# A Study of Internet Connectivity for Mobile Ad Hoc Networks in NS 2

Alex Ali Hamidian

January 2003

Department of
Communication Systems
Lund Institute of Technology, Lund University

Department of Communication Systems
Lund Institute of Technology, Lund University
Box 118
S-221 00 Lund
Sweden

LUTEDX (TETS-5477)/1-41/(2003)&local4

**Abstract**

Ad hoc networking allows portable devices to establish communication independent of a central infrastructure. However, the fact that there is no central infrastructure and that the devices can move randomly gives rise to various kind of problems, such as routing and security. In this thesis the problem of routing is considered.

There are several ad hoc routing protocols, such as AODV[1], DSR[2], OLSR[3] and ZRP[4], that propose solutions for routing within a mobile ad hoc network. However, since there is an interest in communication between not only mobile devices in an ad hoc network, but also between a mobile device in an ad hoc network and a fixed device in a fixed network (e.g. the Internet), the ad hoc routing protocols need to be modified.

In this thesis the ad hoc routing protocol AODV is used and modified to examine the interconnection between a mobile ad hoc network and the Internet. For this purpose Network Simulator 2, NS 2, has been used. Moreover, three proposed approaches for gateway discovery are implemented and investigated.

[1] Ad hoc On-Demand Distance Vector
[2] Dynamic Source Routing
[3] Optimized Link State Routing Protocol
[4] Zone Routing Protocol

# Acknowledgment

I would like to thank my supervisor, Professor Ulf Körner, for his continual encouragement, help and guidance throughout the work.

I would also like to thank, Ph.D. student Anders Nilsson, for sharing his knowledge and for his constant support during the course the project.

Moreover, I wish to thank Ph.D. students Shabnam Aprin and Eligijus Kubilinskas for giving feedback on the report.

Finally, I would like thank my three brothers, Reza, Amir and Arash, and especially my parents Susanne and Davoud for their support and understanding.

Ali Hamidian
Lund Institute of Technology
January, 2003

# Contents

# Chapter 1

# Introduction

The use of the Internet grew explosively in the 90s. Today, many expect one to be able to connect to the Internet. For example, email has become an important way for people from different parts of the world to keep in touch with each other. It is also an excellent way for scientists around the world to collaborate and share ideas with each other. However, to be able to connect to the Internet one has to find a stationary computer with a modem or a network card. This limits one's possibilities to connect to the Internet. Therefore, it is desirable to have access to the Internet from portable devices such as mobile phones, laptops or personal digital assistants (PDAs).

In view of the increasing demand for wireless information and data services, providing faster and more reliable mobile access is becoming an important concern. The widely deployed and successful mobile communication standard *global system for mobile communication* (GSM) has spoiled us by our expecting to reach, and be reached, by everyone at (almost) every place. Nowadays, not only mobile phones, but also laptops and PDAs are used by people in their professional and private lives. These devices are used separately for the most part; i.e. their applications do not interact. Sometimes, however, a group of mobile devices form a spontaneous, temporary network as they approach each other. This allows e.g. participants at a meeting to share documents and presentations. These kind of spontaneous, temporary networks are referred to as *mobile ad hoc networks* (MANETs) (sometimes just called *ad hoc networks*) or *multihop wireless networks*, and are expected to play an important role in our daily lives in the near future.

A mobile ad hoc network is a network formed and functioning without any established infrastructure or centralized administration and consists of mobile nodes that use a wireless interface to communicate with each other. These mobile nodes serve as both hosts and routers so they can forward packets on behalf of each other. Hence, the mobile nodes are able to communicate beyond their transmission range by supporting multihop communication.

The issue of routing in a mobile ad hoc network becomes a challenging task since the mobile nodes are free to move randomly. Ad hoc routing protocols can be classified into three classes[1]: proactive, reactive and hybrid routing protocols. In proactive rout-

[1]There are other ways to categorize ad hoc routing protocols, for example: Flat routing, Hierarchical

ing the routing table of every node is updated periodically. On the contrary, reactive routing is performed on-demand, i.e. the sending node searches for a route to the destination node only when it needs to communicate with it. Hybrid routing uses a mixture of these two routing approaches. That is, proactive routing is used in a limited area around the mobile node and reactive routing is used outside this area.

Mobile Ad Hoc Networking (MANET) is the name of a working group in the Internet Engineering Task Force (IETF) and it serves as a meeting place for people dealing with MANET approaches. The primary focus of the working group is to develop and evolve MANET routing specifications and introduce them to the Internet Standards track. The goal is to support networks scaling up to hundreds of routers according to the official web page [17].

The layout of the thesis is as follows:
Chapter 2 describes the project goals. Chapter 3 gives an overview of the concept of mobile ad hoc networks in general. In addition, it presents some of several promising ad hoc routing protocols. Chapter 4 discusses interworking between mobile ad hoc networks and fixed networks (e.g. the Internet). It also discusses some problems that occur when these different networks are integrated and finally it presents conceivable solutions. Chapter 5 considers three approaches for gateway discovery and discusses advantages and disadvantages of them. Chapter 6 describes the simulation environment used in this project. Chapter 7 explains the simulation scenario and examine the results. Chapter 8 summarizes and concludes the thesis.

Some details of the implementation of the Internet draft "Global Connectivity for IPv6 Mobile Ad Hoc Networks" are presented in appendix A. Appendix B presents the implementation of the three discovery methods examined in this project. Finally, a list of acronyms can be found at the end of this report.

## 1.1 Project Description

The ad hoc routing protocol AODV [12] is one of the promising routing protocols investigated by the MANET working group. It can be used in a mobile ad hoc network to route packets between mobile nodes. However, it cannot provide Internet access to the mobile nodes because it does not support routing between a fixed network like the Internet and a mobile ad hoc network. In the Internet draft "Global Connectivity for IPv6 Mobile Ad Hoc Networks" a solution is presented where the AODV protocol is modified in such a way that it can route packets not only within a mobile ad hoc network, but also to a fixed, wired network. This project aims to implement this solution in a simulation environment. For this purpose, the simulation tool Network Simulator 2 (NS 2) [16], has been used.

The goal of the thesis project is twofold:

- To modify the source code of AODV in NS 2 in accordance with the Internet draft "Global connectivity for IPv6 Mobile Ad Hoc Networks" which presents a solution where AODV is used to provide Internet access to mobile nodes.
- To implement and compare different approaches for gateway discovery.

routing and Geographic position assisted routing [7].

# Chapter 2

# Background

While much research has been done on routing protocols for autonomous mobile ad hoc networks during the last few years, there has not been much work published in the field of Internet access for mobile nodes in a mobile ad hoc network. There are some works where Mobile IP [11] is used to provide Internet access to the mobile nodes.

One solution is presented in the master's thesis "MIPMANET - Mobile IP for Mobile Ad Hoc Networks" [10]. This solution provides Internet access by using tunneling and Mobile IP with foreign agent care-of addresses. Mobile nodes that want Internet access register with a foreign agent and tunnel all packets destined for the Internet to the registered foreign agent. The foreign agent decapsulates the packets and forwards them to the destination. The ad hoc routing protocol AODV is used within the mobile ad hoc network and to deliver packets between mobile nodes and foreign agents.

In "Global Connectivity for IPv4 Mobile Ad hoc Networks" (often simply referred to as "Global4") [1] a solution is presented where AODV cooperates with the Mobile IP protocol. Mobile IP is used for mobile node registrations with a foreign agent, while AODV is used for routing within the mobile ad hoc network and for obtaining routes to the foreign agent. In this solution, the foreign agent discovery mechanism is incorporated into the ad hoc routing protocol.

There are also some works in which mobile IP is not used. The paper "Wireless Multihop Internet Access: Gateway Discovery, Routing and Addressing" [15] discusses interesting issues like gateway discovery and different kinds of routing policies. The master's thesis, "Gateway Detection and Selection for Wireless Multihop Internet Access" [2], (which reminds of the former paper) discusses gateway detection and selection in more detail.

The leading and most promising work in the field is the Internet draft "Global Connectivity for IPv6 Mobile Ad Hoc Networks" (often simply referred to as "Global6" compared to "Global4" mentioned above) [14]. Hence, in this project, the necessary parts of this draft have been implemented in NS 2, in order to provide Internet access to mobile nodes. However, some issues are not considered in this draft. These issues are discussed and conceivable solutions are presented in Section 4.5.

This thesis also considers gateway discovery. In particular, a solution for implementing the different approaches has been presented. In "Gateway Detection and Selection

for Wireless Multihop Internet Access" [2] and "Wireless Multihop Internet Access: Gateway Discovery, Routing and Addressing" [15] gateway discovery is discussed but none of them goes into deep with it and they do not consider the question of duplicated broadcast messages.

In the Internet draft "Global6", the term *Internet Gateway* is used instead of the term *gateway* that is used throughout this text. The reason to why the shortened term have been used is that no other kind of gateway is of importance in this project.

# Chapter 3

# Mobile Ad Hoc Networking

This chapter gives an overview of Mobile Ad Hoc Networking. Section 3.1 introduces the protocol stacks used in the Internet and MANET and compares them with the Open Systems Interconnection (OSI) model. Section 3.2 describes the different routing concepts. In Section 3.3 and 3.4 two reactive routing protocol are presented. Section 3.5 presents a proactive routing protocol and finally, in Section 3.6, a hybrid routing protocol is described.

## 3.1 The Protocol Stack

In this section the protocol stack for mobile ad hoc networks is described. This gives a comprehensive picture of, and helps to better understand, mobile ad hoc networks. Figure 3.1 shows the protocol stack which consists of five layers: *physical layer, data link layer, network layer, transport layer* and *application layer*. It have similarities to the TCP/IP protocol suite. As can be seen the OSI layers for *session*, *presentation* and *application* are merged into one section, the application layer.

On the left of Figure 3.1, the OSI model is shown. It is a layered framework for the design of network systems that allows for communication across all types of computer systems.

In the middle of the figure, the TCP/IP suite is illustrated. Because it was designed before the OSI model, the layers in the TCP/IP suite do not correspond exactly to the OSI layers. The lower four layers are the same but the fifth layer in the TCP/IP suite (the application layer) is equivalent to the combined session, presentation and application layers of the OSI model.

On the right, the MANET protocol stack - which is similar to the TCP/IP suite - is shown. The main difference between these two protocol stacks lies in the network layer. Mobile nodes (which are both hosts and routers) use an ad hoc routing protocol to route packets. In the physical and data link layer, mobile nodes run protocols that have been designed for wireless channels. Some options are the IEEE standard for wireless LANs, IEEE 802.11, the European ETSI standard for a high-speed wireless LAN, HIPERLAN 2, and finally an industry approach toward wireless personal

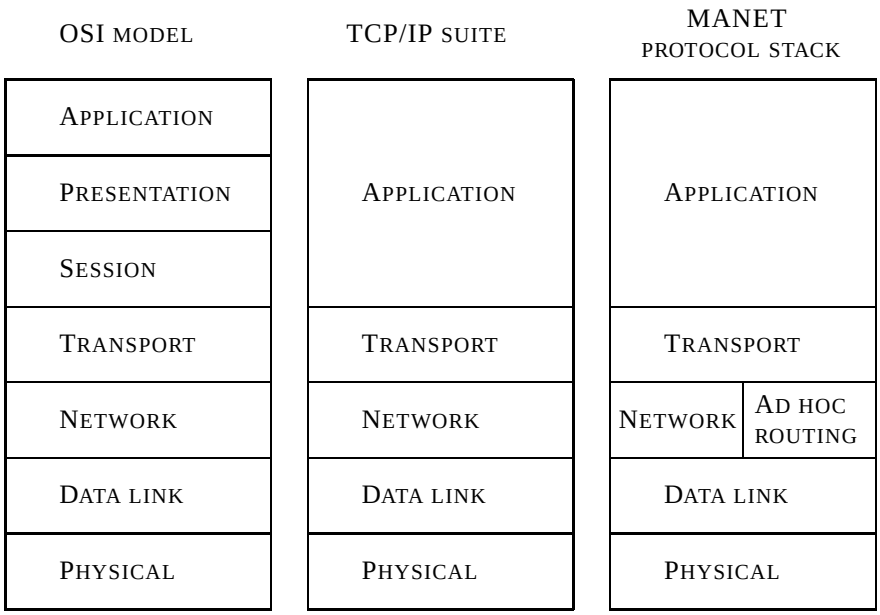


Figure 3.1: The OSI model, TCP/IP suite and MANET protocol stack.

area networks, i.e. wireless LANs at an even smaller range, Bluetooth [13]. In the simulation tool used in this project, the standard IEEE 802.11 is used in these layers.

This thesis focuses on ad hoc routing which is handled by the network layer. The network layer is divided into two parts: Network and Ad Hoc Routing. The protocol used in the network part is Internet Protocol (IP) and the protocol used in the ad hoc routing part is Ad hoc On-Demand Distance Vector (AODV). Other ad hoc routing protocols that can be used in this part of the network layer are discussed in Sections 3.4, 3.5 and 3.6. One of the reasons to why AODV has been used in this study is that it is one of the most developed routing protocols for mobile ad hoc networks. A second reason is that the Internet draft "Global6" [14] uses AODV as an example when illustrating how to extend the route discovery messaging of a reactive routing protocol for discovering gateways.

In the transport layer the User Datagram Protocol (UDP), is used in this study. The Transmission Control Protocol (TCP) is not used because there are some research showing that TCP does not perform well in mobile ad hoc networks. One reason to this is that in wired networks, lost packets are almost always due to congestion but in mobile ad hoc networks lost packets are more often caused by other reasons like route changes or transmission errors [6], [13].

### 3.1.1 Interworking

Whenever a mobile node is to send packets to a fixed network, it must transmit the packets to a gateway [14]. This will be discussed in more detail later in Chapter 4, but here the protocol stacks involved during communication between a mobile ad hoc network and the fixed Internet node are shown. A gateway acts as a bridge between a

MANET and the Internet. Therefore, it has to implement both the MANET protocol stack and the TCP/IP suite, as shown in the middle of Figure 3.2. Although the figure shows that all the layers are implemented for the gateway, it does not necessarily need all of the layers.

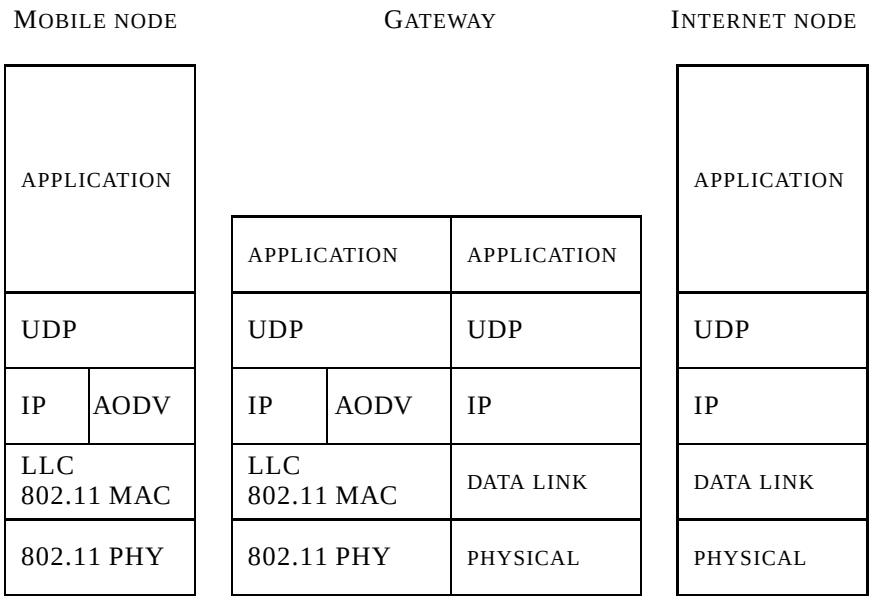


Figure 3.2: The protocol stacks used by mobile nodes, gateways and Internet nodes.

The protocol stack used by the mobile node is the MANET protocol stack discussed previously and shown on the right of Figure 3.1. The fixed Internet node uses the TCP/IP suite. A gateway, that must be able to translate between these two "languages", must understand the both architectures.

## 3.2 Proactive, Reactive and Hybrid Routing Protocols

Traditional distance-vector and link-state routing protocols [4] are proactive in that they maintain routes to all nodes, including nodes to which no packets are sent. For that reason they require periodic control messages, which leads to scarce resources such as power and link bandwidth being used more frequently for control traffic as mobility increases. One example of a proactive routing protocol is Optimized Link State Routing Protocol (OLSR) [3]. OLSR, which has managed to reduce the utilization of bandwidth significantly, is described in Section 3.5.

Reactive routing protocols, on the other hand, operate only when there is a need of communication between two nodes. This approach allows the nodes to focus either on routes that are being used or on routes that are in process of being set up. Examples of reactive routing protocols are Ad hoc On-Demand Distance Vector (AODV) [12], and Dynamic Source Routing (DSR) [9]. AODV is described in Section 3.3 and DSR in Section 3.4.

Both proactive and reactive routing have specific advantages and disadvantages that make them suitable for certain types of scenarios. Proactive routing protocols have their routing tables updated at all times, thus the delay before sending a packet is minimal. However, routing tables that are always updated require periodic control messages that are flooded through the whole network - an operation that consumes a lot of time, bandwidth and energy. On the other hand, reactive routing protocols determine routes between nodes only when they are explicitly needed to route packets. However, whenever there is a need for sending a packet, the mobile node must first find the route if the route is not already known. This route discovery process may result in considerable delay.

Combining the proactive and reactive approaches results in a hybrid routing protocol. A hybrid approach minimizes the disadvantages, but also the advantages of the two combined approaches. The Zone Routing Protocol (ZRP) [5] is such a hybrid reactive/proactive routing protocol. Each mobile node proactively maintains routes within a local region (referred to as the routing zone). Mobile nodes residing outside the zone can be reached with reactive routing. ZRP is discussed in Section 3.6.

## 3.3 Ad hoc On-Demand Distance Vector (AODV)

Ad hoc On-Demand Distance Vector, AODV, is a distance vector routing protocol that is reactive [12]. The reactive property of the routing protocol implies that it only requests a route when it needs one and does not require that the mobile nodes maintain routes to destinations that are not communicating. AODV guarantees loop-free routes by using sequence numbers that indicate how new, or fresh, a route is.

AODV requires each node to maintain a routing table containing one route entry for each destination that the node is communicating with. Each route entry keeps track of certain fields. Some of these fields are:

- Destination IP Address: The IP address of the destination for which a route is supplied
- Destination Sequence Number: The destination sequence number associated to the route
- Next Hop: Either the destination itself or an intermediate node designated to forward packets to the destination
- Hop Count: The number of hops from the Originator IP Address to the Destination IP Address
- Lifetime: The time in milliseconds for which nodes receiving the RREP consider the route to be valid
- Routing Flags: The state of the route; up (valid), down (not valid) or in repair

### 3.3.1 Route Discovery

Whenever a source node desires a route to a destination node for which it does not already have a route, it broadcasts a *route request* (RREQ) message to all its neighbours. The neighbours update their information for the source and create *reverse route entries* for the source node in their routing tables. A neighbour receiving a RREQ may send a *route reply* (RREP) if it is either the destination or if it has an unexpired route to the destination. If any of these two cases is satisfied, the neighbour unicasts a RREP back to the source. Along the path back to the source, intermediate nodes that receive the RREP create *forward route entries* for the destination node in their routing tables. If none of the two cases mentioned is satisfied, the neighbour rebroadcasts (forwards) the RREQ.

Each mobile node keeps a cache where it stores the source IP address and ID of the received RREQs during the last PATH_DISCOVERY_TIME seconds (see Section A.1.2). If a mobile node receives another RREQ with the same source IP address and RREQ ID during this period, it is discarded. Hence, duplicated RREQs are prevented and not forwarded.

When searching for a route to the destination node, the source node uses the *expanding ring search* technique to prevent unnecessary network-wide dissemination of RREQs. This is done by controlling the value of the time to live (TTL) field in the IP header. The first RREQ message sent by the source has TTL=TTL_START (see Section A.1.2). The value of TTL defines the maximal number of hops a RREQ can move through the mobile ad hoc network, i.e. it decides how far the RREQ is broadcasted. In other words, it implies that the RREQ which is broadcasted by the source, is received only by mobile nodes TTL hops away from the source (and of course all mobile nodes less than TTL hops away from the source). Apart from setting the TTL, the timeout for receiving a RREP is also set. If the RREQ times out without reception of a corresponding RREP, the source broadcasts the RREQ again. This time TTL is incremented by TTL_INCREMENT, i.e. the TTL of the second RREQ message is TTL_START + TTL_INCREMENT. This continues until a RREP is received or until TTL reaches TTL_THRESHOLD. If TTL reaches TTL_THRESHOLD a RREQ is sent with TTL=NET_DIAMETER, which disseminate the RREQ widely, throughout the MANET. Broadcasting a RREQ with TTL=NET_DIAMETER is referred to as a *network-wide search*. If a source node does a network-wide search and still does not receive a RREP, it may try again to find a route to the destination node, up to a maximum of RREQ_RETRIES times.

### 3.3.2 Route Maintenance

When a link in a route breaks, the node upstream of the break invalidates all its routes that use the broken link. Then, the node broadcasts a *route error* (RERR) message to its neighbors (TTL is set to one). The RERR message contains the IP address of each destination which has become unreachable due to the link break. Upon reception of a RERR message, a node searches its routing table to see if it has any route(s) to the unreachable destination(s) (listed in the RERR message) which use the originator of the RERR as the next hop. If such routes exist, they are invalidated and the node broadcasts a new RERR message to its neighbors. This process continues until the source receives

a RERR message. The source invalidates the listed routes as previously described and reinitiates the route discovery process if needed.

## 3.4 Dynamic Source Routing (DSR)

Dynamic Source Routing, DSR, is a reactive routing protocol that uses *source routing* to send packets [9]. It is reactive like AODV which means that it only requests a route when it needs one and does not require that the nodes maintain routes to destinations that are not communicating. It uses source routing which means that the source must know the complete hop sequence to the destination.

Each node maintains a route cache, where all routes it knows are stored. The route discovery process is initiated only if the desired route cannot be found in the route cache.

To limit the number of route requests propagated, a node processes the route request message only if it has not already received the message and its address is not present in the route record of the message.

As mentioned before, DSR uses source routing, i.e. the source determines the complete sequence of hops that each packet should traverse. This requires that the sequence of hops is included in each packet's header. A negative consequence of this is the routing overhead every packet has to carry. However, one big advantage is that intermediate nodes can learn routes from the source routes in the packets they receive. Since finding a route is generally a costly operation in terms of time, bandwidth and energy, this is a strong argument for using source routing. Another advantage of source routing is that it avoids the need for up-to-date routing information in the intermediate nodes through which the packets are forwarded since all necessary routing information is included in the packets. Finally, it avoids routing loops easily because the complete route is determined by a single node instead of making the decision hop-by-hop.

### 3.4.1 Route Discovery

Route Discovery is used whenever a source node desires a route to a destination node. First, the source node looks up its route cache to determine if it already contains a route to the destination. If the source finds a valid route to the destination, it uses this route to send its data packets. If the node does not have a valid route to the destination, it initiates the route discovery process by broadcasting a *route request* message. The route request message contains the address of the source and the destination, and a unique identification number.

An intermediate node that receives a route request message searches its route cache for a route to the destination. If no route is found, it appends its address to the route record of the message and forwards the message to its neighbors. The message propagates through the network until it reaches either the destination or an intermediate node with a route to the destination. Then a *route reply* message, containing the proper hop sequence for reaching the destination, is generated and unicast back to the source node.

### 3.4.2 Route Maintenance

Route Maintenance is used to handle route breaks. When a node encounters a fatal transmission problem at its data link layer, it removes the route from its route cache and generates a *route error* message. The route error message is sent to each node that has sent a packet routed over the broken link. When a node receives a route error message, it removes the hop in error from its route cache.

Acknowledgment messages are used to verify the correct operation of the route links. In wireless networks acknowledgments are often provided as e.g. an existing standard part of the MAC protocol in use, such as the link-layer acknowledgment frame defined by IEEE 802.11. If a built-in acknowledgment mechanism is not available, the node transmitting the message can explicitly request a DSR-specific software acknowledgment to be returned by the next node along the route.

## 3.5 Optimized Link State Routing Protocol (OLSR)

Optimized Link State Routing Protocol, OLSR, is another routing protocol developed for mobile ad hoc networks [3]. It is a proactive protocol, which means that the mobile nodes exchange topology information with each other regularly. As mentioned earlier in Section 3.2, there is a big disadvantage of proactive routing protocols. To keep the routing tables updated the network is flooded and every mobile node receives the same message from each of its neighbors. Thus, bandwidth and energy are wasted for useless messages. To avoid too many redundant retransmissions, the flooding process is optimized in OLSR. In OLSR, only some selected nodes forward the broadcast messages during the flooding process. These selected nodes are referred to as *multipoint relays* (MPRs).

### 3.5.1 Multipoint Relays

The use of Multipoint Relays (MPRs), as the only nodes that forward broadcast messages, substantially reduces the message overhead as compared to a classical flooding mechanism, where every node retransmits each message when it receives the first copy of the message. Another optimization is achieved by minimizing the set of links flooded in the network. As contrary to the classic link state algorithm, a mobile node declares only the MPR links to its neighbor nodes, rather than all links to all neighbors. In summary, multipoint relaying allow to reduce the utilization of bandwidth in two following ways:

1. The number of redundant retransmissions when flooding the network is greatly reduced.

2. Redundant topology advertisements are reduced.

## 3.6 Zone Routing Protocol (ZRP)

Zone Routing Protocol, ZRP, is a routing protocol that is designed for mobile ad hoc networks [5]. It is a hybrid protocol that is part proactive and part reactive.

The proactive part, uses a modified distance vector scheme within the *routing zone* of each node. The routing zone is determined by a *zone radius*, which is the minimum number of hops it should take to get to any node. Thus, each node has a routing zone, which is composed of nodes within its local area. This proactive component is called *Intrazone Routing Protocol* (IARP). The reactive component is called *Interzone Routing Protocol* (IERP), and uses queries to get routes when a node is to send a packet to a node outside of its routing zone.

ZRP uses a method called *bordercasting* in which a node asks all nodes on the border of its routing zone to look for the node outside of its routing zone.

### 3.6.1 Intrazone Routing Protocol (IARP)

The Intrazone Routing Protocol (IARP) proactively maintains routes to destinations within a local neighborhood, which is referred to as a routing zone. More precisely, a node's routing zone is defined as a collection of nodes whose minimum distance in hops from the node in question is no greater than a parameter referred to as the zone radius. Note that each node maintains its own routing zone. An important consequence is that the routing zones of neighboring nodes overlap.

### 3.6.2 Interzone Routing Protocol (IERP)

The operation of the reactive Interzone Routing Protocol (IERP) is quite similar to standard route discovery process of reactive routing protocols. An IERP route discovery is initiated when no route is locally available to the destination of an outgoing data packet. The source generates a route query message, which is uniquely identified by a combination of the source node's address and request number. The query is then relayed to a subset of neighbors as determined by the bordercast algorithm. Upon receipt of a route query message, a node checks if the destination lies in its zone or if a valid route to it is available in its route cache. If the destination is found, a route reply is sent back to the source. If not, the node bordercasts the query again.

### 3.6.3 Bordercast Resolution Protocol (BRP)

Since the topology of the local zone of each mobile node is known (this information is provided by IARP), global route discovery is simplified. Rather than broadcasting a route query from neighbor to neighbor, ZRP uses a concept called *bordercasting*. Bordercasting means that the route query is directed toward regions of the network that have not yet been covered by the query. A covered node is the one that belongs to the routing zone of a node that has received a route query. Hence, the route query traffic is reduced by directing route queries outwards from the source and away from covered routing zones.

# Chapter 4

# Internet Connectivity for Mobile Ad Hoc Networks

This chapter investigates interworking between mobile ad hoc networks and the Internet. Section 4.1 motivates the need of Internet connectivity for MANETs . In Sections 4.2 and 4.3 the extended route request and route reply messages of the reactive ad hoc routing protocol AODV are described. Section 4.4 describes how a mobile node can obtain a default route. Section 4.5 discusses some important issues that must be considered when trying to integrate a MANET to the Internet.

## 4.1 Motivation

Although an autonomous, stand-alone mobile ad hoc network is useful in many cases, a mobile ad hoc network connected to the Internet is much more desirable. So far, most of the research concerning mobile ad hoc networking has been done on protocols for autonomous mobile ad hoc networks. However, during the last few years, some work has been done concerning the integration of mobile ad hoc networks and the Internet.

In this thesis the access to the Internet from a multihop wireless network is investigated. To achieve this network interconnection, *gateways* that understand the protocols of both the mobile ad hoc network stack and the TCP/IP suite (see Figure 3.2) are needed. All communication between a mobile ad hoc network and the Internet must pass through the gateways.

The Internet draft “Global Connectivity for IPv6 Mobile Ad Hoc Networks” [14] describes how to provide Internet connectivity to mobile ad hoc networks. In particular, it explains how a mobile node and a gateway should operate. Further, it proposes and illustrates how to apply a method for discovering gateways. In the case for reactive routing protocols, the idea is to extend the route discovery messaging, so that it can be used for discovering not only mobile nodes but also gateways.

## 4.2 The Extended Route Request

The extended RREQ message contains exactly the same fields with the same functions as the ordinary RREQ message, except for a flag. This flag is called *Internet-Global Address Resolution Flag* and is referred to as the I-flag. Hence, the RREQ message extended with the I-flag is referred to as the RREQ_I message throughout this text. Figure 4.2 shows the format of the RREQ_I message.

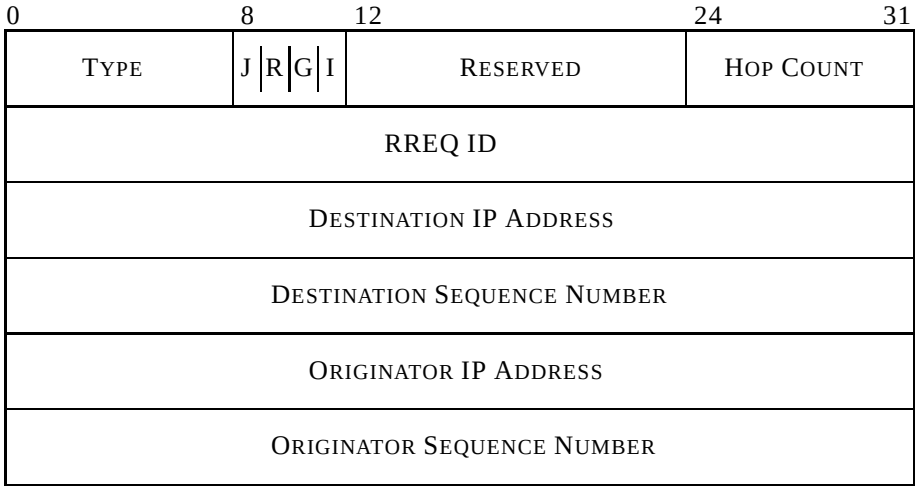


Figure 4.1: The format of a Route Request message extended with the I-flag.

The I-flag is used for global address resolution and it indicates that the source node requests global connectivity. The RREQ_I message plays the same role as the *router solicitation* message of ICMP. Section 5.2 describes how the RREQ_I message is used to reactively discover a gateway.

## 4.3 The Extended Route Reply

The extended RREP message contains exactly the same fields with the same functions as the ordinary RREP message, except for a flag. This flag is the same flag that has extended the RREQ message to the RREQ_I message, namely the *Internet-Global Address Resolution Flag* (or the I-flag). Hence, the RREP message extended with the I-flag is referred to as the RREP_I message throughout this text. Figure 4.3 shows the format of the RREP_I message.

The I-flag is used for global address resolution and, if set, it indicates that this RREP contains information about a gateway. The RREP_I message plays the same role as the *router advertisement* message of ICMP. Section 5.1 describes why the RREP_I message cannot be used to proactively discover a gateway. Instead Section 5.3 describes how RREP_I messages can be used by the gateways to proactively advertise information about themselves in a limited zone around the gateway.

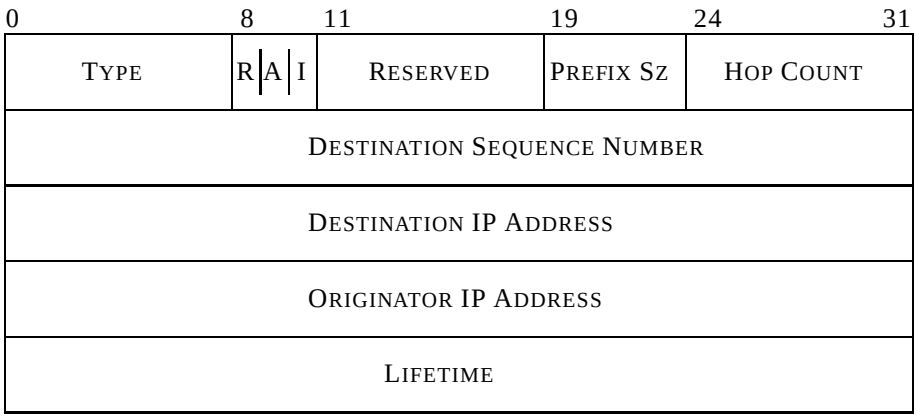


Figure 4.2: The format of a Route Reply message extended with the I-flag.

## 4.4 Obtaining a Default Route

A mobile node needs to learn the location and address of a gateway to be able to have access to the Internet. In other words, the mobile node needs a route to a gateway, which it uses as its default route, to be able to send packets to the Internet. This gateway information can be obtained in a few different ways:

- By relying on periodic advertisement messages broadcasted by the gateway (proactive gateway discovery)
- By sending a RREQ_I to the ALL_MANET_GW_MULTICAST address (reactive gateway discovery)
- By sending a RREQ which is received by a gateway

When a mobile node discovers a gateway, i.e. when it receives some message that, among other things, contains the address of the gateway, it creates a default route with the address of the gateway as the next hop. Chapter 5 describes three different methods for gateway discovery.

## 4.5 Problems and Conceivable Solutions

Assume that a mobile node (S) wants to communicate with another node (D) and that S does not have any route to D in its routing table. Hence, S does not know whether D is a mobile node (located within the MANET) or a fixed node (located on the Internet). Using AODV (see Section 3.3) as the ad hoc routing protocol, S broadcasts a RREQ, requesting for a route to D. If D is a mobile node, the node itself or another mobile node with a fresh route to it will unicast back a RREP to S. However, if D is a fixed node, no mobile node will send a reply to S. So, how can S find a route to D if D is a fixed node? According to “Global6”, if S broadcasts a RREQ but no corresponding RREP is received, S assumes that D is a fixed node. Hence, the packets are sent to the Internet by using the default route.

### 4.5.1 Mobile Nodes versus Fixed Nodes

As already said, if a mobile node (S) broadcasts a RREQ but does not receive any corresponding RREP, S assumes that the destination (D) is a fixed node located on the Internet. But how many RREQs does S have to send, without receiving any corresponding RREP, before it can assume that D is located on the Internet? This issue is not discussed in "Global6".

As described in 3.3.1, S uses *expanding ring search* to find a route to D. To be absolutely sure that D is not a mobile node located within the MANET, S must do, at least, one *network-wide search*. Since a network-wide search consumes a lot of time and link bandwidth, it is not a good idea to do this search more than once. The idea can be summarized:

*A mobile node assumes that a destination node is a fixed node located on the Internet, if the mobile node has done one network-wide search without receiving any corresponding RREP for the destination node.*

In this study, the expanding ring search of AODV is used, without any modifications, as described in Section 3.3.1. It should be mentioned that, using the expanding ring search technique results in a considerable route discovery delay if the destination is a fixed node. Modifying the TTL_START, TTL_INCREMENT and TTL_THRESHOLD parameters can decrease the route discovery delay if the destination is a fixed node, but at the same time, the modification can result in increased routing overhead if the destination is a mobile node. The modification could for example be to increase TTL_START. Because, assuming the destination is a fixed node, increasing TTL_START would result in less number of broadcasted RREQs (and consequently less delay) before the source assumes that the destination is a fixed node. Thus, different approaches are preferable depending on whether a mobile node is to communicate mostly with the MANET or the Internet.

### 4.5.2 Gateway Operation upon Reception of RREQs

According to "Global6", when a gateway receives a RREQ, it looks in its routing table searching for the destination IP address specified in the RREQ message. If the address is not found in the routing table, the gateway has to send a RREP_I back to the originator of the RREQ. On the other hand, if the gateway finds the host route in its routing table, it should *not* unicast back a RREP_I to the originator of the RREQ "because the destination is then assumed to be inside the manet" ([14] Section 9.2). However, if the host route is found, not sending neither a RREP_I nor a RREP back to the originator of the RREQ, is not a good idea. The gateway must send a RREP and optionally also a RREP_I back to the originator of the RREQ. After pointing out the problem to the authors of the draft, they agreed on changing this. Hence, in the implementation used in this project:

*If a gateway receives a RREQ and finds the host route in its routing table, the gateway unicasts a RREP - and optionally also a RREP_I - back to the originator of the RREQ.*

In this way, a mobile node may obtain a default route although it has not requested this route. If the mobile node is to communicate with the Internet later, this default route

can be used and hence, the mobile node does not have to send another request message in order to find a route to a gateway.

Another issue that must be considered is how a gateway should react when it receives several RREQs for the *same* destination. As already said, a gateway should send a RREP_I if it receives a RREQ and it does not find the destination address in its routing table. But since expanding ring search is used, a gateway may receive several RREQs for the same destination address. The question is, should the gateway reply *every* RREQ with a RREP_I or only some of them?

The chief advantage of sending a RREP_I for every received RREQ is that the route to the gateway and the default route is updated. The chief disadvantage is that network resources are used. However, since the RREP_Is are unicast and not broadcasted to the requesting node, there will not be that much traffic generated. Hence, in the implementation used in this project:

*A gateway replies every received RREQ with a RREP_I.*

### 4.5.3 The Routing Table

Another issue that is worth discussing is how the routing table should change after a network-wide search without receiving any corresponding RREP. Assume that a source mobile node has done a network-wide search, without receiving any corresponding RREP. Hence, the source node assumes that the destination node is a fixed node located on the Internet.

According to "Global6" (Section 8.3), the source node sends its data packets using the default route. What the source node actually has to do is to create a route entry for the destination node in its routing table, see Figure 4.3. If the route entry for the fixed destination node would not be created in the routing table, the source node would not find the address to the fixed node in its routing table when the next data packet would be generated and hence, the source would have to do another time consuming network-wide search.

Although it is necessary for the source node to create a route entry for the fixed node in its routing table, there is a disadvantage. The disadvantage is that a mobile node will have to create a route entry for *every* fixed node that it wants to communicate with, in its routing table. One might think that a mobile node already has to create a new route entry for every mobile node in the mobile ad hoc network it communicates with, so there should not be anything strange about that. The problem is, however, that the number of fixed nodes is much greater than the number of mobile nodes. If a mobile node desires to communicate with many fixed nodes, its routing table will grow rapidly. However, this is not as alarming as it sounds. In AODV the routes that are not used will expire and eventually be deleted after a certain time, preventing the routing table to grow without control.

To summarize the idea:

*Although a default route is used, a mobile node has to create a new route entry in its routing table, not only for every mobile node, but also for every fixed node that it communicates with.*

| DESTINATION ADDRESS | NEXT HOP ADDRESS |
|---|---|
| FN (0.0.1) | DEFAULT (-10) |
| DEFAULT (-10) | GATEWAY (1.0.0) |
| GATEWAY (1.0.0) | MN_A (1.0.3) |

Figure 4.3: The routing table of a mobile node after creation of a route entry for a fixed node. The values in the parentheses are examples of IP addresses used in NS 2.

Figure 4.3 shows how the routing table of a mobile node (S) should look like after creation of a route entry for a fixed node. If S wants to communicate with the fixed node FN, S sends its data packets to MN_A. When MN_A receives the data packets it searches its routing table to see if it has a valid route to FN. If a valid route to FN is found, the data packets are sent to the next hop specified by the route entry. On the other hand, if a valid route is not found, the packets would normally be dropped because MN_A does not know to which node the packets should be forwarded. "Global6" does not mention how this case should be handled. In the implementation used in this project, if MN_A does not find a valid route to FN and if the destination is a fixed node located on the Internet, MN_A creates a (or updates the) route entry for FN in its routing table. Next, it forwards the data packets to a gateway which forwards them toward their destination. To summarize the idea:

*If an intermediate mobile node receives a data packet, it searches its routing table looking for a valid route to the destination. If a valid route to the destination is not found and the destination is a fixed node located on the Internet, the intermediate mobile node creates a new route entry (or updates the old invalid route entry) for the fixed node and forwards the data packet toward the gateway.*

### 4.5.4 Intermediate Node Operation upon Reception of RREQs

According to "Global6", when an intermediate mobile node receives a RREQ_I message, it *must not* send a RREP_I to the originator of the request message, even if the intermediate node has a route to a gateway. Instead, the intermediate mobile node rebroadcasts the received RREQ_I message. So far everything is correct, but the draft does not mention how an intermediate mobile node should react when it receives a RREQ message destined for a fixed node. The idea used in the implementation used in this study, is described below.

When an intermediate mobile node receives a RREQ message, it searches its routing table for a route to the destination. If the destination is a fixed node, the intermediate node *must not* send a RREP back to the originator of the request message even if the route is found. Because if the intermediate node sends a RREP back to the originator of the RREQ message, the originator thinks that the destination is a mobile node that can be reached via the intermediate node. It is important for the originator of the RREQ to know that the destination is a fixed node and not a mobile node, because sometimes

these are processed differently. Hence, in the implementation used in this study:

*If an intermediate mobile node receives a RREQ message destined for a fixed node, it must not send a RREP back to the originator of the RREQ even if the intermediate mobile node knows a route to the destination.*

### 4.5.5 Unreachable Gateway

An interesting issue to consider is what a mobile node should do if it cannot reach any gateway, although the destination is a fixed node. This issue is not discussed in "Global6".

Assume that a mobile node (MN) is sending data packets to a fixed node through a gateway (GW). Assume further that MN moves away from GW such that GW becomes unreachable for MN, i.e. MN cannot reach GW or any other gateway - not even through another intermediate mobile node. What shall MN do?

In the implementation used in this study, MN broadcasts a RREQ_I message to the ALL_MANET_GW_MULTICAST address (see Section 4.4), i.e. the IP address for the group of all gateways in the mobile ad hoc network. However, since GW is unreachable for MN, the RREQ_I message is not received by GW (or any other gateway). MN uses the expanding ring search technique when it broadcasts RREQ_I messages, but not even a RREQ_I message with the TTL value set to NET_DIAMETER is received by any gateway, because MN cannot reach any intermediate mobile node that can forward the RREQ_I message on its behalf.

After doing a network-wide search without receiving any corresponding RREP_I message from any gateway, MN pauses for a while. When the pause is finished, MN does another network-wide search and pauses again if no RREP_I is received. This procedure continues until MN moves close to a gateway or an intermediate mobile node so it can receive a RREP_I from a gateway. When a gateway is found, MN sends its data packets to the fixed node through the found gateway.

Letting the mobile node to broadcast RREQ_I messages until it finds a gateway might not be the best solution. An alternative solution would be to drop all buffered data packets destined for the destination and send an ICMP Destination Unreachable message to the application. There might exist better solutions than the two mentioned above, but due to lack of time this issue was not investigated further. To summarize the idea behind the implementation used in this study:

*If a mobile node cannot reach any gateways, it broadcasts RREQ_I messages until it finds one.*

# Chapter 5

# Gateway Discovery

The question of whether the configuration phase with the gateway should be initiated by the gateway (proactive method), by the mobile node (reactive method) or by mixing these two approaches (hybrid proactive/reactive method) has been discussed lately. In the following, the mechanisms of these three approaches are discussed. Proactive gateway discovery is discussed in Section 5.1, reactive gateway discovery is discussed in Section 5.2 and finally, hybrid gateway discovery is discussed in Section 5.3. The question of packet formats is also considered.

## 5.1 Proactive Gateway Discovery

The proactive gateway discovery is initiated by the gateway itself. The gateway periodically broadcasts a *gateway advertisement* (GWADV) message which is transmitted after expiration of the gateway's timer, ADVERTISEMENT_INTERVAL (see Table 7.2). The time between two consecutive advertisements must be chosen with care so that the network is not flooded unnecessarily. All mobile nodes residing in the gateway's transmission range receive the advertisement.

Upon receipt of the advertisement, the mobile nodes that do not have a route to the gateway create a route entry for it in their routing tables. Mobile nodes that already have a route to the gateway update their route entry for the gateway. Next, the advertisement is forwarded by the mobile nodes to other mobile nodes residing in their transmission range. To assure that all mobile nodes within the mobile ad hoc network receive the advertisement, the number of retransmissions is determined by NET_DIAMETER defined by AODV (see A.1.2). However, this will lead to enormously many unnecessary duplicated advertisements. A conceivable solution to the problem that occurs due to these duplicated advertisements, is presented in Section 5.1.1.

Although the problem of duplicated broadcast messages can be solved, one disadvantage remain. This disadvantage, which is general for all proactive approaches, is the fact that the message is flooded through the whole mobile ad hoc network periodically. This a very costly operation. Limited resources in a mobile ad hoc network, such as power and bandwidth, will be used a lot.

### 5.1.1 Duplicated Broadcast Messages

The problem of duplicated broadcast messages in mobile ad hoc networks is well known. In AODV, RREQ messages are broadcasted. To avoid duplicated RREQs, a RREQ ID is used (see Section 3.3.1). When a RREQ is received by a mobile node, it first checks to determine whether it already has received a RREQ with the same originator IP address and RREQ ID. If such a RREQ already has been received, the node discards the newly received RREQ.

In this thesis, the idea of comparing the RREQ ID with the originator IP address is used to solve the problem of duplicated advertisements. An advertisement is approximately a RREP_I message and since this message does not contain any field similar to the RREQ ID field in RREQ messages, a new AODV message has been introduced: the gateway advertisement (GWADV) message. This new AODV message is basically a RREP message extended with one field from the RREQ message, namely the RREQ ID field. Figure 5.1 illustrates the GWADV message format which can solve the problem of duplicated broadcast messages.

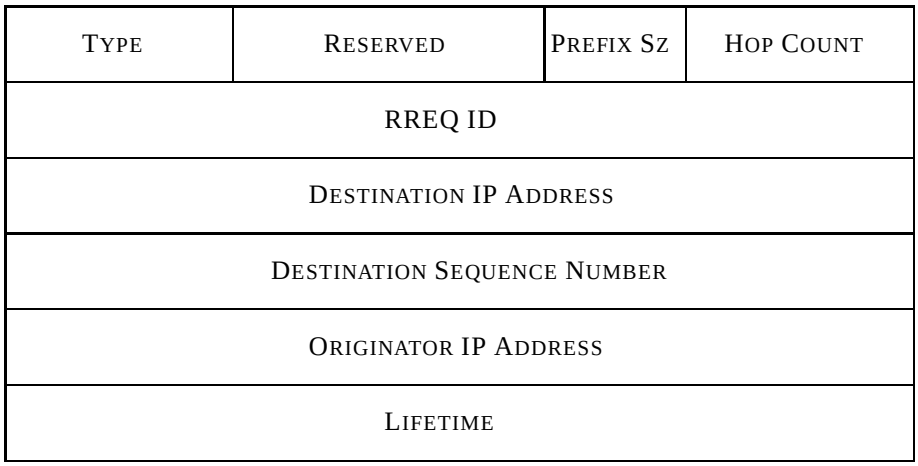


Figure 5.1: The format of a gateway advertisement (GWADV) message.

When a mobile node receives a GWADV, it first checks to determine whether a GWADV with the same originator IP address and RREQ ID already has been received during the last BCAST_ID_SAVE seconds (see Section A.1.2). If such a GWADV message has not been received, the message is rebroadcasted. Otherwise, if such a GWADV message has been received, the newly received GWADV is discarded. Hence, duplicated GWADVs are not forwarded and the advertisement is flooded through the whole network without causing too much congestion. However, the disadvantage with this solution is the fact that a new AODV message is introduced which requires AODV to be modified.

It is worth mentioning that the mobile nodes randomize their rebroadcasting of the GWADV in order to prevent synchronization and subsequent collisions with other nodes' rebroadcasts.

## 5.2 Reactive Gateway Discovery

The reactive gateway discovery is initiated by a mobile node that is to initialize or update information about the gateway. The mobile node broadcasts a RREQ_I (see Figure 4.2) to the ALL_MANET_GW_MULTICAST address (see A.1.2), i.e. the IP address for the group of all gateways in a mobile ad hoc network. Thus, only the gateways are addressed by this message and only they process it. Intermediate mobile nodes that receive the message just forward it by broadcasting it again. Since the message format is RREQ, which has a RREQ ID field as discussed in Section 5.1.1, duplicated RREQ_Is are discarded. Upon receipt of a RREQ_I, a gateway unicasts back a RREP_I which, among other things, contains the IP address of the gateway.

The advantage of this approach is that RREQ_Is are sent *only* when a mobile node needs the information about reachable gateways. Hence, periodic flooding of the complete mobile ad hoc network, which has obvious disadvantages as discussed in 5.1, is prevented. The disadvantage of reactive gateway discovery is that the load on forwarding mobile nodes, especially on those close to a gateway, is increased.

## 5.3 Hybrid Gateway Discovery

To minimize the disadvantages of proactive and reactive gateway discovery, the two approaches can be combined. This results in a hybrid proactive/reactive method for gateway discovery. For mobile nodes in a certain range around a gateway, proactive gateway discovery is used. Mobile nodes residing outside this range use reactive gateway discovery to obtain information about the gateway.

The gateway periodically broadcasts a RREP_I message (see Figure 4.3) which is transmitted after expiration of the gateway's timer, ADVERTISEMENT_INTERVAL (see Table 7.2). All mobile nodes residing in the gateway's transmission range receive the RREP_I. Upon receipt of the message, the mobile nodes that do not have a route to the gateway create a route entry for it in their routing tables. Mobile nodes that already have a route to the gateway update their route entry for the gateway. Next, the RREP_I is forwarded by the mobile nodes to other mobile nodes residing in their transmission range. The maximal number of hops a RREP_I can move through the mobile ad hoc network is ADVERTISEMENT_ZONE (see Table 7.2). This value defines the range within which proactive gateway discovery is used.

When a mobile node residing outside this range needs gateway information, it broadcasts a RREQ_I to the ALL_MANET_GW_MULTICAST address. Mobile nodes receiving the RREQ_I just rebroadcast it. Upon receipt of this RREQ_I, the gateway unicasts back a RREP_I.

# Chapter 6

# Network Simulator 2

As mentioned earlier in the text, Network Simulator 2 [16] is used as the simulation tool in this project. NS was chosen as the simulator partly because of the range of features it provides and partly because it has an open source code that can be modified and extended. In this project, the Internet draft "Global6" has been implemented in NS 2 so that AODV can be used to provide Internet access for mobile nodes. In addition, three methods for gateway discovery have been implemented and tested in simulations.

There are several different versions of NS and at the current time the latest version is ns-2.1b9a while ns-2.1b10 is under development. The latest version of ns has been used in this study.

This chapter describes the simulation environment in Section 6.1 and the application used for animation in Section 6.2.

## 6.1 Network Simulator (NS)

Network Simulator (NS) is an object-oriented, discrete event simulator for networking research. NS provides substantial support for simulation of TCP, routing and multicast protocols over wired and wireless networks [16]. The simulator is a result of an ongoing effort of research and development. Even though there is a considerable confidence in NS, it is not a polished and finished product yet and bugs are being discovered and corrected continuously.

NS is written in C++, with an OTcl[1] interpreter as a command and configuration interface. The C++ part, which is fast to run but slower to change, is used for detailed protocol implementation. The OTcl part, on the other hand, which runs much slower but can be changed very quickly, is used for simulation configuration. One of the advantages of this split-language programming approach is that it allows for fast generation of large scenarios. To simply use the simulator, it is sufficient to know OTcl. On the other hand, one disadvantage is that modifying and extending the simulator requires programming and debugging in both languages simultaneously.

[1] Object Tool Command Language

### 6.1.1 Marc Greis' Tutorial

The very first thing to do for a new user of NS, is to read Marc Greis' tutorial. There is a link to this tutorial on the web page of NS which can be found at www.isi.edu/nsnam/ns/. The purpose of this tutorial is to make it easier for new NS users to use NS and NAM[2], to create their own simulation scenarios for these tools and to eventually add new functionality to NS.

### 6.1.2 NS by Example

On the web page of NS, there is a link to another tutorial for NS. This tutorial has been written by Jae Chung and Mark Claypool and its purpose is to give new users some basic idea of how the simulator works, how to setup simulation networks, where to look for further information about network components in simulator codes, how to create new network components and so on. In particular, it explains the linkage between the two languages used in NS, namely C++ and OTcl. One can find some very good examples and brief explanations in this tutorial, which is the second tutorial to study after reading Marc Greis' tutorial.

### 6.1.3 NS Manual, NS Search and NS Mailing List

In the NS Manual [16] one can find the answer to many questions. A link to this Manual can be found on the web page of NS. However, if no answer can be found in the Manual, the NS mailing list archives should be searched. The archive keeps all previous emails sent to the ns-users mailing list. The ns-users mailing list should be used if an answer still (after looking in the Manual and searching the archives) has not been found. Everyone that has subscribed will receive this email and will hopefully reply.

## 6.2 Network Animator (NAM)

Network Animator (NAM) is an animation tool for viewing network simulation traces and real world packet traces [16]. It supports topology layout, packet level animation and various data inspection tools.

Before starting to use NAM, a trace file need to be created. This trace file is usually generated by NS. It contains topology information, e.g. nodes and links, as well as packet traces. During a simulation, the user can produce topology configurations, layout information and packet traces using tracing events in NS.

Once the trace file is generated, NAM can be used to animate it. Upon startup, NAM will read the tracefile, create topology, pop up a window, do layout if necessary and then pause at time 0. Through its user interface, NAM provides control over many aspects of animation. In Figure 6.2 a screenshot of a NAM window is shown, where the most important functions are explained.

[2]Network Animator, see Section 6.2

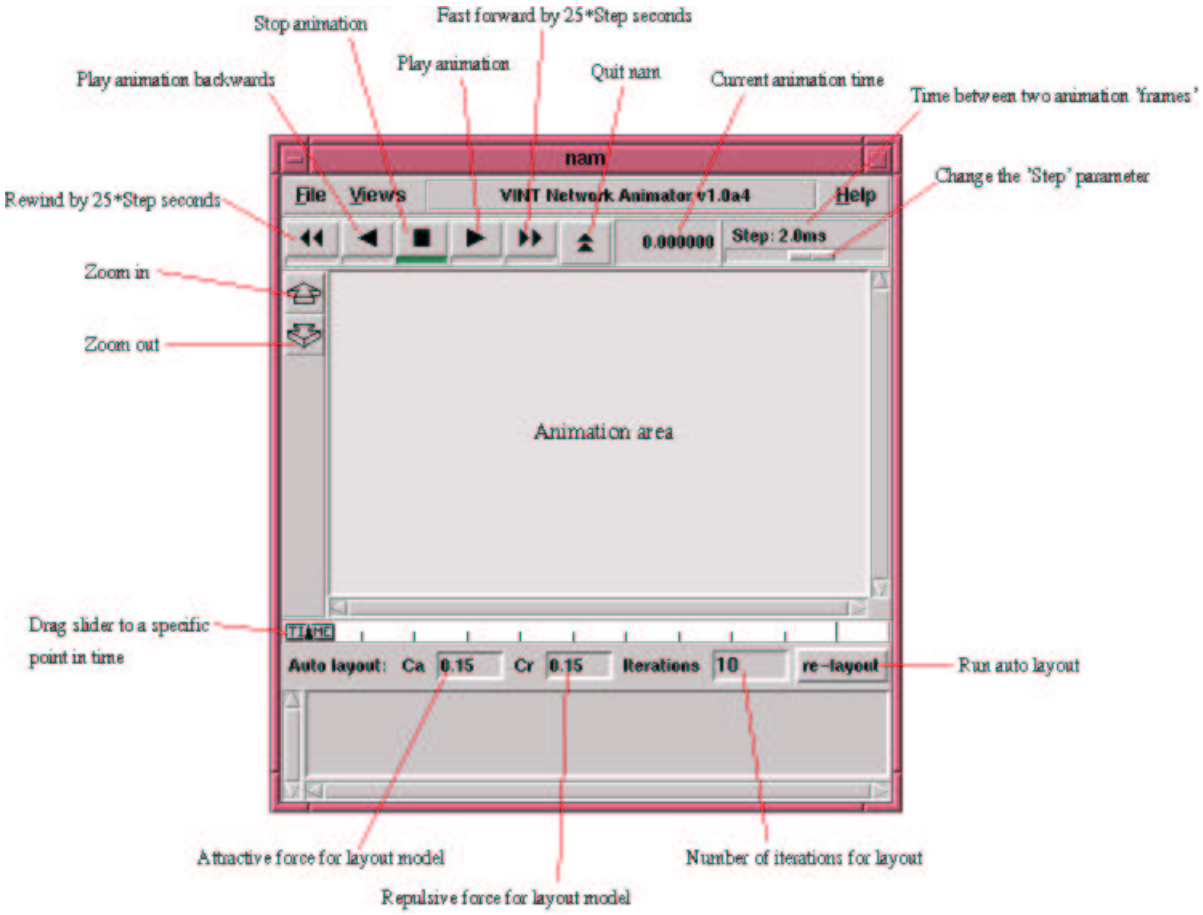


Figure 6.1: Screenshot of a NAM window explaining the most important functions.

Although the NAM software contain bugs, as do the NS software, it works fine most of the times and causes only little trouble. NAM is an excellent first step to check that the scenario works as expected. NS and NAM can also be used together for educational purpose and to easily demonstrate different networking issues.

# Chapter 7

# Simulation

To be able to evaluate the implementation of the Internet draft “Global Connectivity for IPv6 Mobile Ad Hoc networks” in NS 2, some simulation scenarios must be run. This chapter describes what have been simulated, how the simulations have been set up and finally it presents the results of the simulations.

The simulations were conducted on an Intel Pentium IV processor at 1.7 GHz, 256 MB of RAM running Linux Red Hat 7.2.

## 7.1 Simulation Setup

This section describes the scenario, the movement model and the communication model used in this study. Moreover, it presents the parameters used in the simulations.

### 7.1.1 Scenario

The studied scenario consists of 15 mobile nodes, 2 gateways, 2 routers and 2 hosts. The topology is a rectangular area with 800 m length and 500 m width. A rectangular area was chosen in order to force the use of longer routes between nodes than would occur in a square area with equal node density. The two gateways are placed on each side of the area; their x,y-coordinates in meters are (100,250) and (700,250). All simulations are run for 900 seconds of simulated time.

Five of the 15 mobile nodes are constant bit rate traffic sources. They are distributed randomly within the mobile ad hoc network. The time when the five traffic sources start sending data packets is chosen uniformly distributed within the first ten seconds of the simulation. After this time the sources continue sending data until one second before the end of the simulation. The destination of each of the sources is one of the two hosts, chosen randomly.

A screenshot of the simulation scenario is shown in Figure 7.1. The five mobile nodes that are marked with a ring, are the sources. The two hexagonal nodes are the gateways and the four square nodes are the two hosts and the two routers.

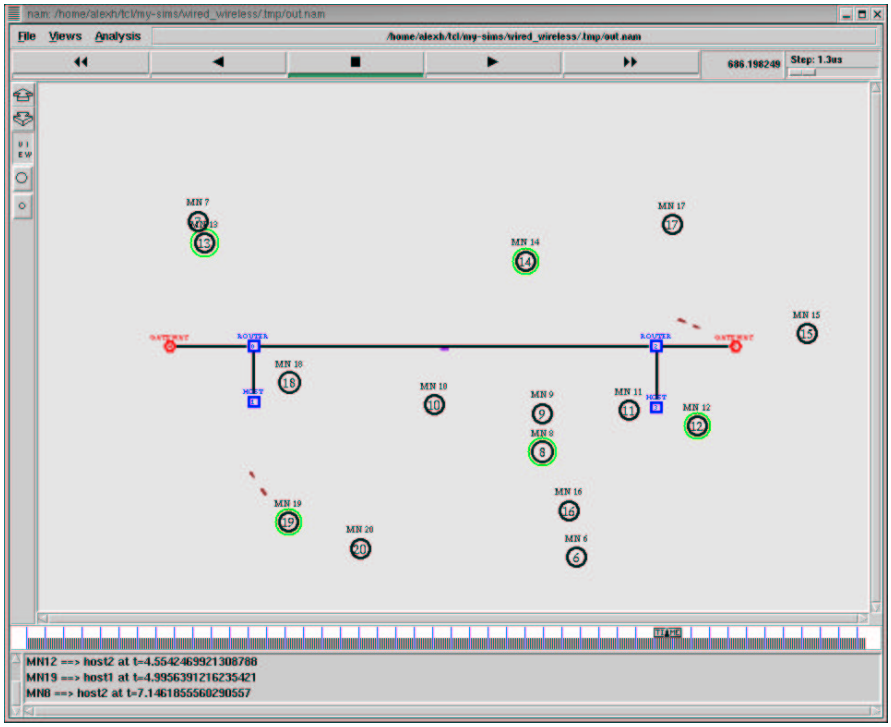


Figure 7.1: Screenshot of the simulation scenario.

### 7.1.2 Movement Model

The mobile nodes move according to the "random waypoint" model [8]. Each mobile node begins the simulation by remaining stationary for *pause time* seconds. It then selects a random destination in the defined topology area and moves to that destination at a random speed. The random speed is distributed uniformly between zero (zero not included) and some *maximum speed*. Upon reaching the destination, the mobile node pauses again for *pause time* seconds, selects another destination, and proceeds there as previously described. This movement pattern is repeated for the duration of the simulation.

The movement patterns are generated by CMU's[1] movement generator (setdest). The chosen values for pause time and maximum speed are shown in Table 7.1.

### 7.1.3 Communication Model

In the scenario used in this study, five mobile nodes communicate with one of two fixed nodes (hosts) located on the Internet through a gateway. As the goal of the simulations was to compare the different approaches for gateway discovery, the traffic source was chosen to be a constant bit rate (CBR) source. Each source mobile node generates packets every 0.2 seconds in this study. In other words, each source generates 5 packets

[1] Carnegie Mellon University

per second. Since each packet contain 512 bytes of data, the amount of generated data is 5*512*8 bit/s = 20 kbit/s, for each source.

The traffic connection pattern is generated by CMU's traffic generator (cbrgen.tcl). The main parameters in cbrgen.tcl are "connections" (number of sources) and "rate" (packet rate); see Table 7.1.

### 7.1.4 Parameters

The parameters that are common for all simulations are given in table 7.1 and the parameters that are specific for some simulations are shown in table 7.2.

| **Parameter** | **Value** |
|---|---|
| Transmission range | 250 m |
| Simulation time | 900 s |
| Topology size | 800 m x 500 m |
| Number of mobile nodes | 15 |
| Number of sources | 5 |
| Number of gateways | 2 |
| Traffic type | constant bit rate |
| Packet rate | 5 packets/s |
| Packet size | 512 bytes |
| Pause time | 5 s |
| Maximum speed | 10 m/s |

Table 7.1: General parameters used in all simulations.

The transmission range is the maximum possible distance between two communicating mobile nodes. If the distance between two mobile nodes is larger than 250 m they cannot communicate with each other directly.

| **Parameter** | **Value** |
|---|---|
| ADVERTISEMENT_INTERVAL | varied from 2-60 seconds |
| ADVERTISEMENT_ZONE | 3 hops |

Table 7.2: Specific parameters used in some simulations.

ADVERTISEMENT_INTERVAL is used when proactive and hybrid discovery methods are used (see Sections 5.1 and 5.3). ADVERTISEMENT_ZONE is used for hybrid gateway discovery method and defines the range within which proactive gateway discovery is used.

## 7.2 Performance Metrics

The second goal of this project was to "implement and compare different approaches for gateway discovery". The implementation details are presented in appendix B, Sections B.1, B.2 and B.3. Comparing the different methods is done by simulating them

and examining their behavior. In the simulations in the following section, the effect of different gateway advertisement intervals are evaluated.

In comparing the gateway discovery approaches, the evaluation has been done according to the following three metrics:

- The packet delivery ratio is defined as the number of received data packets divided by the number of generated data packets.
- The end-to-end delay is defined as the time a data packet is received by the destination minus the time the data packet is generated by the source.
- The overhead is defined as the total number of AODV messages transmitted during the simulation. For AODV messages sent over multiple hops, *each* transmission of the message (each hop) counts as one transmission.

## 7.3 Simulation Results

In this section the effect of varying gateway advertisement intervals is evaluated. Since gateway advertisements are not sent in the reactive gateway discovery approach, the results for this approach are constant and independent of the advertisement interval. Each data point is an average value of 10 runs with the same communication model, but different randomly generated movement patterns.

### 7.3.1 Packet Delivery Ratio

Figure 7.2 shows the packet delivery ratio with advertisement intervals between 2 and 60 seconds. As the figure shows, the packet delivery ratio is very high (above 99.8 %) for all three gateway discovery approaches. The figure also shows that the difference between the three approaches are very small. However, the proactive and hybrid approaches have some larger packet delivery ratio than the reactive approach, especially with short advertisement intervals. The reason is that the short advertisement intervals reslut in more gateway information (RREP_I and GWADV packets).

As described in Sections 5.1 and 5.3 a mobile node that receive a RREP_I or a GWADV message, update its route entry for the gateway. Therefore, it is more likely for the mobile nodes to have fresher and shorter routes to a gateway and thereby minimizing the risk for link breaks. Link breaks can result in lost data packets since the source continues to send data packets until it receives a RERR message from the mobile node that has a broken link. The longer the route is (in number of hops), the longer time it can take before the source receive a RERR and hence, more data packets can be lost.

When the advertisement interval increases, a mobile node receives less gateway information and consequently it does not update the route to the gateway as often as for short advertisement intervals. Therefore, the positive effect of periodic gateway information is decreased as the advertisement interval increases.

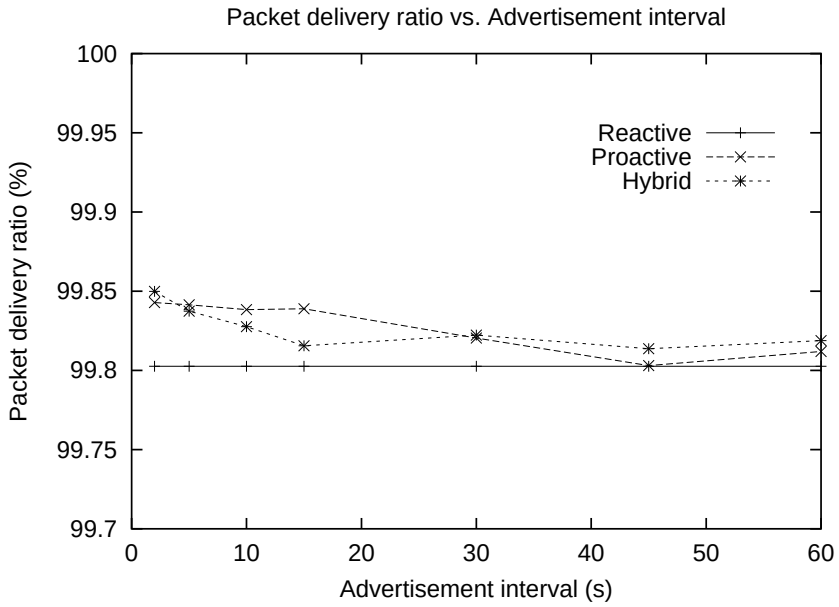


Figure 7.2: Packet delivery ratio

### 7.3.2 Average End-to-end Delay

Figure 7.3 shows the average end-to-end delay with advertisement intervals between 2 and 60 seconds. As the figure shows, the average end-to-end delay is less for the proactive and hybrid approaches than for the reactive approach. The reason is that the periodic gateway information sent by the gateways allow the mobile nodes to update their route entries for the gateways more often, resulting in fresher and shorter routes. With the reactive approach a mobile node continues to use a route to a gateway until it is broken. In some cases this route can be pretty long (in number of hops) and even if the mobile node is much closer to another gateway it does not use this gateway, but continues to send the data packets along the long route to the gateway further away until the route is broken. Therefore, the end-to-end delay increases for these data packets, resulting in increased average end-to-end delay for all data packets.

The figure also shows that the average end-to-end delay is decreased slightly for short advertisement intervals when the advertisement interval is increased. At the first thought this might seem unexpected. However, it can be explained by the fact that very short advertisement intervals result in a lot of control traffic which lead to higher processing times for data packets at each node. Moreover, since the AODV messages are prioritized over data packets, these have to wait in the routing queue until the AODV messages are sent, resulting in higher end-to-end delay.

### 7.3.3 AODV Overhead

Figure 7.4 shows the AODV overhead with advertisement intervals between 2 and 60 seconds. The AODV overhead is dominated by the periodically broadcasted RREP_I

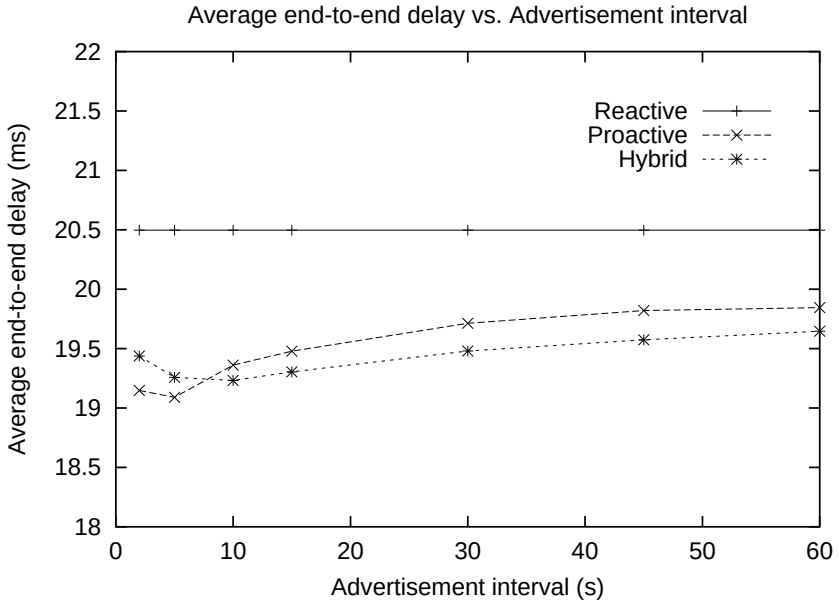


Figure 7.3: Average end-to-end delay

and GWADV messages. As the figure shows, the AODV overhead is larger for the proactive and hybrid approaches than for the reactive approach, especially for short advertisement intervals. This is an expected result since the proactive and hybrid approaches periodically broadcast gateway information no matter if the mobile nodes need them or not, while the reactive approach broadcasts gateway information only when a mobile node sends a request for it. Moreover, the figure shows that the AODV overhead decreases for the proactive and hybrid approaches as the advertisement interval increases. This is due to less frequent gateway information transmissions.

Finally it can be noticed that the overhead for the hybrid approach is much greater than for the proactive approach when the advertisement interval is short. This is due to duplicated messages as described in Section 5.1.1. In the hybrid method, gateways broadcast RREP_I messages which are forwarded by mobile nodes until the TTL (time to live) value for the messages are decreased to zero. Hence, there are some amount of duplicated RREP_I messages, i.e. a mobile node can receive the same RREP_I several times. On the other hand, in the proactive method, gateways broadcast GWADV messages which are forwarded by mobile nodes *only* if they have not forwarded the messages before. Hence, there are no duplicated broadcast messages generated when the proactive approach is used.

In the simulations where hybrid gateway discovery has been used, the TTL value has been set to ADVERTISEMENT_ZONE which is defined as 3 in this study. This implies that a RREP_I message is received by all mobile nodes within a range of 3 hops from the gateway. The discussion above and Figure 7.4 illustrates why RREP_I messages cannot be used for a proactive gateway discovery method unless it is modified. Because in the proactive approach the TTL value would have to be set to NETWORK_DIAMETER, which equals 30 hops in the AODV implementation in NS. The

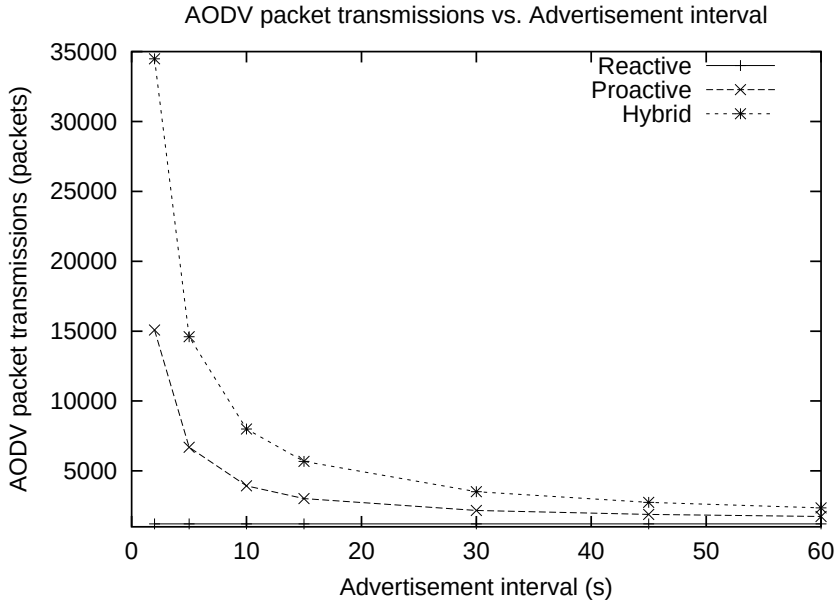


Figure 7.4: AODV overhead

figure shows namely that a lot of duplicated RREP_I messages are generated when TTL is set to 3; then one can imagine how much overhead a TTL value of 30 would have generated.

# Chapter 8

# Conclusions

The thesis has considered the Internet access of mobile nodes in a mobile ad hoc network. The ad hoc routing protocol AODV has been extended to route packets, not only within a MANET but also between a wireless MANET and the wired Internet. To be able to achieve this, some nodes must act as a mixture of a mobile node and a fixed node. The communication between the wireless and the wired network must pass through these nodes, which are referred to as gateways. In this project, three methods for detection of these gateways have been presented, implemented and compared. The three methods for gateway detection are referred to as reactive, proactive and hybrid gateway discovery. The comparison between these methods provides us useful information.

Regarding the packet delivery ratio, the result is largely the same, regardless of which gateway discovery method is used. As for the average end-to-end delay, the proactive and hybrid methods perform slightly better than the reactive method. Concerning the routing overhead, when the advertisement interval is short the reactive method generates much less overhead than the proactive method, which in turn generates much less overhead than the hybrid method. When the advertisement interval increases, all three methods generate virtually the same routing overhead.

The results presented are valid for the specific scenario used in this project. Therefore, one cannot tell which of the gateway discovery methods is the best one for every possible scenario. There are many factors that can be changed and their impact should be investigated. Unfortunately the scope of this project made it impossible to deal with more than a part of these interesting issues. The aim in future work will be to examine them in greater detail. For example, changing the number of mobile nodes and the size of the topology changes the mobile node density. Its impact should be investigated. Another issue that should be examined is the impact of the number of gateways and the distance between them. Certain other questions of interest are the number of traffic sources, the number of packets sent per second, the size of the data packets, and the speed of the mobile nodes.

# Appendix A

# Implementation of "Global Connectivity for IPv6 Mobile Ad Hoc networks" in NS 2

The following sections describe some details about the implementation of the Internet draft "Global Connectivity for IPv6 Mobile Ad Hoc Networks" in NS 2. The main modifications has been done in the ad hoc routing protocol AODV. The most important modifications and extensions are presented in this appendix.

## A.1 Modifying the AODV Implementation in NS2

The AODV related files in the NS distribution are aodv.{h,cc}, aodv_packet.h, aodv_rqueue.{h,cc}, aodv_rtable.{h,cc}, aodv_logs.{h,cc} and aodv.tcl. The main code is implemented in aodv.cc and the functions are declared in aodv.h. In aodv_packet.h, the AODV message formats (RREQ, RREP, RERR and HELLO) are defined. Moreover the new message format GWADV (gateway advertisement) has been added and defined in this file.

The main modifications has been done in aodv.cc. The modifications have been done mostly in accordance with "Global6". Here, the most important modifications are explained.

The functions are explained in a logical order: when a mobile node is to send a data packet to a destination, it tries to find a route to the destination (rt_resolve). If the mobile node does not have any valid route to the destination it broadcasts a RREQ message (sendRequest). The RREQ message is eventually received by the destination or another node which knows a route to the destination (recvRequest). The node sends a RREP/RREP_I message back to the originator of the RREQ (sendReply). The originator of the RREQ receives the RREP/RREP_I message (recvReply) and starts sending data packets to the destination (find_send_entry if the destination is a fixed node).

### A.1.1 Modifications in aodv.cc

- **void AODV::rt_resolve(Packet *p)**
  This function is invoked in two situations. First, when a mobile node is to send a data packet and second, when an intermediate mobile node receives a data packet which it must forward toward its destination.

  If the function is invoked by a (source) mobile node that wants to send data packets to some destination node, there is a check to determine if the destination is a fixed node and the route to the fixed node is invalid. If this is the case, the (source) mobile node broadcasts a RREQ_I message to discover a gateway. Otherwise, the mobile node acts as described in the AODV draft without any modifications.

  If the function is invoked by an intermediate node which has received a data packet which must be forwarded, the packet is processed differently depending on if the intermediate node is a mobile node or a gateway. If the intermediate node is a mobile node and it has a default route, the data packet is destined for a fixed node. Therefore, the intermediate mobile node updates its route entry for the fixed node and forwards the packet toward the gateway. On the other hand, if the intermediate node is a gateway, it has received a data packet from a fixed node destined for a mobile node. Consequently, the gateway broadcasts a RREQ message to discover a route to the destination.

- **void AODV::sendRequest(nsaddr_t dst, u_int8_t flag)**
  This function is invoked when a mobile node needs to find a route to a destination node by broadcasting a RREQ. The RREQ is broadcasted according to the expanding ring search algorithm described in Section 3.3.1. However, when a RREQ has been broadcasted through the whole network, i.e. when a mobile node has done a network-wide search without receiving any corresponding RREP, it assumes that the destination node is a fixed node located on the Internet. First, the mobile node updates its route entry for the fixed node. Then, it checks for buffered packets destined for the fixed node. In case there are such packets, they are forwarded toward the gateway.

  This function is also invoked when a mobile node needs to find a route to a gateway by broadcasting a RREQ_I. The RREQ_I is broadcasted in the same way as a RREQ message. A mobile node needs to find a route to a gateway when it detects a link break and the destination of the route is a fixed node.

- **void AODV::recvRequest(Packet *p)**
  This function is invoked when a mobile node receives a RREQ or a RREQ_I message. The message is processed differently depending on if the node is a mobile node or a gateway. If the node is a mobile node, the code runs without any modifications, i.e. the node tries to unicast back a RREP to the originator of the RREQ message. However, in case the node is a gateway, a RREP_I is unicast back to the originator of the RREQ message.

- **void AODV::sendReply(nsaddr_t ipdst, u_int32_t hop_count, nsaddr_t rpdst, u_int32_t rpseq, u_int32_t lifetime, double timestamp, u_int8_t flag)**

This function is invoked by a node that has received a RREQ or a RREQ_I message and either it is the destination or has a fresh route to the destination. The function just unicasts back a RREP or RREP_I message (depending on the value in the "flag" field) to the originator of the RREQ or RREQ_I. See comments about function recvRequest.

- **void AODV::recvReply(Packet *p)**
  This function is invoked when a mobile node receives a RREP or a RREP_I message. The message is processed differently depending on if it is a RREP or a RREP_I. If the message is a RREP, the code runs without any modifications. However, if the message is a RREP_I the mobile node saves the address of the gateway and creates a (or updates the) default route with the address of the gateway as the next hop. If the mobile node already has a route to another gateway than the originator of the newly received RREP_I message, it performs gateway selection with the number of hops to the gateways as metric.

  Then the mobile node checks if it has any packets queued in its buffer destined for a fixed node. If such packets exist and there exists a valid default route, all packets queued in the buffer are forwarded toward the gateway.

  If hybrid gateway discovery method is used, RREP_Is are broadcasted by the gateway periodically. When a mobile node receives any of these RREP_Is it updates its route entry for the gateway that originated the RREP_I and rebroadcasts it so other mobile nodes can receive the gateway information. If a gateway receives a RREP_I, it discards the message and does not rebroadcast it, to reduce unnecessary network load.

- **rt_entry* AODV::find_send_entry(rt_entry *rt)**
  A function that searches the routing table and returns the correct route that a packet should be sent to, i.e. it finds the correct next hop that the packets should be forwarded to. This is needed since default routes, which are not valid next hops, have been introduced. The routing table is searched for a correct next hop only if the destination is a fixed node, otherwise (if the destination is a mobile node) the function just return the next hop as indicated in the routing table.

  As an example, take a look at Figure 4.3 in Section 4.5.3. If this function is not invoked, the packets destined for FN will be sent to next hop "DEFAULT" which is defined as -10 in aodv.h. Since there is no node with this address, the packets will be dropped. Therefore, this function is invoked to find the correct next hop, MN_A in this example. Hence, the packets are forwarded to MN_A which forwards them to the gateway which forwards them toward FN.

### A.1.2 Modifications in aodv.h

In this file default values for some important parameters associated with AODV protocol operations are defined. These values does not always match the default values given in the AODV Internet draft. Some of the interesting values are given below:

| Parameter | Value |
|---|---|
| MY_ROUTE_TIMEOUT | 10 seconds |
| ACTIVE_ROUTE_TIMEOUT | 10 seconds |
| REV_ROUTE_LIFE | 6 seconds |
| BCAST_ID_SAVE | 6 seconds |
| NETWORK_DIAMETER | 30 hops |
| NODE_TRAVERSAL_TIME | 0.03 seconds |
| RREQ_RETRIES | 2 |
| MAX_RREQ_TIMEOUT | 10 seconds |
| TTL_START | 1 |
| TTL_INCREMENT | 2 |
| TTL_THRESHOLD | 7 |

Table A.1: Default values for some important parameters associated with AODV protocol operations.

Upon receipt of a RREP a mobile node sets the lifetime of the route to MY_ROUTE_TIMEOUT.

ACTIVE_ROUTE_TIMEOUT is the lifetime of an active route.

REV_ROUTE_LIFE is the lifetime of a reverse route created when an intermediate mobile node receives a RREQ originated by another mobile node. This parameter is replaced by PATH_DISCOVERY_TIME in the AODV draft.

BCAST_ID_SAVE defines how long time the RREQ ID should be saved. After BCAST_ID_SAVE seconds the RREQ ID is deleted. The RREQ ID is stored in order to prevent duplicated RREQs and GWADVs being forwarded. In the Sections 3.3.1 and 5.1.1 it is described in more detail how this is done. This parameter is replaced by PATH_DISCOVERY_TIME in the AODV draft.

When a mobile node is to do a network-wide search, it sets TTL in the IP header of the RREQ message to NETWORK_DIAMETER. This parameter is called NET_DIAMETER in the AODV draft.

NODE_TRAVERSAL_TIME is the time it takes for a node to process a packet.

RREQ_RETRIES defines the number of times to redo a network-wide search before timing out for MAX_RREQ_TIMEOUT sec.

A mobile node has to wait MAX_RREQ_TIMEOUT seconds after doing network-wide search RREQ_RETRIES times. This parameter is not defined in the AODV draft.

TTL_START, TTL_INCREMENT and TTL_THRESHOLD are parameters used for the expanding ring search described in Section 3.3.1.

DEFAULT is the address of the default route and ALL_MANET_GW_MULTICAST is the multicast address of all the gateways in the MANET. The values are just arbitrary values and they are negative so they cannot be mixed with the address of a mobile node.

GWINFO_LIFETIME is the lifetime of a RREP_I sent by a gateway. This value is defined in the Internet draft “Global6”.

| Parameter | Value |
|---|---|
| DEFAULT | -10 |
| ALL_MANET_GW_MULTICAST | -20 |
| GWINFO_LIFETIME | 10 seconds |
| ADVERTISEMENT_INTERVAL | varied from 2-60 seconds |
| ADVERTISEMENT_ZONE | 3 hops |

Table A.2: Some important parameters associated with the gateway operation.

ADVERTISEMENT_INTERVAL is the interval between two consecutive gateway information messages.

ADVERTISEMENT_ZONE is the zone within which mobile nodes receive the gateway information message. Hence, this value limits the gateway information message propagation.

### A.1.3 Modifications in aodv_packet.h

- **struct hdr_aodv_request**
  A field for flags has been added to this RREQ message. In that field the I-flag, mentioned in Section 4.2, has been defined.

- **struct hdr_aodv_reply**
  A field for flags has been added to this RREP message. In that field the I-flag, mentioned in Section 4.3, has been defined.

- **struct hdr_aodv_advertisement**
  This new AODV message, which has been named gateway advertisement (GWADV), is basically a RREP message extended with one field from the RREQ message, namely the RREQ ID field. Figure 5.1 illustrates the GWADV message format. This message is periodically broadcasted by the gateway to advertise its address to the mobile nodes in the mobile ad hoc network. GWADV messages have been introduced to solve the problem of duplicated broadcast messages discussed in Section 5.1.1.

# Appendix B

# Implementation of Three Gateway Discovery Methods in NS 2

The following sections present the implementation of the three discovery methods examined in this project. The main part of the implementation has been done in aodv.cc.

## B.1 Implementation of Proactive Gateway Discovery Method

```
void AODV::sendAdvertisement() {
  /*
    Only gateways broadcast GWADV messages
  */
  if(index != thisnode->base_stn()) {
    //I'm not gateway; return
    return;
  }

  //Allocate a GWADV message
  Packet *p = Packet::alloc();
  struct hdr_cmn *ch = HDR_CMN(p);
  struct hdr_ip *ih = HDR_IP(p);
  struct hdr_aodv_advertisement *ad = HDR_AODV_ADVERTISEMENT(p);

  ad->ad_type = AODVTYPE_ADVERTISEMENT;
  ad->ad_hop_count = 1;
  seqno++;
  if(seqno%2) seqno++;
  ad->ad_dst_seqno = seqno;
  ad->ad_src = index;
```

```
  ad->ad_lifetime = (1 + ALLOWED_HELLO_LOSS) * (u_int32_t)
    ADVERTISEMENT_INTERVAL;
  ad->ad_bcast_id = ad_bid++;

  ch->ptype() = PT_AODV;
  ch->size() = IP_HDR_LEN + ad->size();
  ch->iface() = -2;
  ch->error() = 0;
  ch->addr_type() = NS_AF_NONE;
  ch->prev_hop_ = index;

  ih->saddr() = index;
  ih->daddr() = IP_BROADCAST;
  ih->sport() = RT_PORT;
  ih->dport() = RT_PORT;
  //The GWADV is flooded through the whole MANET
  ih->ttl_ = NETWORK_DIAMETER;

  Scheduler::instance().schedule(target_, p, 0.0);
}
```

## B.2 Implementation of Reactive Gateway Discovery Method

```
void AODV::sendRequest(nsaddr_t dst, u_int8_t flag) {

  // Allocate a RREQ message
  Packet *p = Packet::alloc();
  struct hdr_cmn *ch = HDR_CMN(p);
  struct hdr_ip *ih = HDR_IP(p);
  struct hdr_aodv_request *rq = HDR_AODV_REQUEST(p);
  aodv_rt_entry *rt = rtable.rt_lookup(dst);
  assert(rt);

  /*
    Return if
    1. route is up
    2. RREQ_I has already been sent
    3. network-wide search has been done 3 times

    rt_req_cnt is the number of times we did network-wide search.
    RREQ_RETRIES is the maximum number we will allow broadcasting RREQs befor
    going to a long timeout.
  */
  if(rt->rt_flags == RTF_UP) {
    assert(rt->rt_hops != INFINITY2);
    Packet::free((Packet *)p);
    return;
  }
```

```
if(rt->rt_req_timeout > CURRENT_TIME) {
  Packet::free((Packet *)p);
  return;
}

if((rt->rt_req_cnt > RREQ_RETRIES)) {
  rt->rt_req_timeout = CURRENT_TIME + MAX_RREQ_TIMEOUT;
  rt->rt_req_cnt = 0;
  Packet *buf_pkt;
  while ((buf_pkt = rqueue.deque(rt->rt_dst))) {
    drop(buf_pkt, DROP_RTR_NO_ROUTE);
  }
  Packet::free((Packet *)p);
  return;
}

//...OMITTED CODE NOT RELEVANT FOR REACTIVE GATEWAY DISCOVERY...//

// Determine the TTL to be used this time.
if(rt->rt_last_hop_count < INFINITY2) {
  rt->rt_req_last_ttl = max(rt->rt_req_last_ttl, rt->rt_last_hop_count);
}

if (0 == rt->rt_req_last_ttl) {
  // First time query broadcast
  ih->ttl_ = TTL_START;
}
else {
  // Expanding ring search
  if (rt->rt_req_last_ttl < TTL_THRESHOLD)
    ih->ttl_ = rt->rt_req_last_ttl + TTL_INCREMENT;
  else {
    // network-wide broadcast
    ih->ttl_ = NETWORK_DIAMETER;
    rt->rt_req_cnt += 1;
  }
}

// remember the TTL used  for the next time
rt->rt_req_last_ttl = ih->ttl_;

// PerHopTime is the roundtrip time per hop for route requests.
// Also note that we are making timeouts to be larger if we have done
// network wide broadcast before.
rt->rt_req_timeout = 2.0 * (double) ih->ttl_ * PerHopTime(rt);
if (rt->rt_req_cnt > 0)
  rt->rt_req_timeout *= rt->rt_req_cnt;
rt->rt_req_timeout += CURRENT_TIME;
```

```
  // Don't let the timeout to be too large, however .. SRD 6/8/99
  if (rt->rt_req_timeout > CURRENT_TIME + MAX_RREQ_TIMEOUT)
    rt->rt_req_timeout = CURRENT_TIME + MAX_RREQ_TIMEOUT;
  rt->rt_expire = 0;

  // Fill out the RREQ message
  ch->ptype() = PT_AODV;
  ch->size() = IP_HDR_LEN + rq->size();
  ch->iface() = -2;
  ch->error() = 0;
  ch->addr_type() = NS_AF_NONE;
  ch->prev_hop_ = index;

  ih->saddr() = index;
  ih->daddr() = IP_BROADCAST;
  ih->sport() = RT_PORT;
  ih->dport() = RT_PORT;

  // Fill up some more fields
  rq->rq_type = AODVTYPE_RREQ;
  rq->rq_hop_count = 1;
  rq->rq_bcast_id = bid++;
  rq->rq_dst = dst;
  rq->rq_dst_seqno = (rt ? rt->rt_seqno : 0);
  rq->rq_src = index;
  seqno += 2;
  assert ((seqno%2) == 0);
  rq->rq_src_seqno = seqno;
  rq->rq_timestamp = CURRENT_TIME;
  //The I-flag is set for RREQ_I messages
  rq->rq_flags = flag;
  Scheduler::instance().schedule(target_, p, 0.);
}
```

## B.3 Implementation of Hybrid Gateway Discovery Method

```
void AODV::sendReply_I() {
  /*
    Only gateways broadcast RREP_I messages
  */
  if(index != thisnode->base_stn()) {
    //I'm not gateway; return
    return;
  }

  //Allocate a RREP_I message
  Packet *p = Packet::alloc();
  struct hdr_cmn *ch = HDR_CMN(p);
```

```
  struct hdr_ip *ih = HDR_IP(p);
  struct hdr_aodv_reply *rp = HDR_AODV_REPLY(p);

  rp->rp_type = AODVTYPE_RREP;
  //The I-flag is set for RREP_I messages
  rp->rp_flags = RREP_IFLAG;
  rp->rp_hop_count = 1;
  rp->rp_dst = index;
  seqno++;
  if(seqno%2) seqno++;
  rp->rp_dst_seqno = seqno;
  rp->rp_lifetime = (1 + ALLOWED_HELLO_LOSS) *  (u_int32_t)
    ADVERTISEMENT_INTERVAL;

  ch->ptype() = PT_AODV;
  ch->size() = IP_HDR_LEN + rp->size();
  ch->iface() = -2;
  ch->error() = 0;
  ch->addr_type() = NS_AF_NONE;
  ch->prev_hop_ = index;

  ih->saddr() = index;
  ih->daddr() = IP_BROADCAST;
  ih->sport() = RT_PORT;
  ih->dport() = RT_PORT;
  //TTL is limited in order to avoid too much advertisement duplication
  ih->ttl_ = ADVERTISEMENT_ZONE;

  Scheduler::instance().schedule(target_, p, 0.0);
}
```

# Bibliography


[1] Belding-Royer E.M.; Sun Y.; Perkins C. *Global Connectivity for IPv4 Mobile Ad Hoc Networks*, IETF Internet Draft, Nov. 2001. Work in progress.

[2] Bernard M. *Gateway Detection and Selection for Wireless Multihop Internet Access*, Master's thesis, (Olching, Germany), 2002.

[3] Clausen T.; Jacquet P.; Laouiti A.; Minet P.; Muhlethaler P.; Qayyum A.; Viennot L. *Optimized Link State Routing Protocol*, IETF Internet Draft, July 2002. Work in progress.

[4] Comer D.E. *Internetworking With TCP/IP Volume I: Principles, Protocols, and Architectures*, fourth edition, 2000.

[5] Haas Z.J.; Pearlman M.R.; Samar P. *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*, IETF Internet Draft, July 2002. Work in progress.

[6] Holland G.; Vaidya N. *Analysis of TCP Performance over Mobile Ad Hoc Networks*, in Proceedings of IEEE/ACM MOBICOM, 1999.

[7] Hong X.; Xu K.; Gerla M. *Scalable Routing Protocols for Mobile Ad Hoc Networks*, IEEE Network, July/August 2002.

[8] Johnson D.B.; Maltz D.A. *Dynamic Source Routing in Ad Hoc Wireless Networks*, in: Mobile Computing (Imielinski T. and Korth H.), chapter 5, pages 153-181. Kluwer Academic Publishers, 1996.

[9] Johnson D.B.; Maltz D.A.; Hu Y.; Jetcheva J.G. *The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)*, IETF Internet Draft, Feb 2002. Work in progress.

[10] Jönsson U.; Alriksson F.; Larsson T.; Johansson P.; Maguire G.Q. *MIPMANET - Mobile IP for Mobile Ad Hoc Networks*, in Proceedings of the Workshop on Mobile Ad Hoc Networking and Computing (MobiHoc), Boston, USA, 2000.

[11] Perkins C. *IP Mobility for IPv4, Revised*. draft-ietf-mobileip-rfc2002-bis-08.txt, September 2001.

[12] Perkins C.; Belding-Royer E.M.; Das S. *Ad hoc On-Demand Distance Vector (AODV) Routing*, IETF Internet Draft, Jan 2002. Work in progress.

[13] Schiller J. *Mobile Communications*, 2000.

[14] Wakikawa R.; Malinen J.; Perkins C.; Nilsson A.; Tuominen A.J. *Global Connectivity for IPv6 Mobile Ad Hoc Networks*, IETF Internet Draft, November 2001. Work in progress.

[15] Xi J.; Bettstetter C. *Wireless Multihop Internet Access: Gateway Discovery, Routing and Addressing*, in Proceedings of the International Conference on Third Generation Wireless and Beyond (3Gwireless'02), San Francisco, USA, May 2002.

[16] McCanne S.; Floyd S. ns Network Simulator. http://www.isi.edu/nsnam/ns/. Fall K; Varadhan K., and the VINT project. The ns manual.

[17] The MANET web page, http://www.ietf.org/html.charters/manet-charter.html.

# Acronyms

| | |
|---|---|
| AODV | Ad hoc On-demand Distance Vector |
| CBR | Constant Bit Rate |
| DSR | Dynamic Source Routing |
| ETSI | European Telecommunications Standards Institute |
| GSM | Global System for Mobile communications |
| GWADV | GateWay ADVertisement |
| HIPERLAN 2 | HIgh-PERformance Local Area Network type 2 |
| ICMP | Internet Control Message Protocol |
| IEEE | Institute of Electrical and Electronics Engineers |
| IETF | Internet Engineering Task Force |
| IP | Internet Protocol |
| IPv6 | Internet Protocol version 6 |
| LAN | Local Area Network |
| LLC | Logical Link Control |
| MAC | Media Access Control |
| MANET | Mobile Ad hoc NETwork |
| MPR | MultiPoint Relay |
| NAM | Network AniMator |
| NS | Network Simulator |
| OLSR | Optimized Link State Routing protocol |
| OSI | Open Systems Interconnection |
| OTcl | Object Tool command language |
| PDA | Personal Digital Assistant |
| RREP | Route REPly |

| | |
|---|---|
| RREQ | Route REQuest |
| TCP | Transmission Control Protocol |
| TTL | Time To Live |
| UDP | User Datagram Protocol |
| ZRP | Zone Routing Protocol |