently discarded. Verifying that the Neighbor Discovery message has a hop limit of 255 provides protection from network attacks that are based on Neighbor Discovery and launched from off-link nodes. If a message has a hop limit of 255, a router could not have forwarded the message from an off-link node.

2.3.2 Neighbor Discovery Message Exchanges

The Neighbor Discovery protocol provides message exchanges for the following processes:

. Address resolution.

. DAD : Duplicate Address Detection.

. Router discovery.

. Redirect function.

2.4 IPv6 Stateless Address Autoconfiguration

A highly useful aspect of IPv6 is its ability to automatically configure itself without the use of a stateful configuration protocol, such as Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [49]. The IPv6 SAA [47] is based on NDP [82] which is specified for links that support a native form of multicast or broadcast. Some other links are covered by extension documents, but there is no extension for ad hoc networks yet.
The IPv6 SAA basically consists of three phases:

1. Construction of a link-local address for the use on the local link.

2. The Duplicate Address Detection.

3. Construction of a site-local address for the use on the site.

The use of a global address is not part of the SAA. Therefore, Statefull Address Autoconfiguration DHCPv6 [49] may be used.

The process begins with the construction of a link-local address that is based on the interface identifier and a well-known link-local prefix. It is first considered as a *tentative address*. The IEEE 64-bit Extended Universal Identifier (EUI-64) is used for this purpose. Although the EUI-64 number is designed to be globally unique, this cannot be guaranteed, because of manufacturers using unregistered 802.x addresses, support for changing the MAC-address by the user or obscure network devices, which choose a MAC-address randomly. The DAD process is needed to detect and handle duplicate addresses. Therefore, the node issues a *Neighbor Solicitation (NS)* message (Figure 2.11) with the well-known *unspecified address* as source IP address. This address must never be assigned to a node and indicates the absence of an address [45].

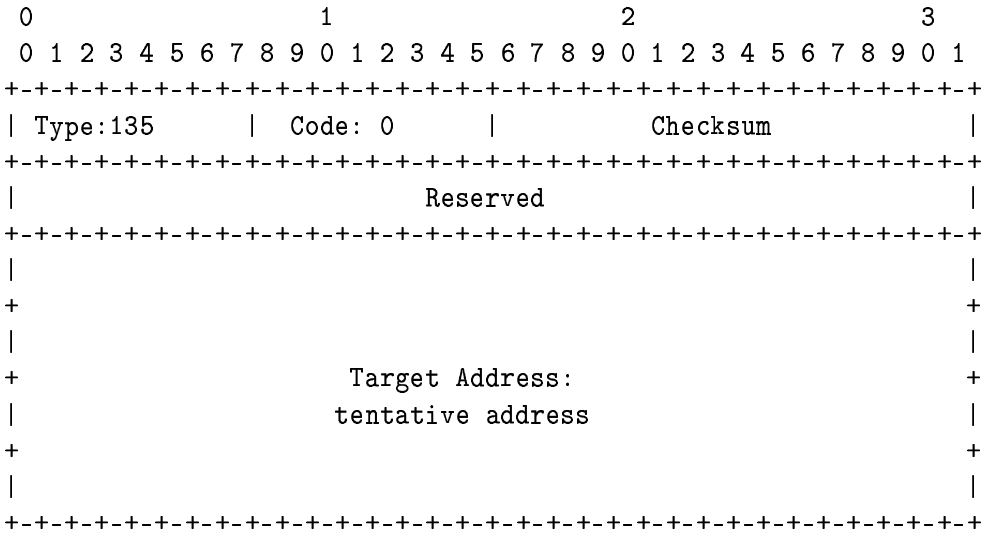


Figure 2.11: Neighbor Solicitation message

The *solicited-node multicast address* is used as destination IP address, which is based on a well-known prefix and the last 24 bits of the tentative address. The tentative address is used as solicitation target address and a hop limit of 255 is

used to limit the message on the local link. NDP messages with a hop limit of less than 255 are discarded. This prevents hackers from introducing NDP messages to the local link from outside.
If the address is already in use by another node, this node responds with a *Neighbor Advertisement (NA)* message carrying the all-nodes multicast address as destination IP address. An address conflict is recognized, if the sender receives a NA message in reply to the NS message or if a NS message with the same solicitation target address is received, indicating that another node with the same tentative address currently performs the DAD.
If Routing Advertisement messages containing a subnet ID are received, hosts construct a site-local address using the link-local address, a well-known site-local prefix and the announced subnet ID.

2.5 IPv6 Transition Technologies

The transition from IPv4 to IPv6 will take years, and organizations or hosts within organizations might continue to use IPv4 indefinitely. Therefore, although migration is the long-term goal, equal consideration must be given to the interim coexistence of IPv4 and IPv6 nodes.

RFC 1752 [36] defines the following transition criteria:

- Existing IPv4 hosts can be upgraded at any time, independent of the upgrade of other hosts or routers.
- Hosts that use only IPv6 can be added at any time, without dependencies on other hosts or routing infrastructure.
- IPv4 hosts on which IPv6 is installed can continue to use their IPv4 addresses and do not need additional addresses.
- Little preparation is required to either upgrade IPv4 nodes to IPv6 or deploy new IPv6 nodes.

The inherent lack of dependencies between IPv4 and IPv6 hosts, IPv4 routing infrastructure, and IPv6 routing infrastructure requires several mechanisms that allow seamless coexistence.

2.5.1 Node Types

RFC 4213 [37] defines the following node types:

. **IPv4-only Node :** A node that implements only IPv4 (and has only IPv4 addresses). This node does not support IPv6.

. **IPv6-only Node :** A node that implements only IPv6 (and has only IPv6 addresses). This node is able to communicate only with IPv6 nodes and applications.

. **IPv6/IPv4 Node :** A node that implements both IPv4 and IPv6. This node is IPv6-enabled if it has an IPv6 interface configured.

. **IPv4 Node :** A node that implements IPv4 (it can send and receive IPv4 packets). An IPv4 node can be an IPv4-only node or an IPv6/IPv4 node.

. **IPv6 Node :** A node that implements IPv6 (it can send and receive IPv6 packets). An IPv6 node can be an IPv6-only node or an IPv6/IPv4 node.

For coexistence to occur, the largest number of nodes (IPv4 or IPv6 nodes) can communicate using an IPv4 infrastructure, an IPv6 infrastructure, or an infrastructure that is a combination of IPv4 and IPv6. True migration is achieved when all IPv4 nodes are converted to IPv6-only nodes. However, for the near future, practical migration is achieved when as many IPv4-only nodes as possible are converted to IPv6/IPv4 nodes.

2.5.2 Address Compatibility

To aid in the migration from IPv4 to IPv6 and the coexistence of both types of hosts, the following addresses are defined :

. **IPv4-compatible Addresses :** The IPv4-compatible addresses, 0:0:0:0:0:0:w.x.y.z or ::w.x.y.z (where w.x.y.z is the dotted decimal representation of a public IPv4 address), is used by IPv6/IPv4 nodes that are communicating using IPv6. IPv6/IPv4 nodes support both IPv4 and IPv6 protocols. When an IPv4-compatible address is used as an IPv6 destination, the IPv6 traffic is automatically encapsulated with an IPv4 header and sent to the destination using the IPv4 infrastructure.

. **IPv4-mapped Addresses :** The IPv4-mapped address, 0:0:0:0:0:FFFF:w.x.y.z or ::FFFF:w.x.y.z, is used to represent an IPv4-only node to an IPv6 node. It is used only for internal representation. The IPv4-mapped address is never used as a source or destination address of an IPv6 packet.

. **6over4 Addresses :** Each 6over4 address comprises a valid 64-bit unicast address prefix and the interface identifier ::WWXX:YYZZ (where WWXX:YYZZ is the colon-hexadecimal representation of w.x.y.z, a unicast IPv4 address assigned to an interface). An example of a link-local 6over4 address based on the IPv4 address of 131.107.4.92 is FE80::836B:45C. 6over4 addresses represent a host that use the automatic tunneling mechanism defined in RFC 2529 [38].

. **6to4 Addresses :** The 6to4 address is used for communicating between two nodes running both IPv4 and IPv6 over an IPv4 routing infrastructure. The 6to4 address is formed by combining the prefix 2002::/16 with the 32 bits of a public IPv4 address of the node, forming a 48-bit prefix. 6to4 is a tunneling technique described in RFC 3056 [39].

2.5.3 Transition Mechanisms

The following mechanisms are used to help IPv6 coexist with an IPv4 infrastructure and to provide an eventual transition to an IPv6-only infrastructure:

. **Dual IP Layer :** The dual IP layer is an implementation of the TCP/IP suite of protocols that includes both an IPv4 Internet layer and an IPv6 Internet layer. IPv6/IPv4 nodes use this mechanism to communicate with both IPv4 and IPv6 nodes. Figure 2.12 shows a dual IP layer architecture.

. **IPv6 over IPv4 Tunneling :** IPv6 over IPv4 tunneling is the encapsulation of IPv6 packets with an IPv4 header so that IPv6 packets can be sent over an IPv4 infrastructure. Within the IPv4 header:

    1. The IPv4 Protocol field is set to 41 to indicate an encapsulated IPv6 packet.
    2. The Source and Destination fields are set to IPv4 addresses of the tunnel endpoints. The tunnel endpoints are either manually configured as part of the tunnel interface or are automatically derived from the sending interface, the next-hop address of the matching route, or the source and destination IPv6 addresses in the IPv6 header.

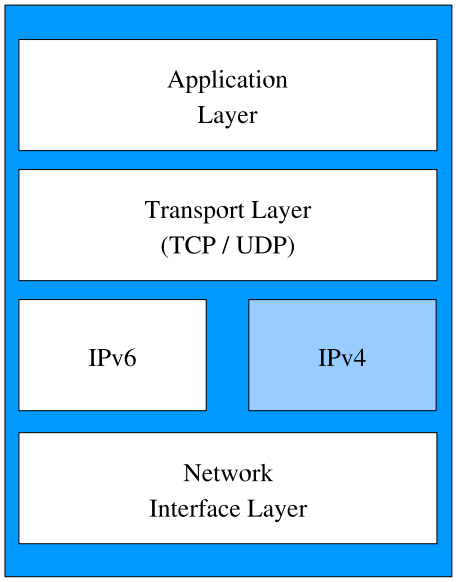


Figure 2.12: A Dual IP Layer Architecture

. **DNS Infrastructure :** Because most users refer to network resources by name rather than by address, successful coexistence requires upgrading the DNS infrastructure. Upgrading the DNS infrastructure consists of populating the DNS servers with records to support IPv6 name-to-address and address-to-name resolution. After the sending host obtains addresses using a DNS name query, the node must select which addresses to use for communication.

RFC 3596[52] defines the changes that need to be made to the Domain Name System (DNS) to support hosts running IP version 6 (IPv6). The changes include a resource record type to store an IPv6 address, a domain to support lookups based on an IPv6 address, and updated definitions of existing query types that return Internet addresses as part of additional section processing. The extensions are designed to be compatible with existing applications and, in particular, DNS implementations themselves.

The DNS infrastructure must contain the following resource records (populated either manually or dynamically) for the successful resolution of domain names to addresses:

1. A records for IPv4-only and IPv6/IPv4 nodes.
2. AAAA records for IPv6-only and IPv6/IPv4 nodes.

The DNS infrastructure must contain the following resource records (populated either manually or dynamically) for the successful resolution of address to domain names (reverse queries):

1. PTR records in the IN-ADDR.ARPA domain for IPv4-only and IPv6/IPv4 nodes.
2. PTR records in the IP6.ARPA domain for IPv6-only and IPv6/IPv4 nodes (optional).

Chapter 3

Adapting OLSR to IPv6

In this chapter we will describe the changes needed to adapt *OLSR* protocol to IPv6. These changes include addressing issues, packet diffusion in IPv6 networks, and, essentially, autoconfiguration adapted to *OLSR*.

The issues to address are :

- *Addressing:* IPv6 introduces several changes, some more conceptual than others. Changes include the diffusion of data packets (broadcast or multicast) and existing multiple addresses of interfaces.
- *Protocol changes:* the *OLSR* specification [55] gives the protocol messages format for IPv4 packets, but some additional changes are proposed.
- *Neighbor discovery:* it is described how the neighbor discovery mechanism of IPv6 still operates properly.
- *Autoconfiguration:* loosely related to addressing, the ability for an IPv6 node to self-configure its addresses yields numerous challenges and had been the subject of elaborate research.

Sections 3.1 and 3.2 tackle the IPv6 addressing issues, and the following issues are later tackled in a separate section each.

3.1 IPv6 ad hoc addressing issues

As said previously, several changes are required because of the various novelties introduced by IPv6 itself.

In the following, we first present the addresses that should be used by *OLSR* to diffuse its control packets, and then we introduce the routable addresses to use with some implementation issues.

3.1.1 Interface addresses

The IPv6 Addressing Architecture specification [80] defines several addresses that a host should recognize; this includes its link-local addresses and its assigned unicast addresses.

Section 3.3 describes in more detail the proposed system and algorithm. Let's consider the conceptual problem however, here: on the one hand, a MANET node is expected to have at least one link-local address, but this address is not routable (so no routing protocol can set routing tables using them) ; on the other hand, a MANET node may get a global address by the IPv6 autoconfiguration mechanism such as router advertisement messages, but a MANET node is not guaranteed to be connected to an IP gateway in the general case.

The solution chosen here is to use site-local addresses: it is the classical solution of creating a third intermediary address, with a well-defined prefix, and completed with the suffix obtained by the IPv6 stateless autoconfiguration process. The IPv6 stateless autoconfiguration process, specified by [47] and modified with the algorithm described in Section 3.5, yields a link-local address with the well-known link-local prefix (`FE80::/64`). It has ("typically") a suffix of 64 bits based on the EUI-64 identifier described in [80].

More precisely, the chosen solution in this chapter is to consider a MANET as a single site local network, and to use a site-local prefix with an agreed fixed 16 bits `OLSR_SUBNET` subnet. And then an *OLSR* node will perform link-local address autoconfiguration, and upon success, will automatically configure for each of its OLSR interfaces, the site-local address with that subnet (`FEC0:0:0:OLSR_SUBNET::/64`), and will run the *OLSR* protocol using it.

Additional addresses may be obtained by the standard router advertisements, if the MANET of a node is connected to one or several gateways: such interfaces addresses are configured and advertised as described in Section 3.3.

Since link-local addresses would never appear in the messages of a properly autoconfigured node, with the previous design, our algorithm still uses them for interfaces which are not yet configured, in the stateless configuration process, in a manner which is identical to [60]: a range of temporary addresses is used exclusively during autoconfiguration.

3.1.2 OLSR diffusion addresses

The *OLSR* protocol works by diffusing its control packets to its direct neighbors (nodes within radio range). Those packets are processed locally, and some of them, if they are destined to the entire network, are retransmitted by a subset of neighbors called *MPRs*. With the MPR flooding process, messages will reach all the nodes in the network.

With IPv4, sending packets to one hop neighbors could be achieved by using the IPv4 broadcast address (`255.255.255.255`).

In IPv6, no such broadcast address exists, and the question is which diffusion address should be used. We propose that a multicast address ALL_OLSR_NODES is used for the destination address, in order to reach all the nodes present on the link (in other words, here, within radio reach) to get the same effect as in IPv4. ALL_OLSR_NODES could be taken as ALL_LINK_NODES (`FF02::1`).

In addition, since a node has several interfaces addresses, we propose that the site-local addresses (see previous section), which are always active in our proposal, are used as source addresses.

3.2 Diffusing non *OLSR* packets

Since *MANETs* are multi-hop routing networks, in order to flood packets to all the nodes, retransmissions are usually needed. With *OLSR*, packets are retransmitted hop by hop to the direct neighborhood using the *MPRs*. On the other hand, for many applications, a direct multicast on the local "link" is performed and such packets are never routed. It is also the case for most of IPv6 messages for neighbor discovery and autoconfiguration for instance. This relies on the assumption that being on the same network is equivalent to being on the same link, an assumption which doesn't hold in MANET networks. As a result, in a multi-hop network, by default, this kind of messages will not be delivered to all the nodes.

Several solutions exist in the literature, (including adding an extra addressing layer like in ANANAS [54]). In the context of *OLSR*, we propose the two following solutions to diffuse non-*OLSR* packets to all nodes :

1. Encapsulate the packets in specific *OLSR* messages and use the *MPR* flooding.
2. Use of a new multicast address which we call ALL_MANET_NODES, instead of the ALL_LINK_NODES.

Details are provided in the next sections.

Note that when a gateway is diffusing Router Advertisements messages, associated HNA messages should be sent at the same time.

3.2.1 Encapsulation of non-OLSR messages

The first solution is conceptually easy, as it uses directly the optimized *MPR* flooding mechanism of *OLSR*. If the node does not implement the Message Type of the received message [1], the message should be processed and forwarded according to the default forwarding algorithm described in [55].

3.2.2 MANET-scope Multicast

In the second solution, all *MANET* nodes must join this ALL_MANET_NODES group address to receive the flooding messages and it should have a site scope to allow the routing by intermediate nodes. The flooding mechanism in the ad hoc network is based on retransmission of the received packet at most once ; pure flooding [2] could be used.

This implies also that a mechanism to control the process of repetition is used, in order to avoid useless transmissions. This, could be done, by adding a sequence number in each packet and a duplicate table in each node, as for *OLSR* protocol packets. A node would maintain a duplicate set to prevent transmitting the same packet twice. Unfortunately, IPv6 packets do not contain a sequence number. A known idea, which we adopt, is to create a new option for the IPv6 hop-by-hop extension header, to add a packet sequence number and also possibly the IP address of the intermediate transmitter.

3.3 Changes to the *OLSR* routing protocol

Detailed *OLSR* messages and algorithms description can be found in Chapter 1 and in [55].

[1] That is the case of the new non-OLSR messages introduced in our IPv6 solution.

[2] pure flooding: repetition of a packet exactly once by each node in the network, so that all nodes are reached.

3.3.1 OLSR packet format

The essential change needed for the existing *OLSR* packet format is to replace the IPv4 addresses with the IPv6 addresses in all messages, as highlighted in the *OLSR* specification itself [55].

The other change to *OLSR* packet formats, described in the next section, is either a change of the "Multiple Interface Declaration" messages or the addition of another type of messages: "Multiple Address Declaration".

3.3.2 Multiple interface addresses

In IPv6, an interface can have several addresses. As described in Section 3.1, in our proposal an *OLSR* node will have for each interface:

- A link-local address usually obtained by autoconfiguration. Let's add that, before autoconfiguration is completed, this address is *temporary* used as the source address for *OLSR* packets.
- A site-local address, derived from the link-local address in a well-know fixed subnet `OLSR_SUBNET` for site-local prefix. This address is permanently used as the source for all *OLSR* packets once autoconfiguration is completed.
- Any number (possibly zero) of additional global or site local unicast addresses which are automatically or manually configured.

Temporary use of link-local addresses

As described in more details later in Section 3.5.1, OLSR is modified in such a way that link-local addresses are used by a node which has not yet completed autoconfiguration of its interfaces. The idea being that only local topology should be updated according to those informations. This is achieved by modifying the processing rules of the *OLSR* specification [55] as follows:

- Link sensing and Neighbor discovery is done as in [55].
- MPR computation ignores all links to one-hop neighbors involving link-local addresses. This is to prevent other nodes to immediately rely on the incoming node for the purposes of MPR flooding.
- Routing table computation ignores all links involving one link-local address.

- "Topology Control" message generation should also ignore all links involving one link-local address.

Multiple Address Declaration messages

Another issue is that the *OLSR* specification [55] does not directly consider the case of having several addresses associated with one interface.

A simple remark can be used: an interface address which is never used as a source address for protocol packets is strictly equivalent to an interface which has physically no neighbor. Hence, conceptually all the extra addresses (that is all the other addresses than the site-local addresses, which are used by *OLSR*, and the link-local addresses which must not be routed), could be declared in the "Multiple Interface Declaration" of the *OLSR* protocol, and this stays also true for IPv4.

However, also because of the proactive part of duplicate address detection (Section 3.5.2), our solution requires extra information to be added, so preferably a new type of message, "Multiple Address Declaration" message is introduced.

The "Multiple Addresses Declaration" message contains:

- An "identifier" of the node (see Section 3.5.2)
- The "main address" (which is the site-local IPv6 address of one interface).
- The list of site-local IPv6 *OLSR* addresses of the node's interfaces on which the *OLSR* protocol is running.
- The list of global unicast IPv6 addresses.

and those messages should be processed like "Multiple Interface Declaration" message.

Notice that "Multiple Address Declaration" message should be sent only once autoconfiguration has been completed.

3.4 Neighbor discovery

In IPv6 nodes (hosts and routers) use Neighbor Discovery Protocol [82] to determine the MAC addresses for neighbors known to reside on attached links and to quickly purge cached values that become invalid. Hosts also use Neighbor Discovery to find neighboring routers that are willing to forward packets on their

behalf. Finally, nodes use the protocol to actively keep track of which neighbors are reachable and which are not and to detect changed MAC addresses. When a router or the path to the router fails, a host actively searches for functioning alternates. We note that neighbor discovery does not mean learning information about all other nodes connected to the same link but only those with which the node is willing to communicate.

Routing table in *OLSR* indicates the next hop for each reachable destination in the network. This next hop is one of the direct neighbors. This means that the neighbor solicitation for address resolution will work without any modification. The node uses link local broadcast, and the destination will reply. With coherent routing tables, the address resolution works correctly. In *OLSR*, "gateways" declare themselves to the entire network by periodically transmitting Host and Network Association (HNA) message, containing pairs of (network address, netmask) corresponding to the connected hosts and networks. The neighbor discovery is adapted to *OLSR*, and consequently, we do not introduce any modification to the classical procedure.

3.5 Autoconfiguration: Duplicate Address Detection

In this section, a relatively simple approach to autoconfiguration and Duplicate Address Detection(DAD) is selected. The main underlying idea is that preventing address collision, requires one assumption among the following three:

- an assumption of some unique identifiers being allocated at one level or another (MAC addresses, administrative level, unique "network identifiers", ...) - from which some other identifiers might be derived.
- an assumption of a certain number of scenarios: but not all possible scenarios.
- an assumption that collision detection could be, by nature, probabilistic (with a probability of collision which could be set to a tolerable level) - and thus not absolutely guaranteed.

In this section, the assumption chosen is the third one: duplicate address detection is done in a probabilistic way, and then a two-part approach is used:

- Checking immediately whether a newly created site-local address is already in use, with a certain probability of false-negatives.

- Checking periodically whether the currently used site-local addresses are used by other nodes, also with a certain probability of false-negatives, low but not absolutely nil.

The first parts avoid to immediately collide with existing nodes in the network, the second part ensures detection of duplicates, in all the scenarios, including the merging scenarios although with a delay proportional to the period of checks.

IPv6 Stateless Address Autoconfiguration [47] is based on several steps: after the creation of a link-local address, the node must check whether the address is already in use by another interface of another node somewhere in the network (the Duplicate Address Detection mechanism). In wired environment, this means that all the links of the attached interfaces of the node are probed. If the address is not unique the process is interrupted, otherwise the autoconfiguration was successful and the address may be safely used. In a *MANET* network, the nodes on the links of the attached interfaces would include only the nodes with an interface within radio reach of the transmitter and not all the participating nodes. Hence, the uniqueness of the address is not guaranteed if the classical DAD procedure is applied.

In the following, we propose a new and simple algorithm to perform autoconfiguration in an IPv6 *OLSR* network which includes two parts: the first part, in Section 3.5.1, is a Duplicate Address Detection following the philosophy of the IPv6 DAD and reactive protocols of sending a request to the whole network and waiting for a (possible) answer; the second part, in Section 3.5.2, is performing Duplicate Address Detection in the spirit of the proactive protocols, checking periodically for duplicate addresses. Section 3.5.3 details what should be done upon detection of a collision.

For clarity, the algorithm have been presented with one single interface: the autoconfiguration on several interfaces is easily derived, by waiting for the autoconfiguration process to finishing on all interfaces before considering that autoconfiguration is completed.

3.5.1 Duplicate Address Detection: reactive probing

The reactive probing algorithm works in several steps; the main idea is the use of the MPR flooding mechanism to perform a diffusion of a DAD message to the entire network.

However, the difficulty is that the incoming node performing the address autoconfiguration is not yet in the *OLSR* network, and as a consequence, MPR

flooding mechanism cannot work.

The precise reasons are:

- No neighbor node is MPR of the incoming node, so none of its neighbors will repeat its messages.
- Even if all its one hop neighbors were to retransmit the messages by some means, an answer sent by MPR flooding might fail to reach the incoming node [3].

Hence in this algorithm a new node proceeds by using the intermediary step of "partly" joining the network. Partly means that the *OLSR* protocol is modified so that to exchange informations so as to maintain proper local topology information only (one-hop, two-hop, and MPR information), as specified in Section 3.3. Once the local topology is known well enough, the link-local address is diffused to the whole network within a DAD message and the new node waits for an answer. The details of the reactive probing algorithm are precised in the following.

When a new node joins a MANET network, it performs the following tasks :

1. It creates a link-local address.
2. The node performs the classical IPv6 DAD (Duplicate Address Detection) procedure in its neighborhood.
3. If the link-local address is unique in the immediate neighborhood, the node assigns it to its interface. Now the node, considered as a pending node, starts running *OLSR* and calculates its *MPRs* by exchanging Hello messages using link-local addresses. In this intermediate step, the new node must not be declared by its neighbors to the entire network and must not be chosen as a *MPR*.
4. The node chooses one of its already configured neighbors (*MPRs*): it will be its **D**esignated **R**outer (**DR**).
5. In this step, the **D**esignated **R**outer performs the DAD procedure in the entire network on behalf of the pending node using one of the methods of Section 3.2 for diffusing non *OLSR* packets.

[3] this stems from the fact that in some topologies, some of its two hop neighbors might not have have selected an MPR covering it

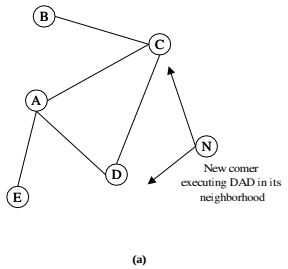


Figure 3.1: A newcomer in the network

6. If the address is not duplicated, the pending node will now create a site-local address and no longer be pending. The site-local address will be used by the node in the *OLSR* processing. Note that the link-local address will expire from the table of its neighbors.

7. The node informs its **D**esignated **R**outer of the end of this configuration process.

8. Now, the new node can integrate the network.

9. Then, it can possibly wait for an IPv6 Router Advertisement message, to complete its set of addresses, or send an IPv6 Router Solicitation message to explicitly check for existing gateways.

Example

We give now an example to illustrate the autoconfiguration procedure. In Figure 3.1, a newcomer (node N) is turned on and tries to configure itself. We suppose that the other nodes of the network have already configured their interfaces and are running *OLSR* protocol. The node N first sets its interface to a link-local address [84]. Then, it starts performing the classical DAD on the local link (which means the direct neighborhood).

If no reply is received, the node N starts running *OLSR* protocol (Figure 3.2). With the exchange of *HELLO* messages, the node N calculates its *MPRs* and chooses one of them to perform DAD in the network on behalf of it (node D).

Node D diffuses a request in the network to check if this address is used by another node. Notice here that if it exists another node having the same link-local address and running the DAD procedure at the same time, this situation will be

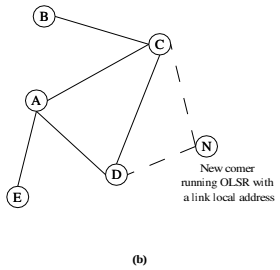


Figure 3.2: A newcomer starts running *OLSR* with link-local address

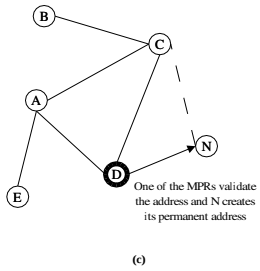


Figure 3.3: A newcomer configures its site-local address and integrates the network

detected by their respective *DRs(MPRs)*. In Figure 3.3, the node D validate the link-local address of the newcomer. Finally, the node N, can configure its site-local address.

3.5.2 Duplicate Address Detection: proactive checking

Under our assumptions, there are some scenarios in which the initial duplicate address detection is not sufficient to ensure that the IPv6 addresse is unique in the MANET network: for instance when a merge of two previously disconnected MANET networks occurs.

Our algorithm relies on the *node identifiers* (NID) diffused along with "Multiple Address Declaration" (see Section 3.3.2).

A *node identifier* is a sequence of bits of fixed length L which is randomly generated. Hence, we are using the standard idea that the probability of two nodes having the same node identifier is low, and the probability of at least one

address collision with N nodes, which is the well known "birthday problem", can be set arbitrarily low by choosing a value of L large enough [4].

A node performs proactive checking, by simply checking Multiple Address Declaration messages from *other* nodes, more precisely MAD messages with a node identifier different from its own. If in such a message, the site-local address of one of its interfaces is found to be advertised, then an address collision has been detected and the conflict must be handled (Section 3.5.3).

MPR calculation, as defined in [55], is based on the assumption that there is no address duplication in the network. Hence, when a merge of two or more separately configured MANETs occurs, address duplications may happen and the MPR calculation may be affected resulting in MAD messages being not propagated throughout the entire network. In such a situation, the duplicated addresses may not be detected and the network remains corrupted. To handle this problem, a more elaborate MPR mechanism is proposed in Chapter 4. The new MPR algorithm continue to work even in the presence of address duplications.

3.5.3 Duplicate Address Detection: collision resolution

In the case where a node detects that another node is using the same address, the node will generate, this time randomly, a new link-local address, and starts again the autoconfiguration process with this new address. Note that this will interrupt running networked applications.

A number of refinements can be adopted, namely:

- The node with the lowest node identifier (considering the sequence of bits in lexicographical order), only, may regenerate a new address. This avoids having both nodes possibly regenerating new addresses.

- Additionally, a fixed part of the first bits of the identifier may be set to the "priority" of the node to lower the probability that important nodes would have to change their addresses.

[4] different birthday probability, denoted $Q_1(N, 2^L)$, can be estimated as $(1 - \frac{N}{2^{L+1}})^{(N-1)}$ with an error bound given by [66] for instance.

3.5.4 State transitions of an OLSR node running autoconfiguration

Figure 3.4 depicts the state transitions of an OLSR mobile node running autoconfiguration during its ad hoc session. The protocol is as follows:

1. When an OLSR node switches to ad hoc mode, it sends a Neighbor Solicitation(NS) message in its neighborhood and changes from *Un-Initialized* state to *Waiting* state;

2. The OLSR node stays in the *Waiting* state and repeats diffusing NS messages with new link-local addresses in its neighborhood for less than or equal to k times while receiving Neighbor Advertisement(NA) packets;

3. If no response received for the NS message, the node configures its interface with this locally unique link-local address and can partly join the network. The node changes to *Pending* state;

4. The node returns to *Waiting* state if a conflict is detected by its *Designated Router*. Otherwise, a *UA-Confirm* message is received and the OLSR node will now create its site-local address and switches to *Configured* state;

5. During *Configured* state, the OLSR node periodically diffuses its MAD messages to the whole network and sends back replies(NA packets), in case of conflict, on receipt of NS packets from other nodes;

6. If the OLSR node receives a MAD message with different node identifier(NID) than its own but containing at least one of the node's site-local addresses, it switches to *Conflict Resolution* state. After the conflict is resolved, the node returns to *Configured* state;

7. When the OLSR node ends its session, it switches out of ad hoc mode and changes to *Un-Initialized* state.

3.6 Conclusion

In this chapter we presented the issues and necessary changes to adapt *OLSR* to IPv6: this included IPv6 stateless autoconfiguration, but also a number of relatively less explored issues such as IPv6 addressing issues, and changes to the OLSR protocol itself.

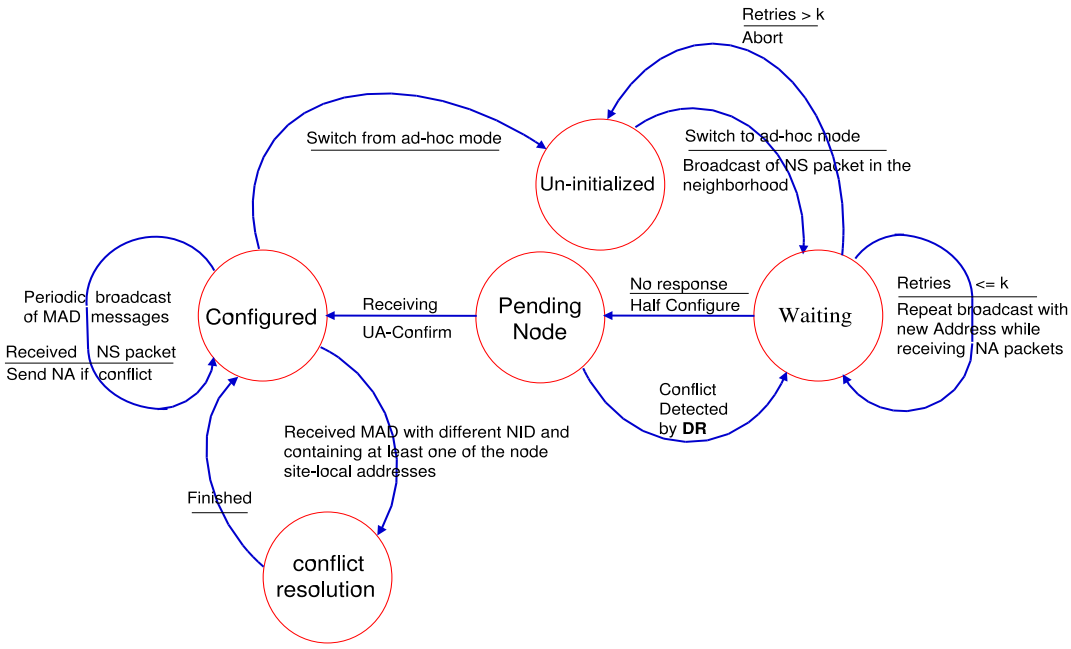


Figure 3.4: The finite state machine for an OLSR node running autoconfiguration

The autoconfiguration procedure that is proposed relies on two procedures; first, immediate request to the entire network, and second periodic checking (using node identifiers). The first procedure uses MPR flooding, and thus requires special treatments and algorithms that are detailed, in order for a node to join "partly" the OLSR network without having a unique unicast address yet.

Parallelly, changes to OLSR protocol were proposed. They include mechanisms to automatically create routable site-local addresses, definitions of multicast addresses for OLSR messages, proposition for flooding IPv6 (multicast) packets to the entire network (as intended to be the application).

Part II

Autoconfiguration schemes for OLSR

Mobile Ad hoc NETworks (MANETs) are infrastructure-free, highly dynamic wireless networks, where central administration or configuration by the user is very difficult. Most of MANET routing protocols assume that mobile nodes in ad hoc networks are configured a priori with a unique IP address before joining a manet. For small scale MANETs, it may be possible to allocate free IP addresses manually. However, the procedure becomes impractical for a large-scale or open system where mobile nodes are free to join and leave. In hardwired networks nodes usually rely on a centralized server and use a dynamic host configuration protocol, like DHCP [48] [49], to acquire an IP address. Such a solution cannot be deployed in MANETs due to the unavailability of any centralized DHCP server, the instability of links, mobility of the nodes, and the openness of the mobile ad hoc networks. Therefore, more overhead, and also more complexity, results from performing DAD (Duplicate Address Detection) when comparing to the protocols in wired networks, such as DHCP [49] and SAA [47].
Numerous dynamic addressing schemes for ad hoc networks have been proposed. These approaches differ in a wide range of aspects, such as address format, the usage of centralized servers or full decentralization, hierarchical structure or flat network organization, and explicit or implicit duplicate address detection.

The following chapter describes in detail a set of autoconfiguration rules we proposed for OLSR. The implementation details and the performance analyses of this autoconfiguration protocol are given in Chapter 5.

Chapter 4

Advanced autoconfiguration solutions for OLSR

Before describing the autoconfiguration protocol we proposed for OLSR, I first present a short related work in the following section. This related work includes some previously proposed autoconfiguration protocols for ad hoc networks.

4.1 Autoconfiguration in ad hoc networks

Numerous studies have been carried out on autoconfiguration protocols and the related issue of duplicate address detection in ad hoc networks. These studies have been proposed within the IETF or published in academic papers. In this section, I will review autoconfiguration scenarios and the different conditions where address duplications may occur. Then, I will present some proposed autoconfiguration protocols for ad hoc networks.

4.1.1 Address autoconfiguration scenarios

Before describing some proposed autoconfiguration algorithms, I first highlight some scenarios where address duplications may occur and which allow to discriminate between the different algorithms.

The first one is the simplest scenario; a mobile node joins and then leaves a *MANET*. An unused IP address is allocated to the node on its arrival and becomes free on its departure.

A more complicated scenario is the following: because nodes are free to move arbitrarily during their session in the *MANET*, one or more configured nodes can leave the transmission range of others for a while, and the network becomes partitioned. In both partitions, the nodes still use the previously allocated IP addresses. If a new node arrives at one partition and is assigned an IP address belonging to the other partition, IP address conflict happens when these two partitions merge later.

Another scenario is when two independent *MANETs* merge. Because the two *MANETs* are configured separately, and the address allocation in one *MANET* is independent of the other, there may be some duplicated addresses in both of them.

Most of the other scenarios, could be regarded as special cases of the three scenarios described above.

4.1.2 Address autoconfiguration in ad hoc networks

Numerous autoconfiguration protocols and the related issue of duplicate address detection in ad hoc networks have been suggested. These protocols can be divided into the three following categories:

- Conflict-free algorithms: In this approach, a set of nodes in the network are responsible for address allocation. The nodes taking part in address allocation have disjoint address pools. The Dynamic Configuration and Distribution Protocol (DCDP) [64] is a conflict-free allocation algorithm. When a new mobile node joins the *MANET*, an address pool is divided into halves between itself and a configured node. This algorithm takes into account network partition and merge. Conflicts will however occur during the merge if two or more of the separately configured *MANETs* taking part in the merge begin with the same reserved address range. Another conflict-free allocation algorithm, called the "prophet allocation protocol" has been proposed for large scale *MANETs* in [65]. The idea is that every mobile node executes a stateful function *f(n)* to get a unique IP address. *f(n)* is a function of a state value called the *seed* which is updated for each node in the network.

- Best-effort algorithms: The Distributed Dynamic Host Configuration Protocol (DDHCP) [63] is one example of a conflict-free allocation algorithm. In this algorithm, the nodes responsible for allocation try to assign an unused IP address to a new node – unused, that is, to the best of their knowledge.

Then the new node performs DAD to guarantee that it is an unallocated IP address. DDHCP maintains a global allocation state, so IP addresses which have been used, and addresses which have not yet been allocated, are known. When a new node joins the network, one of its neighbors chooses an unused address for it. The same unused IP address in the global address pool could be assigned to more than one node arriving at almost the same time. This is the reason why DAD is still performed by a node after getting an IP address. This algorithm takes into account network partition and merger, and works well with proactive routing protocols.

- Conflict-detection algorithms: The algorithms related to this approach perform the DAD (Duplicate Address Detection) to ensure the uniqueness of the allocated IP address. The general procedure is that a node generates a tentative address and then performs DAD within its neighborhood (radio range of the node). If the address is unique, the DAD is performed again over the whole network and a unique IP address is constructed. Examples of such approaches include [60] and [61].

  Conflict-detection algorithms can also be divided into two categories which differ in when, and how duplicate addresses are detected.

  The ADAD (Active Duplicate Address Detection) mechanisms distribute additional control information in the network to prevent address duplication as, for instance, in [60] and [61].

  In contrast, PDAD (Passive Duplicate Address Detection) algorithms [62], try to detect duplicates without disseminating additional control information in the network. The idea behind this approach is to continuously monitor routing protocol traffic to detect duplicates rather than sending additional control packets for this purpose. However, in [62] a so-called Address Conflict Notification (ACN) message is introduced for the purpose of conflict resolution.

The previous approaches can be summarized as shown in Figure 4.1.

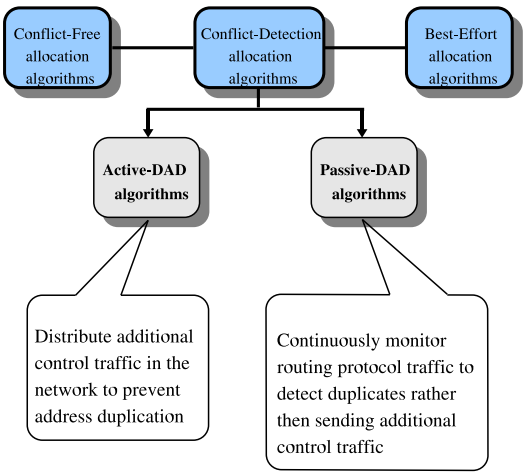


Figure 4.1: Duplicate Address Detection approaches for ad hoc networks

In the following sections we describe in detail the autoconfiguration protocol we proposed for OLSR. This protocol is divided into three steps and each step is detailed in a separate section.

4.2 Overview

In this section we are proposing a new autoconfiguration algorithm for OLSR based on a DAD approach. The main idea is that preventing address collision requires an assumption that some unique identifiers (*Node-ID*) are allocated to each node in the network. This new autoconfiguration algorithm is optimized to reduce the bandwidth utilization. As a matter of fact this approach uses the OLSR's MPR optimization to broadcast a new control packet(*MAD: Multiple Address Decalaration*) used by the DAD procedure. Moreover this approach is very simple and a formal proof of correctness is given.

Our proposed autoconfiguration algorithm for OLSR is based on three principles:

- *Initial address assignment*: an IP address is selected by the arriving node and the node can join the ad hoc network.
- *Duplicate Address Detection*: each node checks that there is not another node with the same address.

- *Conflict resolution*: when a node detects that another node is using the same address, it will select a new address.

In our approach, address assignment is relatively simple: In this first step, an IP address is selected by the arriving node and this latter can join the ad hoc network. Numerous schemes can be used to select this IP address. For instance the node can perform a random selection in a well known pool of addresses, without special message exchanges with its neighbors; another technique consists of one of its neighbors selecting the address on behalf of the arriving node. In Section 4.3.1 we discuss in detail various ways to assign an IP address to an arriving node.

After this first step has been performed, the second step can take place. The aim of this step is to detect potential address duplications on run. To perform this task a *Duplicate Address Detection* algorithm is started on this newly configured node. This DAD algorithm allows the newly configured node to state whether the selected address is duplicated or not in a proactive manner. The proposed DAD procedure is based on a special control packet called MAD(Multiple Address Declaration): it is emitted by each node, and includes one identifier and all the addresses of the node. This message is periodically transmitted to the entire network. The identifier of each node is assumed to be unique.

The central idea is that if there is a conflict between two nodes:

- One of the nodes in conflict will receive the MAD message from the other node in conflict: the MAD message received will include the address of the receiving node but it will have the identifier of the other node.
- The receiving node will deduce that the MAD message is not its own message and was sent by another node, hence that there is a conflict.

Because MAD messages should be sent to the whole network, and because OLSR has an optimized mechanism, called MPR-flooding, to transmit information to the whole network, it is natural to reuse this mechanism for MAD messages. However, the presence of conflicts may introduce deficiencies in the mechanism. These deficiencies could result in failure to detect the duplication of addresses. As an illustration of such possible situations, we give the following example.

Figure 4.2 shows two conflicting nodes $X1$ and $X2$ ($X1$ and $X2$ have the same address 5), in the 2-hop neighbors of node Z. In this configuration, node Y and node Z are not chosen as MPRs, then, the "Multiple Address Declaration" messages diffused respectively by node $X1$ and node $X2$ can not be propagated throughout the entire network. In our scenario, node Z could not calculate its

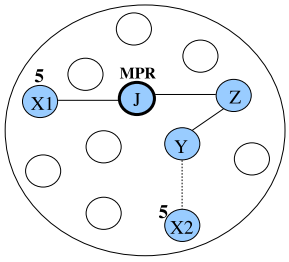


Figure 4.2: Address duplicate scenario

MPR set correctly, because MPR calculation is based on the assumption that there is no address duplication in the 1-hop and 2-hop neighbors. Consequently, node $X1$ and node $X2$ will not detect the address conflict, and the network remains corrupted.

Hence, an important contribution of our work is to introduce changes to the MPR-flooding mechanism, so that MAD messages are propagated effectively, and, equally important, that these changes allow duplicate address detection in all possible cases of conflicts. This modified version of MPR flooding takes into consideration the possibility that MAD originator addresses might be duplicated. It is only useful for "Multiple Address Declaration" message, but could be used in general for any kind of message which includes the node identifier. This version of MPR flooding is called *Duplicate Address Detecting MPR Flooding* or *DAD-MPR Flooding*.

4.3 Address assignment and resolution of conflicts

4.3.1 Initial address assignment

There are two ways to allocate an address to an arriving node. The first way is to allocate a random address to this node and then to rely on the DAD algorithm to discover any potential address duplication. If this address is duplicated another random address will be chosen. This allocation process terminates when the selected address is conflict free. The second way is for the new node to ask for the help of one of its neighbors to get an IP address. Since a configured node

receives the MAD messages of all the nodes in the network, it can maintain a pool of not already used addresses. It can give such an address to the requesting node. In principle there will be no duplication with this scheme except if, due to MAD messages loss, the proposed address is duplicated or in the case of simultaneous requests in different locations of the network. These two schemes can be simply analyzed. Let us denote by N_n the number of nodes in the network. These nodes have a unique and non duplicated addresses. Let us denote by N_a the total number of addresses in the pool of addresses. In the first technique, the probability that the chosen address will be duplicated is thus : $p = \frac{N_n}{N_a}$. If one denotes by D_1 the duration to detect a duplication, the average time to obtain a non duplicated address can be simply expressed as

$$\sum_{i \geq 1} (1-p) i D_1 p^{i-1} = D_1 \frac{1}{1-p} = D_1 \frac{1}{1-p} = D_1 \frac{1}{(1-\frac{N_n}{N_a})}$$

In the second scheme, to take into account the effect of transient packets loss we will suppose that a fraction h of the N_n configured nodes will not be known by a configured node. Thus when a neighbor replies to a requesting node with an address, the probability that the chosen address will be duplicated is : $p = \frac{hN_n}{N_a - N_n(1-h)}$. If one denotes by D_2 the duration to request an address and to detect a potential duplication, the average time to obtain a non duplicated address can be expressed as

$$\sum_{i \geq 1} (1-p) i D_2 p^{i-1} = D_2 \frac{1}{(1-\frac{hN_n}{N_a - N_n(1-h)})}$$

In Figure 4.3 we show the duration for a requesting node to obtain a not duplicated address. To take into account, in the second scheme, the duration of the exchanges between the requesting node and one of its neighbors we have supposed that $D_2 = 2D_1$. We have also assumed that 10% of the MAD packets are lost, thus we can assume that $h = 0.1$. We are considering an address pool of 256 addresses; $N_a = 256$. We can see in Figure 4.3 that except when there are few configured nodes, the second approach offers better performances. When there are a few configured nodes the probability of choosing a duplicated address is small and the overhead induced by requesting an address to a neighbor is predominant. When there are numerous configured nodes, the probability of choosing a duplicated address increases and the second scheme performs better than the first.

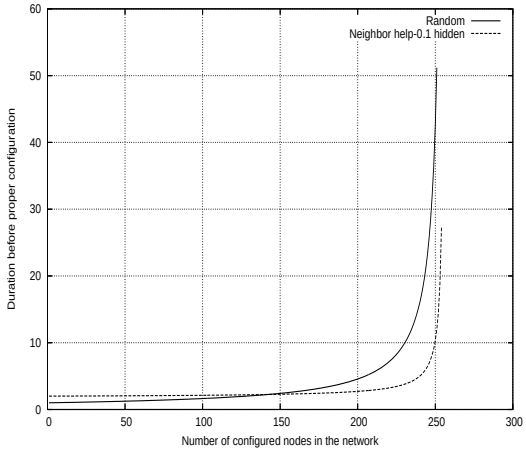


Figure 4.3: Duration for a first address assignment

4.3.2 Pool of addresses

The pool of addresses could be for local use only. For example, it could be reserved by the IANA [1] authority for local MANET forwarding (i.e, those addresses must not be forwarded outside the MANET network, nor reached from outside). A second possibility consists in relying on some machines which will announce the prefix to use for address autoconfiguration for this MANET network. These machines could be connected to the internet, and act as gateways. In this case, the addresses may be global addresses, and could be seen from outside.

4.3.3 Resolution of a conflict

When two nodes A_1 and A_2 are configured with the same IP address, and if we assume that there is no packet loss, each of these two nodes will receive the MAD message of the other node. Thus the nodes where the conflict lies are bound to discover the conflict. A simple rule to solve this conflict will be that the node in conflict with the smallest identifier changes its address. Since this node knows via the reception of the MAD control messages the already assigned addresses, the new address must be selected at random among the addresses that the node believes not to have been already assigned.

Additionally, if a fixed part of the first bits of the node identifier is set to the “priority” of the node, this will lower the probability that "important" nodes

[1] Internet Assigned Numbers Authority.

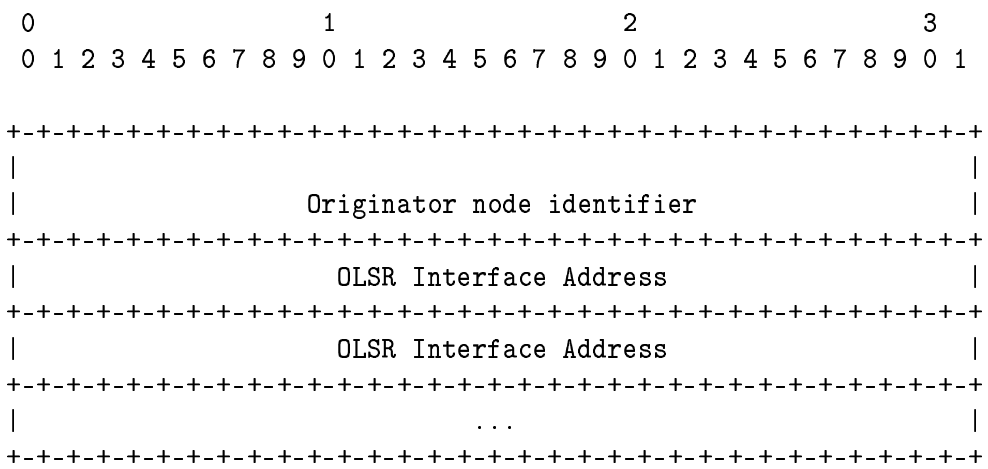


Figure 4.4: MAD message

would have to change their addresses.

4.4 Duplicate Address Detection

In order to detect address conflicts, each node diffuses a special message that we call MAD for "Multiple Address Declaration" to the entire network. This control message includes the node addresses and the node identifier (Figure 4.4). The node identifier is a sequence of bits of fixed length L which is randomly generated. Hence we are using the standard idea that the probability of two nodes having the same node identifier is low, and the probability of at least one address collision with N nodes, which is the well known "birthday problem" [66], can be set arbitrarily low by choosing a large enough value of L.

A node detects an address conflict when it receives a MAD message having the same address as its own, but with a different identifier. Actually other nodes, not involved in the conflict, will detect the address duplication by receiving two MAD messages having the same address, but holding different node identifiers as shown in Figure 4.5. These nodes could announce the detected conflict using a special control message as is done in the Pacman protocol [62]. However this approach may induce broadcast storm since many nodes may announce the conflict and special care must be taken to avoid this effect. For that reason we do not recommend this way.

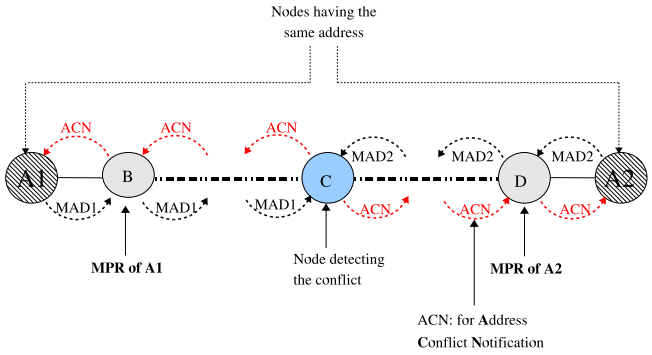


Figure 4.5: Conflict Notification

To save the channel bandwidth the MAD packets should be broadcasted using the MPR flooding mechanism of *OLSR*. However, applying OLSR relaying optimization rules as they are defined, may not be sufficient to ensure diffusion in some conflictual cases.

DAD-MPR flooding mechanism should have the property that it will continue to work even in the presence of duplicate addresses in the network. This property means that, in the absence of packet loss, the DAD-MPR Flooding will allow a MAD packet to reach all the nodes in the network.

The DAD-MPR flooding protocol comprises two parts: the first part describes some autoconfiguration rules for OLSR nodes with single interfaces, and the second part describes some additional autoconfiguration rules to handle OLSR nodes with multiple interfaces.

4.4.1 DAD-MPR Flooding in a single interface OLSR network

4.4.1.1 MAD relaying rules for a single duplicated address

First, let us assume that there are only two nodes with a duplicated address in the network. Let us denote by A_1 and A_2 these two nodes with same address A, but different node identifiers ID_{A_1} and ID_{A_2} and let us denote by d the distance, in number of hops, between node A_1 and node A_2. Let us assume that node A_1 and node A_2 send MAD messages M_1 and M_2 respectively. Moreover we will assume that there is permanent packet loss between a node and its neighbors.

Thus a packet broadcast by a node will eventually reach all its neighbors. This assumption is true in a connected MANET when the packet is sent periodically and the network load is kept below a reasonable level.

We intend to add special rules to the MPR flooding algorithm to handle the MAD messages. With these additional rules we must be in a position to prove that if there is no packet loss the MAD message of A_1 will be received by A_2 and vice versa the MAD message of A_2 will be received by A_1. The main problem here comes from the MPR calculation as we have already seen in the Figure 4.2. As a matter of fact, the MPR calculation is done on addresses so the MPR set of a node may not be properly calculated if it has address duplication in its 2-hop neighborhood. Therefore, the MPR set may not properly cover the 2-hop neighborhood of this node. In such a case, control messages using the MPR optimization may not be propagated to all nodes in the network.

Let us denote by N_i the set of nodes which are exactly at distance i of A_1 and at distance $d-i$ of A_2, for $i \in \{1, ..., d-1\}$. Those nodes are precisely the nodes which are on a shortest path from A_1 to A_2. Hence the sets N_i are never empty since a shortest path exists. An example of such sets is illustrated in Figure 4.6.

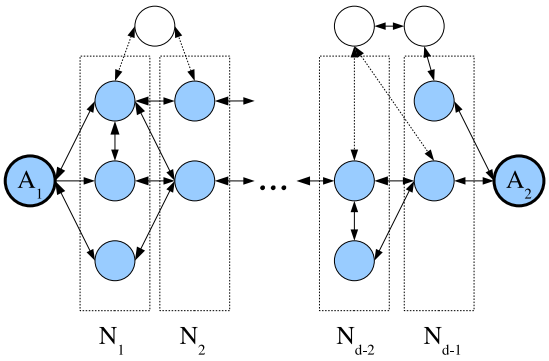


Figure 4.6: Example of Sets N_i

Then several cases can occur, depending on the distance d:

- $d \geq 5$

  In Figure 4.7 nodes A_1 and A_2 have the same address, and they are 5 hops away from each other. In this case nodes A_1 and A_2 and all the intermediary nodes do not have duplications in their 1-hop and 2-hop neighbors according to our assumption of only two nodes with a duplicated address. Hence all the intermediary nodes calculate their MPR set properly. So, using the MPR flooding the messages M_1 and M_2 will be correctly propagated to the

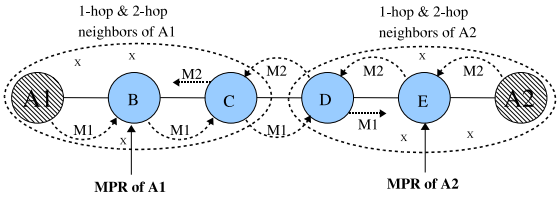


Figure 4.7: Single interface nodes: $d = 5$

nodes A_2 and A_1 respectively and the conflict will be detected. With the same consideration we can show that, in this case, all the MAD messages will reach all the nodes in the network.

- $d = 4$

  In Figure 4.8, nodes A_1 and A_2 do not have duplications in their 1-hop and

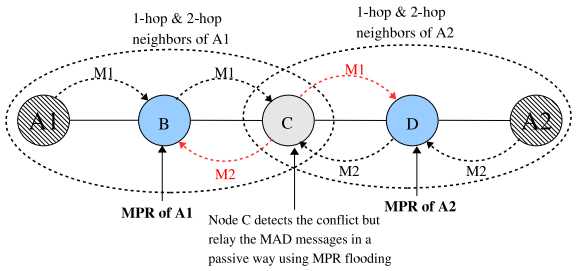


Figure 4.8: Single interface nodes: $d = 4$

2-hop neighbors according to our initial assumption. Therefore, the MAD messages diffused by A_1 and A_2 will reach node C. Node C can detect the conflict by examining the node identifiers contained in the received MAD messages. These messages should be relayed by C, because it has been chosen as a MPR by B and D. In our case, this is not trivial. In fact, messages generated by A_1 and A_2 may have close sequence numbers, which

may prevent one of the two sets of messages from being relayed by C (due to possible existence of a duplicate tuple indicating that such a message having the same sequence number and originator, has been received and processed). We need here to add an extra rule to allow MAD message relaying. We modify the MAD duplicate message detection, which will be based on the originator address, the message sequence number, plus the node identifier. Hence, A_1 and A_2 MAD messages will be forwarded by node C in all cases and reach node B and node D. Notice here that the MPR calculation of node C is affected due to the presence of duplicated addresses in its 2-hop neighbors (A_1 and A_2). Node C chooses only one node between B and D as a MPR to cover its 2-hop neighbors. The chosen MPR, should act like node C as described before to relay the MAD messages. Following this rule, we are sure that one of the two nodes (A_1 and A_2) receives the MAD messages generated by the other node and hence can detect the conflict. We call this rule *Rule 1*.
We will see with the next case, where $d = 3$, another rule that allows the two conflicting nodes to detect the address duplication.

- $d = 3$

  In case of distance $d = 3$, nodes B and C (Figure 4.9) do not need to choose

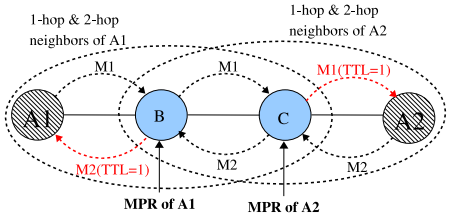


Figure 4.9: Single interface nodes: $d = 3$

a MPR to cover respectively their 2-hop neighbors A_2 and A_1 since they have the same address. Address A is considered as a one hop neighbor. In contrast, node A_1 chooses node B as a MPR to reach node C, and node A_2 chooses node C as a MPR to reach node B. In this situation, and thanks to the new rule described previously, the A_1 MAD messages will reach node C, and the ones generated by A_2 will arrive at node B. But, B and C do not choose each other as a MPR, consequently, A_1 can not receive MAD messages coming from A_2, and A_2 MAD messages will never reach node A_1. To tackle this problem, we add another rule that we call ,*Rule 2*, to enable MAD relaying for such situations, as follows: if a given node N receives a MAD message from a neighbor M, and M did not select N as a MPR, then,

node N will repeat this message if it detects that one of its 1-hop neighbors has the same address as the one contained in the MAD message. The MAD TTL value is put to 1 to avoid the transmission of the MAD message beyond the conflicting nodes.

- $d = 2$

  In Figure 4.10, the node B detects the duplication because the nodes $A1$ and $A2$ are in its 1-hop neighborhood, it proceeds by the same manner as in the case of $d = 3$. Thus node A_1 will receive the MAD message of A_2; and node A_2 will receive the MAD message of A_1.

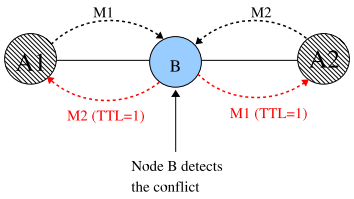


Figure 4.10: Single interface nodes: $d = 2$

- $d = 1$

  This is the simplest case and because node A_1 and node A_2 are in the radio range of each other, the conflict will be detected by both of them. See Figure 4.11.

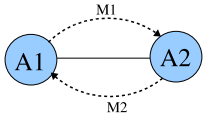


Figure 4.11: Single interface nodes: $d = 1$

In the previous analysis, we considered the case of a unique couple of nodes having the same address in the network. Notice that, if we have a single couple of nodes having the same address in a 2-hop neighborhood of each node in the network, the previous reasoning continues to work correctly. In fact, with the previous reasoning, any 2-hop conflict is detected and fixed. Hence, the MPR flooding mechanism will work correctly. Consequently, MAD messages can be delivered to any other node in the network beyond the previously MPR corrupted area and finally other possible conflicts can be resolved.

The general case of multiple conflicts is treated in Section 4.4.1.2.

4.4.1.2 Case of multiple conflicts

By multiple conflicts we mean, that we may have more than a single duplicated address in the network, knowing that a duplicated address, could be shared by several nodes at the same time. In the case of duplicated address shared by more than two nodes in the network, conflicts are detected and fixed couple by couple by applying the previous rules cited in Section 4.4.1.1. Eventually, this kind of conflict is resolved. Nevertheless, the previous rules are not sufficient for the special case of loops as depicted in Figure 4.12. In fact, in Figure 4.12, each

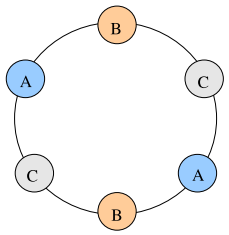


Figure 4.12: Case of multiple conflicts

node considers that it has only two neighbors at 1-hop distance and no 2-hop neighbors (i.e the network seen by node A is composed by direct neighbors B and C). None of the nodes present in this network will be elected as a MPR. Hence, MAD messages will not be relayed and never reach other conflicting nodes or at least a neighbor of a conflicting node. In that case the previous rules will not ensure the relaying of MAD messages between nodes in conflict.

To handle multiple conflicts, we add a third rule to the classical MPR flooding mechaninsm. The property that we add is actually simple. We weaken the relaying condition for nodes who are in the 1-hop neighborhood of a node who is sending a MAD message. When these neighbor nodes receive a MAD message,

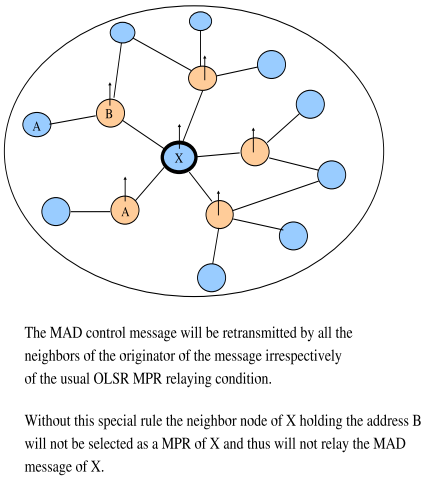


Figure 4.13: All the 1-hop neighbors of the originator of the MAD message will relay the message

they must relay it irrespectively of the relaying conditions of the OLSR MPR flooding algorithm (Figure 4.13).

We call this rule *Rule 3*. Actually, this rule covers the *Rule 2*. In fact, when a node $N1$ receives a MAD message from a neighbor node $N2$, and if the node $N1$ detects that one of its 1-hop neighbors, say N, has the same address as the one contained in the MAD message but with a different node identifier, $N1$ may act as if the MAD message was orginated from node N and hence, it applies the *Rule 3*. That is why we merge these two rules into a single rule in Section 4.4.1.3.

With these three rules, we will be in the position to prove the correctness of the DAD-MPR flooding algorithm. More precisely in the absence of packet loss a MAD message will finally reach all the nodes in the network.

Proof of correctness of DAD-MPR flooding algorithm with multiple conflicts

Let's denote: A, B, C, D, ... the nodes and '1', '2', ... the addresses. Let's denote 'A{1}' the node 'A' with the address '1' and so on ...
In this part, the conflicts with distance ≤ 2 are obviously resolved. In addition, the proof given in Section 4.4.1.1, can be adapted to the context of multiple conflicts by applying *Rule 2* and *Rule 3*. Thus any conflict at distance ≤ 3 is detected and then resolved.

Case of d = 4

Lemma 1: in a permanent 4-hop conflict represented by the topology A{1}–B{2}–C{3}–D{4}–E{1}, neither B nor D chooses C as MPR. In other terms, in a 4-hop conflict between two given nodes, the node(s) at the center of the conflict isn't chosen as MPR by the neighbors of the conflicting nodes. A node "in the center of conflict" is defined to be exactly 2-hop away from the both nodes in conflict. At least one such node exists by definition of "4-hop conflict".

Proof: by contradiction.

Assume there is a permanent 4-hop conflict: A{1}–B{2}–C{3}–D{4}–E{1} and C is MPR of B for instance (case with D is symmetrical).

Then:

- The node A originates one MAD message
- The node B retransmits it: because it is a 1-hop neighbor of the MAD message originator (*Rule 3*)
- The node C retransmits it: because it is a MPR of B
- The node D retransmits it: because it detects the conflict and is one hop away from the other node in conflict (*Rule 2*).

which results in the conflict being resolved, which itself is contradictory with the "permanent 4-hop conflict" hypothesis.

Lemma 2 : in a permanent 4-hop conflict which can be represented by the topology : A{1}–B{2}–C{3}–D{4}–E{1}, there must be some nodes X and Y such as the topology includes: X{4}–Y–B{2}–C{3}–D{4}–E{1} and Y is MPR of B.

Proof:

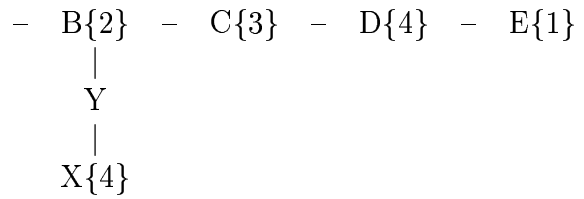
Assume there is a permanent 4-hop conflict which can be represented by the topology: A{1}–B{2}–C{3}–D{4}–E{1}. *Lemma 1* shows that C is not MPR of B (nor D, incidentally). And then since D{4} is a 2-hop neighbor of B via C, proper MPR selection in B requires that:

(a) Some node with address 4 is covered by another 1-hop neighbor, chosen as MPR by B.

(b) OR some node with address 4 is a neighbor of B.

The last case (b) is impossible because, otherwise, there would be a topology such as X{4}–B{2}–C{3}–D{4}–E{1}, which is a 3-hop conflict [2].
So (a) must be verified : let us denote X{4} the other 2-hop neighbor of B with address 4, and Y the 1-hop neighbor to reach it (the MPR chosen by B) [hence a part of the topology is represented by B{2}–Y–X{4}].

Then the topology includes:



Which proves the lemma. [Notice here that, the proof is still valid if Y happens to be A].

Theorem: there are no permanent 4-hop conflicts.

Proof: by contradiction.

Assume there is a permanent 4-hop conflict which can be represented by the topology: A{1}–B{2}–C{3}–D{4}–E{1}. Then according to *lemma 2*, there exist nodes X and Y such that: X{4}–Y–B{2}–C{3}–D{4}–E{1} and Y is MPR of B.

Now by noticing that a subgraph of this topology is : X{4}–Y–B{2}–C{3}–D{4}–... (a 4-hop conflict with address 4), we can apply *lemma 2* on this topology again: thus, there exist nodes U and V such that: U{3}–V–Y–B{2}–C{3}–D{4} (and V is MPR of Y) is part of the graph.

Now notice that the topology includes the subgraph of a 4-hop conflict with address 3: U{3}–V–Y–B{2}–C{3}, but this time with the noteworthy fact that Y is MPR of B{2} (this fact comes exclusively from the first application of *lemma 2*). But this fact is in contradiction with the *lemma 1* applied to the 4-hop conflict between U{3} and C{3}: indeed the *lemma 1* indicates (proves) that Y shall choose neither V, nor B{2} as MPR. The contradiction shows that the hypothesis that "there is a permanent 4-hop conflict" is impossible, hence the theorem.

Case of d $\geq$ 5

We have just shown that all conflicts between nodes at distance $d = 4$ are resolved. Thus after a given time, there will not be any 2-hop conflict remaining for the MPR selection on the route between two nodes at five hops aways or more. Thus the classical rules of the MPR flooding are sufficient to ensure that

[2] Recall that any conflict with distance ≤ 3 is resolved.

the MAD messages of two nodes at distance $d \geq 5$ will be exchanged between these two nodes. In other words, conflict between nodes at distance $d \geq 5$ are thus detected with MAD messages.

To be completly rigorous we have shown the previous results with the hidden assumption that when a conflict is detected, it is then successfully resolved. Thus to obtain the previous results we have to detect and resolve the conflicts at distance $d = 2$, then $d = 3$, then again $d = 4$ and finally at distance $d \geq 5$. This asssumption might not be true, if the resolution of a conflict leads to another conflict. This might happen when there is only a very small fraction of free available addresses in the pool. To overcome this problem we can use random address assignment. In such a case the process should eventually terminate with a network without address conflict.

4.4.1.3 Specification of the DAD-MPR Flooding algorithm

Let us recall the assumptions here.

Each node A periodically sends a MAD message M including:

- The originator address of A, $Orig_A$, in the OLSR message header.
- The message sequence number, $mssn$, in the OLSR message header.
- The node identifier ID_A (a string of bits) in the message itself.

The message is propagated by MPR flooding to the other nodes ; but for DAD-MPR Flooding, the duplicate table of OLSR is modified, so that it also includes the node identifier list in the duplicate tuple. That is, a duplicate tuple, includes the following information:

- The originator address (as in OLSR standard duplicate table).
- The message sequence number (as in OLSR standard duplicate table).
- The list of node identifiers.

The detailed algorithm for DAD-MPR Flooding is the following:

- When a node B receives a MAD message M from node C with originator $Orig_A$, with message sequence number $mssn$, and with node identifier ID_A, it performs the following tasks:

1. **If** a duplicate tuple exists with the same originator $Orig_A$, the same message sequence number, and ID_A is in the list of node identifiers, **Then**, the message is ignored (it has already been processed). The algorithm stops here.
2. **Else** one of the following situations occurs :
   (a) A duplicate tuple exists with the same originator $Orig_A$ and the same message sequence number, but ID_A is not in the list of node identifiers: then, a conflict is detected (address $Orig_A$ is duplicated). ID_A is added to the list of node identifiers.
   (b) No duplicate tuple exists with the same originator $Orig_A$, and the same message sequence number $mssn$. A new one is created with the originator address, message sequence number and list of node identifiers containing only ID_A.
3. The MAD messages should be relayed if one or more of the following rules are met:
   (a) C had chosen this current receiving node, B, as a MPR (as in normal MPR flooding).
   (b) The node B has a link (symmetric or asymmetric) with the originator address, $Orig_A$, contained in the MAD message M (*Rule 2* and *Rule 3*).

4.4.2 DAD-MPR Flooding in a multi-interface OLSR network

4.4.2.1 Interfaces and Addresses

In OLSR [55], a node may have several interfaces which participate in the OLSR network. This situation results in more difficult algorithms, processing, and the need for more additional terminology (addressed in the OLSR specifications).

Figure 4.14 is an example of a network with three nodes X, Y, and Z where two nodes X and Y have multiple interfaces (X^1, X^2 and Y^1, Y^2 respectively).

Each OLSR node has a different address for each of its interfaces. This address is called the *Interface Address*. For instance, in Figure 4.14, the interface address of the interface X^1 of the node X is `@1`. Each node arbitrarily chooses one unique interface address as its *Main Address*, which will be used as the originator address of the messages of the node. In Figure 4.14, node X has chosen the address of

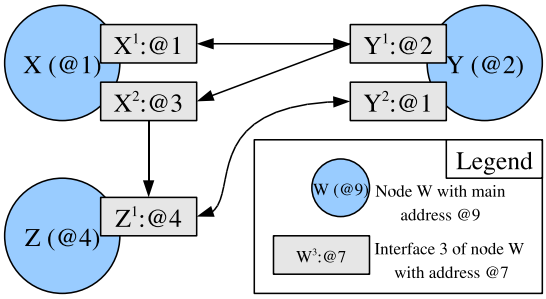


Figure 4.14: Example of links between neighbor nodes X, Y and Z

interface X^1, `@1`, as its main address, node Y has chosen the one from Y^1, `@2`, and node Z has chosen `@4` from Z^1.

In the rest of this document, the same conventions will be used: nodes are denoted with letters such as X, interfaces are denoted with node names with indices such as X^1, and addresses in general (main addresses or interface addresses) are denoted with the prefix `@`, such as address `@1`.

4.4.2.2 Links and Neighbors

In contrast with the situation where OLSR nodes have a unique interface, in the context of multiple interfaces, the distinction between "links" and "neighbors" is necessary. A *Link* represents the physical connection between two interfaces (of different nodes), i.e. the fact that the packets from one interface reach another interface. Links can be symmetric or asymmetric: a link is *Symmetric* when communication is possible in both directions, that is, the packets sent on each interface will reach the other one and vice-versa. In the opposite case, the link is unidirectional, communication is only possible from one given interface to the other, and the link is then called *Asymmetric*. In Figure 4.14, the arrows are meant to specify in which directions the traffic flows: the links between interfaces X^1 and Y^1, between Z^1 and Y^2 are symmetric ; while the links (Y^1, X^2) and (X^2, Z^1) are asymmetric.

Based on the existing links, the term *Neighbor* is used to denote the fact that one node has at least one interface which has a link with one interface to the Neighbor node. It is a *Symmetric Neighbor*, when there is at least one such link which is symmetric, otherwise it is a *Not-Symmetric Neighbor* (from the terminology of [55]). In Figure 4.14, X and Y are symmetric neighbors due to the link (X^1, Y^1), Y and Z are symmetric neighbors with the link (Y^2, Z^1) while X and Z are not-symmetric neighbors with the asymmetric link (X^2, Z^1).

4.4.2.3 Details of OLSR Protocol

In this section, the link sensing and neighbor discovery protocols of OLSR are described in detail, as such details affect the proofs.

The link sensing and neighbor discovery automatically detect the links, the neighbors and the 2-hop neighbors. In a node X, the information obtained by each interface about links is maintained inside a *Link Set*, where each link is associated to a *Link Tuple* which includes the following information:

- the interface address of the node (termed *local interface address*).
- the interface address of the other end of the link, i.e. the interface address of the neighbor, that was heard (termed *neighbor interface address*).
- the *main address* of the neighbor.
- the status of the link: symmetric, asymmetric, lost[3]

The *Neighbor Set* holds information in *Neighbor Tuples* which is deduced from the information obtained from the link set and from exchanged messages:

- Any neighbor which has a link (i.e. a link in the node's link set) has a neighbor tuple.
- The neighbor tuple includes the *Neighbor Status*: symmetric or not symmetric. A neighbor is symmetric if and only if there is at least a link tuple to it with status symmetric.
- The neighbor tuple includes the main address of the neighbor.

The *2-Hop Neighbor Set* holds information in *2-Hop Neighbor Tuples* obtained from the received *Hello* messages. A 2-hop neighbor tuple includes:

- the main address of the neighbor
- the main address of the 2-hop neighbor

As an example of such sets for the topology in Figure 4.14, the tables for some nodes are given: tables 4.1 and 4.2 represent the link set for nodes X and Y, table 4.3 represents the neighbor set for node Y, and table 4.4 represents the 2-hop neighbor set for node Z.

[3]The "lost" status is ignored in this article: indeed, it can appear transiently only as a result of topology changes or multiple message losses, which are conditions excluded from the hypothesis made for the formal proofs.

Information	For (Y^1,X^1)	For (Y^1,X^2)
local interface address	@1	@3
neighbor interface address	@2	@2
neighbor main address	@2	@2
status	sym	asym

Table 4.1: Example of Link Set (for X)

Information	For (X^1,Y^1)	For (Z^1,Y^2)
local interface address	@2	@1
neighbor interface address	@1	@4
neighbor main address	@1	@4
status	sym	sym

Table 4.2: Example of Link Set (for Y)

Information	For X	For Z
neighbor main address	@1	@4
status	sym	sym

Table 4.3: Example of Neighbor Set (for Y)

Information	For X
neighbor main address	@2
two hop main address	@1

Table 4.4: Example of 2-Hop Neighbor Set (for Z)

These sets are filled as a result of exchanging *Hello* messages. The generation and processing of *Hello* messages is done as follows:

Generation of Hello messages

- A HELLO message is generated by node X for each interface X^i.
- The HELLO message includes the information from the link tuples on this interface X^i, with additional information of the neighbor (as a way of compressing redundant information):

  (neighbor interface address, link status, neighbor status, neighbor MPR status)

  The neighbor interface address and link status originate directly from the link tuple, the neighbor status from the associated neighbor tuple, and the neighbor MPR status from the MPR list (and announces whether or not the neighbor has been selected as an MPR).
- Because *Hello* messages are used for 2-hop neighbor discovery, neighbors which do not have a link to the interface where the HELLO is generated (but to some other interfaces) are still advertised with the information that there is no link.

***Processing of Hello messages**: Link Sensing*
When a node X receives a *Hello* message from node Y:

- The interface address of the sender `@y` and the interface address of the receiver `@x` are known.
- The *Hello* message contains the main address, denoted `@Y`, of Y, as part of the message header.
- If the interface address of the receiver is itself in the HELLO message with status symmetrical or asymmetrical, the receiver deduces that a HELLO message had been transmitted previously exactly in the opposite direction. Hence a link tuple will be created (or updated) indicating that the link (`@y`, `@x`) is symmetrical (and to a neighbor with the main address `@Y`).
- If the interface address of the receiver is not itself in the HELLO message, the receiver deduces that the link is asymmetrical (at least for now), and a link tuple will be created (or updated) indicating that the link (`@y`, `@x`) is asymmetrical (and to a neighbor with the main address `@Y`).

***Processing of Hello messages**: Neighbor Discovery*
When a node X receives a *Hello* message from node Y:

- The *Hello* message contains the main address, denoted `@Y`, of Y, as part of the message header.
- After the link between X and Y (`@x`, `@y`) has been updated, node X checks again the neighbor status of node Y: if at least one link exists between X and Y which is symmetrical, then Y is a symmetric neighbor. Otherwise it is a "not-symmetric" neighbor.

***Processing of Hello messages**: 2-Hop Neighbor Discovery*
When a node X receives a *Hello* message from node Y:

- The *Hello* message contains the main address, denoted `@Y`, of Y, as part of the message header.
- This *Hello* message contains the neighbors of the node Y, which are by definition the 2-hop neighbors of X.
- The small caveat is that the message contains the addresses of the *interfaces* of the neighbors of Y. Hence, X first converts these interface addresses into main addresses, using the MID/MAD information base.
- After converting the interface address into a main address `@Z`, node X knows that there is a 2-hop neighbor with the main address `@Z`, reachable by the neighbor with the main address `@Y`: it will ensure that a proper 2-hop tuple with these addresses exists.

Let us now consider the case of nodes with multiple interfaces. The flooding rules for MAD messages described in Section 4.4.1 are not sufficient to handle address duplicates in a network with nodes that have multiple interfaces.

In the following we introduce other modifications to the OLSR MPR flooding algorithm than those introduced in Section 4.4.1. The aim of these modifications is to ensure that MAD messages are actually propagated throughout a network of nodes having multiple interfaces. This new MPR flooding algorithm for MAD messages diffusion is detailed in the following section.

4.4.2.4 MAD relaying rules and multiple interfaces

In this section we need the following definitions:

Definition 1: Two nodes are in conflict if, at least, there exists an interface address shared by the two nodes.

Definition 2: A node X has a non conflicting symmetric neighborhood if each of its symmetric neighbors is not in conflict with a symmetric or asymmetric neighbor of the node X.

Due to the specific processing involved when $OLSR$ uses multi-interfaces nodes, we need to slightly modify the relaying rules proposed in Section 4.4.1. We propose to use the following rules:

Rule 1: When a node X receives a MAD message and if node X has a symmetric or asymmetric link with a node Y with the same main address as the address contained in the MAD message, then node X relays this MAD message. When relaying the MAD message the $hop-count$ field is set to one.
The $hop-count$ field is set to 1 to handle the case of wrong calculation by node Y of its 2-hop neighbors identifiers due to late delivery, for some reason, of MAD messages. In Figure 4.15, if h=0 this means that the MAD message is originating

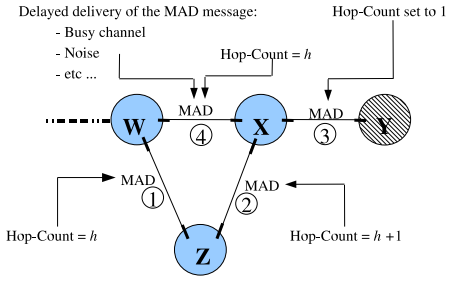


Figure 4.15: Multi-interfaces nodes: $hop-count$ field set to 1

from node W. In such a case node X receives the MAD message from node Z with $hop-count = 1$. But node X is also a neighbor of node W and should, in principle, receive the MAD message directly from node W with $hop-count = 0$ and before the one relayed by node Z. The node X relays this MAD message to node Y with $hop-count$ field $= 2$. Hence, the mapping between 2-hop neighbors main addresses and their corresponding identifiers may be affected within node Y. This is why it is necessary to set the $hop-count$ field of a MAD message to 1 before its retransmission by a neighbor of the main address contained in the MAD message.

Rule 2: When a node X receives a $HELLO$ message from a node Y, this $HELLO$ contains interface addresses of 2-hop neighbors of X(1-hop neighbors of Y). To convert such addresses into main addresses the node X uses MAD messages that are exclusively relayed by Y, and that originate from these 2-hop neighbors (that is, received with a hop-count field equal to 1). Rule 2 will actually avoid inconsistent main address conversions for 2-hop neighbors in node X and hence, will lead to a correct calaculation of its MPRs.

An example illustrating Rule 2 is presented here. In Figure 4.16, node X

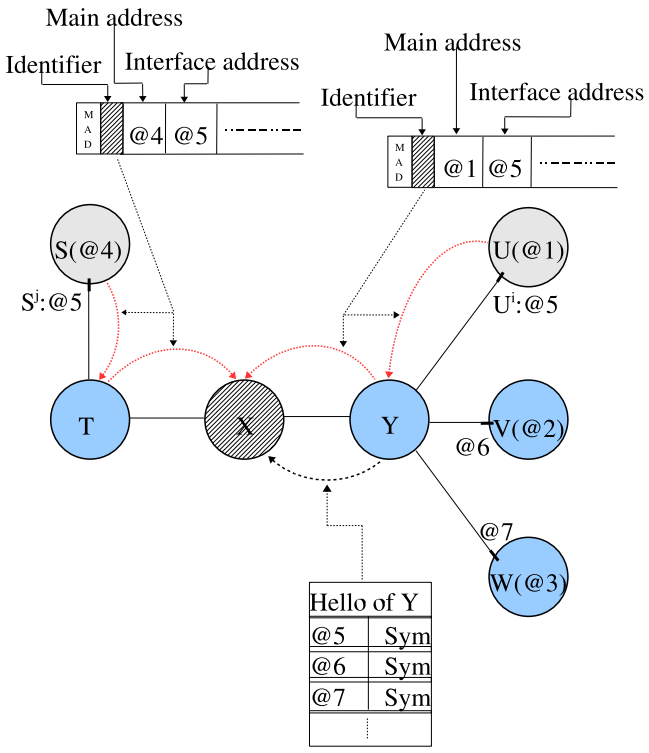


Figure 4.16: Multi-interfaces nodes: an illustration of Rule 2

receives a $HELLO$ message from node Y. This $HELLO$ contains the interface address, U^i:@5, of the 2-hop neighbor of X, node U. The question here is, what is the main address of @5 since this address is duplicated and shared between the interfaces U^i and S^j ? To convert this address into its actual main address, node X should use the MAD message relayed by Y and originating from node U. Otherwise, node X uses the MAD message relayed by T and originating from node S and, therefore, the conversion will be wrong. Applying Rule 2, node X constructs a 2-hop tuple for the address U^i:@5("main address of $Y \rightarrow$ main

address of U(@1)").
Now when the node X receives a $HELLO$ message from node T, it proceeds in the same manner to bind the address S^j:@5 to its actual main address. Node X uses the MAD message coming from T to construct a 2-hop tuple for the address S^j:@5("main address of $T \rightarrow$ main address of S(@4)").

4.4.2.5 DAD-MPR flooding algorithm and proof of correctness

We assume that there can be an arbritrary number of nodes having multiple interfaces with a duplicated address in the network. We also assume that each node in the network picks a globally unique random identifier.

With the previous rules, we will be in the position to prove the correctness of the DAD-MPR flooding algorithm. More precisely in the absence of packet loss a MAD message will finally reach all the nodes in the network.

To prove the correctness of the proposed algorithm, we consider all the cases of the distance d between the closest conflicting nodes in the network.

- **d = 1**

  Lemma 1: If the neighbor table of a node A contains a symmetric neighbor X, then there exists physically at least one symmetric link between A and X.

  Proof:

  - Because node A has node X as a symmetric neighbor in its neighbor table, node A has necessarily received a $HELLO$ message from node X indicating that A is a symmetric or asymmetric neighbor of node X.
  - Because X has sent such a message, it necessarily has an entry in its neighbor table for a symmetric or asymmetric neighbor B with an interface address @1, indicating that X has received a $HELLO$ message from B. Then two cases can occur:
    1. Node B is actually node A and in this case the lemma is verified.
    2. Or nodes B and A are two asymmetric neighbors of node X, with the same address @1 as explained in Figure 4.17. By applying the relaying rule, Rule 1, the conflict will be detected and in this case it is node A that changes its address.

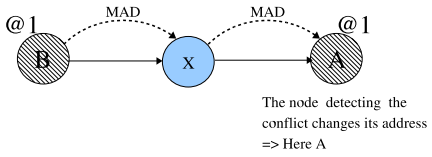


Figure 4.17: Multi-interfaces nodes: Lemma 1

Lemma 2: Running the DAD-MPR flooding algorithm ensures that each node in the network has a non conflicting 1-hop symmetric neighborhood (i.e the detected conflicts will be resolved).

Proof: Proof by contradiction
We assume that a symmetric neighbor, say $X1$, of a node A is in conflict with a symmetric or asymmetric neighbor, say $X2$, of the same node A (Figure 4.18).

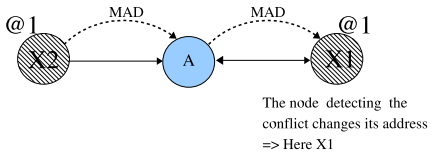


Figure 4.18: Multi-interfaces nodes: Lemma 2

- By applying Rule 1, node A receives and relays the MAD messages of node $X2$.
- The symmetric neighbor $X1$ detects the conflict and changes its address.

Hence the 1-hop neighborhood of node A contains no duplications.

- **d = 2**

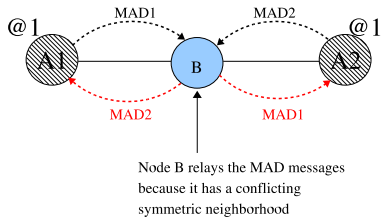


Figure 4.19: Multi-interfaces nodes: $d = 2$

This case is proved by applying Lemma 2, because such a situation implies that a node, say B, as in Figure 4.19 has a conflicting symmetric neighborhood.

- **d = 3**

  As shown in Figure 4.20, the MAD message sent by A is relayed by B and

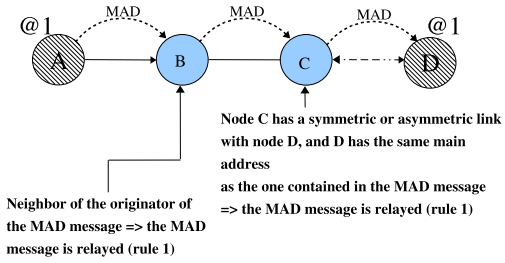


Figure 4.20: Multi-interfaces nodes: $d = 3$

C according to Rule 1. Thus node D changes its address.

- **d = 4**

  Let us now suppose that the nodes holding some duplicated addresses are 4 hops away from each other. For notation convenience we call these nodes

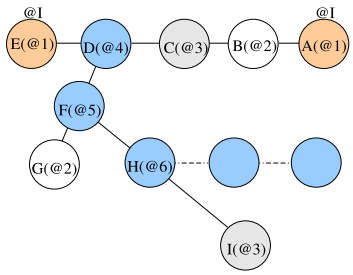


Figure 4.21: Multi-interfaces nodes: $d = 4$

node A and node E, see Figure 4.21. By definition there is at least one path of 4 hops from node A to node E. We can assume that the three nodes on this path have non duplicated addresses, otherwise we will fall in the previous cases of a distance between nodes with address duplication of 3 hops or less. We know that in this case the duplication is detected and then resolved.

If node D selects node C as its multipoint relay then the node E's MAD message will reach node A. Thus the address duplication between nodes A and E will be detected and then resolved.

Lemma 3: If the address conflicts between A and E are not conflicts of main addresses, they do not introduce defaults in MPR selection of node C.

Proof: By hypothesis, A and E have different main addresses, and they will send MAD messages, each with different (originator) main address. B and D, the neighbors of, respectively, A and E, will send HELLO messages, which may include a conflicting interface address `@I` of A and E respectively. However, because of Rule 2, C will use the MAD coming from B (i.e. originating from A), to convert the address `@I` in HELLO messages of B into a main address. The address obtained is then the main address of A, and C deduces that it has a 2-hop neighbor with the main address of A via node B. Similarily, using MAD messages coming from D and HELLO messages originated by D, C will deduce that it has a 2-hop neighbor with the main address of E via node D. Because the 2-hop neighbors A and B have different main addresses, the MPR calculation in C will cover them properly, hence the lemma is proved.

Lemma 4: If D doesn't select C as its MPR there necessarily exists another neighbor F of D that D has selected as a MPR to reach a node with a main address @2.

Proof: We have to consider the following two cases, either this node is actually B or it is not.

1. If it is B then the MAD message reaches node B and node A according to Rule 1.
2. If it is not B, it is another node denoted G that is in conflict with B (notice that the conflict is on the main address).

Lemma 5: If no (more) conflicts are detected on topology (E, D, C, B, A) then there exists a node F which is chosen as a MPR by node D to reach a node G with the main address @2.

Proof: Applying Lemma 4, either the conflict $A \leftrightarrow E$ is detected and then resolved or there are such nodes (F and G).
By hypothesis, the conflict between A and E is not detected and hence not resolved.

We prove by contradiction that 4-hop conflicts cannot occur. Let us assume a stable network situation where no (more) conflicts are resolved by MAD detection, and let us assume there is a 4-hop conflict on a network (E,D,C,B,A).

1. Applying Lemma 5 on the network (E,D,C,B,A), there must be a node F chosen as a MPR by D to reach a node G with the main address @2 (and $G \neq B$).
2. There is now a 4-hop conflict between B and G. By hypothesis, this conflict is not detected. Lemma 5 can be applied to the nodes (G,F,D,C,B) and hence there must be a node H chosen as a MPR by F to reach a node I with the main address @3.
3. But then because F is a MPR of D, the MAD messages from C will reach I : a conflict will be detected, which contradicts the hypothesis.

- **d $\geq$ 5**

  According to the previous lemmas, we know that conflicts between nodes at distance $d \leq 4$ will be detected and resolved. Thus the 2-hop neighborhood of any node will not contain address conflicts. According to Rule 2, the interface addresses of 2-hop neighbors of any node are correctly mapped into their corresponding main addresses. Therefore the MPR set of any node is correctly computed. Hence the MAD messages of two nodes in conflict at a distance $d \geq 5$ will be simply relayed according to the MPR flooding.

4.4.2.6 Specification of the DAD-MPR Flooding

Each node A periodically sends a message M that includes:

- The originator address of A, $Orig_A$, in the OLSR message header.
- The list of interface addresses of node A in the message itself.
- The message sequence number, $mssn$, in the OLSR message header.
- The node identifier ID_A (a string of bits) in the message itself.

The duplicate table of OLSR is modified, so that it also includes the node identifier list in the duplicate tuple. That is, a duplicate tuple, includes the following information:

- The originator address (as in OLSR standard duplicate table).
- The message sequence number (as in OLSR standard duplicate table).
- The list of node identifiers.
- The list of interface addresses from which the MAD message was received.

The DAD-MPR flooding algorithm for multi-interfaces nodes is the following:

- When a node B receives a MAD message M on its interface B^i from node C with originator $Orig_A$, with message sequence number $mssn$, and with node identifier ID_A, it performs the following tasks:
  1. **If** a duplicate tuple exists with the same originator $Orig_A$, the same message sequence number $mssn$, the ID_A is in the list of node identifiers, and the interface address of B^i is in the list of interface addresses from which the message has already been received, **Then**, the message is ignored (it has already been processed). The algorithm stops here.
  2. **Else** one of the following situations occurs :
     (a) A duplicate tuple exists with the same originator $Orig_A$, the same message sequence number $mssn$, ID_A is in the list of node identifiers, but the address of B^i is not in the list of interface addresses from which the MAD message has already been received: then, the message must be processed. The address of B^i is added to the list of interface addresses from which the message has already been received.

(b) A duplicate tuple exists with the same originator $Orig_A$ and the same message sequence number, but ID_A is not in the list of node identifiers: then, a conflict is detected (address $Orig_A$ is duplicated). ID_A is added to the list of node identifiers. B^i is added to the list of interface addresses from which the MAD message has already been received if it does not already exist in this list.

(c) No duplicate tuple exists with the same originator $Orig_A$, and the same message sequence number $mssn$. A new one is created with the originator address $Orig_A$, message sequence number $mssn$, list of interface addresses from which the MAD message has already been received containing only the address of B^i, and list of node identifiers containing only ID_A.

3. The MAD messages should be relayed if one or more of the following rules are met:

(a) C had chosen this current receiving node ,B, as a MPR.

(b) Node B has a symmetric or asymmetric link with a node Y with the same main address as the address contained in the MAD message M (Rule 1). The $hop - count$ field of the MAD message is set to 1 before its retransmission.

- When a node X receives a $HELLO$ message from a node Y, this $HELLO$ contains the interface addresses of the 2-hop neighbors of X. Node X uses MAD messages relayed by Y, and that originate from these 2-hop neighbors, to convert such addresses into main addresses.

4.4.2.7 Alternate MAD relaying rules

Section 4.4.2.4 presented a solution for relaying MAD messages, which relied on two rules: the first rule was a rule for repeating the *MAD* messages, from and to neighbors ; the second rule was a rule for MPR calculation.

One issue with this previous approach is that 2-hop conflicts must be resolved before one can be sure that the MAD messages are successfully transmitted over the entire network. An ideal property would be that the MAD messages reach all the nodes in the network irrespectively of potential address duplications. This property can be achieved if the MPR flooding continues to work in the presence of address duplication. One solution is therefore to base the selection of MPRs not on addresses but on node identifiers. With the assumption that node identifiers are globally unique in the network, one can be sure that there will not be identifier duplications at two hops from a given node and thus the selection of MPRs will

be correct. This solution can be simply implemented, the selection of the MPRs must follow the principle defined in the OLSR protocol except that the basis for selection must be the node identifiers i.e. the 2-hop coverage must be considered not on the addresses but on the node identifiers.

This is achieved in this section by providing an alternative to the second rule. Rule 2 is replaced by a *Rule* 2^{bis}:

> **R**ule 2^{bis}: The MPR calculation is modified, by using node identifiers. Each address is converted into a node identifier (using a method described later): as a result the node computing its MPR set, has its 1-hop and 2-hop topology represented by links between node identifiers.

(a) **Proof**

The previous proofs (for different distances) for Rule 1 and Rule 2, apply for Rule 1 and Rule 2^{bis} except for the case of the distance d = 4. In the case of d = 4, however, because the MPR calculation is performed on node identifiers, and because node identifiers are theoretically unique , there can be no node identifier duplication, and no defective MPR selection. Therefore, the DAD messages from a conflicting node will reach the other conflicting node which is 4 hops away, and hence no such conflict can persist indefinitly.

(b) **Method of address conversion to node identifiers**

The goal is to obtain the 1-hop neighborhood and the 2-hop neighborhood with node identifiers in place of addresses.

Converting main addresses of 1-hop neighbors to node identifiers is easily done: when receiving the *MAD* messages from neighbors, the main address can be identified to be the address of a neighbor, and the node identifier is given. Hence the node may record the information mapping the main addresses of neighbors to their identifiers.

Converting main addresses of 2-hop neighbors to node identifiers is less direct: however the information is obtained thanks to the fact that *MAD* messages from 2-hop neighbors are always retransmitted by 1-hop neighbors. Such *MAD* messages are identified by the fact that they arrive with a $hop - count$ field (corrected by Rule 1 if necessary), equal to 1 ; in the following, they are called *2-hop MAD messages*. The receiver node can thus maintain the information *2-Hop Identifier Table*: (1-hop neighbor main address, 2-hop neighbor addresses list, 2-hop neighbor identifier) in or in addition to, the MID/MAD information base. Now taking advantage of the fact that conflicts at distance

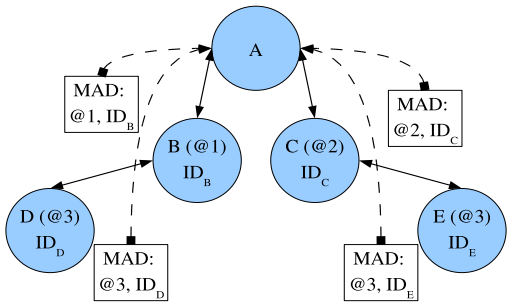


Figure 4.22: Example of topology

1 to 3 are resolved anyway by Rule 1, it is known that 1-hop neighbors will retransmit the 2-hop neighbors MAD messages (2-hop MAD messages for the receiver) that all have different addresses: otherwise there would be a 2-hop conflict, necessarily resolved. Thus the mapping deduced from the *2-Hop Identifier Table*, "(neighbor main address, 2-hop neighbor main address) $\rightarrow$ 2-hop neighbor identifier" is unique (and corresponds to reality).

Note, that the information contained in the 2-Hop Identifier Table, should also be used for *Hello* messages processing (converting interface addresses into main addresses) so that the 2-hop neighbor main address in the 2-hop tuple is the actual main address.

(c) **Example**

An example of such a conversion method is presented here. In the topology of Figure 4.22, 5 nodes, A, B, C, D, and E are present. Node A is considered. It receives:

- *MAD* messages from neighbor `@1` with identifier ID_B
- *MAD* messages from neighbor `@2` with identifier ID_C
- *MAD* messages through neighbor `@1` from originator `@3` with identifier ID_D (2-hop *MAD* message).
- *MAD* messages through neighbor `@2` from originator `@3` with identifier ID_E (2-hop *MAD* message).

Now *Hello* messages from node B include the information: originator address `@1` ; link with `@3` symmetric (and also interface address of A symmetric).

When A receives such *Hello* messages, it understands that it has a symmetric neighbor, with address `@1`, and also that one of its 2-hop neighbors has the address `@3`. Because A received the *MAD* messages through `@1` with identifier

ID_D for `@3`, it will assume that the identifier corresponding to that `@3` is ID_D. It also knows from previous MAD messages from B, that the identifier for neighbor address `@1` is ID_B.

Hence it deduces that it has a 2-hop neighbor with identifier ID_D through the neighbor with identifier ID_B. Now even though E has the same address `@3` as D, the same method will make A realize that it has a 2-hop neighbor with identifier ID_E through the neighbor with identifier ID_C. This is the basis for a safe MPR calculation on identifiers.

4.5 Conclusion

The autoconfiguration procedure proposed in this chapter relies on an efficient and proven duplicate address detection algorithm. A special control message MAD (Multiple Address Declaration) conveys a random identifier with the address of the node. This control message uses the OLSR genuine MPR-flooding algorithm to reach all the nodes in the network, however special rules have been added to ensure that even with address duplications the MAD messages will be propagated throughout the entire network. A formal proof of correctness of our duplicate address detection scheme is given. Simple approaches to allocate an address to a newly arriving node or to solve conflicts are also provided. Finally, alternate relaying rules for MAD messages using node identifiers instead of addresses are presented.

Chapter 5

Implementation, simulation, and performance analyses

5.1 Implementation of the DAD-MPR flooding protocol

DAD-MPR flooding protocol is implemented as an extention to OLSR to support autoconfiguration. This implementation is based on the implementation of NOA-OLSR [59], a totally different autoconfiguration algorithm that was developed at Niigata University, itself based on OOLSR [1] the INRIA object oriented re-implementation of OLSR protocol in the C++ programming language. The generation and processing of MAD (Multiple Address Declaration) message is mainly based on the implementation of OLSR MID (Multiple Interface Declaration) message. Only the autoconfiguration rules related to single interfaces are implemented.

5.2 Control overhead of the DAD-MPR Flooding protocol

The overhead of the proposed autoconfiguration protocol can be rather easily evaluated. The main part of this overhead resides in the sending of MAD messages.

In this section, we show that the cost of MAD messages is bounded by the

[1] http://hipercom.inria.fr/OOLSR/.

cost of other existing OLSR messages (HELLO messages and TC messages) multiplied by a given factor. Also we show that the factor is proportional to the MAD message rate. We then argue that the MAD message rate (fully adjustable) is expected to be one order of magnitude lower than the rate of other OLSR messages: this gives a hint about the reason why MAD message overhead is, at worst comparable, and in general much lower, than the cost of OLSR messages. In the last part, simulations are made.

5.2.1 Analytic bound of the cost of MAD messages

The overhead of the OLSR routing protocol without MAD messages is well known, and using results of [67] (with a slightly different notation), the OLSR overhead as a number of bytes is given by:

$$overhead = T_h + T_t = \tau_h L_h N + \tau_t L_t o \rho N^2 \tag{5.1}$$

with the different parameters described in table 5.1.

The cost of MAD messages can be evaluated and bounded in a similar way: it is, at most, the cost of the retransmission of a MAD message by all the neighbors of one node due to Rule 3, plus the cost of retransmission of a MAD message by MPR-flooding to the entire network. More precisely we have:

$$T_m \leq \text{overhead of neighbors} + \text{overhead of MPR-flooding}$$
$$T_m \leq \tau_m L_m N N_n + \tau_m L_m N(oN)$$

It is possible to bound each term of this sum using the overhead of standard OLSR messages. Indeed, for the first term:

$$\tau_m L_m N N_n \leq \tau_m L_m N(\frac{L_h}{L_a}) = \frac{\tau_m}{\tau_h}\frac{L_m}{L_h}(\frac{L_h}{L_a})(\tau_h L_h N)$$

$$\tau_m L_m N N_n \leq T_h \frac{\tau_m}{\tau_h}\frac{L_m}{L_a}$$

and for the second term:

$$\tau_m L_m N(oN) = \frac{\tau_m}{\tau_t}\frac{L_m}{L_t}\frac{1}{\rho}(\tau_t L_t o \rho N^2)$$

$$\tau_m L_m N(oN) = T_t \frac{\tau_m}{\tau_t}\frac{L_m}{L_t}\frac{1}{\rho}$$

Varia- ble	Meaning
δ	average degree of a node
N	number of nodes in the network
τ_h	Hello message rate
L_h	size of Hello messages
τ_t	TC message rate
L_t	size of TC messages
o	broadcast optimization factor, $\frac{1}{\delta} \leq o \leq 1$
ρ	proportion of nodes which are MPR of at least one node
τ_m	MAD message rate
L_m	size of MAD messages
T_h	total overhead of Hello messages (in bytes)
T_t	total overhead of TC messages (in bytes)
T_m	total overhead of MAD messages (in bytes)
L_a	size of one address
N_n	avg. number of neighbors of one node

Table 5.1: Notation used for control overhead

Hence the overhead of MAD messages is bounded by

$$T_m \leq \alpha T_h + \beta T_t \tag{5.2}$$

where

$$\alpha = \frac{\tau_m}{\tau_h}\frac{L_m}{L_a} \tag{5.3}$$

$$\beta = \frac{\tau_m}{\tau_t}\frac{L_m}{L_t}\frac{1}{\rho} \tag{5.4}$$

Or also, compared to the total standard OLSR overhead:

$$T_m \leq max(\alpha, \beta)(T_h + T_t) \tag{5.5}$$

For networks of similar density the optimization factor ρ is expected to be similar, hence α and β stay constant even when the area of the network grows.

Additionally, the rate of MAD messages τ_m can be adjusted: the overhead due to MAD messages is proportional to the MAD message rate. Hence there is a direct tradeoff between MAD message overhead and the promptness of duplicate address detection.

5.2.2 Discussion of the cost of MAD messages

In the following, the properties of α and β are discussed using intuition, but no proof: the Section 5.2.3 will later justify and quantify the intuitions highlighted in this section.

The first observation is based on the fact that the DAD procedure may not need to be as fast to resolve conflicts as OLSR is fast to adjust to topology changes. Indeed, the standard OLSR messages should be sent with a rate at least of the order of magnitude of the rate of topology changes. The MAD messages should be sent with a rate at least of the order of magnitude of the rate of address changes or the rate of newcomers arrival in the network (or lower). Intuitively, in a MANET, there should be markedly more changes of topology per second than changes of addresses per second - or newcomers entry in the network per second. This translates into the fact that, arguably, the ratio of the rates $\frac{\tau_m}{\tau_h}$ and $\frac{\tau_m}{\tau_t}$ (appearing in the expressions of α and β) are both expected to be much lower than 1.

The second observation is, considering again the expression of α (equation 5.3) and β (equation 5.4), one can see that, for instance, α is the product of the previous

(small) ratio of rates by the product of the ratio of sizes $\frac{L_m}{L_a}$. Now, considering that a MAD message includes an identifier which is a few times larger than an address, we estimate that the ratio would be larger than 1, typically 4 to 10. Arguably, the product of this second ratio (large but reasonnable) by the first one (quite small), could typically yield an α value still lower than 1 or close to 1.

Similarily for β, typical values of ρ are on the order 0.5 to 0.9, and hence, $\frac{1}{\rho}$ should be less than 2. Furthermore, the ratio of the size of the messages $\frac{L_m}{L_t}$ is even smaller that the previous ratio $\frac{L_m}{L_a}$, since a TC message includes several addresses. Hence this ratio, is typically of the order of magnitude of 1, and once again, the product of the ratio would yield a β which is much lower than 1.

Our conclusion is for a sensible choice of parameters, a rapid analysis of the structure of the equations 5.3 and 5.4 shows that one can expect both of them to be lower than 1 and hence have an MAD overhead lower than existing OLSR protocol overhead.

5.2.3 Simulation results

The overhead of the proposed autoconfiguration mechanism for *OLSR*, the overhead of MAD messages, has been simulated (using a simulator written in the Ocaml programming language) in order to evaluate its performance.

An idealized simulation model was choosen in order to evaluate precisely its impact: a given number of nodes are placed in a square area ; there is no mobility, and a unit disk graph[2] is used. There is no MAC, hence no contention or collision, and the transmission delay is uniform. The radio range is a parameter of the simulation.

5.2.3.1 Cost of MAD flooding

In our simulations, the first interesting parameter is the average cost of one diffusion of *MAD* message. Indeed, in MAD diffusion, not all, but only a fraction of the nodes will retransmit the MAD message, hence an interesting parameter is the fraction of the nodes of the network which retransmit on average a MAD message (in our graph, the fraction is expressed in percentage of the nodes of the network). Note that the size of one MAD message is fixed and that all nodes are diffusing MAD messages periodically, hence one can easily derive the absolute

[2]In a unit disk graph, each node is identified with a disk of unit radius $r = 1$ in the plane, and is connected to all nodes within (or on the edge of) its corresponding disk.

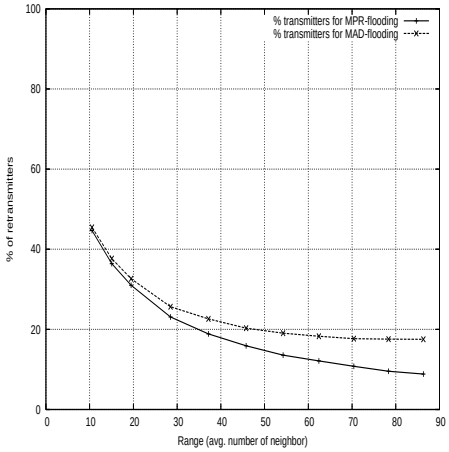


Figure 5.1: Fraction of the nodes in the network which retransmit the MAD message

overhead counted in messages (or bytes) per second.

Figure 5.1 depicts **the proportion of the nodes which retransmit one MAD message**, for increasing density: the simulated network comprises 1000 nodes and for each simulation, a different radio range is set in order to modify the density of the neighboring nodes. The result of MPR-flooding is given on the same figure for reference and is actually the quantity oN of the Section 5.2.1.

As shown in that section, the evaluation of the MAD-flooding is a little more complex than just evaluating oN: indeed it is the overhead of the MPR-flooding, plus the extra transmission of the neighbors of the initial originator of the MAD message - which are not MPR. The average number of the neighbors which are not MPRs of a given node is thus expected to be: avg number of neighbors per node - avg number of MPRs per node. Hence an estimate would be:

$$\begin{aligned} &\text{MAD-flooding overhead} = \text{MPR-flooding overhead} \\ &+ (\text{avg number of neighbors per node - avg number of MPRs per node}) \end{aligned}$$

In order to verify the estimate, using the previous MPR-flooding simulation results, we have calculated the corresponding MAD-flooding overhead. Precisely, for each simulation, we have two results: the overhead when actually simulating the MAD-flooding, and the overhead estimated from the MPR-flooding. Figure 5.2 depicts both the simulation results and the estimate. As they are close, the Figure 5.3 is more precise by depicting the ratio of the two quantities: we see that in our simulation set, the estimate is within 5 % of the simulation results,

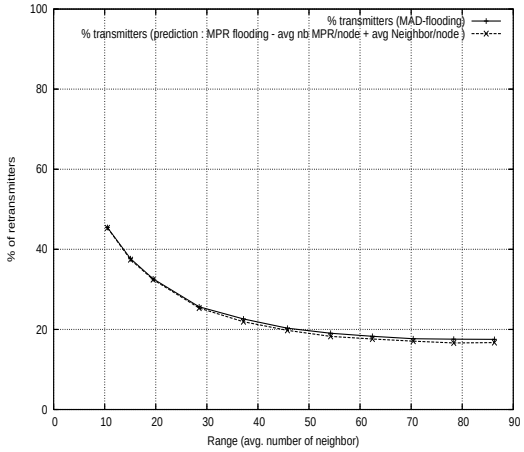


Figure 5.2: Comparison between actual results and prediction

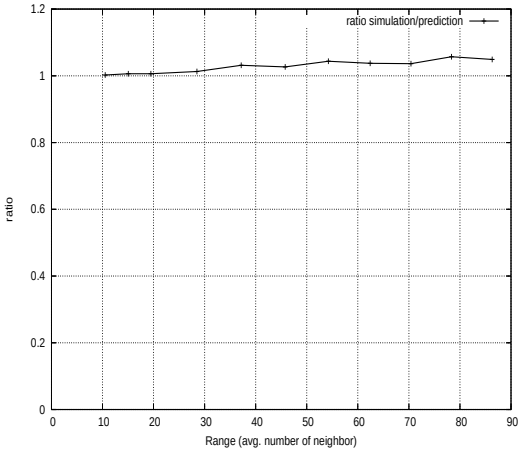


Figure 5.3: Ratio of overhead in simulation vs estimated overhead

for the overhead of MAD message.

The general conclusion is that the additional overhead generated by MAD messages remains limited compared to one classical MPR-flooding: the cost of the MAD flooding is similar to the cost of MPR-flooding, with the exception that all the direct neighbors of the originator will retransmit (instead of only the MPR) in the first step.

It also appears that the estimate of the cost of the MAD-flooding computed from the measured cost of MPR-flooding is close to the actual measured MAD-flooding, showing that the estimate is quite precise.

5.2.3.2 Comparison of the cost of MAD overhead with OLSR overhead

In the previous section, the cost of MAD-flooding was given, by simulation, as the percentage of nodes in the network that retransmit one MAD message.

This allows to compute the cost of MAD messages with an absolute value in bytes per second. But another interesting result is the relative cost of MAD messages compared to standard OLSR messages, in the spirit of the analytical results of Section 5.2.1.

In order to do so, more parameters have to be defined. The following are used:

- Hello message rate, $\tau_h = 0.5\ s^{-1}$ (as specified in RFC 3626)
- TC message rate, $\tau_t = 0.2\ s^{-1}$ (as specified in RFC 3626)

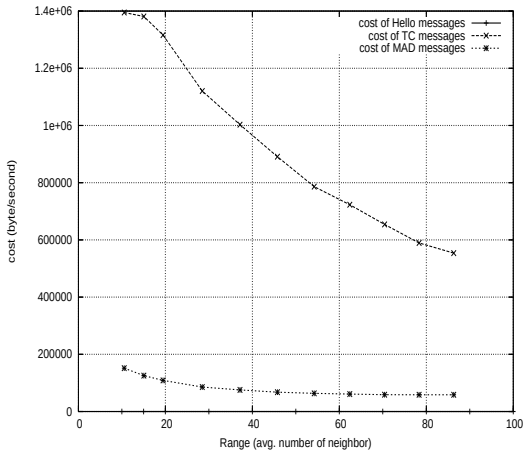


Figure 5.4: Cost of control messages in bytes

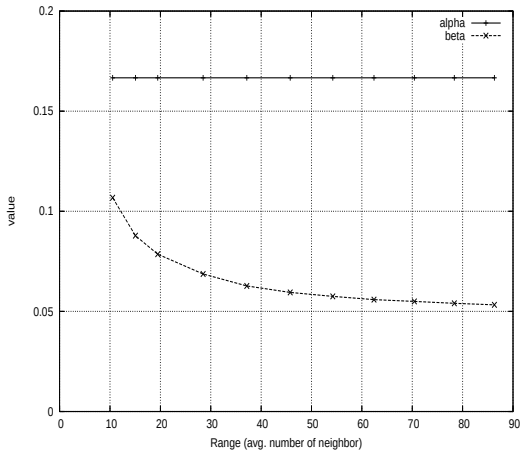


Figure 5.5: Evaluation of parameters α and β

- MAD message rate, $\tau_t = 1/60\ s^{-1}$ (one per minute)
- Size of address, $L_a = 4$ bytes
- Size of identifier, 16 bytes (hence $L_m = L_a + id_{size} = 20$)

Furthermore, the overhead of MAC, IP, UDP or OLSR message headers is also not taken into account.

The results, using the same simulations as in previous Section 5.2.3.1, are represented on Figure 5.4 [3].

An observation is that in this network the TC message overhead dominates and Hello message overhead is insignificant. More importantly, that in comparison, the MAD message overhead is limited.

Furthermore, the bound in equation 5.2, is dominated by the part due to TC messages (with factor β). As a consequence, β is almost exactly the ratio between MAD message cost and TC message cost ; this is confirmed by calculation performed with simulation results. The Figure 5.5, shows the values of α and β for the corresponding simulations. The value of β appears to stay between 0.05 and 0.12, which indicates that, for these simulations, MAD message overhead is at most 5 % to 12 % of the TC message overhead (which is the main part of the OLSR protocol overhead), computed in bytes, something which is quite reasonnable and show the efficiency of the proposed DAD algorithm.

[3] The curve for the cost of HELLO messages is on the X axis.

5.3 Convergence of the DAD-MPR flooding protocol

To evaluate the latency of address duplication detection using our DAD-MPR flooding protocol, we have simulated the merger of more than two networks generating massive address duplications. The nodes in each network are randomly placed in a square area of 1.0 x 1.0. The simulations were performed on MANETs with nodes moving from their starting points to a chosen destination in such a way that the length of the intersection area after merge is equal to a given value l (see Figure 5.6).

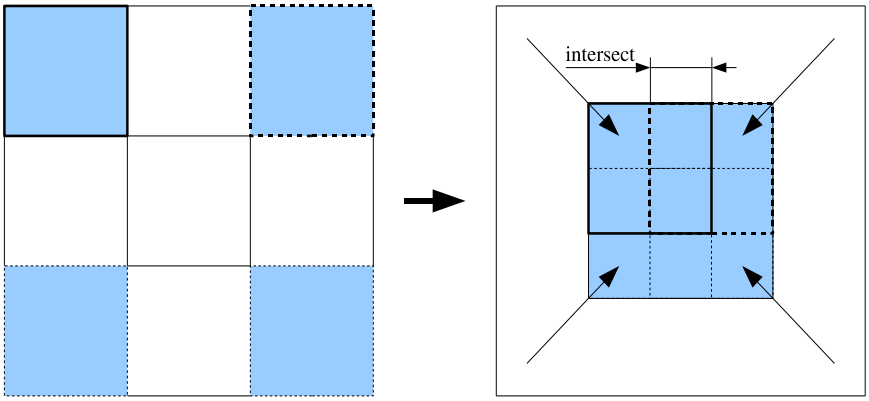


Figure 5.6: Length of the intersecting area after merge

As an illustration, Figure 5.7 and Figure 5.8 show the positions of the nodes of 3 networks of 40 nodes each before and after the merger. The length of the intersection square area l is equal to 0.70.

The code used for simulations is compiled in a specific library (*libolsr_static_plugee.a*). It contains the core files for the OLSR protocol implementation with autoconfiguration and a simple simulator support. The code used for running simulations is composed of three modules, *oolsrsimple.cc*, *liboolsr.py*, and *madAutoConfSimul.py* (see Figure 5.9). The module *oolsrsimple.cc* is a wrapper used to use *libolsr_static_plugee.a* library and the *Python* functions there. The library *liboolsr.py* is a small *Python* library to create basic simulations, and *madAutoConfSimul.py* is the *Python* program used to make simulations.

The simulations were run for a period of 100 s. No mergers were simulated in the first 30 seconds to allow the pre-configured nodes in each network to calculate

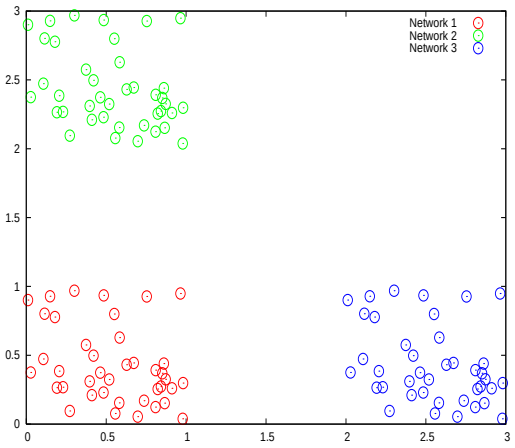


Figure 5.7: Position of the nodes before the merger

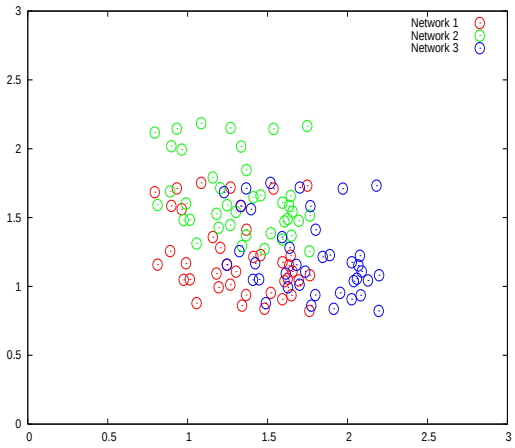


Figure 5.8: Position of the nodes after the merger

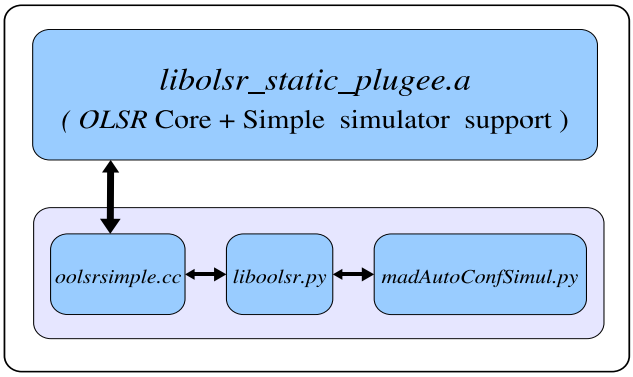


Figure 5.9: Architecture of the simulator for autoconfiguration

their MPR sets and to set up their routing tables. There is no MAC, hence no contention or collision, the transmission delay is uniform, and there is no mobility after merge. The parameters of the simulation are: the radio range (R: 0 to 1) of each node, the number of nodes (N) in each network, the number of networks merging ($nb\text{-}part$: 1 to 4), and the length of the intersecting square (l: 0 to 1) after merge. In the following, we discuss the challenge of detecting all duplicated addresses in a reasonable time when a merger of several networks containing duplicates occurs. The main parameter to compute is the duration our DAD-MPR flooding algorithm takes to detect all address conflicts after the merger. We dis-

cuss several merger scenarios by varying the values of the simulation parameters cited above.

At the begining each network is a copy of each other and every node x of each network m, m$\geq$2, has the same address as the node x of the network 1 and has the same position but translated.

Figure 5.10 shows the duration of address duplications detection, when 2, 3, and 4 networks merge by varying the number of nodes in each network. The radio range R of each node is set to 0.40 and the length of the intersecting area l is set to 0.70. We notice here that all the durations of address conflicts detection are $\leq$ 5s (one period of MAD message). In this case, either the networks form a 1-hop network after merge (all nodes are 1-hop neighbors of each other) and one period of MAD message suffices to the MAD messages to reach all the nodes, or the MPR sets in the networks computed before the merger continue to cover all the nodes after the merger. These situations occur when the radio range of the nodes and the length of the intersecting area are relatively high as it is the case here.

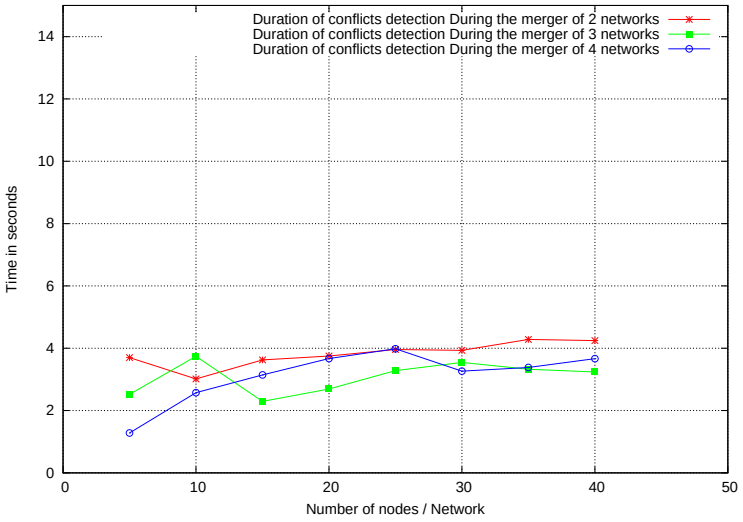


Figure 5.10: Duration of conflicts detection / Number of nodes N

Figure 5.11 shows the duration of address duplications detection during the merger of two networks of 50 nodes each, by varying the radio range of the nodes. The length of the intersecting area l is set to 0.70. As we can see on this figure, there exist address conflicts detection durations $>$ 5s. This happens when the resulting network after the merger is a multi-hop network and the MPR sets in the original networks do not cover all the nodes after the merger. In fact, some of

the MAD messages need to wait for the recalculation of the MPRs in the resulting network and hence to wait, at least, for the second period of MAD message after the merger to be propagated to the whole network. In addition, when a node detects an address conflict and changes its address, it must declare itself with the new address to the other nodes. Consequently, the neighbor tables, the MPR sets, and the topology tables are updated by at least its neighbors taking into account this new address, and therefore generating more latency in propagating the MAD messages and detecting the remaining address conflicts. That is why the duration of duplicate addresses detection can be more than 10s (two periods of MAD message) in some conflictual cases. Finally, when the radio range of the nodes is high enough, the network tends to become a 1-hop network and the address duplications are detected during the first MAD message period.

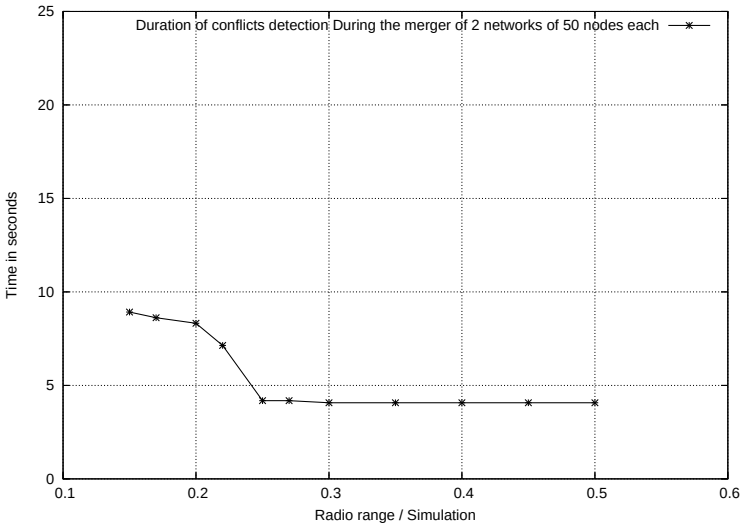


Figure 5.11: Duration of conflicts detection / Radio range R

To get more accurate results by simulations and confirm the robustess of our autoconfiguration protocol, it is better that the resulting network after the merger contains at least 5 hops. That way, our DAD-MPR flooding protocol can be faced to more complicated scenarios. For this purpose we set the length of the intersecting area l to 0.0, in such a way that there exists at least a link between the original networks after the merger (see Figure 5.12 and Figure 5.13).

Also, it was not easy in our simulations to generate connected network topologies because of the random character of the positions of the nodes in the square area of 1.0 x 1.0. Therefore, to increase the chances to get a connected network topology, we vary the number of nodes N in each network and the radio range

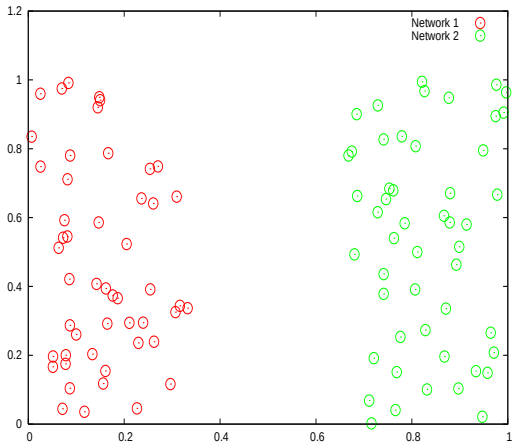


Figure 5.12: Position of the nodes before the merger ($l = 0.0$)

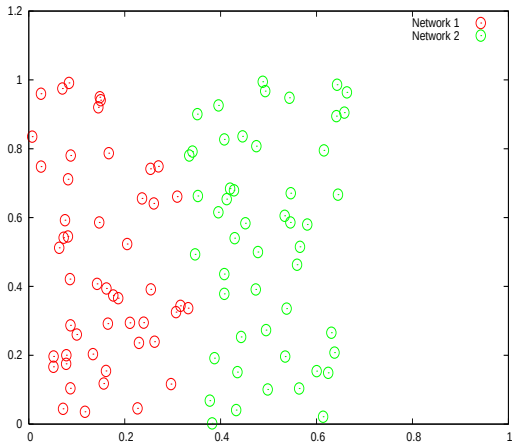


Figure 5.13: Position of the nodes after the merger ($l = 0.0$)

of the nodes R in such a way that the density of neighbors D in the network is kept equal to 10 [4]. Let us denote by S the surface of the square area where the nodes are randomly placed, and by S_x the surface of the area covered by the radio range R of node x (see Figure 5.14). Hence, we have $S_x{=}\pi R^2$ and $\frac{S}{\pi R^2} D{=}N$. Therefore, since the surface of the square area where the nodes are placed is of size 1.0x1.0, the interaction between the radio range R and the number of nodes N in the network to keep the value of D equal to 10 is expressed as $R{=}\sqrt{\frac{D}{\pi N}}$. This equation is illustrated in Figure 5.15.

The numbers of nodes used in the following simulations and their corresponding values of the radio range parameter R for $D{=}10$ are presented in Table 5.2.

Figure 5.16 shows the durations of address duplications detection during the merger of two networks with a density of neighbors $D{=}10$ and by varying the radio range of the nodes. The length of the intersecting area l is set to 0.0. Here the durations of conflicts detection vary from 10.59s for $R{=}0.15$ to 3.59s for $R{=}0.50$ which are reasonable values. One can notice that the only difference between this figure and Figure 5.11 is that in this figure the durations of address conlicts detection for the values of $R{\leq}0.35$ are a little bit higher than those reported in Figure 5.11. This is due to the fact that the numbers of nodes after the merger in

[4] An approach to topology control based on the principle of maintaining the number of neighbors of every node equal or slightly below a specific value k, was proposed in [68]. The value of k that guarantees connectivity of the network with high probability was estimated. Setting $k = 9$ produces a network which is connected with probability at least 0.95 for numbers of nodes in the range 50-500. The sizes of the networks in our simulations belong to this range.

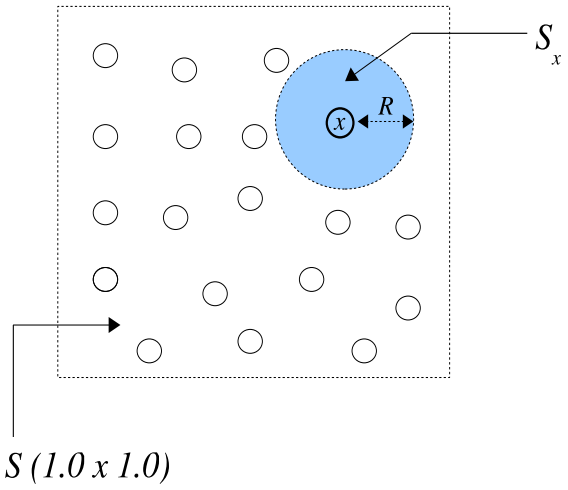


Figure 5.14: The area covered by the radio range R

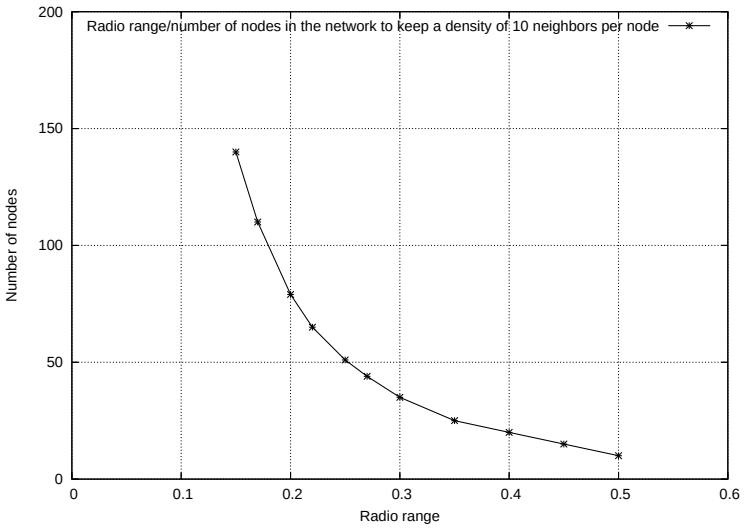


Figure 5.15: Number of nodes N / Radio range R

this figure vary from 280 nodes to 100 nodes for $R \leq 0.25$ and are higher than the one in Figure 5.11 which is fixed and equal to 100 nodes. However, the durations of conflicts detection for $R > 0.35$ are almost the same in both figures because

Radio range R	*Number of nodes N*
0.15	140
0.17	110
0.20	79
0.22	65
0.25	51
0.27	44
0.30	35
0.35	25
0.40	20
0.45	15
0.50	10

Table 5.2: Correspondance between R and N

the networks tend to become a 1-hop networks and the address duplications are detected during the first MAD message period.

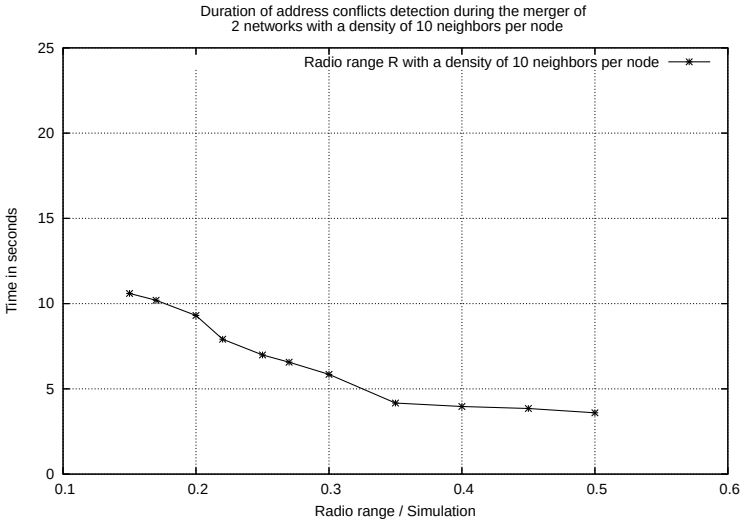


Figure 5.16: Radio range R with *D=10*

Now rather than considering fully duplicated networks, we compute the durations of address duplications detection by varying the number of duplicated addresses after the merger.

Figure 5.17 shows the durations of address duplications detection during the

merger of 2 networks of 50 nodes each. The radio range R is set to 0.25 and l is set to 0.0. The number of address conflicts vary from 5 to 50 addresses. In this figure the durations of conflicts detection vary from 3.14s for 5 address conflicts to 5.80s for a network containing 50 address conflicts.

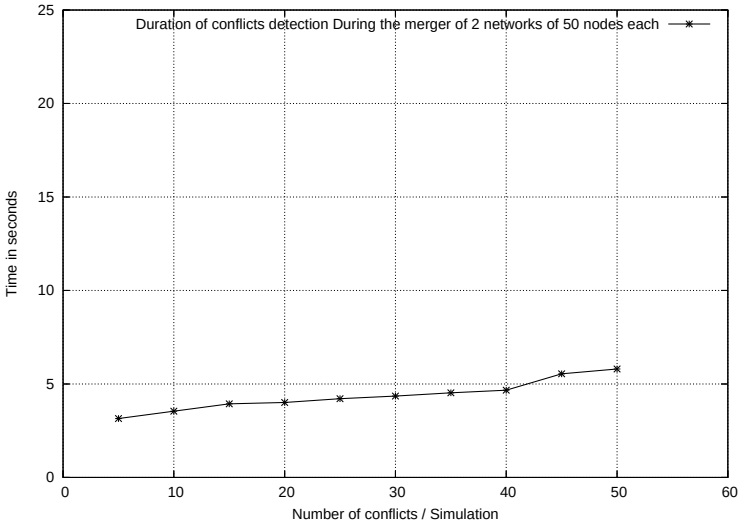


Figure 5.17: Duration of conflicts detection / Number of conflicts

The same simulation parameters as in Figure 5.16 are used in Figure 5.18; however, the durations of address conflicts detection reported in Figure 5.18 are function of the approximative number of hops in the network after the merger. This number of hops is calculated by dividing the length of the square area where the nodes are placed by the radio range R. We see here that even for networks containing more than 6 hops, the duration of address conflicts detection remains limited ($\approx$ 10.5s).

5.4 Conclusion

In the first section of this chapter we have presented how our autoconfiguration protocol is implemented and integrated with OLSR.
In the sections after, we have presented a detailed analysis of the overhead induced by our autoconfiguration algorithm (MAD message overhead). This overhead has also been simulated and the results show that it remains limited compared to the overhead of classical OLSR control messages. In order to test the convergence of our protocol, we have simulated the merger of more than two networks generating

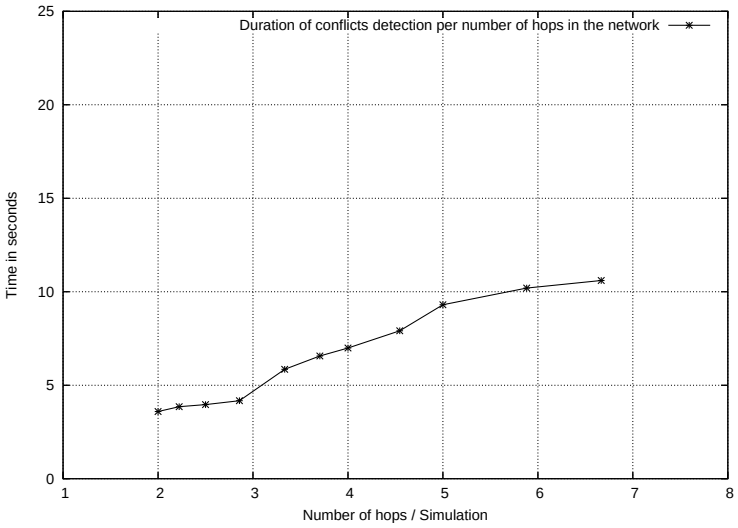


Figure 5.18: Duration of conflicts detection / Number of hops

massive address duplications. The main parameter of interest is the duration our DAD-MPR flooding algorithm takes to detect and resolve all address conflicts after the merger. The results of the simulation experiments show that DAD-MPR flooding protocol can efficiently detect and resolve address duplications within seconds, even if the addresses in the networks are fully duplicated.

Part III

Securing ad hoc networks

Confidentiality, integrity and authentication are more relevant issues in ad hoc networks than in wired fixed networks. One way to address these issues is the use of symmetric key cryptography, relying on a secret key shared by all the members of the network. But establishing and maintaining such a key (also called the session key) is a non-trivial problem. We show that Group Key Agreement (GKA) protocols are suitable for establishing and maintaining such a session key in these dynamic networks. We take an existing GKA protocol, which is robust to connectivity losses and discuss all the issues for the smooth functioning of this protocol in ad hoc networks. We give implementation details and network parameters, which significantly reduce the computational burden of using public key cryptography in such networks.

Chapter 6

Dynamic Group Key Agreement in Ad hoc Networks

6.1 Introduction

A Mobile Ad hoc NETwork (MANET) is a collection of mobile nodes connected via a wireless medium forming an arbitrary topology. Implicit herein is the ability for the network topology to change over time as links in the network appear and disappear. To maintain the network connectivity, a routing protocol must be used. An important security issue is that of the integrity of the network itself. Quite a lot of studies have been already done to resolve security issues in existing routing protocols (see [99],[107],[88],[87]).

An orthogonal security issue is that of maintaining confidentiality and integrity of data exchanged between nodes in the network. The task of ensuring end-to-end security of data communications in MANETs is equivalent to that of securing end-to-end security in traditional wired networks. Many studies have been carried out to solve this problem. One widespread solution is to create a virtual private network (VPN) in a tunnel between the two communicating nodes. IPSec is a well known security architecture which allows such VPNs to be built between two communicating nodes. However this solution requires a different secret key for each end-to-end connection. Moreover the VPN solution can only handle unicast traffic. An alternative solution is the use of a shared secret key. There are many issues with such an approach. First this key must be distributed among the network nodes. Second this key must be renewed often to avoid it being compromised. A solution to these two issues is the use a Group Key Agreement protocol, which relies on the principles of the public key cryptography.

A Group Key Agreement protocol (GKA) is a key establishment technique in which a shared secret is derived by more than two participants as a function of information publicly contributed by each of them. It is especially well suited to moderate sized groups with no central authority to distribute keys. An authenticated group key agreement protocol provides the property of key authentication (also called implicit key authentication), whereby each participant is assured that no other party besides the participants can gain access to the computed key. GKA protocols are different from group key distribution (or key transport) protocols wherein one participant chooses the group key and communicates it to all the others. GKA protocols help in deriving keys which are composed of each one's contribution. This ensures that the resulting key is fresh (for a given session) and is not favorable to one participant in any way. The following security goals can be identified for any GKA protocol.

1) **Key Secrecy**: The key can be computed only by the participants.
2) **Key Independence**: Knowledge of any set of group keys does not lead to the knowledge of any other group key not in this set (see [92]).
3) **Forward Secrecy**: Knowledge of some long term secret does not lead to the knowledge of past group keys.

An important advantage of a group key agreement protocol over a simple group key distribution scheme is forward secrecy. This property can be particularly interesting in situations where some nodes are likely to be compromised (e.g. in military scenarios). In such scenarios, using a GKA, the knowledge of the long term secret of this node does not compromise all past session keys. From a functional point of view, it is desirable to have procedures to handle the dynamism in the network. These procedures enable efficient merging or partitioning of two groups in the network.

6.2 Related Work

Key establishment protocols for networks can be broadly classified into three classes: *Key transport using symmetric cryptography*, *Key transport using asymmetric cryptography* and *Key agreement using asymmetric cryptography*. In key transport protocols, one participant chooses the group key and securely transfers it to other participants using a priori shared secrets (symmetric or asymmetric). These protocols are not suitable for ad hoc networks for two reasons; firstly, they require a single trusted authority to distribute keys and secondly, compromise of

the a priori secret of any participant breaches the security of all past group keys, thus failing to provide forward secrecy. Thus GKA protocols are more relevant since they provide this forward secrecy property.

Most group key agreement protocols are derived from the two-party Diffie-Hellman key exchange protocol. GKA protocols, not based on Diffie-Hellman, are few and include the protocols of Pieprzyk and Li [106], Tzeng and Tzeng [111] and Boyd and Nieto [93]. Both protocols of Pieprzyk and Li [106] and Boyd and Nieto [93] fail to provide *forward secrecy* while the protocol of Tzeng and Tzeng [111] is quite resource-intensive and prone to certain attacks [93]. Forward Secrecy is a very desirable property for key establishment protocols in ad hoc networks, as some nodes can be easily compromised due to low physical security of nodes. Thus it is essential that the compromise of one single node does not compromise all past session keys. We summarize and compare in Table 6.1 existing GKA protocols based on the Diffie-Hellman protocol. We compare the unauthenticated versions of the protocols, as most achieve authentication by using digital signatures in a very similar manner and thus have similar added costs for achieving authentication. We compare the efficiency of these protocols based on the following parameters:

- **Number of synchronous rounds**: In a single synchronous round, multiple independent messages can be sent in the network. The total time required to run a round-efficient GKA protocol can be much less than other GKA protocols that have the same number of total messages but more rounds. This is because the nodes spend less time waiting for other messages before sending their own.
- **Number of messages:** This is the total number of messages (unicast or broadcast) exchanged in the network to derive the group key. For multiple-hop ad hoc networks, the distinction between unicast and broadcast messages is important as the latter can be much more energy consuming (for the whole network) than the former.
- **Number of exponentiations**: All Diffie-Hellman based GKA protocols require a number of modular exponentiations to be performed by each participant. Relative to all cryptographic operations, a modular operation is the most computationally intensive operation and thus gives a good indication of the computational cost for each node.

Communication costs still remain the critical factor for choosing energy-efficient protocols for most ad hoc networks. A modular exponentiation (which is most efficiently done using elliptic curve cryptography) can be performed in a few tens

	Expo per U_i	Messages	Broadcasts	Rounds
ITW [100]	m	$m(m-1)$	0	$m-1$
GDH.1 [110]	$i+1$	$2(m-1)$	0	$2(m-1)$
GDH.2 [110, 96]	$i+1$	$m-1$	1	m
GDH.3 [110]	3	$2m-3$	2	$m+1$
Perrig [105]	$\log_2 m+1$	m	$m-2$	$\log_2 m$
Dutta [98]	$\log_3 m$	m	m	$\log_3 m$

Table 6.1: Comparison of non constant rounds GKA protocols

	Expo per U_i	Messages	Broadcasts	Rounds	Structure	FS
Octopus [90]	4	$3m-4$	0	4	Hypercube	Yes
BDB [97, 101]	3	$2m$	m	2	Ring	Yes
BCEP [95]	$2^{\dagger}$	$2m$	0	2	None	No
Catalano [94]	$m+1$	$2m$	0	2	None	Yes
KLL [102]	3	$2m$	$2m$	2	Ring	Yes
NKYW [104]	$2^{\ddagger}$	m	1	2	None	Yes
STR [109, 103]	$(m-i)^{*}$	m	1	2	Skewed tree	Yes
Ours (AGDH)	2^{**}	m	1	2	None	Yes

$\dagger$: m exponentiations for the base station.
$\ddagger$: $m+1$ exponentiations and m-1 inverse calculations for the parent node.
$*$: Up to $2m$ exponentiations for the sponsor node.
$**$: m exponentiations for the leader.

Table 6.2: Comparison of constant round GKA protocols

of milliseconds on most palmtops, whereas message propagation in multi-hop ad hoc networks can be easily of the order of few seconds and has energy implications for multiple nodes in the network. As can be seen in Table 6.1, most existing GKA protocols require $O(m)$ rounds of communication for m participants in the protocol. Such protocols do not scale well in ad hoc networks. Even tree-based GKA protocols with $O(\log m)$ rounds can be quite demanding for medium to large sized ad hoc networks. Therefore constant-round protocols are better suited for ad hoc networks.

Among the constant round protocols (see Table 6.2), Octopus [90], BDB [101] and KLL [102] require special ordering of the participants. This results in messages sent by some participants being dependent on that of others. In such a case, failure of a single node can often halt the protocol. Thus such protocols are not robust enough to adapt well to the dynamism of ad hoc networks. The BCEP protocol

[95] involves a base station, and fails to provide forward secrecy if the long-term secret of the base station is revealed. The Bresson and Catalano protocol [94] is computationally demanding with $O(m)$ exponentiations for each participant. Another drawback is that if any participant's message is lost in the first round, the whole protocol is brought to a halt, as the secret sharing scheme implies all m contributions are required to compute the key. Thus only the protocols NKYW and STR (described below in detail) seem to be usable in MANETs.

NKYW[104]: The original paper proposes this protocol for ad hoc networks composed of devices with unequal computational powers. In the first round, each participant M_i unicasts its contribution $g^{r_i}, i \in [1, n-1]$ to a fixed node M_n, called the parent node. The parent node chooses random r and r_n and computes $w = g^r$, $x_n = g^{rr_n}$ and $x_i = (g^{r_i})^r$ for each received g^{r_i}. It broadcasts w and $\{x_n * \Pi_{j \neq i} x_j\}_i$. The key is derived from $\Pi_i x_i$. The protocol remains a bit expensive computationally compared to the protocol that will be described in this chapter.

STR[109, 103]: This protocol was proposed by Steer *et al.* in [109] for static groups. Perrig *et al.* proposed procedures to handle group changes in [103]. Although this protocol has not been cited as a constant round protocol till now, we explain here in detail why this protocol is indeed a constant round protocol. In the first round, each participant M_i broadcasts its contribution g^{r_i} (also known as its blinded key). In the second round, a key-tree as shown in Figure 6.1 where each leaf node represents a participant is constructed using participant IDs or the value of the contributions. The node in the bottom-most, left-most position in the tree is called the sponsor. The sponsor node broadcasts the set of blinded keys for all the intermediate nodes upto the root node. For the case shown in Figure 6.1, the broadcast message is $\{g^{r_1}, g^{r_2}, g^{r_3}, g^{r_4}, g^{g^{r_1 r_2}}, g^{g^{r_3 \cdot g^{r_1 r_2}}}\}$. The group key is $K = g^{r_4 \cdot g^{r_3 \cdot g^{r_1 r_2}}}$. Participant M_i has to perform $m - i$ exponentiations except for the sponsor which has to compute $2m$ exponentiations. The protocol lacks a proof of security against active adversaries.

Both these protocols are computationally more expensive compared to the protocol that will be described in this chapter.

The contributions of this chapter are the following:

- an authenticated dynamic group key agreement protocol is recalled [89],
- the mechanisms that must be used in a MANET to implement this group key agreement protocol are described,
- a precise study of the cryptographic parameters that this group key agreement protocol must use in the context of an ad hoc network is carried out.

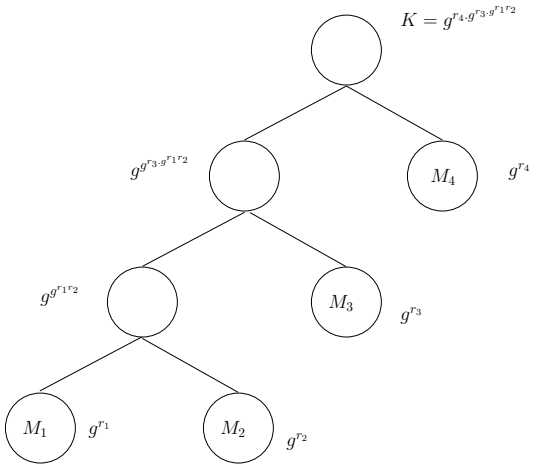


Figure 6.1: The STR Protocol

Finally our adaptation of the group key agreement protocol called AGDH for Asymmetric Group Diffie Hellman, is demonstrated to be one of the very few group key agreement protocols suitable for ad hoc networks.

The following sections are organized as follows:

- Section 6.3 recalls the group key agreement protocol used [89]. We describe the basic functioning of the protocol only,
- Section 6.4 explains how this group key agreement protocol can be implemented in an ad hoc network. The main issues discussed in this section include the election of a leader in the ad hoc network and the actions that must be undertaken to handle splits and mergers in the ad hoc network,
- Section 6.5 discusses the overhead of cryptographic operations.

6.3 Presentation of AGDH

We recall an existing group key agreement protocol in this section. We first illustrate the basic principle of key exchange, followed by a detailed explanation of how it is employed to derive Initial Key Agreement, Join/Merge and Delete/Partition procedures to handle dynamism in ad hoc groups.

6.3.1 Notation

G: A subgroup (of prime order q with generator g) of some group.
U_i: i^{th} participant amongst the n participants in the current session.
U_l: The current group leader ($l \in \{1, \ldots, n\}$).
r_i: A random number (from $[1, q-1]$) generated by participant U_i. Also called the *secret* for U_i.
g^{r_i}: The *blinded secret* for U_i.
$g^{r_i r_l}$: The *blinded response* for U_i from U_l.
$\mathcal{M}$: The set of indices of participants (from $\mathcal{P}$) in the current session.
$\mathcal{J}$: The set of indices of the joining participants.
$\mathcal{D}$: The set of indices of the leaving participants.
$x \leftarrow y$: x is assigned y.
$x \stackrel{r}{\leftarrow} \mathcal{S}$: x is randomly drawn from the uniform distribution S.
$U_i \longrightarrow U_j : \{M\}$: U_i sends message M to participant U_j.
$U_i \stackrel{B}{\longrightarrow} \mathcal{M} : \{M\}$: U_i broadcasts message M to all participants indexed by $\mathcal{M}$.
N_i: Random nonce generated by participant U_i.
$\mathcal{V}_{PK_i}\{msg_i, \sigma_i\}$: Signature verification algorithm which returns 1 if σ_i is a valid signature on message msg_i else 0.

6.3.2 A Three Round Protocol

6.3.2.1 The formal description

Please note that in the following rounds each message is digitally signed by the sender (σ_i^j is signature on message msg_i^j in Tables 6.3- 6.5) and is verified (along with the nonces) by the receiver before following the protocol. Thus we omit to describe these steps which are formally shown in Tables 6.3- 6.5.

Protocol Steps:

Round 1: The chosen group leader, M_l makes a initial request (**INIT**) with his identity, U_l and a random nonce N_l to the group $\mathcal{M}$.

Round 2: Each interested M_i responds to the **INIT** request, with a **IREPLY** message which contains his identity U_i, a nonce N_l and a blinded secret g^{r_i} to M_l (see Table 6.3 for exact message contents).

Round 3: M_l collects all the received blinded secrets, raises each of them to the power of its secret (r_l) and broadcasts them along with the original contributions to the group, i.e. it sends an **IGROUP** message which contains

$\{U_i, N_i, g^{r_i}, g^{r_i r_l}\}$ for all $i \in \mathcal{M} \setminus \{l\}$.

Key Calculation: Each M_i checks if its contribution is included correctly and obtains g^{r_l} by computing $(g^{r_i r_l})^{r_i^{-1}}$. The group key is

$$Key = g^{r_l} * \Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l} = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}.$$

Note:

1) The original contributions g^{r_i} are included in the last message as they are required for key calculation in case of group modifications (see below), and also, because it may be possible that a particular contribution has not been received by some members.

2) Even though $\Pi_{i \in \mathcal{M} \setminus \{l\}} g^{r_i r_l}$ is publicly known, it is included in key computation, to derive a key composed of everyone's contribution. This ensures that the key can not be pre-determined and is unique to this session.

3) Even though the current group leader chooses his contribution after others, he cannot pre-determine the group key.

The protocol is formally defined in Table 6.3. Table 6.4 shows how the protocol is run when a group wants to join an existing group, and Table 6.5 to leave.

6.3.2.2 Example runs of the protocol

We now see how this protocol can be used to derive Initial Key Agreement (IKA), Join/Merge and Delete/Partition procedures for ad hoc networks.

Initial Key Agreement: Secure ad hoc group formation procedures typically involve peer discovery and connectivity checks before a group key is derived. Thus, an $INIT$ request is issued by a participant and all interested peers respond. The responses are collected and connectivity checks are carried out to ensure that all participants can listen/broadcast to the group (see for instance [108]). After the group membership is defined, GKA procedures are implemented to derive a group key. Such an approach is quite a drain on the limited resources of ad hoc network devices. Thus an approach which integrates the two separate procedures of group formation and group key agreement is required. The above protocol fits well with this approach. Round 1 and Round 2 of the above protocol can be incorporated into the group formation procedures. In this way, blinded secrets, g^{r_i}'s, of all potential members, U_i's, are collected before the group composition is defined. When the fully connected ad hoc group is defined, a single broadcast message

Round 1
$l \xleftarrow{r} \mathcal{M}, N_l \xleftarrow{r} \{0,1\}^k$
$U_l \xrightarrow{B} \mathcal{M} : \{msg_l^1 = \{ \textbf{INIT} , U_l, N_l\}, \sigma_l^1\}$
Round 2
$\forall i \in \mathcal{M} \setminus \{l\}, if(\mathcal{V}_{PK_l}\{msg_l^1, \sigma_l\} == 1)$, $r_i \xleftarrow{r} [1, q-1], N_i \xleftarrow{r} \{0,1\}^k$,
$U_i \longrightarrow U_l : \{msg_i = \{ \textbf{IREPLY}, U_l, N_l, U_i, N_i, g^{r_i}\}, \sigma_i\}$
Round 3
$r_l \xleftarrow{r} [1, q-1]$,
$\forall i \in \mathcal{M} \setminus \{l\}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1)$ and N_l is as contributed
$U_l \xrightarrow{B} \mathcal{M} : \{msg_l^2 = \{\textbf{IGROUP}, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M}\setminus\{l\}}\}, \sigma_l^2\}$
Key Computation
$if(\mathcal{V}_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$ and g^{r_i} and N_i are as contri buted
$Key = g^{r_l(1+\sum_{i \in \mathcal{M}\setminus\{l\}} r_i)}$

Table 6.3: IKA

(Round 3 in Table 6.3) from the group leader, U_l, (using contributions of only the participants who have joined) helps every participant to compute the group key. An example is provided below.

Suppose U_1 initiates the group discovery and initially 5 participants express interest and send g^{r_2}, g^{r_3}, g^{r_4}, g^{r_5} and g^{r_6} respectively along with their identities and nonces. Finally only 3 join because of the full-connectivity constraint. Suppose the participants who finally join are U_2, U_4 and U_5. Then the group leader, U_1, broadcasts the following message: $\{g^{r_2}, g^{r_4}, g^{r_5}, (g^{r_2})^{r_1}, (g^{r_4})^{r_1}, (g^{r_5})^{r_1}\}$. On receiving this message, each participant can derive g^{r_1} using his respective secret. Thus the key $g^{r_1(1+r_2+r_4+r_5)}$ can be computed.

Join/Merge: Suppose new participants, U_9 and U_{10} join the group of U_1, U_2, U_4 and U_5 with their contributions g^{r_9} and $g^{r_{10}}$ respectively. Then the previous group leader (U_1) changes its secret to $r_1^{'}$ and sends $g^{r_1^{'}}$, g^{r_2}, g^{r_4}, g^{r_5}, g^{r_9}, $g^{r_{10}}$ to U_{10} (say the new group leader). U_{10} generates a new secret $r_{10}^{'}$ and broadcasts the following message to the group: $\{g^{r_1^{'}}, g^{r_2}, g^{r_4}, g^{r_5}, g^{r_9}, g^{r_{10}^{'}r_1^{'}}, g^{r_{10}^{'}r_2}, g^{r_{10}^{'}r_4}, g^{r_{10}^{'}r_5}, g^{r_{10}^{'}r_9}\}$. And the new key is $g^{r_{10}^{'}(1+r_1^{'}+r_2+r_4+r_5+r_9)}$.

Delete/Partition: When participants leave the group, they send a **DEL** message, the group leader changes its secret contribution and sends an **IKA** Round 3

Round 1
$\forall i \in \mathcal{J}, r_i \stackrel{r}{\leftarrow} [1, q-1], N_i \stackrel{r}{\leftarrow} \{0,1\}^k,$
$U_i \stackrel{B}{\longrightarrow} \mathcal{M} : \{msg_i = \{ \textbf{JOIN}, U_i, N_i, g^{r_i}\}, \sigma_i\}$
Round 2
$\forall i \in \mathcal{J}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1)\ r_l \stackrel{r}{\leftarrow} [1, q-1], l' \stackrel{r}{\leftarrow} \mathcal{M} \cup \mathcal{J}$
$U_l \longrightarrow U_{l'} : \{msg_l = \{ \textbf{JREPLY}, \{U_i, N_i, g^{r_i}\}_{\forall i \in \mathcal{M} \cup \mathcal{J}}\}, \sigma_l\}$
Round 3
$if(\mathcal{V}_{PK_i}\{msg_l, \sigma_l\} == 1), l \leftarrow l', r_l \stackrel{r}{\leftarrow} [1, q-1], \mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{J}$
$U_l \stackrel{B}{\longrightarrow} \mathcal{M} : \{msg_l^2 = \{ \textbf{JGROUP}, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}\}, \sigma_l^2\}$
Key Computation
$if(\mathcal{V}_{PK_l}\{msg_l^2, \sigma_l^2\} == 1)$ and g^{r_i} and N_i are as contributed
$Key = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 6.4: Join/Merge

like message to the group, omitting the leaving participants' contributions. Refer to Table 6.5 and below for an example.

Suppose a participant, U_2, leaves the group of U_1, U_2, U_4, U_5, U_9 and U_{10}. Then the leader, U_{10} changes its secret to $r_{10}^{''}$ and broadcasts $\{g^{r_1^{'}}, g^{r_4}, g^{r_5}, g^{r_9}, (g^{r_1^{'}})^{r_{10}^{''}}, (g^{r_4})^{r_{10}^{''}}, (g^{r_5})^{r_{10}^{''}}, (g^{r_9})^{r_{10}^{''}}\}$ to the group. And the new key is $g^{r_{10}^{''}(1+r_1^{'}+r_4+r_5+r_9)}$.

Round 1
$\forall i \in \mathcal{D}, U_i \longrightarrow U_l : \{msg_i = \{ \textbf{DEL}, U_i, N_i\}, \sigma_i\}$
Round 2
$\forall i \in \mathcal{D}, if(\mathcal{V}_{PK_i}\{msg_i, \sigma_i\} == 1), r_l \stackrel{r}{\leftarrow} [1, q-1], \mathcal{M} \leftarrow \mathcal{M} \setminus \mathcal{D}$
$U_l \stackrel{B}{\longrightarrow} \mathcal{M} : \{msg_l = \{ \textbf{DGROUP}, U_l, N_l, \{U_i, N_i, g^{r_i}, g^{r_i r_l}\}_{i \in \mathcal{M} \setminus \{l\}}\}, \sigma_l\}$
Key Computation
$if(\mathcal{V}_{PK_l}\{msg_l, \sigma_l\} == 1)$ and g^{r_i} and N_i are as contribute d
$Key = g^{r_l(1+\sum_{i \in \mathcal{M} \setminus \{l\}} r_i)}$

Table 6.5: Delete/Partition

6.4 Using this GKA protocol within an ad hoc network

In the following we are considering a multi-hop ad hoc network. We are not assuming any particular property of the routing protocol which ensures the connectivity of the network. We can use reactive protocols as AODV or DSR [69, 73] where the connectivity is created on demand when a route is needed. We can also use proactive protocols as OLSR or TBRPF [55, 70] where synchronous packets are used to maintain the knowledge of the topology. We will assume that we have a broadcast mechanism to flood messages within the ad hoc network. We are not assuming that this flooding mechanism is reliable, but we assume that the network is connected and that the flooding messages finally reach all the nodes in the network[1].

A key point in the GKA protocol described above is the existence of a group leader. Thus it is necessary to have a robust mechanism to elect such a leader in an ad hoc network. This is the first issue that we study.

6.4.1 Election of a group leader

A key requirement is that all members of a group agree on the same group leader. A simple solution is that the group leader periodically broadcasts messages. These messages then serve as a proof for nodes that are within reach of the group leader, that a group leader exists and operates properly. We can simply use the **INIT** message of the GKA protocol to demonstrate the existence and the correct functioning of the group leader. When the other nodes in the network receive this **INIT** message each replies with an **IREPLY** message including their contribution. Using these **IREPLY** messages, the group leader defines a group and sends to all members of the group an **IGROUP** message. The **INIT** message can be seen as an **IGROUP** message when the group is not yet defined. In the following we will only use the term **IGROUP** message.

These **IGROUP** messages are sent periodically; depending on the dynamics of the group, the group leader will send a new **IGROUP** message or exactly the same message as before. If the network only comprises of the group leader, the latter will send periodically empty **IGROUP** messages. It will stop sending this message when a node joins its network by replying to its **IGROUP** message with an **IREPLY** message. The mechanism to elect a group leader simply follows

[1]We mean that synchronous flooded messages will finally reach all the nodes in the network even if there are messages losses

from the property that, in a network with a group leader, periodic messages are broadcast by the group leader and are, in principle, received by the group members. If a node does not receive a message for a fixed period T, known a priori by the network nodes, this node sets a random timer. At the expiration of this timer and if no **IGROUP** message has been received meanwhile, the node becomes the group leader. It then sends an empty **IGROUP** message.

There may be a collision on **IGROUP** messages if two nodes or more have selected the same value for their random timer. In such a case, there may be **IGROUP** messages generated by two (or more) group leaders. To select a group leader, we can use additional rules. The first rule is that when a group leader A receives an **IGROUP** message from a group leader B which has a smaller ID than its own ID, the group leader A just stops sending its periodic messages. The group members that will receive periodic messages from more than two group leaders will only consider the message issued by the group leader with the smallest index. Thus if an **IGROUP** message showing a larger ID than a previously received **IGROUP** message is received, then this message is simply discarded and no **IREPLY** message is issued. On the contrary if an **IGROUP** message showing a smaller ID is received then the node issues a **IREPLY** message.

Another issue is how the GKA protocol takes into account the dynamism of an ad hoc network. For instance a node may leave the network without being able to send the group leader a message pointing out its departure from the network. This issue is handled in the next subsection

6.4.2 Handling join and withdrawal of a node

A node which joins the network will receive the periodic **IGROUP** message of the group leader. It will just have to send a **JREPLY** message, with its contribution, to join the group. The group leader will incorporate this new contribution in its next **IGROUP** message. Actually there is no need in the protocol to differentiate between **JREPLY** and **IREPLY**. Thus, for simplicity's sake, we will only keep the **IREPLY** message.

In an ad hoc network, the only conceivable way for the group leader to be sure that a node remains in a group is to receive a message from it. Thus to handle the dynamism of a group, the group leader will use the periodic reception of the **IREPLY** messages. The period with which an **IREPLY** message is sent by a member of the group should be the same for all the nodes of the group. If the group leader does not receive a **IREPLY** message for a given number of periods (greater than 1 to handle possible packet loss), the lack of reception of

these messages should be handled in the same way as the reception of a **DEL** message. In such a case the group leader will change its own contribution in the **IGROUP** message and will re-send the **IGROUP** message.

When a node deliberately wishes to withdraw from a group it can use the **DEL** message to announce this wish to the group leader. Upon the reception of such a message the group leader will change its own contribution in the **IGROUP** message and will re-send the **IGROUP** message. The use of the **DEL** message will speed up the taking into account of the node withdrawal.

6.4.3 Handling merge or split of groups

The merger of groups (two or more) leads group leaders to receive **IGROUP** messages from other group leaders. The scheme used in the group leader election can be used to resolve the conflict. When the conflict is resolved only one group leader remains in the group. If a group splits, a part of the group will remain without a group leader. The technique used in the group leader election can be used in the subgroups without a leader to elect a new leader.

6.4.4 Renewing its contribution

The group leader and group members will have to renew their contribution periodically. For the group leader, the change of its contribution or of some member of the group will lead to a change in the content of the **IGROUP** message. To simplify we can assume that the group leader and the group members change their contribution at the same rate.

We have given all the principles of the protocol. We specify the details of the whole protocol in the next section.

6.4.5 Implementation issues

We will consider a given period T. To simplify, this period will be used both by the group leader and by the members of the group as a period to send their GKA messages.

A node can be in one of the following two states : **member state** or **group leader state**. A node in a member state will start the process to become a group leader if it has not received an **IGROUP** message for a duration kT. A

node which has not received any message from a group leader for a duration kT with $k \geq 2$ will suppose that there is no group leader and starts the procedure to become a leader. Since a node may not have received a packet of the group leader because this packet has been lost, k must be selected so that the probability that $k-1$ successive transmissions of a GKA message are lost is small. Then, to become a group leader, the node selects a random integer i_r between 1 and a given number l (backoff window size) and initializes a timer at $i_r t_{rtd}$, where t_{rtd} is a predefined duration computed to be at least the round trip delay of a message throughout the ad hoc network. With such a figure for t_{rtd} we can be sure that if two nodes draw different integers i_r and $i_{r'}$, the node having selected the larger integer will receive the **IGROUP** message of the other node and then will stop its election process. The backoff window size l must be chosen with respect to the total number of nodes in the network so that the probability that two nodes choose the same integer is small. This back-off procedure is performed to avoid possible multiple group leader candidates, for instance, when a group is set up or split into two subgroups.

When a node in the member state sends its first **IGROUP** message, it enters the group leader state, see Figure 6.2. In the group leader state, a node

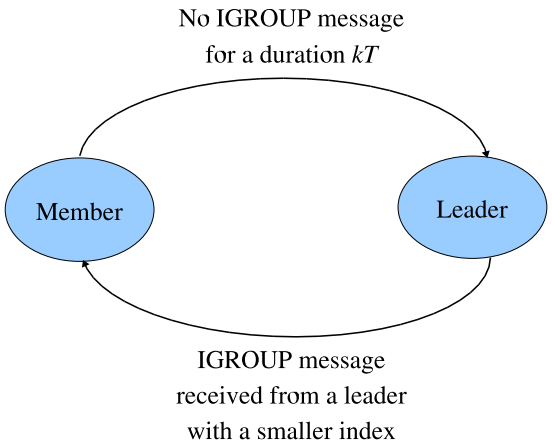


Figure 6.2: Transition between the member and the leader state

must collect **IREPLY** messages and form the corresponding **IGROUP** message. When there is a change in the group (arrival or withdrawal) the group leader must change its contribution. Additionally, irrespective of the modification of the composition of the group, the group leader must change its contribution periodically, to maintain the security of the session key.

When a group leader is elected, the latter may choose to wait additional periods before sending a **IGROUP** containing the contributions of the group members. Doing so, the group leader may avoid unnecessary changes to the session key due to the lack of receipt of all contributions in time.

In the group leader state, a node will also look out for **IGROUP** messages from another group leader. If it receives such a message from another group leader holding a smaller node index, the node changes its state to the member state. In the group member state, a node will have to send **IREPLY** messages periodically. Like the group leader, a group member must change its contribution periodically with a period P (see Figure 6.3). We will assume that P is a large multiple of T.

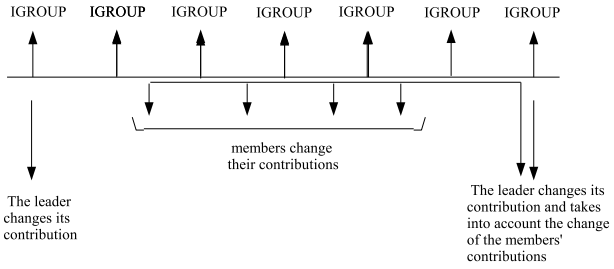


Figure 6.3: Sending **IGROUP** and **IREPLY** messages

To simplify the procedure and to avoid unnecessary computations we can assume that the group leader does not instantly include a new contribution of a group member in the **IGROUP** message, instead it will wait for the change of its own contribution to take into account all new contributions of nodes. This is possible since the contribution of the node member is included in the **IGROUP** message.

Both **IGROUP** and **IREPLY** messages must be sent periodically for each interval T. To reduce the probability of collision of these messages, we add a jitter to times when the GKA messages shall be sent by the group members and the group leader.

In Table 6.6, we have given examples of figures for our GKA protocol. We can notice that l and t_{rtd} will heavily depend on the number of nodes in the network and on the topology of the network.

Parameter	Value	Constraint
P: key renew period	20 min	
T: period of IGROUP messages	5s	
k: number of messages losses before assuming a node leaves	3	large enough to be sure that the message is not simply lost
l: backoff window	20	large enough to avoid collision during the group leader election
t_{rtd}: backoff slot for leader election	100 ms	more than a round trip delay

Table 6.6: Protocol parameters

6.5 Computational overhead

Table 6.7 describes the cost, on an average small device (COMPAQ iPAQ), of elliptic curve cryptography which is more efficient than classical cryptography relying on bigger groups. Basically, for a security level of 2^{80}, such a device can perform almost 100 operations per second. Thus the latency of elliptic curve exponentiation is 10 msec per device, except for the leader whose computational cost grows linearly with the size of the group. Thus there is concern for this particular node. Assuming that the leader devotes half its time to cryptogaphic operations, managing a group of size 50 will impose a delay of 1 second before being able to send the blinded response.

The above computational load on the group leader is in the case where the group leader receives all the blinded secrets at once, and has to give the blinded response also at once. In practice, the group leader will receive the blinded secrets at different time slots. It is then possible to perform operations in a batch: the group leader can generate its own secret in advance, and compute on the fly the blinded reponses $(g^{r_i})^{r_0}$ upon reception of each blinded secret g^{r_i}. He can also stepwise compute the product $(g^{r_1})^{r_0} \cdots (g^{r_m})^{r_0}$, where m is the index of the last received contribution. When he has to broadcast the `IGROUP` message, all the computationaly-intense cryptographical operations, necessary to generate the blinded responses, have already been performed.

Group	Size of contributions	blindings/second=recoveries/second
Modular Field	1024 bits	10
Elliptic curve	160 bits	93

Table 6.7: Performance of elliptic curve cryptography, versus a classical group (modular integers)on a iPAQ, StrongARM-1110, using the `openssl` implementation, for a security level of 2^{80}. Blinding means computing g^{r_i}, and recovering means computing g^{r_0} from the blinded response $g^{r_i r_0}$ of the leader .

6.6 Conclusion

We have discussed a group key agreement protocol for handling ad hoc groups of small to moderate size. We have fully specified the implementation details needed for actual use of the protocol, relying on known network techniques such as self election, periodic broadcast, back-off techniques. The protocol is robust in the sense that connectivity losses do not impair its functioning. Our experiments show that the computational cost of public key cryptography is kept reasonably low. If we consider the constraints in ad hoc networks; of no network structure, high dynamism, restricted bandwidth, the protocol presented here is among the few GKA protocols suitable for ad hoc networks.

Conclusions and Future Work

Rapid advancements in wireless technology have resulted in the vast proliferation of wireless communications networks both in the commercial and military sectors. Mobile ad hoc networks are an example of these wireless communications networks which are growing in popularity due to the abundance of low-cost mobile devices, the speed and convenience of deployment, and the independance of network infrastructure. In such an IP-based network, IP address assignment to mobile devices is one of the most important network configuration parameters. Furthermore, if successful, prevasive computing will lead to the proliferation of networked devices on a scale that we have never experienced before. Even if the nodes were static, manually configuring potentially billions of devices would be too time-consuming and error-prone.

IPv6 is a protocol which offers on one hand a big address space which is necessary to configure these billions of devices, and on the other hand it offers a new well designed protocol stack which implements the autoconfiguration feature. For these reasons, we were interested in *Part I* of this thesis in adapting OLSR protocol to IPv6. We presented the issues and necessary changes to adapt OLSR to IPv6. These changes mainly include the IPv6 stateless address autoconfiguration, but also a number of relatively less explored issues such as IPv6 addressing issues, and changes to the OLSR protocol itself.

In *Part II* of this thesis, we presented a more elaborate autoconfiguration algorithm for both OLSR protocol versions, single interface OLSR network and multi-interfaces OLSR network. The proposed autoconfiguration algorithm relies on two procedures; first, immediate request for an IP address to the entire network and second, periodic checking of address duplications (using node identifiers). For this purpose, a special control message MAD (Multiple Address Declaration) is periodically broadcast by each node to the whole network. The MAD message conveys a random identifier with the address of the node. This control message uses the OLSR MPR-flooding algorithm to reach all the nodes in the network; however, special rules have been added to ensure that even with address duplications the MAD messages will be propagated throughout the entire network.

A detailed analysis of the overhead induced by our autoconfiguration algorithm (MAD message overhead) is presented, and we have shown by simulations that this overhead remains limited. In order to test the convergence of our autoconfiguration protocol, we have simulated the merger of networks generating massive address duplications. We computed the duration our autoconfiguration algorithm takes to detect and resolve all address conflicts after the merger. The results of the simulation experiments show that the autoconfiguration protocol can efficiently detect and resolve address duplications within seconds.
As a future work, we intend to modify our autoconfiguration protocol in such a way that it will be in conformance with the charter of autoconfiguration working group within the IETF. In fact, an autoconfiguration mechanism should be a fully IPv6 solution and should not be dependent on any specific MANET routing protocol.

Another core issue with mobile ad hoc networks is security. Because of the open nature of mobile ad hoc networks, the challenge of maintaining confidentiality and integrity of data exchanged between nodes in the network is not an easy task. One way to address these issues is the use of symmetric key cryptography, relying on a secret key shared by all the members of the network. In *Part III* of this thesis, we have shown that Group Key Agreement (GKA) protocols are suitable for establishing and maintaining such a session key in these dynamic networks. We have taken an existing GKA protocol, which is robust to connectivity losses and discuss all the issues for good functioning of this protocol in ad hoc networks. We have fully specified the implementation details and network parameters which significantly reduce the computational burden of using public key cryptography in such networks.
So far, we have focused on designing the architecture of the protocol and its implementation details. Now, our goal is to extend our work by making simulations to confirm its robustess. Another interesting direction for future work may also be to conceive a security model adapted to mobile ad hoc networks. This will allow to build solid formal proofs for security protocols in mobile ad hoc networks.

List of Figures

List of Tables

Bibliography

Publications

[1] S. Boudjit, A. Laouiti, P. Minet, and C. Adjih. OLSR for IPv6 networks. *Proceedings of Med-Hoc-Net 2004*, Bodrum - Turkey, June 2004.

[2] S. Boudjit, A. Laouiti, P. Muhlethaler, and C. Adjih. Duplicate address detection and autoconfiguration in OLSR. *INRIA Research Report RR-5475*, January 2005.

[3] S. Boudjit, A. Laouiti, P. Muhlethaler, and C. Adjih. Duplicate address detection and autoconfiguration in OLSR. *Proceedings of IEEE SNPD/SAWN 2005*, Maryland - USA, May 2005.

[4] S. Boudjit, A. Laouiti, P. Muhlethaler, and C. Adjih. Duplicate address detection and autoconfiguration in OLSR. *JUCS: Journal of Universal Computer Science (accepted for publication)*, to appear on March 2007.

[5] S. Boudjit, A. Laouiti, P. Muhlethaler, and C. Adjih. Duplicate address detection with multiple conflicts and autoconfiguration in OLSR. *Proceedings of IEEE ICSIT 2005*, Algiers - Algeria, July 2005.

[6] S. Boudjit, C. Adjih, A. Laouiti, and P. Muhlethaler. A duplicate address detection and autoconfiguration mechanism for a single-interface OLSR network. *First Asian Internet Engineering Conference AINTEC2005*, Bangkok - Thailand, December 2005.

[7] S. Boudjit, C. Adjih, A. Laouiti, P. Muhlethaler, and Philippe Jacquet. An Advanced Configuration and Duplicate Address Detection mechanism for a multi-interface OLSR Network. *INRIA Research Report RR-5747*, November 2005.

[8] S. Boudjit, C. Adjih. DAD-MPR flooding protocol, Evaluation through Simulation. *Submitted to IFIP-Networking 2007*, December 2006.

[9] B. Raghav, P. Muhlethaler, D. Augot, C. Adjih, S. Boudjit, A. Laouiti. Efficient and Dynamic Group Key Agreement in Ad hoc Networks. *INRIA Research Report RR-5915*, May 2006.

[10] C. Adjih, S. Boudjit, P. Jacquet, A. Laouiti, P. Muhlethaler. Address autoconfiguration in Optimized Link State Routing Protocol. *IETF Draft draft-laouiti-manet-olsr-address-autoconf-00.txt*, February 2005.

[11] C. Adjih, S. Boudjit, P. Jacquet, A. Laouiti, P. Muhlethaler, Pr. Mase. Address autoconfiguration in Optimized Link State Routing Protocol. *IETF Draft draft-laouiti-manet-olsr-address-autoconf-01.txt*, July 2005.

[12] S. Boudjit. Étude et simulation d'un schéma de réservation de bande passante dans les réseaux ad-hoc. *INRIA Research Report RR-4102*, January 2001.

References

[13] ETSI Special Mobile Group (SMG). Global System for Mobile communications (GSM).

[14] IEEE 802.16 working group. Amendment to Recommended Practice for Coexistence of Fixed Broadband Wireless Access Systems. *IEEE Std 802.16.2a*, August 2001.

[15] IEEE 802.16 working group. IEEE Recommended Practice for Local and metropolitan area networks: Coexistence of Fixed Broadband Wireless Access Systems. *IEEE Std 802.16.2-2004*, March 2004.

[16] IEEE 802.16 working group. IEEE Standard for Local and metropolitan area networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems. *IEEE Std 802.16e*, December 2005.

[17] Paul Mühlethaler. 802.11 et les réseaux sans fil. *Eyrolles, 2002.*

[18] ETSI STC-RES10 Committee. Broadband Radio Access Networks(BRAN): High Performance Radio Local Area Networks type 1; Functional specifications, 1998.

[19] ETSI STC-RES10 Committee. Broadband Radio Access Networks(BRAN): High Performance Radio Local Area Networks type 2; system overview, 1997.

[20] IEEE 802.11 working group. IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. *IEEE Std 802.11-1997*, June 1997.

[21] IEEE 802.11 working group. Local and Metropolitan networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High Speed Physical Layer in the 5 GHz band. *IEEE Std 802.11a-1999(R2003)*, IEEE 2003.

[22] IEEE 802.11 working group. Local and Metropolitan networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher Speed Physical Layer (PHY) Extension in the 2.4 GHz band. *IEEE Std 802.11b-1999(R2003)*, IEEE 2003.

[23] IEEE 802.11 working group. Local and Metropolitan networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Further Higher Data Rate Extension in the 2.4 GHz Band. *IEEE Std 802.11g-2003*, June 2003.

[24] IEEE 802.15 working group. Local and metropolitan area networks-Specific requirements - Part 15: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPANs). *IEEE Std 802.15.1-1999*, March 1999.

[25] IEEE 802.15 working group. Local and metropolitan area networks-Specific requirements - Part 15.1a: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks (WPANs). *IEEE Std 802.15.1-2005*, June 2005.

[26] http://www.3gpp.org/. The 3rd Generation Partnership Project (3GPP).

[27] Telecommunications Industry Association (TIA). Interim Standard (IS-95).

[28] Telecommunications Industry Association (TIA). Interim Standard (IS-2000).

[29] L. Kleinrock, and Fouad A. Tobagi. Packet switching in radio channels - Part I: Carrier Sense Multiple Access modes and their throughput delay characteristics. *IEEE transactions on communications, volume COM-23, pages 1400-1416*, December 1975.

[30] Chlamtac, W. Fanta, and K.D. Levin. BRAM: The Broadcast Recognizing Access Method. *IEEE transactions on communications, volume COM-27, pages 1183-1190*, 1979.

[31] L. Kleinrock, and Scholl. Packet switching in radio channels: New Conflict Free Multiple Access schemes. characteristics. *IEEE transactions on communications, pages 1015-1029*, 1980.

[32] J. Postel. Internet Protocol. *IETF RFC 791*, September 1981.

[33] C. Huitema. IPv6 : The New Internet Protocol, (Second edition). *Prentice Hall, November 1997.*

[34] G. Cizault. IPv6, théorie et pratique. *O'Reilly, November 2005.*

[35] S. Deering, R. Hinden. Internet Protocol, Version 6 (IPv6)Specification. *IETF RFC 2460*, December 1998.

[36] S. Bradner, A. Mankin. The Recommendation for the IP Next Generation Protocol. *IETF RFC 1752*, January 1995.

[37] E. Nordmark, R. Gilligan. Basic Transition Mechanisms for IPv6 Hosts and Routers. *IETF RFC 4213*, October 2005.

[38] B. Carpenter, C. Jung. Transmission of IPv6 over IPv4 Domains without Explicit Tunnels. *IETF RFC 2529*, March 1999.

[39] B. Carpenter, K. Moore. Connection of IPv6 Domains via IPv4 Clouds. *IETF RFC 3056*, February 2001.

[40] J. McCann, S. Deering, J. Mogul. Path MTU Discovery for IP version 6. *IETF RFC 1981*, August 1996.

[41] R. Rivest. The MD5 Message-Digest Algorithm. *IETF RFC 1321*, April 1992.

[42] S. Kent. IP Encapsulating Security Payload (ESP). *IETF RFC 4303*, December 2005.

[43] S. Kent. IP Authentication Header (AH). *IETF RFC 4302*, December 2005.

[44] R. Hinden, S. Deering. Internet Protocol Version 6 (IPv6) Addressing Architecture. *IETF RFC 3513*, April 2003.

[45] R. Hinden, S. Deering. IP Version 6 Addressing Architecture. *IETF RFC 4291*, February 2006.

[46] R. Hinden, S. Deering, E. Nordmark. IPv6 Global Unicast Address Format. *IETF RFC 3587*, August 2003.

[47] S. Thomson, T. Narten. IPv6 Stateless Address Autoconfiguration. *IETF RFC 2462*, December 1998.

[48] R. Droms. Dynamic Host Configuration Protocol. *IETF RFC 2131*, March 1997.

[49] R. Droms, Ed., J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). *IETF RFC 3315*, July 2003.

[50] T. Lemon, B. Sommerfeld Node-specific Client Identifiers for Dynamic Host Configuration Protocol Version Four (DHCPv4). *IETF RFC 4361*, February 2006.

[51] Huitema, C. and B. Carpenter. Deprecating Site Local Addresses. *IETF RFC 3879*, September 2004.

[52] S. Thomson, C. Huitema, V. Ksinant, M. Souissi. DNS Extensions to Support IP Version 6. *IETF RFC 3596*, October 2003.

[53] A. Conta, S. Deering, M. Gupta, Ed. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. *IETF RFC 4443*, March 2006.

[54] Guillaume Chelius, Eric Fleury. Ananas : A New Adhoc Network Architectural Scheme. *INRIA Research Report 4354*, January 2002.

[55] C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, P. Muhletaler, P.Minet, A. Qayyum, L. Viennot. Optimized Link State Routing Protocol. *IETF RFC 3626*, October 2003.

[56] J. Moy. Open Shortest Path First Version 2. *IETF RFC 2328*, April 1998.

[57] A. Qayyum, A. Laouiti, L. Viennot. Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks. *HICSS: Hawai Int. Conference on System Sciences*, January 2002.

[58] A. Qayyum, A. Laouiti, L. Viennot. Multipoint Relaying: An Efficient Technique for Flooding in Mobile Wireless Networks. *INRIA Research Report RR-3898*, February 2000.

[59] K. Mase, C. Adjih. No Overhead Autoconfiguration OLSR. *IETF Draft (work in progress)*, May 2005.

[60] C.Perkins, J.Malinen, R.Wakikawa, E.Belding-Royer, and Y.Sun. IP Address Autoconfiguration for Ad Hoc Networks. *Internet Draft, IETF Working Group MANET*, Work in progress, November 2001.

[61] K.Weniger, M.Zitterbart. IPv6 Autoconfiguration in Large Scale Mobile Ad-Hoc Networks. *Proceedings of European Wireless 2002*, February 2002.

[62] K.Weniger. PACMAN: Passive AutoConfiguration for Mobile Ad hoc Networks. *Proceedings of IEEE WCNC 2003*, March 2003.

[63] S.Nesargi, R.Prakash. MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network. *InfoCom 2002*, June 2002.

[64] Archan Misra, Subir Das, Anthony McAuley, and Sajal K.Das. Autoconfiguration, Registration, and Mobility Management for prevasive Computing. *IEEE Personal Communication*, August 2001, pp 24-31.

[65] Hongbo Zhou, Lionel M. Ni, and Matt W.Mutka. Prophet Address Allocation for Large Scale MANETs. *IEEE INFOCOM 2003*, March 2003.

[66] Sayrafiezadeh M. The Birthday Problem Revisited. *Math. Mag. 67*, 1994, pp 220-223.

[67] Laurent Viennot, Philippe Jacquet, Thomas Heide Clausen. Analyzing control Traffic Overhead in Mobile Ad-hoc Network Protocols versus Mobility and Data Traffic Activity. *Proceedings of IFIP Med-Hoc-Net 2002.*

[68] Douglas M.Blough, Giovanni Resta, Mauro Leoncini, Paolo Santi. The K-Neighbors Protocol for Symmetric Topology Control in Ad Hoc Networks. *MobiHoc 2003*, June 2003.

[69] C.Perkins, E.Belding-Royer, and S.Das. Ad Hoc On-Demand Distance Vector(AODV) Routing. *IETF RFC 3561*, July 2003.

[70] R. Ogier, F. Templin, M. Lewis. Topology Dissemination Based on Reverse-Path Forwarding (TBRPF). *IETF RFC 3684*, February 2004.

[71] C.Perkins, and P.Bhagwat. Highly dynamic Destination Sequenced Distance Vector(DSDV) for mobile computers. *In Proceedings of SIGCOMM 94 Conference on Communications Architecture, Protocols and Applications*, August 1994.

[72] C. Hedrick. Routing Information Protocol (RIP). *IETF RFC 1058*, June 1988.

[73] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). *INTERNET-DRAFT, July 2004, Work in progress.*

[74] Zygmunt J. Haas, Marc R. Pearlman, and P. Samar The Zone Routing Protocol (ZRP) for Ad Hoc Networks. *INTERNET-DRAFT, July 2002, Work in progress.*

[75] Zygmunt J. Haas, Marc R. Pearlman, and P. Samar The Interzone Routing Protocol (IERP) for ad hoc networks. *INTERNET-DRAFT, July 2002, Work in progress.*

[76] Zygmunt J. Haas, Marc R. Pearlman, and P. Samar The Intrazone Routing Protocol (IARP) for Ad Hoc Networks. *INTERNET-DRAFT, July 2002, Work in progress.*

[77] Zygmunt J. Haas, Marc R. Pearlman, and P. Samar The Bordercast Resolution Protocol (BRP) for ad hoc networks. *INTERNET-DRAFT, July 2002, Work in progress.*

[78] S. A.Tanenbaum. Computer Networks. *Prentice Hall, 1996.*

[79] Charles E. Perkins, Jari T. Malinen, Ryuji Wakikawa, Anders Nilsson and Antti J. Tuominen. Internet Connectivity for Ad Hoc Networks. *Wireless Communication and Mobile Computing*, number 5, volume 2, August 2002, pp 465-482.

[80] R. Hinden, S. Deering. IP Version 6 Addressing Architecture. *IETF RFC 2373*, July 1998.

[81] R. Hinden, S. Deering. IPv6 Multicast Address Assignments. *IETF RFC 2375*, July 1998.

[82] T. Narten, E. Nordmark, W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). *IETF RFC 2461*, December 1998.

[83] A. Conta, S. Deering. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. *IETF RFC 2463*, December 1998.

[84] R. Hinden, S. Deering. IP Version 6 Addressing Architecture. *IETF RFC 3513*, April 2003.

[85] R. Hinden, S. Deering, E. Nordmark. IPv6 Global Unicast Address Format. *IETF RFC 3587*, August 2003.

[86] IEEE. Guidelines for 64-bit Global Identifier (EUI-64) Registration Authority. *http://standards.ieee.org/db/oui/tutorials/EUI64.html*, March 1997.

[87] C. Adjih, T. Clausen, Anis Laouiti, Paul Muhlethaler, and Daniele Raffo. Securing the OLSR routing protocol with or without compromised nodes. *Rapport INRIA*, RR-5494:55, February 2005. http://www.inria.fr/rrrt/rr-5494.html.

[88] Cedric Adjih, Thomas Clausen, Philippe Jacquet, Anis Laouiti, Paul Muhlethaler, and Daniele Raffo. Securing the OLSR protocol. In *Proceedings of the 2nd IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2003)*, June 25–27 2003.

[89] Raghav Bhaskar, Daniel Augot, Valerie Issarny, and Daniele Sacchetti. A Three Round Authenticated Group Key Agreement Protocol for Ad hoc Networks. *Accepted for publication in Journal on Prevasive and Mobile Computing* , 2006.

[90] K. Becker and U. Wille. Communication complexity of group key distribution. In *Proceedings of 5th ACM Conference on Computer and Communications Security*, pages 1–6. ACM Press, 1998.

[91] M. Boulkenafed and V. Issarny. AdHocFS: Sharing files in WLANs. In *2nd International Symposium on Network Computing and Applications*, pages 156–163. IEEE Computer Society, 2003.

[92] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer-Verlag, 2003.

[93] C. Boyd and J.M.G. Nieto. Round-optimal contributory conference key agreement. In *Public Key Cryptography '03*, pages 161–174. LNCS 2567, 2003.

[94] E. Bresson and D. Catalano. Constant round authenticated group key agreement via distribut ed computation. In *Proceedings of Public Key Cryptography*, pages 115–119. LNCS 2567, 2004.

[95] E. Bresson, O. Chevassut, A. Essiari, and D. Pointcheval. Mutual authentication and group key agreement for low-power mobile devices. In *Proceedings of the 5th IFIP-TC6 International Conference on Mobile and Wireless Communication Networks*, pages 59–62. World Scientific Publishing, 2003.

[96] E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group Diffie Hellman key exchange under standard assumptions. In *Advances in Cryptology - EUROCRYPT '02*, pages 321–326. LNCS 2332, 2002.

[97] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Proceedings of Advances in Cryptology - EUROCRYPT*, volume 839, pages 275–286. LNCS, 1994.

[98] R. Dutta and R. Barua. Dynamic group key agreement in tree-based setting. In *ACISP*, pages 101–112, 2005.

[99] Y. Hu, A. Perrig, and D. Johnson. Ariadne:: A secure on-demand routing protocol for ad hoc networks. In *MobiCom '02: Proceedings of the 8th annual international Conference on Mobile Computing and Networking*, pages 12–23. ACM Press, 2002.

[100] I. Ingemarsson, D. T. Tang, and C.K. Wong. A conference key distribution system. *IEEE Transactions on Information Theory*, 28(5):714–720, 1982.

[101] J. Katz and M. Yung. Scalable protocols for authenticated group key exchange - full version. In *Advances in Cryptology - CRYPTO '03*, pages 110–125. LNCS 2729, 2003.

[102] H-J. Kim, Lee S-M, and D.H. Lee. Constant-round authenticated group key exchange for dynamic groups. In *Proceedings of Advances in Cryptology - ASIACRYPT*, volume 3329, pages 245–259. LNCS, 2004.

[103] Y. Kim, A. Perrig, and G. Tsudik. Group key agreement efficient in communication. *IEEE Transactions on Computers*, 53(7):905–921, July 2004.

[104] J. Nam, J. Lee, S. Kim, and D. Won. DDH based group key agreement for mobile computing. http://eprint.iacr.org/2004/127, 2004.

[105] A. Perrig. Efficient collaborative key management protocols for secure autonomous group communication. In *Proceedings of International workshop on cryptographic techniques and electronic commerce*, pages 192–202, 1999.

[106] J. Pieprzyk and C.-H. Li. Multiparty key agreement protocols. *IEE Proceedings - Computers and Digital Techniques*, 147(4):229–236, 2000.

[107] Ricardo Staciarini Puttini, Ludovic Me, and Rafael Timóteo de Sousa. Certification and authentication services for securing MANET routing protocols. In *Proceedings of the 5th IFIP TC6 International Conference on Mobile and Wireless Communications Networks*, October 2003.

[108] G-C. Roman, Q. Huang, and A. Hazemi. Consistent group membership in ad hoc networks. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*, pages 381–388. IEEE Computer Society, 2001.

[109] D.G. Steer, L. Strawczynski, W. Diffie, and M. Wiener. A secure audio teleconference system. In *Proceedings of Advances in Cryptology - CRYPTO*, volume 403, pages 520–528. LNCS, 1988.

[110] M. Steiner, G. Tsudik, and M. Waidner. Diffie-hellman key distribution extended to group communication. In *Proceedings of 3rd ACM Conference on Computer and Communications Security*, pages 31–37. ACM Press, 1996.

[111] W.-G. Tzeng and Z.-J. Tzeng. Round-efficient conference key agreement protocols with provable security. In *Proceedings of Advances in Cryptology - ASIACRYPT*, volume 1976, pages 614–627. LNCS, 2000.