LULEÅ TEKNISKA UNIVERSITET

1998:362

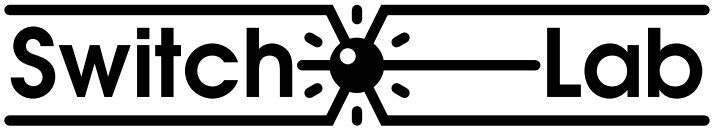
MASTER'S THESIS

Routing Protocols in Wireless Ad-hoc Networks - A Simulation Study

Tony Larsson, Nicklas Hedman

Civilingenjörsprogrammet

1998:362 • ISSN: 1402-1617 • ISRN: LTU-EX--98/362--SE



Master's thesis in Computer Science and Engineering

Routing Protocols in Wireless Ad-hoc Networks - A Simulation Study

Stockholm, 1998

Tony Larsson and Nicklas Hedman

Luleå University of Technology

Supervisor: Per Johansson
Switchlab
Ericsson Telecom AB

Examiner: Mikael Degermark
Department of Computer Science and Electrical Engineering
Division of Computer Communications,
Luleå University of Technology

Abstract

Ad-hoc networking is a concept in computer communications, which means that users wanting to communicate with each other form a temporary network, without any form of centralized administration. Each node participating in the network acts both as host and a router and must therefore be willing to forward packets for other nodes. For this purpose, a routing protocol is needed.

An ad-hoc network has certain characteristics, which imposes new demands on the routing protocol. The most important characteristic is the dynamic topology, which is a consequence of node mobility. Nodes can change position quite frequently, which means that we need a routing protocol that quickly adapts to topology changes. The nodes in an ad-hoc network can consist of laptops and personal digital assistants and are often very limited in resources such as CPU capacity, storage capacity, battery power and bandwidth. This means that the routing protocol should try to minimize control traffic, such as periodic update messages. Instead the routing protocol should be reactive, thus only calculate routes upon receiving a specific request.

The Internet Engineering Task Force currently has a working group named Mobile Ad-hoc Networks that is working on routing specifications for ad-hoc networks. This master thesis evaluates some of the protocols put forth by the working group. This evaluation is done by means of simulation using Network simulator 2 from Berkeley.

The simulations have shown that there certainly is a need for a special ad-hoc routing protocol when mobility increases. More conventional routing protocols like DSDV have a dramatic decrease in performance when mobility is high. Two of the proposed protocols are DSR and AODV. They perform very well when mobility is high. However, we have found that a routing protocol that entirely depends on messages at the IP-level will not perform well. Some sort of support from the lower layer, for instance link failure detection or neighbor discovery is necessary for high performance.

The size of the network and the offered traffic load affects protocols based on source routing, like DSR, to some extent. A large network with many mobile nodes and high offered load will increase the overhead for DSR quite drastically. In these situations, a hop-by-hop based routing protocol like AODV is more desirable.

Preface

This report is the result of our master thesis project carried out at Ericsson Telecom, Switchlab in Stockholm. This master thesis is also the last part of our Master of Science degree at Luleå University of Technology.

Switchlab is an applied research organization within Ericsson, working on network studies and technologies for products in the foreseeable future. This master thesis project has been a cooperation between Switchlab in Stockholm and Ericsson Mobile Data Design (ERV) in Gothenburg. Our master thesis consisted of conducting a simulation study of proposed routing protocols in ad-hoc networks. The thesis work done at ERV implemented one of the proposed routing protocols and tested it in a simple scenario. This has made it possible to share thoughts and ideas with each other.

We would like to thank the following persons: Per Johansson for being our supervisor at Switchlab, Bartosz Mielczarek for contribution of the realistic scenarios and Mikael Degermark for being our Examiner at Luleå University of Technology. Also thanks to Johan Köpman and Jerry Svedlund in Gothenburg for discussions and comments regarding AODV. Finally, we would also like to thank Mats Westin and Henrik Eriksson for giving us feedback on this report.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Wireless communication between mobile users is becoming more popular than ever before. This due to recent technological advances in laptop computers and wireless data communication devices, such as wireless modems and wireless LANs. This has lead to lower prices and higher data rates, which are the two main reasons why mobile computing continues to enjoy rapid growth.

There are two distinct approaches for enabling wireless communication between two hosts. The first approach is to let the existing cellular network infrastructure carry data as well as voice. The major problems include the problem of handoff, which tries to handle the situation when a connection should be smoothly handed over from one base station to another base station without noticeable delay or packet loss. Another problem is that networks based on the cellular infrastructure are limited to places where there exists such a cellular network infrastructure.

The second approach is to form an ad-hoc network among all users wanting to communicate with each other. This means that all users participating in the ad-hoc network must be willing to forward data packets to make sure that the packets are delivered from source to destination. This form of networking is limited in range by the individual nodes transmission ranges and is typically smaller compared to the range of cellular systems. This does not mean that the cellular approach is better than the ad-hoc approach. Ad-hoc networks have several advantages compared to traditional cellular systems. These advantages include:

- On demand setup
- Fault tolerance
- Unconstrained connectivity

Ad-hoc networks do not rely on any pre-established infrastructure and can therefore be deployed in places with no infrastructure. This is useful in disaster recovery situations and places with non-existing or damaged communication infrastructure where rapid deployment of a communication network is needed. Ad-hoc networks can also be useful on conferences where people participating in the conference can form a temporary network without engaging the services of any pre-existing network.

Because nodes are forwarding packets for each other, some sort of routing protocol is necessary to make the routing decisions. Currently there does not exist any standard for a routing protocol for ad-hoc networks, instead this is work in progress. Many problems remain to be solved before any standard can be determined. This thesis looks at some of these problems and tries to evaluate some of the currently proposed protocols.

1.2 Problem description

The objective for this master thesis was to evaluate proposed routing protocols for wireless ad-hoc networks based on performance. This evaluation should be done theoretically and through simulation. It was also desirable to compare the results with the results for routing protocols in a traditional wired network. At the beginning of this master thesis, no implementation of the protocols had been released, so the first main task was to implement some of the protocols.

The thesis also included the goal to generate a simulation environment that could be used as a platform for further studies within the area of ad-hoc networks. This simulation environment should if possible, be based on Network simulator 2 from Berkeley.

The goal of this master thesis was to:

- Get a general understanding of ad-hoc networks.
- Generate a simulation environment that could be used for further studies.
- Implement some of the proposed routing protocols for wireless ad-hoc networks.
- Analyze the protocols theoretically and through simulation.
- Produce a classification of the protocols with respect to applicability in combinations of small/large networks, and mobile/semi-mobile nodes.
- Recommend protocols for specific network scenarios.

1.3 Related work

Many routing protocols have been proposed [2][4][6][8][10][11][12][16][19][22][26], but few comparisons between the different protocols have been made. Of the work that has been done in this field, only the work done by the Monarch[1] project at Carnegie Mellon University (CMU) has compared some of the different proposed routing protocols and evaluated them based on the same quantitative metrics. The result was presented in the article *"A performance comparison of multi-hop ad hoc wireless network routing protocols"* [3] that was released in the beginning of October 1998. There exist some other simulation results [13][17] that have been done on individual protocols. These simulations have however not used the same metrics and are therefore not comparable with each other.

In parallel with our master thesis, a master thesis project in Gothenburg [28] implemented the AODV [19] protocol and tested it in a environment that consisted of 5 computers with wireless interfaces. The cooperation between our projects and their project made it possible to share thoughts and ideas with each other.

1.4 Project organization

The following persons have been involved in this master thesis project:

Simulation study and master thesis authors
M.Sc. Tony Larsson
M.Sc. Nicklas Hedman

Supervisor at Ericsson Telecom AB, Switchlab
Tekn.Lic. Per Johansson

Examiner at Luleå University of Technology
Ph. D. Mikael Degermark

Implementation study at Ericsson Mobile data design (ERV) in Gothenburg
M.Sc. Johan Köpman
M.Sc. Jerry Svedlund

Supervisor at ERV
M.Sc. Christoffer Kanljung

Contribution of realistic scenarios
Ph.D. student at Chalmers University of Technology: Bartosz Mielczarek

[1] MObile Networking ARCHitectures

1.5 Disposition

This report consists of 8 chapters and two appendices. Chapters 1 and 2 explain the concept of ad-hoc networks and routing in general. Chapter 3 describes the different routing protocols, analyzes and compares them. Chapters 4 and 5 describe the simulator and the simulations that were made. Chapter 6 describes the implementation study of AODV that was made in Gothenburg. Chapter 7 concludes the whole report and chapter 8 is the references that we have used. The appendices contain some terminology, details about the implementation of AODV that we did for the simulator and some screenshots of the simulator.

1.6 Abbreviations

AODV	Ad-hoc On-demand Distance Vector
CBR	Constant Bit Rate
CBRP	Cluster Based Routing Protocol
DSDV	Destination Sequenced Distance Vector
DSR	Dynamic Source Routing
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
LAN	Local Area Network
IP	Internet Protocol
MAC	Media Access Protocol
MANET	Mobile Ad-hoc NETworks
OLSR	Optimized Link State Routing Protocol
PDA	Personal Digital Assistant
QoS	Quality of Service
TCP	Transmission Control Protocol
TORA	Temporally Ordered Routing Algorithm
UDP	User Datagram Protocol
WINET	Wireless InterNET
ZRP	Zone Routing Protocol

2 General Concepts

2.1 Wireless ad-hoc networks

2.1.1 General

A wireless ad-hoc network is a collection of mobile/semi-mobile nodes with no pre-established infrastructure, forming a temporary network. Each of the nodes has a wireless interface and communicate with each other over either radio or infrared. Laptop computers and personal digital assistants that communicate directly with each other are some examples of nodes in an ad-hoc network. Nodes in the ad-hoc network are often mobile, but can also consist of stationary nodes, such as access points to the Internet. Semi mobile nodes can be used to deploy relay points in areas where relay points might be needed temporarily.

Figure 1 shows a simple ad-hoc network with three nodes. The outermost nodes are not within transmitter range of each other. However the middle node can be used to forward packets between the outermost nodes. The middle node is acting as a router and the three nodes have formed an ad-hoc network.

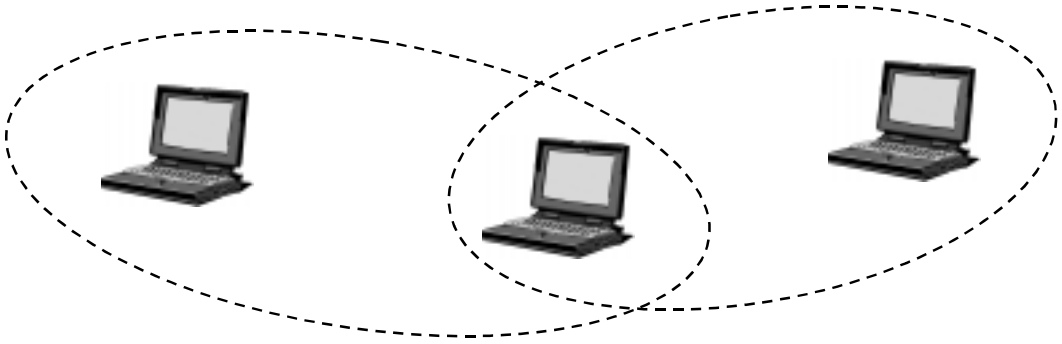


Figure 1: Example of a simple ad-hoc network with three participating nodes.

An ad-hoc network uses no centralized administration. This is to be sure that the network wont collapse just because one of the mobile nodes moves out of transmitter range of the others. Nodes should be able to enter/leave the network as they wish. Because of the limited transmitter range of the nodes, multiple hops may be needed to reach other nodes. Every node wishing to participate in an ad-hoc network must be willing to forward packets for other nodes. Thus every node acts both as a host and as a router. A node can be viewed as an abstract entity consisting of a router and a set of affiliated mobile hosts (Figure 2). A router is an entity, which, among other things runs a routing protocol. A mobile host is simply an IP-addressable host/entity in the traditional sense.

Ad-hoc networks are also capable of handling topology changes and malfunctions in nodes. It is fixed through network reconfiguration. For instance, if a node leaves the network and causes link breakages, affected nodes can easily request new routes and the problem will be solved. This will slightly increase the delay, but the network will still be operational.

Wireless ad-hoc networks take advantage of the nature of the wireless communication medium. In other words, in a wired network the physical cabling is done a priori restricting the connection topology of the nodes. This restriction is not present in the wireless domain and, provided that two nodes are within transmitter range of each other, an instantaneous link between them may form.

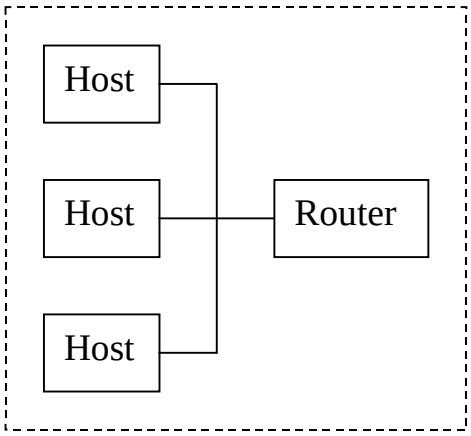


Figure 2: Block diagram of a mobile node acting both as hosts and as router.

2.1.2 Usage

There is no clear picture of what these kinds of networks will be used for. The suggestions vary from document sharing at conferences to infrastructure enhancements and military applications.

In areas where no infrastructure such as the Internet is available an ad-hoc network could be used by a group of wireless mobile hosts. This can be the case in areas where a network infrastructure may be undesirable due to reasons such as cost or convenience. Examples of such situations include disaster recovery personnel or military troops in cases where the normal infrastructure is either unavailable or destroyed.

Other examples include business associates wishing to share files in an airport terminal, or a class of students needing to interact during a lecture. If each mobile host wishing to communicate is equipped with a wireless local area network interface, the group of mobile hosts may form an ad-hoc network.

Access to the Internet and access to resources in networks such as printers are features that probably also will be supported.

2.1.3 Characteristics

Ad-hoc networks are often characterized by a dynamic topology due to the fact that nodes change their physical location by moving around. This favors routing protocols that dynamically discover routes over conventional routing algorithms like distant vector and link state [23]. Another characteristic is that a host/node have very limited CPU capacity, storage capacity, battery power and bandwidth, also referred to as a “thin client”. This means that the power usage must be limited thus leading to a limited transmitter range.

The access media, the radio environment, also has special characteristics that must be considered when designing protocols for ad-hoc networks. One example of this may be unidirectional links. These links arise when for example two nodes have different strength on their transmitters, allowing only one of the host to hear the other, but can also arise from disturbances from the surroundings. Multihop in a radio environment may result in an overall transmit capacity gain and power gain, due to the squared relation between coverage and required output power. By using multihop, nodes can transmit the packets with a much lower output power.

2.2 Routing

Because of the fact that it may be necessary to hop several hops (multi-hop) before a packet reaches the destination, a routing protocol is needed. The routing protocol has two main functions, selection of routes for various source-destination pairs and the delivery of messages to their correct destination. The second function is conceptually straightforward using a variety of protocols and data structures (routing tables). This report is focused on selecting and finding routes.

2.2.1 Conventional protocols

If a routing protocol is needed, why not use a conventional routing protocol like link state or distance vector? They are well tested and most computer communications people are familiar with them. The main problem with link-state and distance vector is that they are designed for a static topology, which means that they would have problems to converge to a steady state in an ad-hoc network with a very frequently changing topology.

Link state and distance vector would probably work very well in an ad-hoc network with low mobility, i.e. a network where the topology is not changing very often. The problem that still remains is that link-state and distance-vector are highly dependent on periodic control messages. As the number of network nodes can be large, the potential number of destinations is also large. This requires large and frequent exchange of data among the network nodes. This is in contradiction with the fact that all updates in a wireless interconnected ad hoc network are transmitted over the air and thus are costly in resources such as bandwidth, battery power and CPU. Because both link-state and distance vector tries to maintain routes to all reachable destinations, it is necessary to maintain these routes and this also wastes resources for the same reason as above.

Another characteristic for conventional protocols are that they assume bi-directional links, e.g. that the transmission between two hosts works equally well in both directions. In the wireless radio environment this is not always the case.

Because many of the proposed ad-hoc routing protocols have a traditional routing protocol as underlying algorithm, it is necessary to understand the basic operation for conventional protocols like distance vector, link state and source routing.

2.2.2 Link State

In link-state routing [23], each node maintains a view of the complete topology with a cost for each link. To keep these costs consistent; each node periodically broadcasts the link costs of its outgoing links to all other nodes using flooding. As each node receives this information, it updates its view of the network and applies a shortest path algorithm to choose the next-hop for each destination.

Some link costs in a node view can be incorrect because of long propagation delays, partitioned networks, etc. Such inconsistent network topology views can lead to formation of routing-loops. These loops are however short-lived, because they disappear in the time it takes a message to traverse the diameter of the network.

2.2.3 Distance Vector

In distance vector [23] each node only monitors the cost of its outgoing links, but instead of broadcasting this information to all nodes, it periodically broadcasts to each of its neighbors an estimate of the shortest distance to every other node in the network. The receiving nodes then use this information to recalculate the routing tables, by using a shortest path algorithm.

Compared to link-state, distance vector is more computation efficient, easier to implement and requires much less storage space. However, it is well known that distance vector can cause the formation of both short-lived and long-lived routing loops. The primary cause for this is that the nodes choose their next-hops in a completely distributed manner based on information that can be stale.

2.2.4 Source Routing

Source routing [23] means that each packet must carry the complete path that the packet should take through the network. The routing decision is therefore made at the source. The advantage with this approach is that it is very easy to avoid routing loops. The disadvantage is that each packet requires a slight overhead.

2.2.5 Flooding

Many routing protocols uses broadcast to distribute control information, that is, send the control information from an origin node to all other nodes. A widely used form of broadcasting is flooding [23] and operates as follows. The origin node sends its information to its neighbors (in the wireless case, this means all nodes that are within transmitter range). The neighbors relay it to their neighbors and so on, until the packet has reached all nodes in the network. A node will only relay a packet once and to ensure this some sort of sequence number can be used. This sequence number is increased for each new packet a node sends.

2.2.6 Classification

Routing protocols can be classified [1] into different categories depending on their properties.

- Centralized vs. Distributed
- Static vs. Adaptive
- Reactive vs. Proactive

One way to categorize the routing protocols is to divide them into centralized and distributed algorithms. In centralized algorithms, all route choices are made at a central node, while in distributed algorithms, the computation of routes is shared among the network nodes.

Another classification of routing protocols relates to whether they change routes in response to the traffic input patterns. In static algorithms, the route used by source-destination pairs is fixed regardless of traffic conditions. It can only change in response to a node or link failure. This type of algorithm cannot achieve high throughput under a broad variety of traffic input patterns. Most major packet networks uses some form of adaptive routing where the routes used to route between source-destination pairs may change in response to congestion

A third classification that is more related to ad-hoc networks is to classify the routing algorithms as either proactive or reactive. Proactive protocols attempt to continuously evaluate the routes within the network, so that when a packet needs to be forwarded, the route is already known and can be immediately used. The family of Distance-Vector protocols is an example of a proactive scheme. Reactive protocols, on the other hand, invoke a route determination procedure on demand only. Thus, when a route is needed, some sort of global search procedure is employed. The family of classical flooding algorithms belongs to the reactive group. Proactive schemes have the advantage that when a route is needed, the delay before actual packets can be sent is very small. On the other side proactive schemes needs time to converge to a steady state. This can cause problems if the topology is changing frequently.

3 Ad-hoc routing protocols

This chapter describes the different ad-hoc routing protocols that we have chosen to simulate and analyze.

3.1 Desirable properties

If the conventional routing protocols do not meet our demands, we need a new routing protocol. The question is what properties such protocols should have? These are some of the properties [5] that are desirable:

Distributed operation
The protocol should of course be distributed. It should not be dependent on a centralized controlling node. This is the case even for stationary networks. The difference is that nodes in an ad-hoc network can enter/leave the network very easily and because of mobility the network can be partitioned.

Loop free
To improve the overall performance, we want the routing protocol to guarantee that the routes supplied are loop-free. This avoids any waste of bandwidth or CPU consumption.

Demand based operation
To minimize the control overhead in the network and thus not wasting network resources more than necessary, the protocol should be reactive. This means that the protocol should only react when needed and that the protocol should not periodically broadcast control information.

Unidirectional link support
The radio environment can cause the formation of unidirectional links. Utilization of these links and not only the bi-directional links improves the routing protocol performance.

Security
The radio environment is especially vulnerable to impersonation attacks, so to ensure the wanted behavior from the routing protocol, we need some sort of preventive security measures. Authentication and encryption is probably the way to go and the problem here lies within distributing keys among the nodes in the ad-hoc network. There are also discussions about using IP-sec [14] that uses tunneling to transport all packets.

Power conservation
The nodes in an ad-hoc network can be laptops and thin clients, such as PDAs that are very limited in battery power and therefore uses some sort of stand-by mode to save power. It is therefore important that the routing protocol has support for these sleep-modes.

Multiple routes
To reduce the number of reactions to topological changes and congestion multiple routes could be used. If one route has become invalid, it is possible that another stored route could still be valid and thus saving the routing protocol from initiating another route discovery procedure.

Quality of service support
Some sort of Quality of Service support is probably necessary to incorporate into the routing protocol. This has a lot to do with what these networks will be used for. It could for instance be real-time traffic support.

None of the proposed protocols from MANET have all these properties, but it is necessary to remember that the protocols are still under development and are probably extended with more functionality. The primary function is still to find a route to the destination, not to find the best/optimal/shortest-path route.

The remainder of this chapter will describe the different routing protocols and analyze them theoretically.

3.2 MANET

IETF has a working group named MANET (Mobile Ad-hoc Networks) [15] that is working in the field of ad-hoc networks. They are currently developing routing specifications for ad-hoc IP networks that support scaling to a couple of hundred nodes. Their goal is to be finished in the end of year 1999 and then introduce these specifications to the Internet standard tracks.

Even if MANET currently is working on routing protocols, it also serves as a meeting place and forum, so people can discuss issues concerning ad-hoc networks. Currently they have seven routing protocol drafts:

- AODV - Ad-hoc On Demand Distance Vector [19]
- ZRP - Zone Routing Protocol [8]
- TORA / IMEP - Temporally Ordered Routing Algorithm / Internet MANET Encapsulation Protocol [6][16][17]
- DSR - Dynamic Source Routing [12][13]
- CBRP - Cluster Based Routing Protocol [11]
- CEDAR - Core Extraction Distributed Ad hoc Routing [26]
- AMRoute – Ad-hoc Multicast Routing Protocol [2]
- OLSR - Optimized Link State Routing Protocol [10]

Of these proposed protocols we have chosen to analyze AODV, DSR, ZRP, CBRP and TORA theoretically. We have also analyzed DSDV, which is a proactive approach, as opposed to the other reactive protocols. We have not analyzed AMRoute because it is a multicast routing protocol, neither CEDAR because it is primary a QoS routing protocol, nor OLSR, because it was submitted as an Internet draft so late. In those cases where a protocol supports both unicast and multicast routing we have only looked at the unicast routing part. Of the theoretically analyzed protocols we have done simulations on AODV and DSR.

3.3 Destination Sequenced Distance Vector - DSDV

3.3.1 Description

DSDV [22] is a hop-by-hop distance vector routing protocol that in each node has a routing table that for all reachable destinations stores the next-hop and number of hops for that destination. Like distance-vector, DSDV requires that each node periodically broadcast routing updates. The advantage with DSDV over traditional distance vector protocols is that DSDV guarantees loop-freedom.

To guarantee loop-freedom DSDV uses a sequence numbers to tag each route. The sequence number shows the freshness of a route and routes with higher sequence numbers are favorable. A route R is considered more favorable than R' if R has a greater sequence number or, if the routes have the same sequence number but R has lower hop-count. The sequence number is increased when a node A detects that a route to a destination D has broken. So the next time node A advertises its routes, it will advertise the route to D with an infinite hop-count and a sequence number that is larger than before.

DSDV basically is distance vector with small adjustments to make it better suited for ad-hoc networks. These adjustments consist of triggered updates that will take care of topology changes in the time between broadcasts. To reduce the amount of information in these packets there are two types of update messages defined: full and incremental dump. The full dump carries all available routing information and the incremental dump that only carries the information that has changed since the last dump.

3.3.2 Properties

Because DSDV is dependent on periodic broadcasts it needs some time to converge before a route can be used. This converge time can probably be considered negligible in a static wired network, where the topology is not changing so frequently. In an ad-hoc network on the other hand, where the topology is expected to be very dynamic, this converge time will probably mean a lot of dropped packets before a valid route is detected. The periodic broadcasts also add a large amount of overhead into the network.

3.4 Ad-hoc On Demand Distance vector - AODV

3.4.1 Description

The Ad Hoc On-Demand Distance Vector (AODV) [19] routing protocol enables multi-hop routing between participating mobile nodes wishing to establish and maintain an ad-hoc network. AODV is based upon the distance vector algorithm. The difference is that AODV is reactive, as opposed to proactive protocols like DV, i.e. AODV only requests a route when needed and does not require nodes to maintain routes to destinations that are not actively used in communications. As long as the endpoints of a communication connection have valid routes to each other, AODV does not play any role.

Features of this protocol include loop freedom and that link breakages cause immediate notifications to be sent to the affected set of nodes, but only that set. Additionally, AODV has support for multicast routing and avoids the Bellman Ford "counting to infinity" problem [27]. The use of destination sequence numbers guarantees that a route is "fresh".

The algorithm uses different messages to discover and maintain links. Whenever a node wants to try and find a route to another node, it broadcasts a Route Request (RREQ) to all its neighbors. The RREQ propagates through the network until it reaches the destination or a node with a fresh enough route to the destination. Then the route is made available by unicasting a RREP back to the source.

The algorithm uses hello messages (a special RREP) that are broadcasted periodically to the immediate neighbors. These hello messages are local advertisements for the continued presence of the node and neighbors using routes through the broadcasting node will continue to mark the routes as valid. If hello messages stop coming from a particular node, the neighbor can assume that the node has moved away and mark that link to the node as broken and notify the affected set of nodes by sending a link failure notification (a special RREP) to that set of nodes.

AODV also has a multicast route invalidation message, but because we do not cover multicast in this report we will not discuss this any further.

Route table management
AODV needs to keep track of the following information for each route table entry:

- Destination IP Address: IP address for the destination node.
- Destination Sequence Number: Sequence number for this destination.
- Hop Count: Number of hops to the destination.
- Next Hop: The neighbor, which has been designated to forward packets to the destination for this route entry.
- Lifetime: The time for which the route is considered valid.
- Active neighbor list: Neighbor nodes that are actively using this route entry.
- Request buffer: Makes sure that a request is only processed once.

Route discovery
A node broadcasts a RREQ when it needs a route to a destination and does not have one available. This can happen if the route to the destination is unknown, or if a previously valid route expires. After broadcasting a RREQ, the node waits for a RREP. If the reply is not received within a certain time, the node may rebroadcast the RREQ or assume that there is no route to the destination.

Forwarding of RREQs is done when the node receiving a RREQ does not have a route to the destination. It then rebroadcast the RREQ. The node also creates a temporary reverse route to the Source IP Address in its routing table with next hop equal to the IP address field of the neighboring node that sent the broadcast RREQ. This is done to keep track of a route back to the original node making the request, and might be used for an eventual RREP to find its way back to the requesting node. The route is temporary in the sense that it is valid for a much shorter time, than an actual route entry.

When the RREQ reaches a node that either is the destination node or a node with a valid route to the destination, a RREP is generated and unicasted back to the requesting node. While this RREP is forwarded, a route is created to the destination and when the RREP reaches the source node, there exists a route from the source to the destination.

Route maintenance
When a node detects that a route to a neighbor no longer is valid, it will remove the routing entry and send a link failure message, a triggered route reply message to the neighbors that are actively using the route, informing them that this route no longer is valid. For this purpose AODV uses a active neighbor list to keep track of the neighbors that are using a particular route. The nodes that receive this message will repeat this procedure. The message will eventually be received by the affected sources that can chose to either stop sending data or requesting a new route by sending out a new RREQ.

3.4.2 Properties

The advantage with AODV compared to classical routing protocols like distance vector and link-state is that AODV has greatly reduced the number of routing messages in the network. AODV achieves this by using a reactive approach. This is probably necessary in an ad-hoc network to get reasonably performance when the topology is changing often.

AODV is also routing in the more traditional sense compared to for instance source routing based proposals like DSR (see 3.5). The advantage with a more traditional routing protocol in an ad-hoc network is that connections from the ad-hoc network to a wired network like the Internet is most likely easier.

The sequence numbers that AODV uses represents the freshness of a route and is increased when something happens in the surrounding area. The sequence prevents loops from being formed, but can however also be the cause for new problems. What happens for instance when the sequence numbers no longer are synchronized in the network? This can happen when the network becomes partitioned, or the sequence numbers wrap around.

AODV only support one route for each destination. It should however be fairly easy to modify AODV, so that it supports several routes per destination. Instead of requesting a new route when an old route becomes invalid, the next stored route to that destination could be tried. The probability for that route to still be valid should be rather high.

Although the Triggered Route Replies are reduced in number by only sending the Triggered Route Replies to affected senders, they need to traverse the whole way from the failure to the senders. This distance can be quite high in numbers of hops. AODV sends one Triggered RREP for every active neighbor in the active neighbor list for all entries that have been affected of a link failure. This can mean that each active neighbor can receive several triggered RREPs informing about the same link failure, but for different destinations, if a large fraction of the network traffic is routed through the same node and this node goes down. An aggregated solution would be more appropriate here.

AODV uses hello messages at the IP-level. This means that AODV does not need support from the link layer to work properly. It is however questionable if this kind of protocol can operate with good performance without support from the link layer. The hello messages adds a significant overhead to the protocol.

AODV does not support unidirectional links. When a node receives a RREQ, it will setup a reverse route to the source by using the node that forwarded the RREQ as nexthop. This means that the route reply, in most cases is unicasted back the same way as the route request used. Unidirectional link support would make it possible to utilize all links and not only the bi-directional links. It is however questionable if unidirectional links are desirable in a real environment. The acknowledgements in the MAC protocol IEEE 802.11 would for instance not work with unidirectional links.

3.5 Dynamic Source Routing - DSR

3.5.1 Description

Dynamic Source Routing (DSR) [3][12][13] also belongs to the class of reactive protocols and allows nodes to dynamically discover a route across multiple network hops to any destination. Source routing means that each packet in its header carries the complete ordered list of nodes through which the packet must pass. DSR uses no periodic routing messages (e.g. no router advertisements), thereby reducing network bandwidth overhead, conserving battery power and avoiding large routing updates throughout the ad-hoc network. Instead DSR relies on support from the MAC layer (the MAC layer should inform the routing protocol about link failures). The two basic modes of operation in DSR are route discovery and route maintenance.

Route discovery

Route discovery is the mechanism whereby a node X wishing to send a packet to Y, obtains the source route to Y. Node X requests a route by broadcasting a Route Request (RREQ) packet. Every node receiving this RREQ searches through its route cache for a route to the requested destination. DSR stores all known routes in its route cache. If no route is found, it forwards the RREQ further and adds its own address to the recorded hop sequence. This request propagates through the network until either the destination or a node with a route to the destination is reached. When this happen a Route Reply (RREP) is unicasted back to the originator. This RREP packet contains the sequence of network hops through which it may reach the target.

In Route Discovery, a node first sends a RREQ with the maximum propagation limit (hop limit) set to zero, prohibiting its neighbors from rebroadcasting it. At the cost of a single broadcast packet, this mechanism allows a node to query the route caches of all its neighbors.

Nodes can also operate their network interface in promiscuous mode, disabling the interface address filtering and causing the network protocol to receive all packets that the interface overhears. These packets are scanned for useful source routes or route error messages and then discarded.

The route back to the originator can be retrieved in several ways. The simplest way is to reverse the hop record in the packet. However this assumes symmetrical links. To deal with this, DSR checks the route cache of the replying node. If a route is found, it is used instead. Another way is to piggyback the reply on a RREQ targeted at the originator. This means that DSR can compute correct routes in the presence of asymmetric (unidirectional) links. Once a route is found, it is stored in the cache with a time stamp and the route maintenance phase begins.

Route maintenance

Route maintenance is the mechanism by which a packet sender S detects if the network topology has changed so that it can no longer use its route to the destination D. This might happen because a host listed in a source route, move out of wireless transmission range or is turned off making the route unusable. A failed link is detected by either actively monitoring acknowledgements or passively by running in promiscuous mode, overhearing that a packet is forwarded by a neighboring node.

When route maintenance detects a problem with a route in use, a route error packet is sent back to the source node. When this error packet is received, the hop in error is removed from this hosts route cache, and all routes that contain this hop are truncated at this point.

3.5.2 Properties

DSR uses the key advantage of source routing. Intermediate nodes do not need to maintain up-to-date routing information in order to route the packets they forward. There is also no need for periodic routing advertisement messages, which will lead to reduce network bandwidth overhead, particularly during periods when little or no significant host movement is taking place. Battery power is also conserved on the mobile hosts, both by not sending the advertisements and by not needing to receive them, a host could go down to sleep instead.

This protocol has the advantage of learning routes by scanning for information in packets that are received. A route from A to C through B means that A learns the route to C, but also that it will learn the route to B. The source route will also mean that B learns the route to A and C and that C learns the route to A and B. This form of active learning is very good and reduces overhead in the network.

However, each packet carries a slight overhead containing the source route of the packet. This overhead grows when the packet has to go through more hops to reach the destination. So the packets sent will be slightly bigger, because of the overhead.

Running the interfaces in promiscuous mode is a serious security issue. Since the address filtering of the interface is turned off and all packets are scanned for information. A potential intruder could listen to all packets and scan them for useful information such as passwords and credit card numbers. Applications have to provide the security by encrypting their data packets before transmission. The routing protocols are prime targets for impersonation attacks and must therefore also be encrypted. One way to achieve this is to use IP-sec [14].

DSR also has support for unidirectional links by the use of piggybacking the source route a new request. This can increase the performance in scenarios where we have a lot of unidirectional links. We must however have a MAC protocol that also supports this.

3.6 Zone Routing Protocol - ZRP

3.6.1 Description

Zone Routing Protocol (ZRP) [8] is a hybrid of a reactive and a proactive protocol. It divides the network into several routing zones and specifies two totally detached protocols that operate inside and between the routing zones.

The Intrazone Routing Protocol (IARP) operates inside the routing zone and learns the minimum distance and routes to all the nodes within the zone. The protocol is not defined and can include any number of proactive protocols, such as Distance Vector or link-state routing. Different zones may operate with different intrazone protocols as long as the protocols are restricted to those zones. A change in topology means that update information only propagates within the affected routing zones as opposed to affecting the entire network.

The second protocol, the Interzone Routing Protocol (IERP) is reactive and is used for finding routes between different routing zones. This is useful if the destination node does not lie within the routing zone. The protocol then broadcasts (i.e. bordercasts) a Route REQuest (RREQ) to all border nodes within the routing zone, which in turn forwards the request if the destination node is not found within their routing zone. This procedure is repeated until the requested node is found and a route reply is sent back to the source indicating the route. IERP uses a Bordercast Resolution Protocol (BRP) [8] that is included in ZRP. BRP provides bordercasting services, which do not exist in IP. Bordercasting is the process of sending IP datagrams from one node to all its peripheral nodes. BRP keeps track of the peripheral nodes and resolves a border cast address to the individual IP-addresses of the peripheral nodes. The message that was bordercasted is then encapsulated into a BRP packet and sent to each peripheral node.

Routing Zone

A routing zone is defined as a set of nodes, within a specific minimum distance in number of hops from the node in question. The distance is referred to as the zone radius. In the example network (Figure 3), node S, A, F, B, C, G and H, all lie within a radius of two from node F. Even though node B also has a distance of 3 hops from node F, it is included in the zone since the shortest distance is only 2 hops. Border nodes or peripheral nodes are nodes whose minimum distance to the node in question is equal exactly to the zone radius. In Figure 3, nodes B and F are border nodes to S.

Consider the network in Figure 3. Node S wants to send a packet to node D. Since node D is not in the routing zone of S, a route request is sent to the border nodes B and F. Each border node checks to see if D is in their routing zone. Neither B nor F finds the requested node in their routing zone; thus the request is forwarded to the respectively border nodes. F sends the request to S, B, C and H while B sends the request to S, F, E and G. Now the requested node D is found within the routing zone of both C and E thus a reply is generated and sent back towards the source node S.

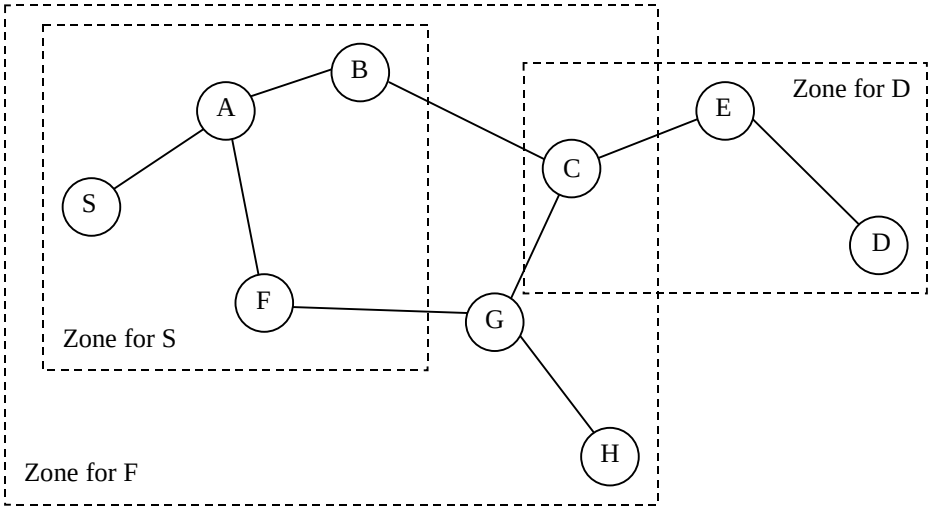


Figure 3: Network using ZRP. The dashed squares show the routing zones for nodes S and D.

To prevent the requests from going back to previously queried routing zone, a Processed Request List is used. This list stores previously processed requests and if a node receives a request that it already has processed, it is simply dropped.

3.6.2 Properties

ZRP is a very interesting protocol and can be adjusted of its operation to the current network operational conditions (e.g. change the routing zone diameter). However this is not done dynamically, but instead it is suggested that this zone radius should be set by the administration of the network or with a default value by the manufacturer. The performance of this protocol depends quite a lot on this decision.

Since this is a hybrid between proactive and reactive schemes, this protocol use advantages from both. Routes can be found very fast within the routing zone, while routes outside the zone can be found by efficiently querying selected nodes in the network. One problem is however that the proactive intrazone routing protocol is not specified. The use of different intrazone routing protocols would mean that the nodes would have to support several different routing protocols. This is not a good idea when dealing with thin clients. It is better to use the same intrazone routing protocol in the entire network.

ZRP also limits propagation of information about topological changes to the neighborhood of the change only (as opposed to a fully proactive scheme, which would basically flood the entire network when a change in topology occurred). However, a change in topology can affect several routing zones.

3.7 Temporally-Ordered Routing Algorithm - TORA

3.7.1 Description

Temporally Ordered Routing Algorithm (TORA) [16][17] is a distributed routing protocol. The basic underlying algorithm is one in a family referred to as link reversal algorithms. TORA is designed to minimize reaction to topological changes. A key concept in its design is that control messages are typically localized to a very small set of nodes. It guarantees that all routes are loop-free (temporary loops may form), and typically provides multiple routes for any source/destination pair. It provides only the routing mechanism and depends on Internet MANET Encapsulation Protocol (IMEP [6]) for other underlying functions.

TORA can be separated into three basic functions: creating routes, maintaining routes, and erasing routes. The creation of routes basically assigns directions to links in an undirected network or portion of the network, building a directed acyclic graph (DAG) rooted at the destination (See Figure 4).

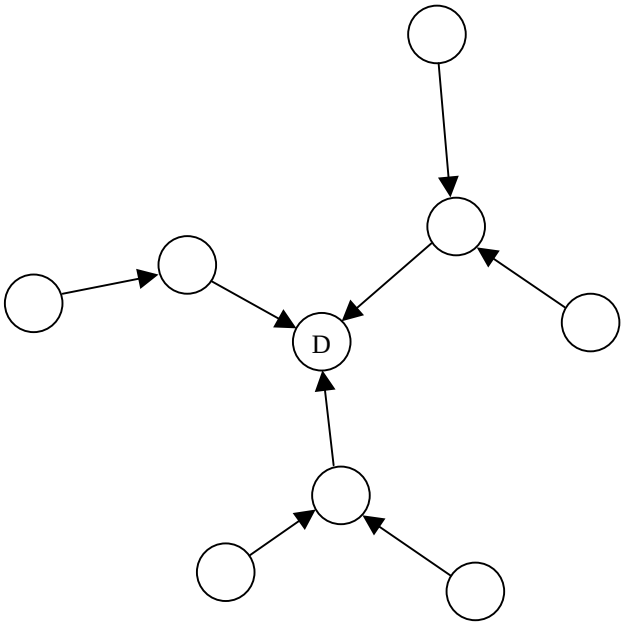


Figure 4: Directed acyclic graph rooted at destination.

TORA associates a height with each node in the network. All messages in the network flow downstream, from a node with higher height to a node with lower height. Routes are discovered using Query (QRY) and Update (UPD) packets. When a node with no downstream links needs a route to a destination, it will broadcast a QRY packet. This QRY packet will propagate through the network until it reaches a node that has a route or the destination itself. Such a node will then broadcast a UPD packet that contains the node height. Every node receiving this UPD packet will set its own height to a larger height than specified in the UPD message. The node will then broadcast its own UPD packet. This will result in a number of directed links from the originator of the QRY packet to the destination. This process can result in multiple routes.

Maintaining routes refers to reacting to topological changes in the network in a manner such that routes to the destination are re-established within a finite time, meaning that its directed portions return to a destination-oriented graph within a finite time. Upon detection of a network partition, all links in the portion of the network that has become partitioned from the destination are marked as undirected to erase invalid routes. The erasing of routes is done using clear (CLR) messages.

3.7.2 Properties

The protocols underlying link reversal algorithm will react to link changes through a simple localized single pass of the distributed algorithm. This prevents CLR packets to propagate too far in the network. A comparison made by the CMU Monarch project has however shown that the overhead in TORA is quite large because of the use of IMEP.

The graph is rooted at the destination, which has the lowest height. However, the source originating the QRY does not necessarily have the highest height. This can lead to the situation, where multiple routes are possible from the source to the destination, but only one route will be discovered. The reason for this is that the height is initially based on the distance in number of hops from the destination.

3.8 Internet MANET Encapsulation Protocol - IMEP

3.8.1 Description

IMEP [5] is a protocol designed to support the operation of many routing protocols in Ad-hoc networks. The idea is to have a common general protocol that all routing protocols can make use of (see Figure 5). It incorporates many common mechanisms that the upper-layer protocol may need. These include:

- Link status sensing
- Control message aggregation and encapsulation
- Broadcast reliability
- Network-layer address resolution
- Hooks for interrouter security authentication procedures

IMEP also provides an architecture for MANET router identification, interface identification and addressing. IMEPs purpose is to improve overall performance by reducing the number of control messages and to put common functionality into one unified, generic protocol useful to all upper-level routing protocols.

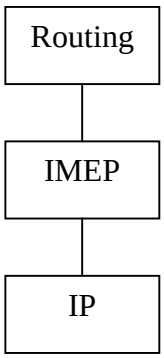


Figure 5: IMEP in the protocol stack.

Of the currently purposed protocols, only TORA and OLSR use IMEP. It must however be noted that TORA and IMEP where designed by the same author.

3.8.2 Properties

The idea to have a general protocol for common basic features is good, but from a performance point of view this is not such a good idea. It adds another layer to the protocol stack. As the work by the CMU Monarch projects has shown [3], IMEP generates a lot of overhead, mainly because of IMEPs neighbor discovery mechanism that generates at least one hello message per second, but also because of the reliable in-order delivery of the packets that IMEP provides.

3.9 Cluster Based Routing Protocol - CBRP

3.9.1 Description

The idea behind CBRP [11] is to divide the nodes of an ad-hoc network into a number of overlapping or disjoint clusters. One node is elected as cluster head for each cluster. This cluster head maintains the membership information for the cluster. Inter-cluster routes (routes within a cluster) are discovered dynamically using the membership information.

CBRP is based on source routing, similar to DSR. This means that intracluster routes (routes between clusters) are found by flooding the network with Route Requests (RREQ). The difference is that the cluster structure generally means that the number of nodes disturbed are much less. Flat routing protocols, i.e. only one level of hierarchy, might suffer from excessive overhead when scaled up.

CBRP is like the other protocols fully distributed. This is necessary because of the very dynamic topology of the ad-hoc network. Furthermore, the protocol takes into consideration the existence of unidirectional links.

Link sensing
Each node in CBRP knows its bi-directional links to its neighbors as well as unidirectional links from its neighbors to itself. To handle this, each node must maintain a Neighbor Table (see Table 1).

Table 1: Neighbor table.

Neighbor ID	Link status	Role
Neighbor 1	Bi/unidirectional link to me	Is 1 a cluster head or member
Neighbor 2	Bi/unidirectional link to me	Is 2 a cluster head or member
…	…	…
Neighbor n	Bi/unidirectional link to me	Is n a cluster head or member

Each node periodically broadcasts its neighbor table in a hello message. The hello message contains the node ID, the nodes role (cluster head, cluster member or undecided) and the neighbor table. The hello messages are used to update the neighbor tables at each node. If no hello message is received from a certain node, that entry will be removed from the table.

Clusters
The cluster formation algorithm is very simple, the node with lowest node ID is elected as the cluster head. The nodes use the information in the hello messages to decide whether or not they are the cluster heads. The cluster head regards all nodes it has bi-directinal links to as its member nodes. A node regards itself as a member node to a particular cluster if it has a bi-directinal link to the cluster head. It is possible for a node to belong to several clusters.

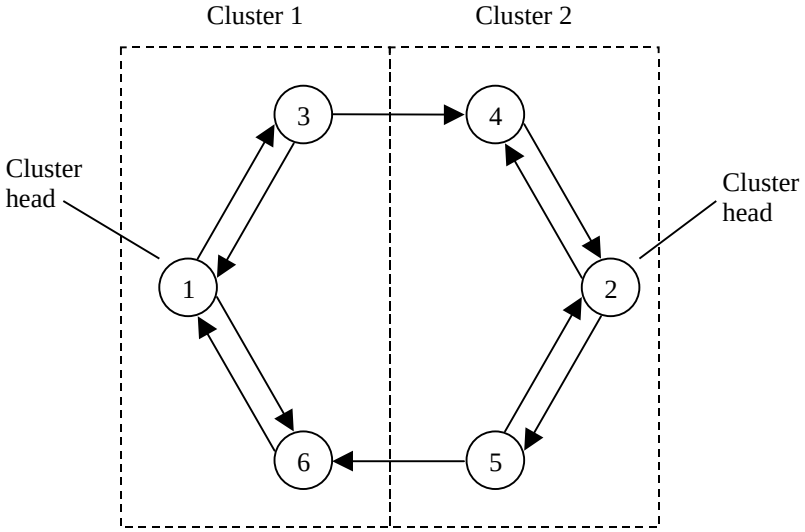


Figure 6: Bi-directional linked clusters.

Clusters are identified by their respective cluster heads, which means that the cluster head must change as infrequently as possible. The algorithm is therefore not a strict "lowest ID" clustering algorithm. A non-cluster head never challenges the status of an existing cluster head. Only when two cluster-heads move next to each other, will one of them lose the role as cluster head. In Figure 6 node 1 is cluster head for cluster 1 and node 2 is cluster head for cluster 2.

Routing

Routing in CBRP is based on source routing and the route discovery is done, by flooding the network with Route Requests (RREQ). The clustering approach however means that fewer nodes are disturbed. This, because only the cluster heads are flooded. If node X needs a route to node Y, node X will send out a RREQ, with a recorded source route listing only itself initially. Any node forwarding this packet will add its own ID in this RREQ. Each node forwards a RREQ only once and it never forwards it to node that already appears in the recorded route.

In CBRP, a RREQ will always follow a route with the following pattern:

```
Source->Cluster head->Gateway->Cluster head->Gateway-> ... ->Destination
```

A gateway node for a cluster is a node that knows that it has a bi-directional or a unidirectional link to a node in another cluster. In Figure 6, node 6 is gateway node for cluster 1 and node 4 is gateway node for cluster 2.

The source unicasts the RREQ to its cluster head. Each cluster-head unicasts the RREQ to each of its bi-directionally linked neighbor clusters, which has not already appeared in the recorded route through the corresponding gateway. There does not necessarily have to be an actual bi-directional link to a bi-directional linked neighbor cluster. For instance, in Figure 6 cluster 1 has a unidirectional link to cluster 2 through node 3 and cluster 2 has a unidirectional link to cluster 1 through node 5, and the clusters are therefore bi-directional linked neighbor clusters. This procedure continues until the target is found or another node can supply the route. When the RREQ reaches the target, the target may chose to memorize the reversed route to the source. It then copies the recorded route to a Route Reply packet and sends it back to the source.

3.9.2 Properties

This protocol has a lot of common features with the earlier discussed protocols. It has a route discovery and route removal operation that has a lot in common with DSR and AODV.

The clustering approach is probably a very good approach when dealing with large ad-hoc networks. The solution is more scalable than the other protocols, because it uses the clustering approach that limits the number of messages that need to be sent. CBRP also has the advantage that it utilizes unidirectional links. One remaining question is however how large each cluster should be. This parameter is critical to how the protocol will behave.

3.10 Comparison

So far, the protocols have been analyzed theoretically. Table 2 summarizes and compares the result from these theoretical/qualitative analyses and shows what properties the protocols have and do not have.

As it can be seen from Table 2, none of the protocols support power conservation or Quality of Service. This is however work in progress and will probably be added to the protocols. All protocols are distributed, thus none of the protocols is dependent on a centralized node and can therefore easily reconfigure in the event of topology changes.

Table 2: Comparison between ad-hoc routing protocols.

	DSDV	**AODV**	**DSR**	**ZRP**	**TORA/ IMEP**	**CBRP**
Loop-free	Yes	Yes	Yes	Yes	No, short lived loops	Yes
Multiple routes	No	No	Yes	No	Yes	Yes
Distributed	Yes	Yes	Yes	Yes	Yes	Yes
Reactive	No	Yes	Yes	Partially	Yes	Yes
Unidirectional link support	No	No	Yes	No	No	Yes
QoS Support	No	No	No	No	No	No
Multicast	No	Yes	No	No	No	No
Security	No	No	No	No	No	No
Power conservation	No	No	No	No	No	No
Periodic broadcasts	Yes	Yes	No	Yes	Yes (IMEP)	Yes
Requires reliable or sequenced data	No	No	No	No	Yes	No

DSDV is the only proactive protocol in this comparison. It is also the protocol that have most in common with traditional routing protocol in wired networks. The sequence numbers were added to ensure loop-free routes. DSDV will probably be good enough in networks, which allows the protocol to converge in reasonable time. This however means that the mobility cannot be too high. The authors of DSDV came to the same conclusions and designed AODV, which is a reactive version of DSDV. They also added multicast capabilities, which will enhance the performance significantly when one node communicates with several nodes. The reactive approach in AODV has many similarities with the reactive approach of DSR. They both have a route discovery mode that uses request messages to find new routes. The difference is that DSR is based on source routing and will learn more routes than AODV. DSR also has the advantage that it supports unidirectional links. DSR has however one major drawback and it is the source route that must be carried in each packet. This can be quite costly, especially when QoS is going to be used.

ZRP and CBRP are two very interesting proposals that divide the network into several zones/clusters. This approach is probably a very good solution for large networks. Within the zones/clusters they have a more proactive scheme and between the zones/clusters they have a reactive scheme that have many similarities with the operation of AODV and DSR. They have for instance a route discovery phase that sends request through the network. The difference between ZRP and CBRP is how the network is divided. In ZRP all zones are overlapping and in CBRP clusters can be both overlapping and disjoint.

None of the presented protocols are adaptive, meaning that the protocols do not take any smart routing decisions when the traffic load in the network is taken into consideration. As a route selection criteria the proposed protocols use metrics such as shortest number of hops and quickest response time to a request. This can lead to the situation where all packets are routed through the same node even if there exist better routes where the traffic load is not as large.

4 Simulation Environment

The simulator we have used to simulate the ad-hoc routing protocols in is the Network Simulator 2 (ns) [7] from Berkeley. To simulate the mobile wireless radio environment we have used a mobility extension to ns that is developed by the CMU Monarch project at Carnegie Mellon University.

4.1 Network Simulator

Network simulator 2 is the result of an on-going effort of research and development that is administrated by researchers at Berkeley. It is a discrete event simulator targeted at networking research. It provides substantial support for simulation of TCP, routing, and multicast protocols.

The simulator is written in C++ and a script language called OTcl[2]. Ns uses an Otcl interpreter towards the user. This means that the user writes an OTcl script that defines the network (number of nodes, links), the traffic in the network (sources, destinations, type of traffic) and which protocols it will use. This script is then used by ns during the simulations. The result of the simulations is an output trace file that can be used to do data processing (calculate delay, throughput etc) and to visualize the simulation with a program called Network Animator (NAM). See Appendix C for a screenshot of NAM. NAM is a very good visualization tool that visualizes the packets as they propagate through the network. An overview of how a simulation is done in ns is shown in Figure 7.

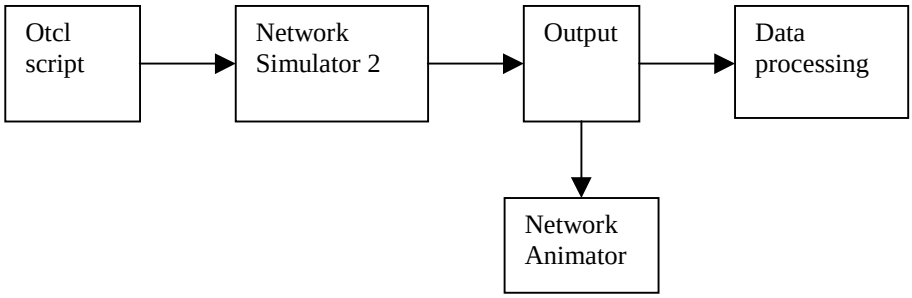


Figure 7: Network simulator 2.

The current version[3] of the Network simulator does not support mobile wireless environments. The Network simulator alone is only intended for stationary networks with wired links. This caused us some problems in the beginning of this master thesis. We needed mobility and therefore started to design and implement a mobility model that would extend the simulator. We also started to implement the AODV protocol. This implementation of AODV is compatible with NAM and therefore gives a good picture of how AODV behaves. It is very easy to follow for instance the route discovery procedure. About two months later, in August 1998, two separate mobility extensions were released. These extensions had everything that we wanted from an extension, so we decided to use one of them. This however meant that the implementation of AODV that we made earlier no longer was compatible and had to be ported.

[2] Object Tool Command Language
[3] Network simulator 2.1b3

4.2 Mobility extension

There currently exist two mobility extensions to ns. These are:

- Wireless mobility extension developed by the CMU Monarch projects [30].
- Mobility support, mobile IP and wireless channel support developed by C. Perkins at Sun Microsystems [18].

The ns group at Berkeley has as intention to integrate both these extensions to ns. This work is however not complete yet. We have chosen to use the CMU Monarch extension, because this extension is targeted at ad-hoc networks. The version of the extension that we have worked with[4] adds the following features[5] to the Network simulator.

Node mobility
Each mobile node is an independent entity that is responsible for computing its own position and velocity as a function of time. Nodes move around according to a movement pattern specified at the beginning of the simulation.

Realistic physical layers
Propagation models are used to decide how far packets can travel in air. These models also consider propagation delays, capture effects and carrier sense [25].

MAC 802.11
An implementation of the IEEE 802.11 Media Access Protocol (MAC) [9] protocol was included in the extension. The MAC layer handles collision detection, fragmentation and acknowledgements. This protocol may also be used to detect transmission errors. 802.11 is a CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) protocol. It avoids collisions by checking the channel before using it. If the channel is free, it can start sending, if not, it must wait a random amount of time before checking again. For each retry an exponential backoff algorithm will be used. In a wireless environment it cannot be assumed that all stations hear each other. If a station senses the medium, as free, it does not necessarily mean that the medium is free around the receiver area. This problem is known as the hidden terminal problem and to overcome these problems the Collision Avoidance mechanism together with a positive acknowledgement scheme is used. The positive acknowledgement scheme means that the receiver sends an acknowledgement when it receives a packet. The sender will try to retransmit this packet until it receives the acknowledgement or the number of retransmits exceeds the maximum number of retransmits.

802.11 also support power saving and security. Power saving allows packets to be buffered even if the system is asleep. Security is provided by an algorithm called Wired Equivalent Privacy (WEP). It supports authentication and encryption. WEP is a Pseudo Random Number Generator (PRNG) and is based on RSAs RC4.

One of the most important features of 802.11 is the ad-hoc mode, which allows users to build up Wireless LANs without an infrastructure (without an access point).

Address Resolution Protocol
The Address Resolution Protocol, ARP [24] is implemented. ARP translates IP-addresses to hardware MAC addresses. This takes place before the packets are sent down to the MAC layer.

Ad-hockey
Ad-hockey is an application that makes it possible to visualize the mobile nodes as they move around and send/receives packets. Ad-hockey can also be used as a scenario generator tool to create the input files necessary for the simulations. This is done, by positioning nodes in a specified area. Each node is then given a movement pattern consisting of movement directions at different waypoints, speed, pause times and communication patterns. Screenshots of ad-hockey can be seen in Appendix C.

[4] Version 1.0.0-beta, released in the middle of August.
[5] At the end of November 1998, the CMU Monarch projects released version 1.1.0 of the extension. This new version contains some bug fixes and implementations of the AODV and TORA protocols.

Radio network interfaces
This is a model of the hardware that actually transmits the packet onto the channel with a certain power and modulation scheme [25].

Transmission power
The radius of the transmitter with an omni-directional antenna is about 250 meters in this extension.

Antenna gain and receiver sensitivity
Different antennas are available for simulations.

Ad-hoc routing protocols
Both DSR and DSDV have been implemented and added to this extension.

4.2.1 Shared media

The extension is based on a shared media model (Ethernet in the air). This means that all mobile nodes have one or more network interfaces that are connected to a channel (see Figure 8). A channel represents a particular radio frequency with a particular modulation and coding scheme. Channels are orthogonal, meaning that packets sent on one channel do not interfere with the transmission and reception of packets on another channel. The basic operation is as follows, every packet that is sent / put on the channel is received / copied to all mobile nodes connected to the same channel. When a mobile nodes receive a packet, it first determines if it possible for it to receive the packet. This is determined by the radio propagation model, based on the transmitter range, the distance that the packet has traveled and the amount of bit errors.

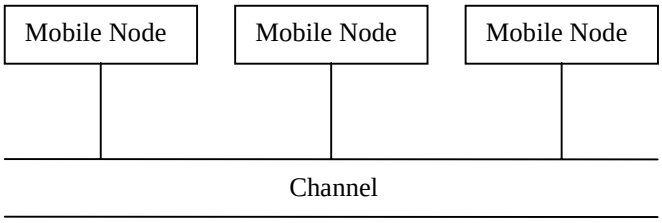


Figure 8: Shared media model.

4.2.2 Mobile node

Each mobile node (Figure 9) makes use of a routing agent for the purpose of calculating routes to other nodes in the ad-hoc network. Packets are sent from the application and are received by the routing agent. The agent decides a path that the packet must travel in order to reach its destination and stamps it with this information. It then sends the packet down to the link layer. The link layer level uses an Address Resolution Protocol (ARP) to decide the hardware addresses of neighboring nodes and map IP addresses to their correct interfaces. When this information is known, the packet is sent down to the interface queue and awaits a signal from the Multiple Access Control (MAC) protocol. When the MAC layer decides it is ok to send it onto the channel, it fetches the packet from the queue and hands it over to the network interface which in turn sends the packet onto the radio channel. This packet is copied and is delivered to all network interfaces at the time at which the first bit of the packet would begin arriving at the interface in a physical system. Each network interface stamps the packet with the receiving interfaces properties and then invokes the propagation model.

The propagation model uses the transmit and receive stamps to determine the power with which the interface will receive the packet. The receiving network interfaces then use their properties to determine if they actually successfully received the packet, and sends it to the MAC layer if appropriate. If the MAC layer receives the packet error- and collision- free, it passes the packet to the mobiles entry point. From there it reaches a demultiplexer, which decides if the packet should be forwarded again, or if it has reached its destination node. If the destination node is reached, the packet is sent to a port demultiplexer, which decides to what application the packet should be delivered. If the packet should be forwarded again the routing agent will be called and the procedure will be repeated.

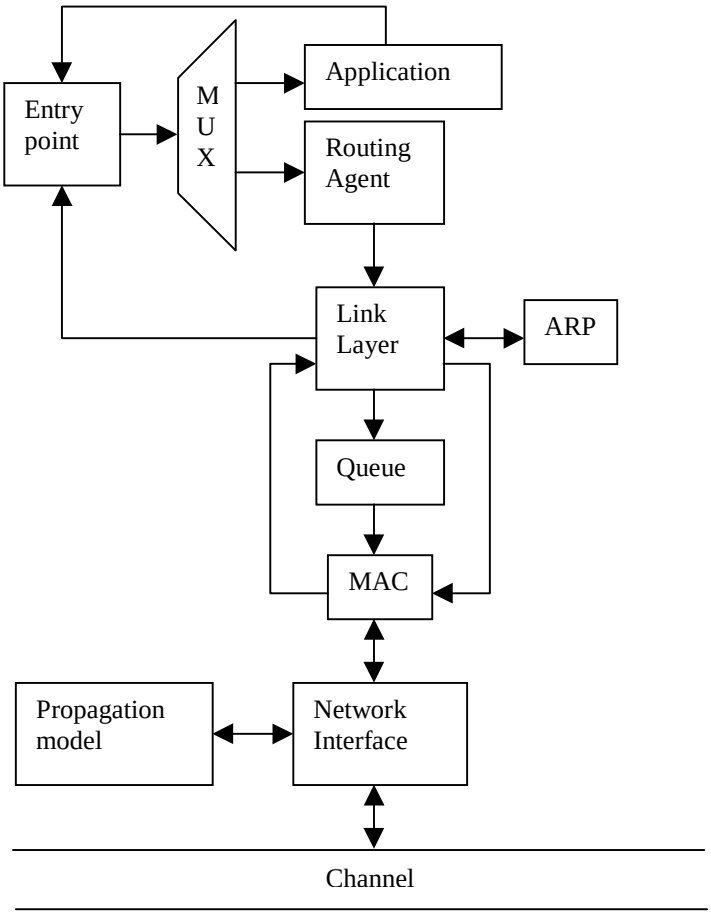


Figure 9: A mobile node.

4.3 Simulation overview

A typical simulation with ns and the mobility extension is shown in Figure 10. Basically it consists of generating the following input files to ns:

- A scenario file that describes the movement pattern of the nodes.
- A communication file that describes the traffic in the network.

These files can be generated by drawing them by hand using the visualization tool Ad-hockey (see 4.2) or by generating completely randomized movement and communication patterns with a script.

These files are then used for the simulation and as a result from this, a trace file is generated as output. Prior to the simulation, the parameters that are going to be traced during the simulation must be selected. The trace file can then be scanned and analyzed for the various parameters that we want to measure. This can be used as data for plots with for instance Gnuplot. The trace file can also be used to visualize the simulation run with either Ad-hockey or Network animator.

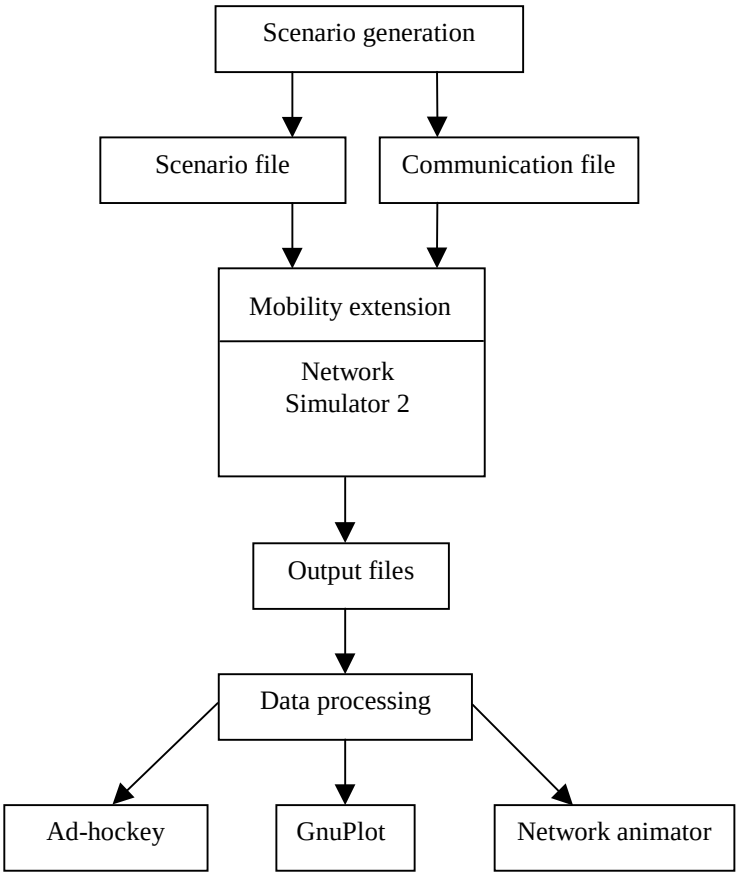


Figure 10: Simulation overview.

4.4 Modifications

To be able to use ns for the simulations, we had to do some modifications. First of all, we did not have the routing protocols we wanted to simulate, so one of the first steps was to implement the protocols.

4.4.1 AODV

We have implemented the AODV protocol (for more details, see appendix B). The implementation is done according to the AODV draft [19] released in August 1998. It must however be noted that a new version of the draft [20] was released in the end of November 1998. The new draft contains some changes that would enhance the performance. These changes that affect the unicast routing part is primarily:

- Reduced or complete elimination of hello messages.
- Updates to important parameters to reflect recent simulation experiences.

To be able to test how the hello messages and link layer support affects the behavior of the protocol we have implemented three versions:

- AODV with only IP-based hello messages
- AODV with only Link Layer notification of broken links
- AODV with both IP-based hello messages and Link layer notification of broken links

The implementation of the different versions has some major differences that will affect the performance. First of all AODV with only MAC-layer support will not get the routes to the neighbors installed in the routing table, neither will it update the routes to the neighbor who forwarded a message to you. Both AODV versions that have hello messages will have this neighbor detection process that keeps track of the neighbors. This means that the protocols with this feature will have more information in the

routing tables. Without this support, buffering of the packets may be necessary while a request is sent out in search for a node that could be a neighbor. It must however be noted that the removal of hello messages somewhat changes the behavior of the AODV protocol. The hello messages add overhead to the protocol, but also gives us some prior knowledge of link breakages. Removing the hello messages makes the protocol completely on-demand, broken links can only be detected when actually sending something on the broken link.

The DSR implementation that was included in the mobility extension used a sendbuffer that buffered all packets that the application sent while the routing protocol searched for a route. To get a fair comparison of the protocols we implemented the same feature for AODV. This buffer can hold 64 packets and packets are allowed to stay in the buffer for 8 seconds.

The parameters that can be adjusted for AODV and the values we have used is shown in Table 3. Some of these parameters are very important and affects the performance of the protocol in drastic ways. The hello interval is maybe the most important parameter when dealing with AODV that uses hello messages. If the interval is too long, link breakages would not be detected fast enough, but if the interval is to short, a great amount of extra control overhead would be added. Most of the parameters in Table 3 are obvious. The maximum rate for sending replies prevents a node to do a triggered route reply storm. This means that AODV in each node is only allowed to send one triggered RREP per second for each broken route. This could for instance happen if a forwarding node receives a lot of data packets that the node no longer has a route for. In this case the node should only send a triggered RREP, as a response to the first data packet and if the node keeps receiving data packets after that, a triggered RREP is only allowed to be sent once per second.

Table 3: Constants used in the AODV implementation.

Parameter	**Value**
Hello interval	1,5 s
Active route timeout	300 s
Route reply lifetime	300 s
Allowed hello loss	2
Request retries	3
Time between retransmitted requests	3 s
Time to hold packets awaiting routes	8 s
Maximum rate for sending replies for a route	1/s

4.4.2 DSR

The DSR implementation that came with the extension uses promiscuous mode (i.e. eavesdropping), which means that the protocol learns information from packets that it overhears. The question is how realistic this is in a real environment. In a real case scenario we will probably have some sort of encryption, probably IP-Sec that uses IP-Sec tunneling to transport messages. We have made some small change to DSR that makes it possible to turn the eavesdropping feature on and off. The parameters that are configurable for DSR are shown in Table 4. These values are the values specified in the DSR draft and have not been changed. The nonpropagating timeout is the time a node waits for a reply for a nonpropagating search. A nonpropagating search is a request that first goes to the neighbors. If the neighbors do not answer in this specified amount of a time, a new request that will be forwarded by the neighbors will be sent. The sendbuffer in the DSR can hold 64 packets and the packets are allowed to stay in the buffer for 30 seconds

Table 4: Constants used in the DSR implementation.

Parameter	**Value**
Time between retransmitted requests	500 ms
Size of source route header carrying n addresses	4n + 4 bytes
Timeout for nonpropagating search	30 ms
Time to hold packets awaiting routes	30 s
Maximum rate for sending replies for a route	1/s

4.4.3 DSDV

The extension also included an implementation of the DSDV protocol. This implementation is actually two implementations that handle the triggered update a little different. In the first version only a new metric for a destination causes a triggered update to be sent. In the second version, a new sequence number for a destination causes a triggered update to be sent. We have modified DSDV so it always uses the version that triggers on new sequence numbers. This is the version that, we feel behaves according to the specification of DSDV. The parameters for DSDV are shown in Table 5 and are as specified in the DSDV paper [22].

Table 5: Constants used in the DSDV implementation.

Parameter	Value
Periodic route update interval	15 s
Periodic updates missed before link declared broken	3
Route advertisement aggregation time	1 s
Maximum packets buffered per node per destination	5

4.4.4 Flooding

We have implemented a simple flooding protocol that simply floods all user data packets to all nodes in the network. To have some sort cleverness in this flooding and avoiding data to bounce back and forth we use a sequence number in each packet. This sequence number is incremented for each new packet. Each node keeps track of (source IP, sequence number) for all destinations and does not process a packet if the packet has a sequence number smaller than the stored sequence number. The idea was to do the simulations on the flooding protocol and compare the results with the results for the routing protocols. After some initial simulations on flooding this plan was abandoned. The simulations took too long to complete. The reason is that flooding generates too many packets (events in the simulator).

4.4.5 The simulator

To the actual simulator (ns + extension) we have added some new features to allow us to make the wanted measurements.

Obstacles

The visualization/scenario generator tool, Ad-hockey, allows the user to place obstacles (lines and boxes) into the scenario. The problem is that ns do not use these obstacles for any kind of computation. Two nodes can communicate, even through a wall. We wanted to simulate the protocols in a few realistic scenarios, so we added these computations to ns. The calculations consisted of two parts. The first part was to store all obstacles in a database that we later could use when calculating the intersection points and the second part was to extend the propagation model with the actual computations. The computations merely consisted of deciding if there existed an intersection point between the straight line from the sending node to the receiving node and any obstacle in the database. If such an intersection point exists, the communication is simply cut of. No fading of the signal, reflections etc is taken into consideration. The model is therefore very simple. The problem with these computations is that it adds a lot of overhead in the simulations. The simulations will take significantly longer time to complete. The extra computation must be done for all packets. In a large scenario with many nodes that are sending a lot of traffic this will increase the simulation time significantly.

Version management

To allow us to test different versions of one protocol simultaneous, we have added a version control to ns. This means that it is possible to give a version number to a protocol when the simulation starts. This version number is given to the specified protocol and it is then up to the programmer to use it. We currently use this feature with both AODV and DSR. The different versions are:

- AODV 1 = AODV with only hello messages.
- AODV 2 = AODV with only MAC-layer feedback.
- AODV 3 = AODV with both hello messages and MAC-layer feedback.
- DSR 1 = DSR with eavesdropping.
- DSR 2 = DSR without eavesdropping.

5 Simulation study

The protocols that we have simulated are DSDV, AODV and DSR. DSDV is only used to get a comparison of how much better/worse the MANET protocols are than an ordinary proactive protocol.

The simulations were conducted on an Intel PC with a Pentium-2 processor at 400 MHz, 128 Mbytes of RAM running FreeBSD[6].

5.1 Measurements

Before we go into the actual simulations, we will discuss which parameters [5] that are interesting to measure when studying routing protocols in an ad-hoc network. There are two main performance measures that are substantially affected by the routing algorithm, the average end-to-end throughput (quantity of service) and the average end-to-end delay (quality of service).

5.1.1 Quantitative metrics

The measurements that we have conducted can be seen from two angels: externally and internally. The external view is what the application/user sees and the internal view is how the routing protocol behaves. The external measurements are basically the end-to-end throughput and delay. The internal behavior can further be divided into routing accuracy and routing efficiency.

- Routing Efficiency: How much of the sent data is actually delivered to the destination? How much routing overhead is required to find routes?
- Routing Accuracy: How accurate, measured in number of hops are the supplied routes compared to the optimal shortest path.

5.1.2 Parameters

The metrics has to be measured against some parameter that describes the characteristic behavior of an ad-hoc network and can be varied in a controlled way. The parameters that we have chosen to simulate with are:

- Mobility, which probably is one of the most important characteristics of an ad-hoc network. This will affect the dynamic topology, links will go up and down.
- Offered network load. The load that we actually offer the network. This can be characterized by three parameters: packet size, number of connections and the rate that we are sending the packets with.
- Network size (number of nodes, the size of the area that the nodes are moving within). The network size basically determines the connectivity. Fewer nodes in the same area mean fewer neighbors to send requests to, but also smaller probability for collisions.

5.1.3 Mobility

Because mobility is an important metric when evaluating ad-hoc networks we need some definition of mobility. There exist many definitions of mobility. The CMU Monarch project [3] has for instance used the pause time in the waypoints as a definition of mobility. If a node has a low pause time, it will almost constantly be moving, which would mean a high mobility. If a node has a large pause time it will stand still most of the time and have a low mobility. We did not think that this mobility definition was good enough, because even if the pause time is low and all nodes are constantly moving, they could all be moving with a very slow speed in the same area.

We have defined mobility a little differently. Our definition is based on the relative movement of the nodes. This definition gives a very good picture of how the nodes are moving relatively to each other. The definition is as follows:

[6] FreeBSD 2.2.6

If several nodes move for a certain time, then the mobility is the average change in distance between all nodes over that period of time. This time is the simulation time T.

Mobility is a function of both the speed and the movement pattern. It is calculated with a certain sampling rate. During the simulations, we have used 0.1 seconds as sampling rate. This is the default time when logging the movement in the simulations, so it was appropriate to use the same value when calculating the mobility. Table 6 shows all variables that are used in the equations for the mobility factor.

Table 6: Mobility variables.

Variable name	**Description**
$dist(n_x, n_y)_t$	the distance between node x and node y at time t
n	number of nodes
i	Index
$A_x(t)$	Average distance for node x to all other nodes at time t
M_x	Average mobility for node x relative to all other nodes during the entire simulation time
T	Simulation time
Δt	Granularity, simulation step
Mob	Mobility for entire scenario

First of all, the average distance from each node to all other nodes has to be calculated. This has to be done at times t = 0, t = 0+X, t = 0+2X, ..., t = simulation time. For the node x at time t the formula is:

$$A_x(t) = \frac{\sum_{i=1}^{n} dist(n_x, n_i)}{n-1} \qquad (5.1)$$

After that, with the use of (5.1), the average mobility for that particular node has to be calculated. This is the average change in distance during a whole simulation. The mobility for node x is:

$$M_x = \frac{\sum_{t=0}^{T-\Delta t} \left| \left(A_x(t) - A_x(t+\Delta t) \right) \right|}{T-\Delta t} \qquad (5.2)$$

Finally, the mobility for the whole scenario is the sum of the mobility for all nodes (5.2) divided with the number of nodes:

$$Mob = \frac{\sum_{i=1}^{n} M_i}{n} \qquad (5.3)$$

The unit for the mobility factor (5.3) is m/s. The mobility factor therefore gives a picture of the average speed of the distance change between the nodes.

Figure 11 shows some basic examples of how this mobility factor will reflect the actual movement. If the nodes are standing still, this will of course lead to a mobility of 0, but this would also be the case when the nodes relative movement is zero, for example when the nodes are moving in parallel with the same speed. It is only when the nodes have a movement relative to each other that the mobility factor will be greater than zero.

Our mobility definition reflects how the mobility affects the dynamic topology, without considering obstacles or surroundings.

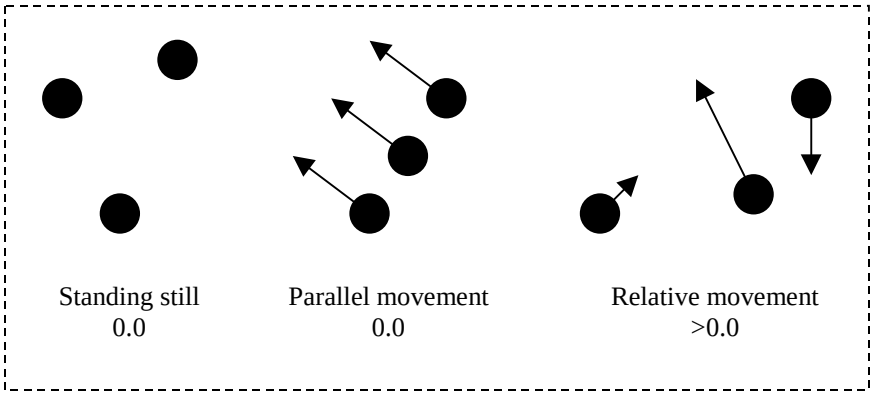


Figure 11: Example of mobility.

The reason for choosing mobility as a parameter in the simulations is first of all that the mobility is one of the most important characteristics of an ad-hoc network. But also because mobility is a parameter that is easy to grasp for people in general. Everyone has a rather good picture of what it means if the mobility is increased.

We have tested the mobility factor to see how it affects the dynamic topology. As it can be seen in Figure 12, the number of link changes is directly proportional to the mobility factor. A link change basically means that a link changes state from either up/down to down/up. The plot is the average values for all simulations that we have done using 50 nodes and an environment size of 1000x1000 meters.

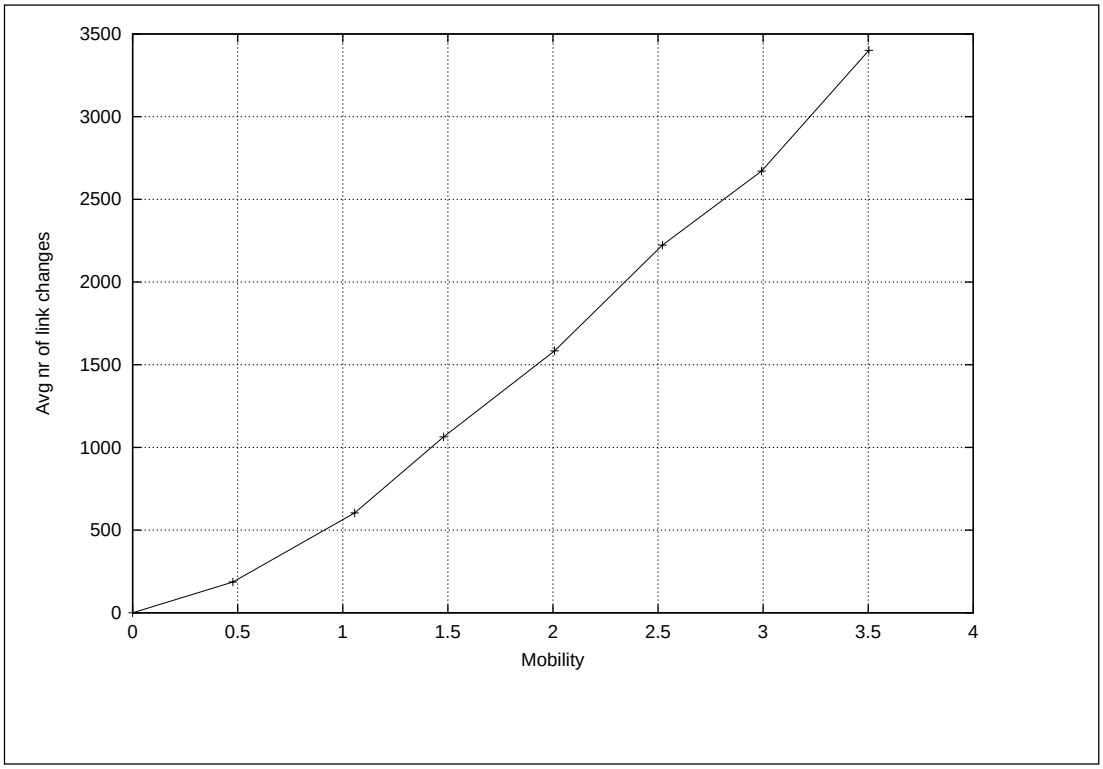


Figure 12: Relation between the number of link changes and mobility.

5.2 Simulation setup

In this chapter we will describe how the simulations were done. We have done 4 different types of simulations:

- Mobility simulations: We vary the mobility to see how it affects the different metrics that we are measuring.
- Offered load simulations: We vary the load that we offer the network, to see how the protocols behave when for instance the load is high.
- Network size simulations: We vary the number of nodes in the network.
- Realistic simulations: A few realistic scenarios were designed. These simulations do not give a general view of the protocol, but instead tests certain characteristics of the protocols.

Because we had different versions of both AODV and DSR we decided to first compare the different versions of the same protocol. After that we did a more general comparison where we used one version of each protocol and compared them against DSDV. The comparisons made are therefore:

- Comparison of AODV with only hello messages, AODV with only link layer support and ADOV with both hello messages and link layer support.
- Comparison of DSR with and without eavesdropping.
- Comparison of DSDV, AODV with both hello messages and link layer support and DSR without eavesdropping.

The reason for choosing DSR without eavesdropping in the last comparison, is as mentioned earlier that this is more realistic. Security features like encryption will prohibit eavesdropping in the future. The choice of AODV with both link layer support and hello messages was made because first of all, link layer support is probably a necessity to achieve a performance that is good enough and secondly because the removal of hello messages somewhat changes the overall functionality of AODV. Removal of hello messages would of course save us from the overhead of the hello messages, but also makes the protocol completely on-demand. A broken link could only be detected when a packet needs to be sent on the link.

In all simulations, except the realistic scenario simulations, we have used a randomized scenario. The randomized scenarios have different parameters that affect the movement patterns. The parameters that can be changed are:

- Maximum speed: Every time a speed is going to be randomized it is randomized in the interval [0,maximum speed].
- Number of nodes: This was constant during the simulations. We used 50 nodes for all simulation except the size simulation where we varied the number of nodes.
- Environment size: Determines the size of the environment. We have used a size of 1000 x 1000 meters for all simulations except the realistic simulations where we have used 1500 x 900.
- Simulation time: The time for which the simulations will be run at. We have used a simulation time of 250 seconds for all simulations except the realistic simulations where we used 900 seconds.
- Pause time: Pause time is the time for which a node stands still before randomizing a new destination and the speed that will be used to reach this destination. We have used a pause time for 1 second in all simulations.

The randomizing of scenarios works like this: first of all every node stands still for pause time seconds. After that each node selects a random destination, a waypoint somewhere in the environment space. Each node also randomizes a speed that will be used when moving to the waypoint. This speed is randomized uniformly in the interval 0 to maximum speed. Every time a node reaches a waypoint, this procedure will be repeated.

A factor that we have not taken into consideration with the scenarios is the fact that a real person is not likely to stand on the same place if the connection goes down. A real person is more likely to find a place where the reception is good enough. The system would be to complex if this factor were included also.

We have assumed bi-directional links during all our simulations, i.e. the links work equally well in both directions. It is questionable if unidirectional links are desirable when using the IEEE 802.11 MAC protocol, because bi-directional links are necessary if 802.11 acknowledgements are supposed to be used.

5.3 Mobility simulations

5.3.1 Setup

The simulations where we varied the mobility where done by randomizing scenario files. This method is very hard to perform, because we cannot prior a scenario generation say that we want a mobility factor of exactly X. Instead we used the maximum speed parameter to control the scenario. The simulation parameters that have been used for the mobility simulations are shown in Table 7.

Table 7: Parameters used during mobility simulations.

Parameter	**Value**
Transmitter range	250 m
Bandwidth	2 Mbit
Simulation time	250 s
Number of nodes	50
Pause time	1 s
Environment size	1000x1000 m
Traffic type	Constant Bit Rate
Packet rate	5 packets/s
Packet size	64 byte
Number of flows	15

The scenario is a very crucial part of the simulation. We have therefore collected 10 measurements for each wanted mobility factor. The mobility factors that we simulated on are: 0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, and 3.5. Because of the hard part of getting scenarios that are precise we have used an interval of ± 0.1 for the above mentioned mobility factors. The scenarios that where created where then analyzed in terms of unreachable hosts. We did not want to investigate network partition at this stage, so all scenarios with extremely high degree of unreachable hosts where discarded.

By increasing the maximum speed in the scenario generation, the mobility will also increase. A mobility factor of 3.5 approximately corresponds to a maximum speed of 20 m/s. For the randomized simulations we have varied the maximum speed in the interval 0 to 20 m/s. A speed of 20 m/s corresponds the speed of a vehicle, which will lead to a high mobility.

We used the same communication pattern for all mobility simulations. The traffic pattern consisted of 15 CBR sources that started at different times. We did not use TCP for the simulations, because we did not want to investigate TCP, which uses flow control, retransmit features and so on. We wanted to get a general view of how the routing protocol behaves. The communication pattern was randomly created. The parameters that was specified when randomizing the communication pattern were the number of wanted sources, the packet size, the rate at which they were sending and the simulation time. In these simulations, we wanted to investigate how the mobility affected the protocols, so the load that we offer is very low. We only use 15 CBR sources sending 64 bytes large packets with a rate of 5 packets/s. The bandwidth of the links are 2 Mbit.

5.3.2 Fraction of received packets

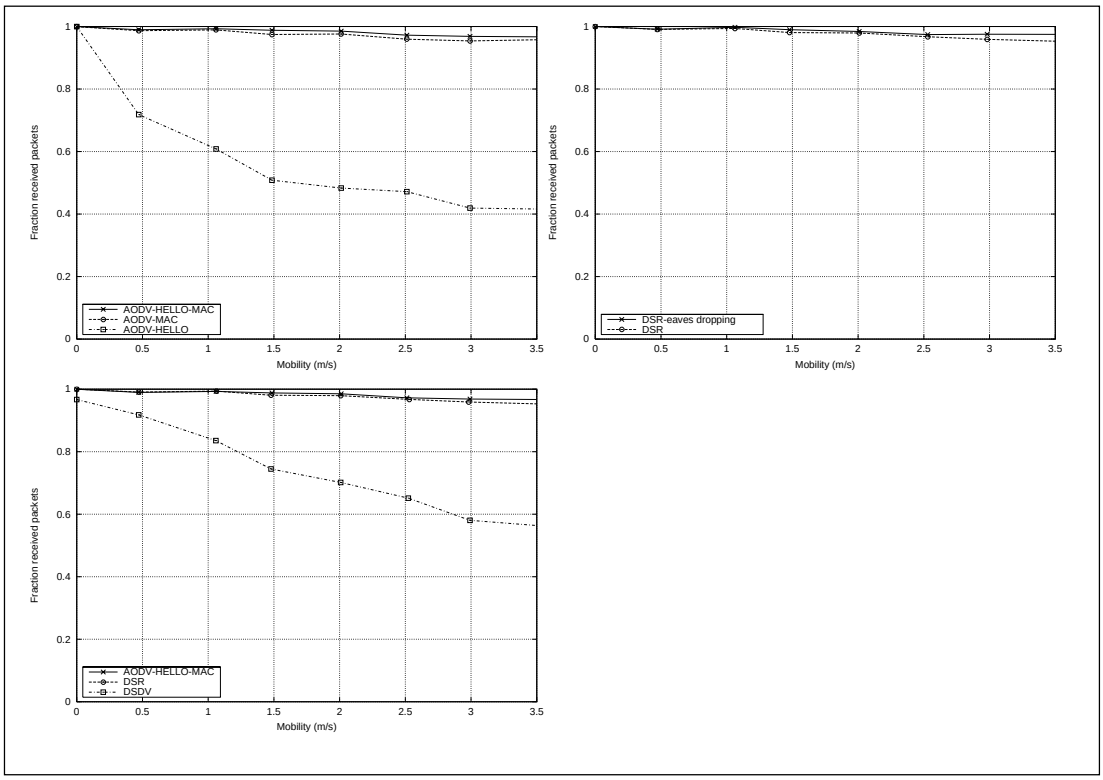


Figure 13: Mobility simulations - fraction of received packets.

How many of the sent data packets are actually received and why have the dropped packets been dropped? Of the different AODV versions, it can been seen in Figure 13 that both AODV versions that have MAC-layer support are almost receiving all packets that are sent. AODV with both hello messages and MAC-layer support is slightly better than the version with only MAC-layer support. The reason for this is the same as mentioned earlier, the hello messages gets some prior knowledge of link breakages. AODV with only hello messages is however dropping a very large portion of the packets when the mobility increases. This large fraction of dropped packets is of course not acceptable and the reason for these drops has to do with the interval of the hello messages. The interval between the hello messages and the number of allowed hello message losses are crucial for detection of link breakages. If the interval is decreased, link breakages are detected earlier, but it would also mean that the control overhead in the network increases. The issue here is to try to find optimal values for these parameters. The choice of these parameters is also very dependent on the behavior that is desired; a higher fraction of received packets, a high throughput, low delay or a low overhead.

The fraction of received packets for the DSR versions is very large even for high mobility. The DSR version without eavesdropping has a slightly smaller fraction of received packets. This difference is however so small that it is negligible. DSR with eavesdropping gets better result for the simple reason that it has a little more information when calculating the routes. A reason for the higher fraction received packets for DSR compared to AODV is that DSR allows packets to stay in the send buffer for as long as 30 seconds, AODV only 8 seconds (our implementation). It must however be noted that the AODV draft [19] does not specify how long a packet is allowed to stay in the sendbuffer.

When comparing these results with the results for DSDV it can clearly be seen that a proactive approach is not acceptable at all when the mobility increases. The fraction received packets drastically goes down to 56-57 %. This value is however for a very large mobility factor (vehicles). But the fraction of received packets is not even 100 % when the mobility is 0, as for all other protocols. The reason for this is that packets are sent before the routing tables have had enough time to converge and the packets are dropped.

The main reasons for dropping packets are that the protocol is sending packets on a broken route that it thinks is valid and that packet in the buffers are dropped because of congestion and timeouts. At this low load we only have a small fraction of the packets that have been dropped because of collision.

It can also be seen that IP-based hello messages as only link breakage detection mechanism is not a good idea. The results are very poor, even DSDV have slightly better results. Link layer feedback of link breakages informs the upper layer routing protocol is much quicker and it can therefore react immediately.

5.3.3 End-to-end delay

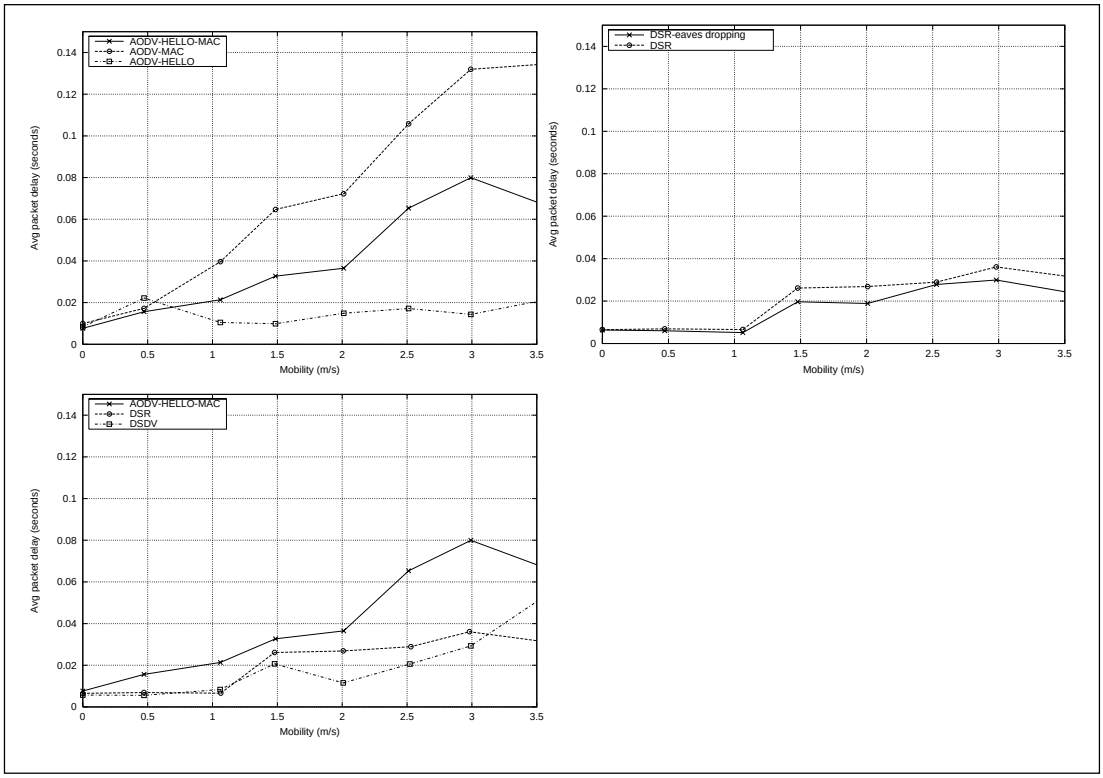


Figure 14: Mobility simulations - delay.

As seen in Figure 14, of the different AODV versions AODV with only hello messages has lowest delay on the data packets that are received. The reason for this is not that it finds routes faster or that the routes are shorter or more optimal, instead AODV with only hello messages is the AODV version that gets significantly fewest packets through the network. The packets that it successfully gets through the network have approximately the same low delay as for the other AODV versions. The difference is that the other AODV versions have a portion of packets that have a higher delay (has been in a buffer a long time and still gets the packets through the network). This affects the average delay, which becomes larger. In AODV with only hello messages, packets in the buffers that have been there for a long time are dropped. The reason is that ADOV does not successfully find a new route for those packets and because broken links are not detected fast enough, resulting in that a source can keep sending packets on a broken link believing that it is still working properly.

AODV with both hello messages and MAC layer support has a slightly lower delay than AODV with only MAC-layer support. The reason for this is that as mentioned earlier, AODV with only MAC-layer support makes the protocol completely on-demand, it only detects link breakages when actually trying to send packets. Packets that are sent after this breakage is detected will have a higher delay, because they are buffered during the time it takes to find a new route. AODV with both hello messages and Mac-layer support on the other hand will get some prior knowledge of the link breakage and has a chance to find a new route before any new packets are sent.

Both DSR versions show a tendency to get higher delay when mobility is increased. The turning point comes at a mobility factor of approximately 1.0. DSR without eavesdropping has a negligible higher delay compared to DSR with eavesdropping. DSDV is the protocol that seems to have lowest delay in these results. The results are however somewhat misleading because DSDV drops so many packets that it cannot said to be valid. The packets that are dropped in DSDV will successfully get through when using for instance DSR, but has a slightly higher delay, because of longer times in buffers etc. These higher delay packets will make the average delay higher for DSR. The same can be said for AODV with only hello messages. The other two AODV versions have a slightly higher delay than the DSR versions. This has probably to do with the source routing concept of DSR. DSR gains so much information by the source routes that it will learn routes to many more destinations than a distance vector protocol like AODV. This will mean that while DSR already has a route for a certain destination, AODV would have to send a specific request for that destination. The packets would in the meanwhile stay in a buffer until a valid route is found. This will take some time and will therefore increase the average delay.

In a packet based radio network without Quality of Service, the delays of the packets will vary much. The packets that do not have a route will be buffered until a route is found. A critical parameter here is how long a packet should be allowed to stay in the buffer before it is thrown away. If the packets are allowed to be in the buffer for a long time, the following situation could happen: A packet is sent, but there does not exist any route to that destination so the packet is buffered and a route request is sent. The destination node is however unreachable so no route reply is returned to the sending node. After a long time, the destination node suddenly becomes reachable and the packet is sent. This packet will have a very large delay. Should this situation be allowed to happen? Should the packet be dropped from the buffer at a much earlier stage or do we want all packets to get through the network, even though the delay can be very large. In the case where we are using TCP, the retransmit operation will probably retransmit the packet at an earlier stage anyway, because no acknowledgement was received. The allowed time for packets to stay in the sendbuffer in DSR is 30 seconds and only 8 seconds for AODV. If a packet is received 30 seconds after it was sent, this will increase the average delay to some degree.

5.3.4 End-to-end throughput

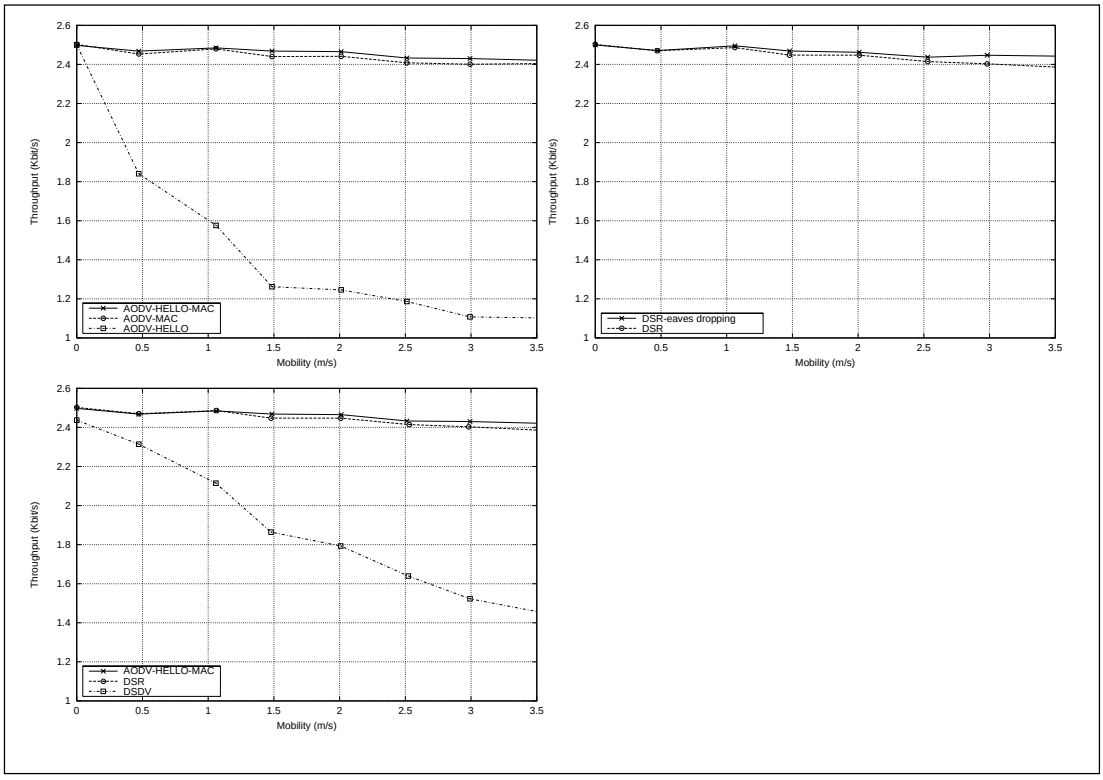


Figure 15: Mobility simulations - throughput.

Figure 15 shows the throughput curves for the different protocols with a packet size of 64 bytes. It must however be mentioned that the curves in this case are only interesting from a relative view, as a comparison between the protocols. We have not tried to maximize the throughput, we have only tried to determine the relative difference in throughput for the different protocols with respect to the mobility factor and the specific load that we have used.

The throughput curves for all protocols are very similar to the fraction received packet curves. This is logical because large packet drops will of course mean lower throughput.

Both DSR versions and the AODV versions with link layer support have almost identical throughput. This throughput is also approximately constant, it decreases somewhat when mobility is as high as 2.5-3.5. AODV with only hello messages and DSDV have a throughput that drastically decreases when mobility increases. AODV with only hello show a very poor result. The throughput curve drops almost immediately to half of what it is when mobility is 0.

5.3.5 Overhead

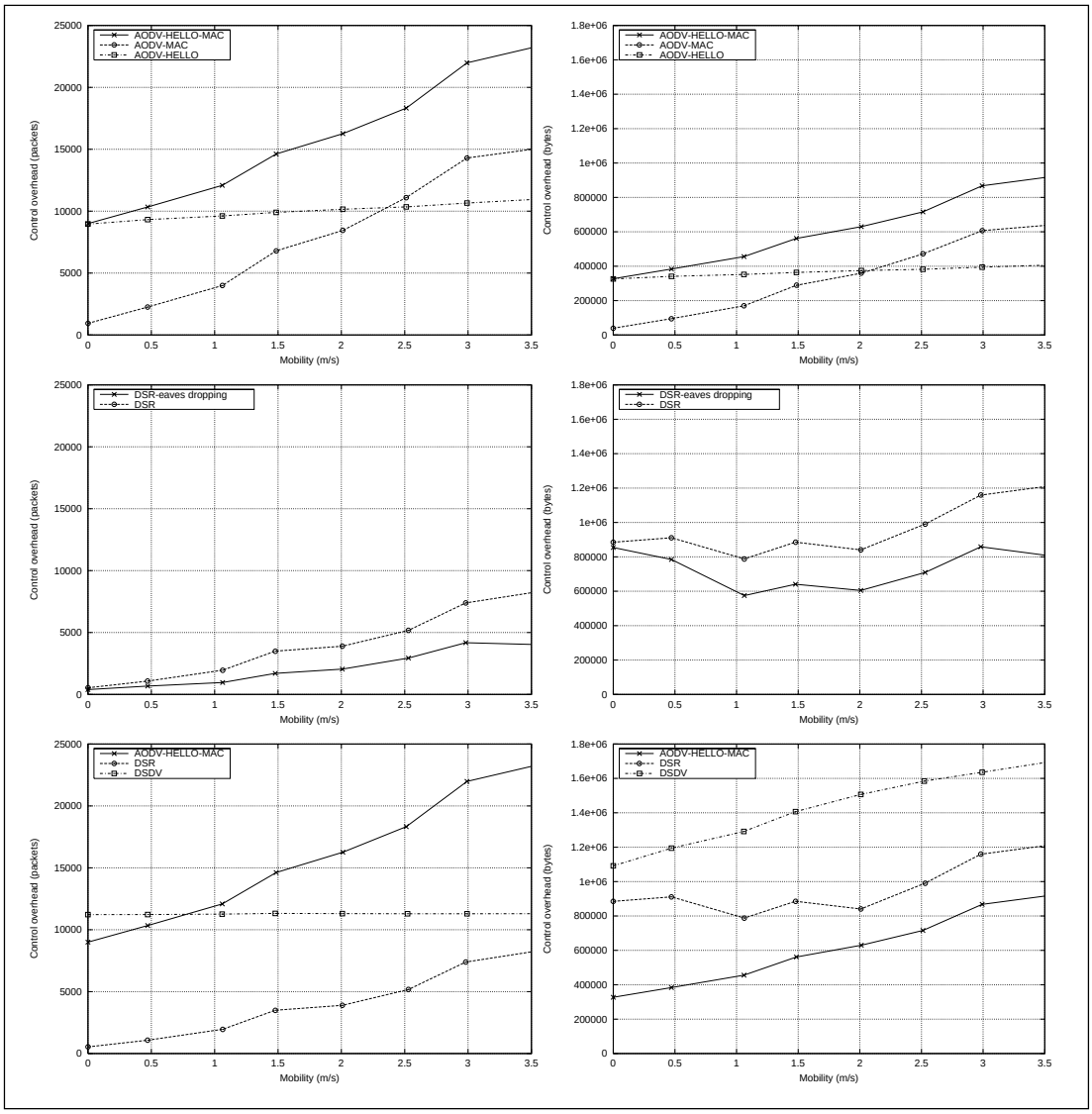


Figure 16: Mobility simulations - overhead.

Because the routing protocol need to send control information to achieve the task of finding routes, it is interesting to see how much control information that is actually sent for each protocol. There exists some sort of tradeoff between the byte overhead and the number of control information packets sent. A large byte

overhead would of course mean a larger part of wasted bandwidth. Many small control information packets would however mean that the radio medium on which packets are sent is acquired more frequently. This can be quite costly in terms of power and network utilization.

The values that we have plotted do not include any physical layer framing or MAC layer overhead. We have only looked at the overhead at the IP-level. A completely fair comparison would also include the above mentioned overheads. We have chosen not to include these for the simple reason that the MAC layer can be different in a real life implementation. We wanted to investigate the overhead generally, not the overhead that is dependent on particularly the IEEE 802.11 MAC protocol.

The results are shown in Figure 16. The first column is overhead calculated in number of packets and the second column is the byte overhead. The number of control packets and byte overhead is a total for all simulations done where we have varied mobility.

Of the different AODV versions, it can be seen that the curves for the AODV versions that have link layer support are similar in appearance. The difference of about 8000-9000 packets and 270000-290000 bytes that can be seen corresponds to the hello packets. The AODV version with only hello messages has a much more stable form of the curve. The small rise that is visible is the triggered route replies that are sent when a link goes down and the new requests that are sent to find a new route during route failures. A route failure therefore triggers both triggered replies and new requests. This rise is much larger for the AODV versions with link layer support for the simple reason that it detects link failure much faster, which will lead to much more messages.

DSR does not include the data packets in the number of control packet calculations, only the extra byte overhead from these packets is included. Worth noting when observing the DSR versions is that the DSR version that does not use eavesdropping has approximately the double amount of control overhead counted in number of messages and about 400000 bytes more of byte overhead than the DSR version with eavesdropping at the highest mobility 3.5. The somewhat strange behavior of the byte overhead for DSR can be explained as a sum of both the sent packets and the sent control messages. As mobility increases, fewer packets will get through the network. Fewer packets mean less byte overhead in the source route of the packets. Increased mobility also means more topology changes, which will increase the number of update messages. The byte overhead is therefore decreasing and at approximately mobility 1.5 the increase of control messages will cause the byte overhead to increase.

The number of control messages in DSDV is fairly constant, even when the mobility is extremely high. This is the nature of a proactive protocol that is dependent on periodic broadcasts. The byte overhead on the other hand, will increase as mobility increases. The reason for this is that the amount of information sent in each update message will be larger as the amount of link changes increases.

5.3.6 Optimal path

One internal aspect of the routing protocol is the routing accuracy, e.g. how good the actual routes are compared to the optimal routes. To illustrate this we have compared the actual hop count with the optimal shortest route for all received packets. We have then, for each protocol, calculated how large fraction the received packets that have been routed through the optimal route, a route with hopcount that is one larger than the optimal, a route with hopcount that is two larger than the optimal and so on. The result is shown in Table 8. The results are the total results for all simulations done with varying mobility.

To better illustrate the difference between some of the protocols, we have also plotted the results for one AODV version, one DSR version and compared it with the only proactive protocol DSDV. It can clearly be seen in Figure 17 that DSDV is the protocol that has the highest degree of optimality, almost 90 % of the received packets have been routed with optimal hop count. The AODV and DSR versions are almost identical. AODV is slightly better. This difference is however so small that it can neglected. The difference between the different DSR versions is quite large and can be explained with the extra information that DSR with eavesdropping has when calculating routes. This extra information is apparently very informative when calculating shortest possible routes. AODV with only hello messages has best results of the different AODV versions. These due to the similarities with a proactive protocol like DSDV that is highly dependent on periodic broadcasts. Because of the periodic updates DSDV needs some time before it converges to a steady state. This happens when we have high movement with a lot of topology changes. Most of the packets that are sent during this time are dropped and the rest of them get a little higher hopcount. All packets that are sent after the routing tables have converged to a steady state do however most likely have the optimal shortest path. AODV with only hello messages have a similar behavior. Link breakages are dependent on the hello messages. This will mean that high movement and frequent topology changes will lead to many packet

drops during this time. The packet that successfully gets through the network have done this during times when the network is somewhat stabile

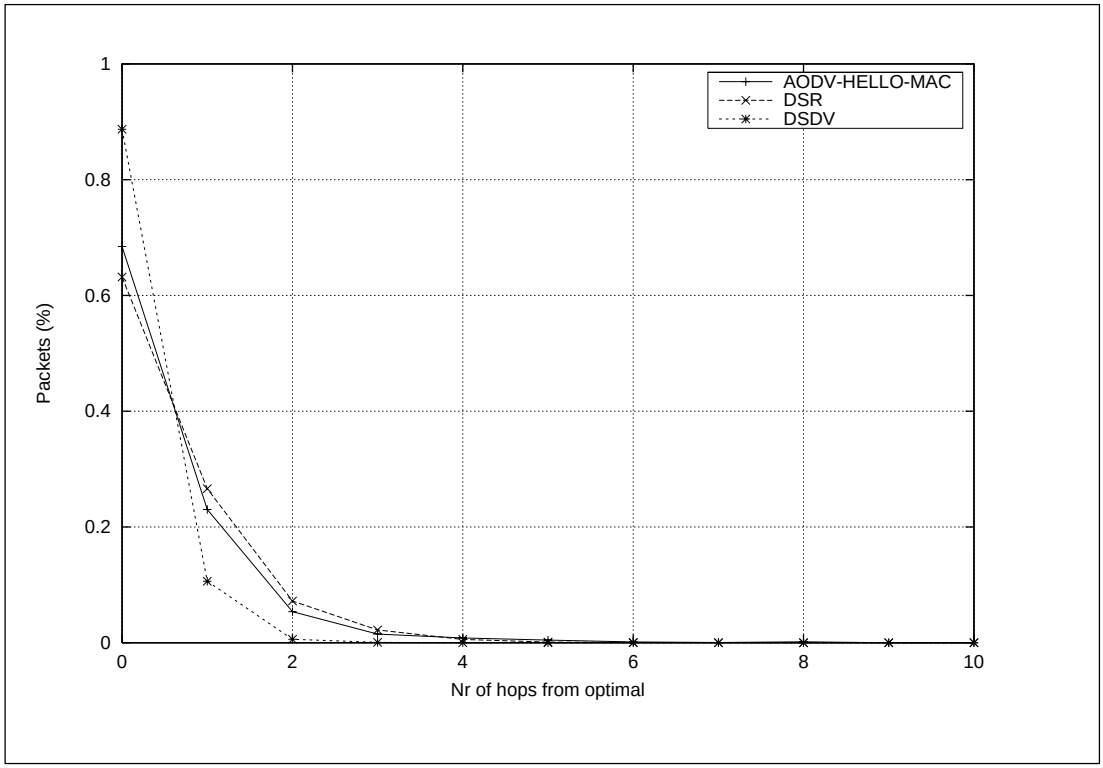


Figure 17: Mobility simulations - optimal path difference.

Because both DSR versions and the AODV versions with link layer support gets significantly more packets through the network when the number of topology changes is large, they will have a little lower fraction of packets that have a optimal route. In times of frequent movement it is easy to get a route that at first is optimal and then the moment later is 1-2 hops longer than another route that become the optimal route. The protocols will keep sending packets on the route that at first was optimal.

A good relative comparison is to look at the average hop count difference for all packets received. DSDV has the smallest average with a hopcount at only 0.13 hops from the optimal path.

Table 8: Optimal path difference for all protocols.

	Number of hops from optimal path (% packets)								
Protocol	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**Average**
AODV 1	79.1%	14.8%	3.68%	1.08%	0.30%	0.55%	0.06%	0.06%	0.292170
AODV 2	55.9%	29.9%	9.04%	2.62%	1.09%	0.56%	0.20%	0.38%	0.682837
AODV 3	68.4%	23.0%	5.37%	1.48%	0.84%	0.47%	0.14%	0.04%	0.459423
DSR 1	81.1%	15.3%	2.73%	0.54%	0.13%	0.08%	0.01%	0.02%	0.234429
DSR 2	63.2%	26.6%	7.21%	2.21%	0.56%	0.13%	0.05%	0.03%	0.512004
DSDV	88.7%	10.6%	0.59%	0.06%	0.00%	0.00%	0.00%	0.00%	0.126577

5.3.7 Summary mobility simulations

The protocols that have link layer support for link breakage detection will be much more stable. The fraction of packets received for these protocols is almost constant at 95 % even when mobility increases. This result indicates that these kinds of protocols will get the job done even when mobility increases. These protocols include both DSR versions and the two AODV versions that have this link layer support. Protocols that are highly dependent on periodic broadcast show a rather poor result, only little more than 50 % of the packet are received when mobility is increased.

Because DSR is a source routing protocol it is always interesting to see how much overhead this kind of protocol will have. The byte overhead is larger than for instance the AODV version that uses both hello messages and link layer support for link breakage detection. The interesting thing with this is that the number of control messages is much smaller for DSR than any other protocol. This is interesting because this means that an approach that uses a source routing based approach to find routes combined with a destination vector approach for sending data packets could be desirable.

5.4 Offered load simulations

We have used these protocols for these simulations:

- AODV with both hello messages and MAC link layer support.
- DSR without eavesdropping
- DSDV

We only used the more realistic version of both AODV and DSR, for the same reason as mentioned in the previous section.

5.4.1 Setup

The offered load simulations where done by varying the load that we offer the network. We had mainly three parameters to adjust the offered load:

- Packet size
- Number of CBR flows
- Rate at which the flows are sending

The mobility simulations that we have done used a packet size of 64 bytes, a rate of 5 packets/s and 15 CBR flows. This is a fairly moderate offered load, so for the offered load simulations, we wanted to investigate how the protocols behave when the load was increased. We could increase the packet size or the number of CBR flows, but the parameter that best describes the load is the rate at which we are sending. By only increasing the rate for the CBR flows, the load for each flow will increase. This also gives some hints of how large the throughput can be. We have used four different offered load cases:

- 5 packets / second (same as the mobility simulations)
- 10 packets / second
- 15 packets / second
- 20 packets / second

The packet size was held constant at 64 bytes and the number of flows at 15. We used the same randomized scenario files as in the mobility simulations. The same communication file was also used, with the exception that we changed the rate for the CBR sources. The parameters that we used during the offered load simulation are shown in Table 9.

Table 9: Parameters used during offered load simulations.

Parameter	Value
Transmitter range	250 m
Bandwidth	2 Mbit
Simulation time	250 s
Number of nodes	50
Pause time	1 s
Environment size	1000x1000 m
Traffic type	Constant Bit Rate
Packet rate	5 packets/s
Packet size	64 byte
Number of flows	15

5.4.2 Fraction of received packets

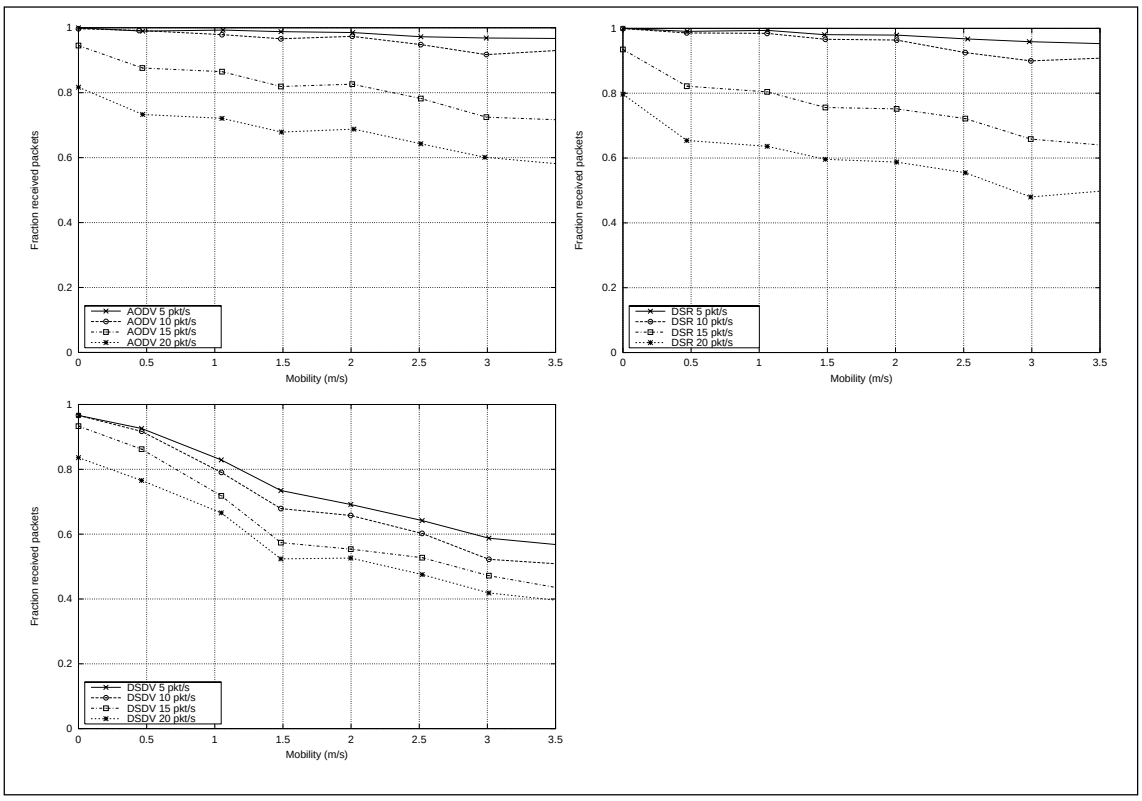


Figure 18: Offered load simulations - fraction of received packets.

At only 5 packets/s both AODV and DSR are rather constant, the fraction of received packets is only decreasing slightly when mobility increases (Figure 18). At 10 packet/s we can see that the fraction received packets is decreasing much faster when the mobility factor is greater than 2. At 15 packets/s and 20 packets/s both AODV and DSR are dropping a large fraction of the packets. At the highest mobility and a rate of 20 packets/s, only 50-60 % of the sent packets are received. The reason is more collisions in the air and congestion in buffers. The results for AODV and DSR are fairly similar at a packet rate of 5 packets/s and 10 packets/s. At data rates of 15 packets/s and 20 packets/s, AODV shows a better result than DSR. At these rates the protocols are however dropping a large fraction of the packets, even at a mobility factor of 0. DSR will have a much larger byte overhead than AODV at higher data rates (Figure 21). The reason for this is the source route in each data packet. This also increases the load on the network and causes more packets to be dropped; thus AODV will get more packets through the network.

DSDV is dropping a large fraction of the packets already at the lowest data rate 5 packets/s. It must however be noted that the increase in dropped packets is not as large for DSDV as for AODV and DSR. At the highest data rate, DSDV is almost as good as DSR.

5.4.3 End-to-end delay

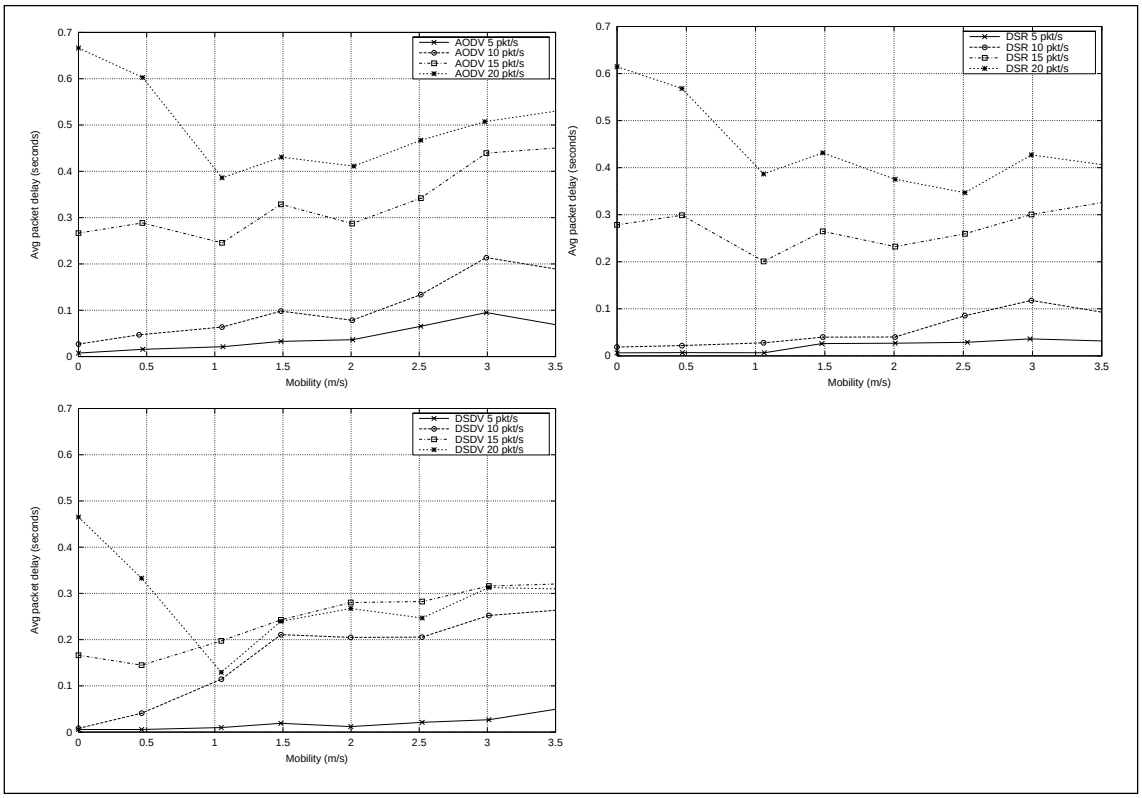


Figure 19: Offered load simulations - average delay.

The delay is also affected by high rate of CBR packets (Figure 19). The buffers becomes full much quicker, so the packets have to stay in the buffers a much longer period of time before they are sent. This can clearly be seen at the highest rate 20 packets/s. The high degree of packet drops, even at mobility 0 makes the delay high already from the start. DSR has a much lower delay compared to AODV. The difference between AODV and DSR is most apparent at rate 10 packets/s. DSDV has the lowest delay of them all. This is however an effect from the large fraction of packet drops that DSDV has, compared to DSR and AODV. The increase in delay for DSDV also comes from the increased time that the packets must stay in the buffers.

The high delay at a mobility factor of 0-1 and a data rate of 20 packets/s that can be seen for all protocols is a result of the extremely high data rate and the low mobility. The high data rate will fill up the buffers very quickly. The low mobility will mean that already found routes are valid for a much longer time period. This means that found routes can be used for more packets. Even the packets that have stayed in the buffer for a long time have a chance to get through. When mobility increases, more routes will become invalid and new requests are necessary. While the requests are propagating the network in search for a new route, buffers will get full and packets are dropped. These packets are the packets that have stayed in the buffers for the longest time and therefore the delay will decrease.

The increase in mobility actually results in a load balancing of the traffic between the nodes; hot spots are “removed” due to mobility.

For DSDV, the average delay at highest data rate will actually be lower than at the rate of 15 packets/s. This is a little strange but has probably something to do with the fact that DSDV only uses a buffer that only has room for 5 packets per flow. At the rate of 15 packets/s and 20 packets/s, when mobility starts to get so high that the topology changes frequently, only 40-60 % of the packets gets through the network. These

topology changes means that the protocol needs more time to converge before the packets can be sent. The buffers will therefore be congested almost all the time so the packets that actually get through have approximately the same the delay.

5.4.4 End-to-end throughput

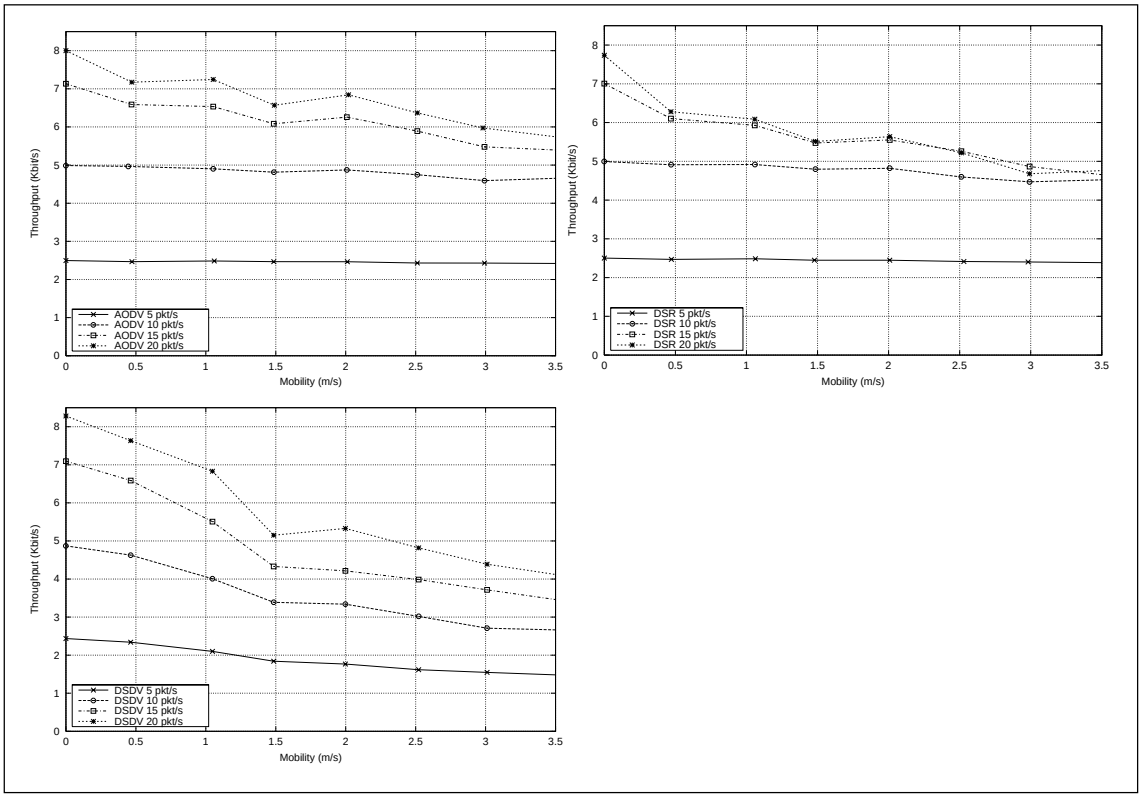


Figure 20: Offered load simulations - average throughput.

At low CBR rates the throughput of DSR and AODV is unaffected of the mobility (Figure 20), it stays constant at 2,5 Kbit/s. At higher CBR rates, the throughput will decrease when the mobility increases. This can already be seen at CBR rate 10 packet/s. The decrease at CBR rate 10 packets/s is however very small. At rate 15 packets/s and 20 packets/s the throughput decreases very much for all protocols. This is however an effect from the large fraction of dropped packets.

The result for AODV is slightly better than for DSR. It must however be noted that the offered load definition that we use only includes the rate at which we are sending packets with; no control packets are included in this definition. The same applies for the throughput, only the data packets are included in the calculations of throughput. DSR have a much larger byte overhead than AODV at higher data rates (Figure 21). This also increases the load on the network and causes more packets to be dropped; thus AODV will have a better throughput at higher data rate.

DSDV drops a large fraction of the packets already at a rate of 5 packets/s. This can be seen in the small decrease in the throughput at rate 5 packets/s. The throughput decreases more and more as the rate increases.

5.4.5 Overhead

In Figure 21, the difference between distance vector and source routing can clearly be seen. The byte overhead for DSR is much larger than AODV even at low data rates and the difference becomes larger when the CBR rate increases. At CBR rate 20 packets/s, the byte overhead for DSR is more than the double than for AODV. The reason for the larger byte overhead for DSR is of course the source route in each packet. The

number of control messages is though smaller for DSR. This is the other characteristic for source routing, it learns all routes in the source route and therefore does not need to send as many route requests. The reason for the increase in number of control packets is the MAC-layer support. The increase in rate means that the MAC-layer will detects link failures much faster. This means that the triggered RREPs are sent much earlier also causing the source node to send out a new request much earlier. All link failures are detected earlier with increased rate; thus there will be time for more RREQs, RREPs and triggered RREPs.

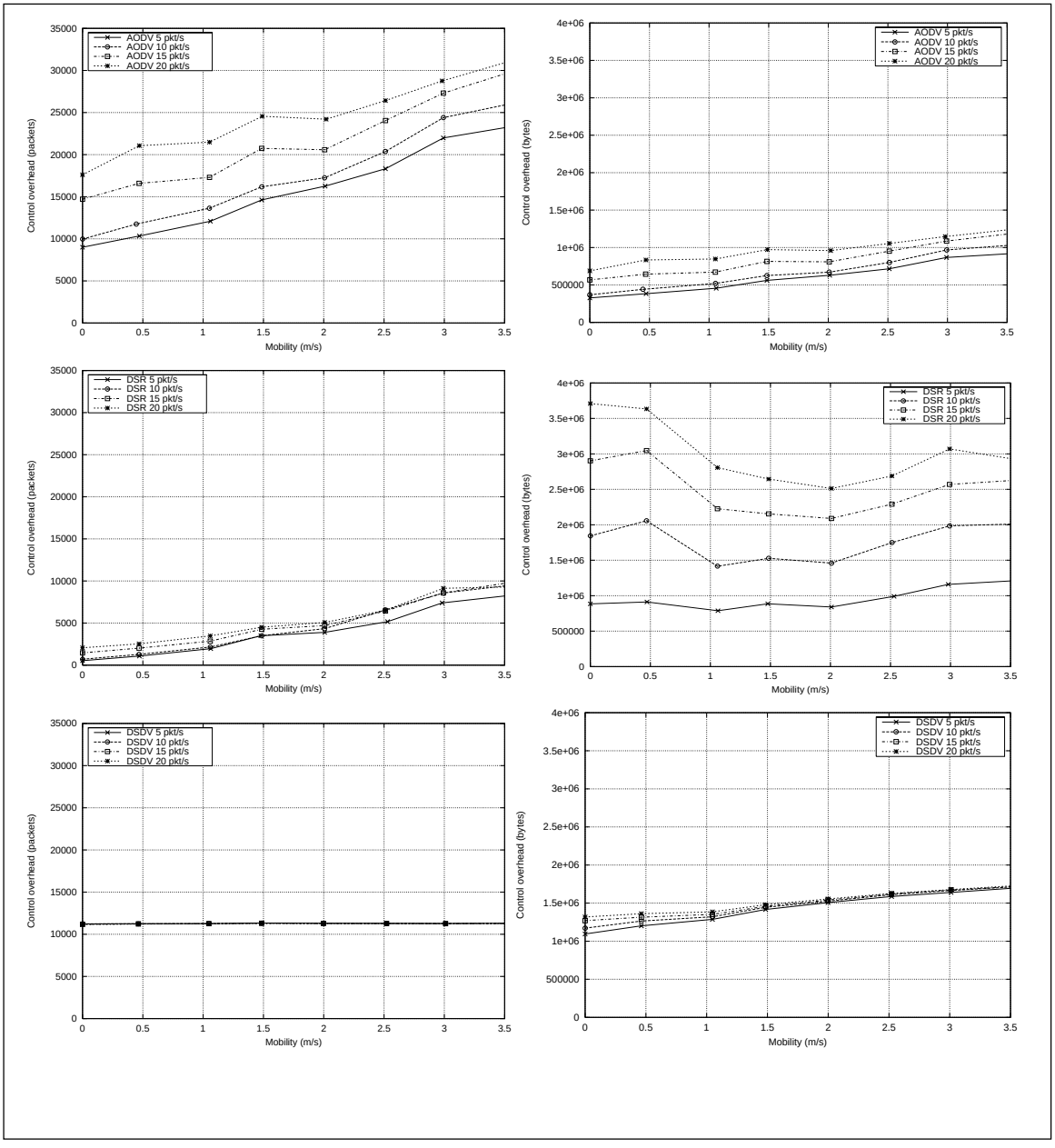


Figure 21: Offered load simulations - overhead.

The amount of control information in DSDV is not affected to any great extent by the data rate for the CBR packets. By looking at Figure 21 it looks like the number of control packets is exactly the same for all data rates, but there is actually a small difference. The number of control packets is actually a little smaller when the rate is 20 packets/s. This difference is about 80 packets. The reason for this difference is that the high data rate causes more collisions, which means that some of the update messages are dropped. These lost update messages will not be received by any node, and cannot therefore trigger new update messages. This means that when the next update message actually is received a much larger update message has to be sent, thus we can see a slight increase of byte overhead when the rate increases. But when the mobility increases more and more packets will be dropped. At the highest data rate, many of the update messages are dropped,

even the packets with a little more information. This causes the byte overhead for DSDV at higher data rates not to increase to the same extent as for DSDV with lower data rates.

5.4.6 Optimal path

Figure 22 illustrates the difference in hopcount from the optimal hopcount at a CBR rate of 20 packets/s. It is the result for all simulations done at this rate. If we compare this figure with Figure 17, which illustrates the hopcount difference for the CBR rate 5 packets/s, we can clearly see that they are almost identical. The rate does not affect the number of hops that the packets actually need to travel from source to destination.

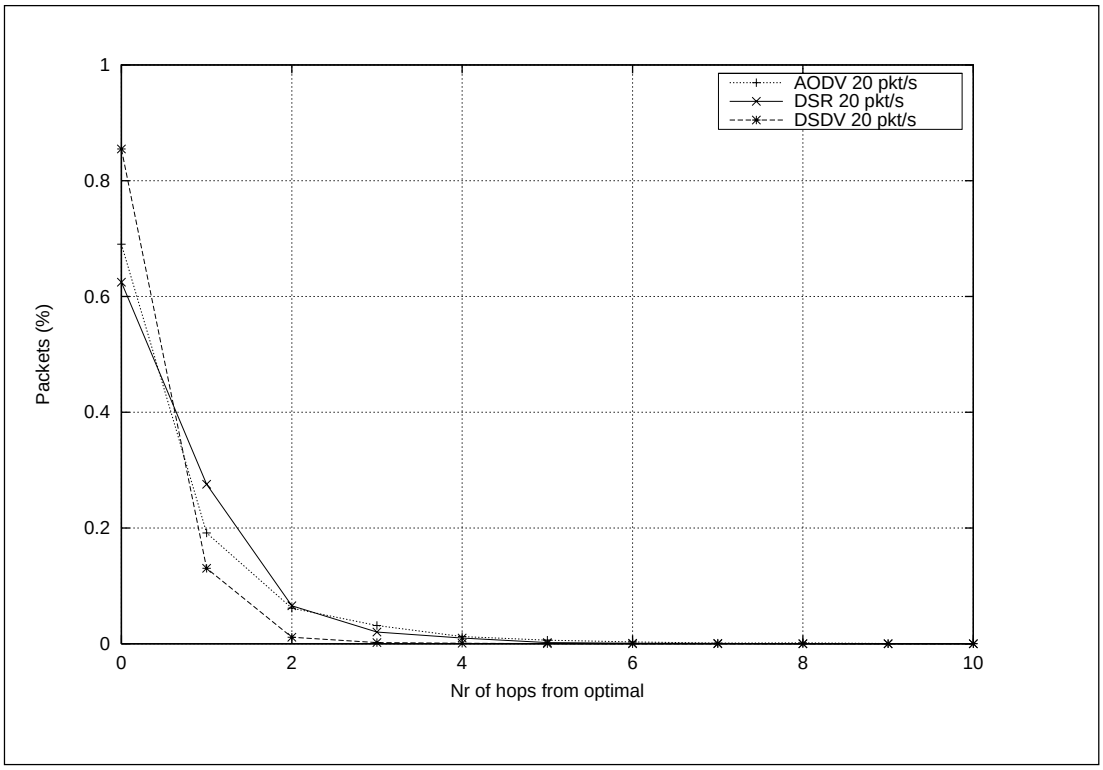


Figure 22: Offered load simulations – optimal path.

5.4.7 Summary offered load simulations

The performance of the protocols differs slightly during different network loads. The most apparent difference is the byte overhead. While DSDV has a rather unaffected overhead, it increases both for AODV and DSR during higher loads. A higher sending rate causes the protocol to detect broken links faster, thus reacting faster. This leads to a slight increase in control packets, which also affects the byte overhead. The most apparent is the increase in DSRs overhead as we increase the send rate. Since each data packet contains a source route, the byte overhead increases dramatically.

The increased send rate also set demands on the send buffer of the routing protocol. Congestion occurs and packets are dropped. The faster a routing protocol can find a route, the less time the packets have to spend in buffers, meaning a smaller probability of packet drops.

5.5 Network size simulations

We did simulations on some of the protocols and varied the number of nodes that participated in the network. We decreased the number of nodes to 35 and 25 nodes. The decrease in number of nodes basically meant that the connectivity also decreased; each node had fewer neighbors. The results from these simulations did not give any new information regarding the performance of the protocols. The relative difference between the protocols was the same.

Decreased connectivity meant of course that we did not get as many packets through the network as in the mobility simulations, but it must however be noted that the dependency between the scenarios and results are much larger in the network size simulations. The worst results for each protocol happened when the mobility was 0. This may sound strange, but the reason for the bad result when the nodes is standing still is the randomized scenarios. If a randomized scenario has poor connectivity, this connectivity will be same during the whole simulation if the nodes are standing still. The nodes are not moving and cannot therefore affect the connectivity. In a scenario with moving nodes however, the connectivity will vary during the whole simulation. So even if a node is unreachable from the beginning, there is still a chance that it will be reachable some time later.

5.6 Realistic scenarios

5.6.1 Setup

The randomized simulations we have done, gives a very good general picture of how the protocols behaves in respect to certain parameters, such as mobility, size and network load. This kind of simulations also has some problems:

- It is hard to identify situations in which the protocols fail or have problems
- It has no connection to a real life situation.
- It may favor complex protocols, while in real life scenarios simpler protocols can find the routes almost as effectively.

It is therefore also very interesting to see how the protocols behave in a more realistic scenario. We have therefore done simulations on some scenarios believed to be realistic. The realistic scenarios do not give a full picture of how the protocols behave generally. Instead they give some sense of weak points in the protocols. The three basic types of scenarios that we have done simulations on are:

- Conference type, with low movement factor.
- Event coverage type, with fairly large movement factor. Could for instance be reporters trying to interview politicians.
- Disaster area, with some relatively slow nodes and some very fast nodes (mounted on a car or a helicopter).

The environment size is 1500 x 900 meters for all realistic scenarios. This size is scaled according to the range of the transmitters. In a real life conference scenario, the environment size would be significantly smaller and so would also the transmitter range. The same thing would apply for the speed of the people moving around. The speed is also scaled to 10 m/s. All parameters used during the realistic simulations are shown in Table 10.

Table 10: Parameters used during realistic simulations.

Parameter	Value
Transmitter range	250 m
Bandwidth	2 Mbit
Simulation time	900 s
Number of nodes	50
Environment size	1500x900 m
Traffic type	Constant Bit Rate
Packet rate	4 packets/s
Packet size	512 byte
Speed of a human	1 m/s
Speed of a mobile node mounted on a vehicle	20 m/s

5.6.2 Conference

This scenario simulates 50 people that are attending a conference, seminar session or some similar activity. It involves communication between some of the people. Parameters specific for the conference scenario are shown in Table 11

Table 11: Parameters used during conference scenario.

Parameter	Value
Number of CBR sources	2
Number of receivers	6
Number of flows	6

The scenario is characterized by:

- Low mobility factor, 10 % of the nodes are moving during any period of time.
- Links are long lasting and involves many hops.
- The traffic is concentrated to a few nodes, typically only the speaker.
- Few obstacles which are far apart. Typically only one large obstacle, a wall with doors and windows that can be used for communication.
- Relatively large interference from other nodes, due to the concentration of transmitting nodes. This can in some cases lead to local congestion.

The scenario basically tests the protocols:

- Ability to respond to local changes for long links.
- Ability to cope with large concentration of traffic.
- Message overhead with low mobility factor.

Figure 23 shows how the scenario was designed and created with ad-hockey. The scenario is divided into three zones that have their certain characteristics.

- Zone 1 - Speaker zone: The speaker moves back and forth. This changes the closest neighbor in the audience.
- Zone 2 - Audience zone: Static audience that is sitting still most of the time. Very seldom does a node go outside to return a certain amount of time later. This will probably result in a link breakage of a long-lasting link.
- Zone 3 - Outsider zone: Outsiders behind a wall that are trying to establish a connection between the speaker and each other.

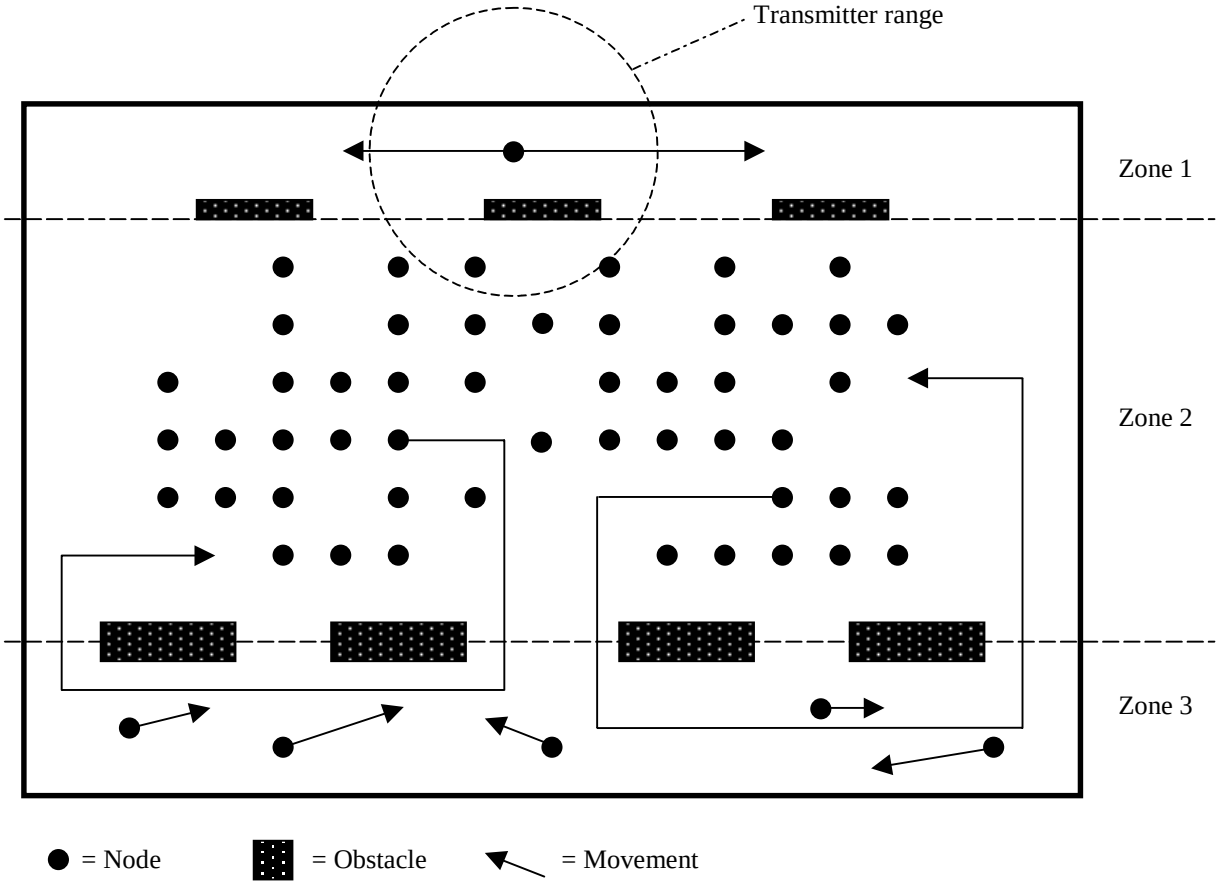


Figure 23: Conference scenario.

The results for this scenario are shown in Table 12 and Table 13. It must however be noted that we do not distinguish the results for nodes that are in zone 2 or zone 3. The calculated mobility factor for this scenario is very low. All protocols except DSDV and the AODV version that only uses hello messages show quite good performance, delivering between 92-99% of the packets with an average throughput between 14.8 – 15.7 Kbit/s. DSDV delivers only 75.6% of the packets with an average throughput of 12.1 Kbit/s and the AODV version with only hello messages delivers 89.3 % of the packets, with a throughput of 14.3 Kbit/s. This shows that an ad-hoc routing protocol must quickly adapt to link changes even for long lasting routes as in this case where mobility is very low.

Table 12: Conference simulation results.

	DSDV	**DSR - 1**	**DSR - 2**	**AODV - 1**	**AODV - 2**	**AODV - 3**
Mobility factor	0.0439350	0.0439350	0.0439350	0.0439350	0.0439350	0.0439350
Received	75.6%	97.1%	98.0%	89.3%	92.3%	94.0%
Throughput	12.1 Kbit/s	15.55 Kbit/s	15.7 Kbit/s	14.30 Kbit/s	14.79 Kbit/s	15.00 Kbit/s
Sent	21510	21510	21510	21510	21510	21510
Average delay	0.052 s	0.210 s	0.23 s	0.26 s	0.29 s	0.39 s
Dropped	5250	614	422	2298	1644	1376
Received packets	16260	20896	21088	19212	19866	20134
Packet overhead	44054	3129	4109	43881	14537	54677
Byte overhead	6406036	3689865	4093220	1660020	610884	2112716
Average hopcount	5.32	5.73	5.79	6.62	6.53	6.45

Both versions of DSR have the lowest packet overhead in this scenario, while all versions of AODV show the lowest byte overhead. The large byte overhead for DSR has to do with the fairly large amount of traffic and that the routes have an average hopcount that is as large as 5-6 hops.

As can be seen in Table 13. DSDV fails to deliver 3599 packets because of route failures. This is due to the slow response time when links go down and the time it takes to find new routes. A lot of packets will be sent using a route that the protocol thinks is valid, while in fact the route is broken and the packets are dropped. Also a lot of packets are dropped in the interface queue.

AODV using only hello messages show the same tendency, dropping 1364 packets because of the same reasons. It is however fewer packets than for DSDV. This has to do with the time between the HELLO messages. The less time between the messages, the faster can the protocol react to broken links and avoid dropping the packets.

In general, most of the packets are dropped in the sendbuffer. Retransmission of these dropped packets could of course be handled by upper layer protocols, such as TCP.

Table 13: Packet drops in conference scenario.

Cause for drop	DSDV	DSR – 1	DSR – 2	AODV –1	AODV – 2	AODV -3
Sendbuffer		519	303	894	1364	1096
Route failures	3599	30	98	1348	201	202
ARP	74	34	4	22	21	14
Interface queue	1567	18	0	4	10	0
End of simulation	10	13	17	30	48	64

5.6.3 Event coverage

This scenario simulates a group of 50 highly mobile people that are changing position quite frequently. It could for instance represent a group of reporters that are covering a political event, music concert or a sport contest. In real life it would be nearly impossible to establish a wired network between the reporters, but they must be in constant communication with each other, enabling a fast reaction. Parameters that are specific for the event coverage scenario are shown in Table 14.

Table 14: Parameters used during event coverage scenario.

Parameter	Value
Number of CBR sources	9
Number of receivers	45
Number of flows	45

The scenario is characterized by:

- Rather high mobility factor. Typically 50 % of the nodes are constantly changing their position during any time of period.
- Every now and then the nodes tend to cluster.
- Links involve a few hops and relatively short-lasting.
- Traffic is spread all over the place.
- Many obstacles. Nodes can usually only communicate with a few nodes. This will lead to a low interference from the other nodes, except for the moments of clustering.

The scenario tests the protocols:

- Ability to respond to fast link changes and fluctuating traffic.
- Message overhead with constant topology updates.

Figure 24 shows how the scenario was designed. 50 % of the nodes are moving randomly with a constant speed of 1 m/s. Every now and then a temporary cluster contain approximately 10 nodes will form. These clusters remain static for a certain amount of time.

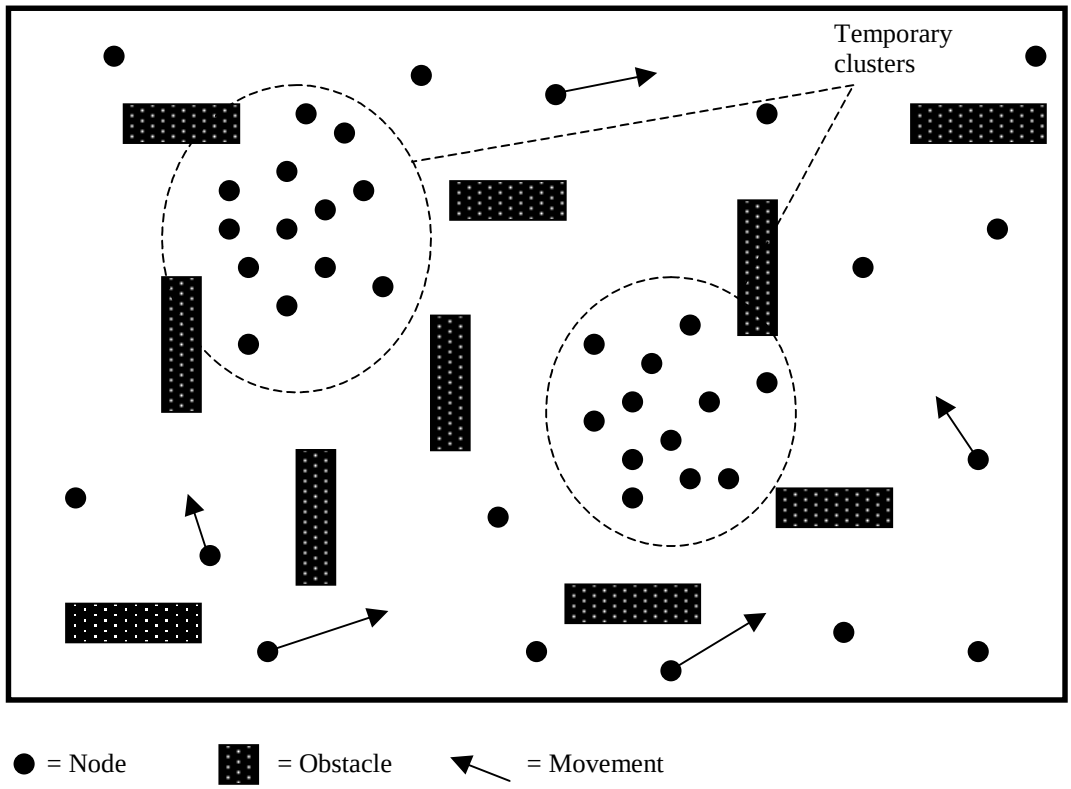


Figure 24: Event coverage scenario.

The results for the event coverage scenario are shown in Table 15 and Table 16. As Table 15 shows, the two protocols that has the lowest fraction of packets received are DSDV and the AODV version that entirely relies on feedback from link-layer. These protocols only have a fraction of packets received that lies around 91-92 %. For all other protocols/versions this value is at least 94 %. The mobility factor for the scenario is 0.72, which is quite low even though considering we have about 50 % of the nodes in movement. The nodes are however only moving with a speed of 1 m/s. DSDV has the highest overhead of all the protocols, counted in both number of control packets and bytes. DSR both with and without eavesdropping have a very low overhead. It must however be noted that the traffic load in this scenario is very low and most of the communication is taking place in a very small area, thus leading to a very low average hopcount of approximately 1.5 for all protocols.

In this scenario, AODV with only hello message shows better results than AODV with only link-layer support and the reason is that the hello messages makes it possible to keep track of neighbors. Most of the communication is taking place within clusters, so to know the neighbors will probably mean that we will not have to make as many requests.

The topology of the network is changing quite frequently. This causes a protocol like DSDV to have a very large overhead. The communication however, take mostly place within the clusters. This makes the job for the on-demand-based protocols much easier. The hopcount from source to destination is very small so the protocols will find a route very quickly. The protocols which are entirely on-demand will have a slightly

higher delay and the protocols that uses some sort of periodic messages (DSDV and the ADOV version that use hello messages) will have lower delay.

Table 15: Event coverage simulation results.

	DSDV	DSR – 1	DSR – 2	AODV – 1	AODV – 2	AODV – 3
Mobility factor	0.721611	0.721611	0.721611	0.721611	0.721611	0.721611
Received	91.4%	97.5%	97.7%	94.5%	92.2%	95.1%
Throughput	14.75 Kbit/s	15.68 Kbit/s	15.71 Kbit/s	15.33 Kbit/s	14.89 Kbit/s	15.36 Kbit/s
Sent	4500	4500	4500	4500	4500	4500
Average delay	0.075 s	0.138 s	0.140 s	0.024 s	0.214 s	0.015 s
Dropped	385	111	102	245	352	219
Received packets	4115	4389	4398	4255	4148	4281
Packet overhead	42415	1056	1354	31342	2722	31443
Byte overhead	10578764	141196	158420	1137752	117904	1142180
Average hopcount	1.46 hops	1.53 hops	1.57 hops	1.55 hops	1.75 hops	1,55 hops

Table 16 shows that most of the packets are dropped due to route failures and in those cases where there exists a sendbuffer, congestion/time-out in the sendbuffer. AODV that only uses link-layer support (AODV 2) has most of the dropped packets due to failed ARP requests and DSDV drops most of its packets at the interface queue.

Table 16: Packet drops in event coverage scenario.

Cause for drop	DSDV	DSR – 1	DSR - 2	AODV - 1	AODV - 2	AODV - 3
Sendbuffer		81	81	100	131	100
Route failures	102	21	19	83	32	15
ARP	48	9	1	61	178	104
Interface queue	235				10	
End of simulation			1	9	1	

5.6.4 Disaster area

This scenario represents some sort of disaster area in a region that lacks any sort of telecommunication infrastructure. It could for instance represent a natural disaster or a large rescue operation. Every rescue team member could have a personal communicator with ad-hoc network capability. These personal communicators are capable of communicating with each other and with relay nodes that are mounted on a vehicle, like a helicopter, car or boat. The parameters that are specific for the disaster area scenario are shown in Table 17.

Table 17: Parameters used during disaster area scenario.

Parameter	Value
Number of CBR sources	38
Number of receivers	87
Number of flows	87

The scenario is characterized by:

- A node movement, where approximately 95 % of the nodes are moving slightly, while the remaining 5 % are changing their position very often.
- Network partitioning now and then.
- Long and short lasting links that only are a few hops.
- Traffic that is spread all over the nodes.
- Many obstacles.
- Low interference from the other nodes.

The scenario tests the protocol:

- Ability to work with both slow and fast changing network topologies.
- Ability to cope with network partitioning.

Figure 25 shows how the scenario was designed. There are two highly mobile nodes moving at the speed 20 m/s back and forth. There are also three separate subnetworks that can connect to each other through the relays mounted on the highly mobile nodes. Within each subnetwork, the movement is randomized at speed 1 m/s.

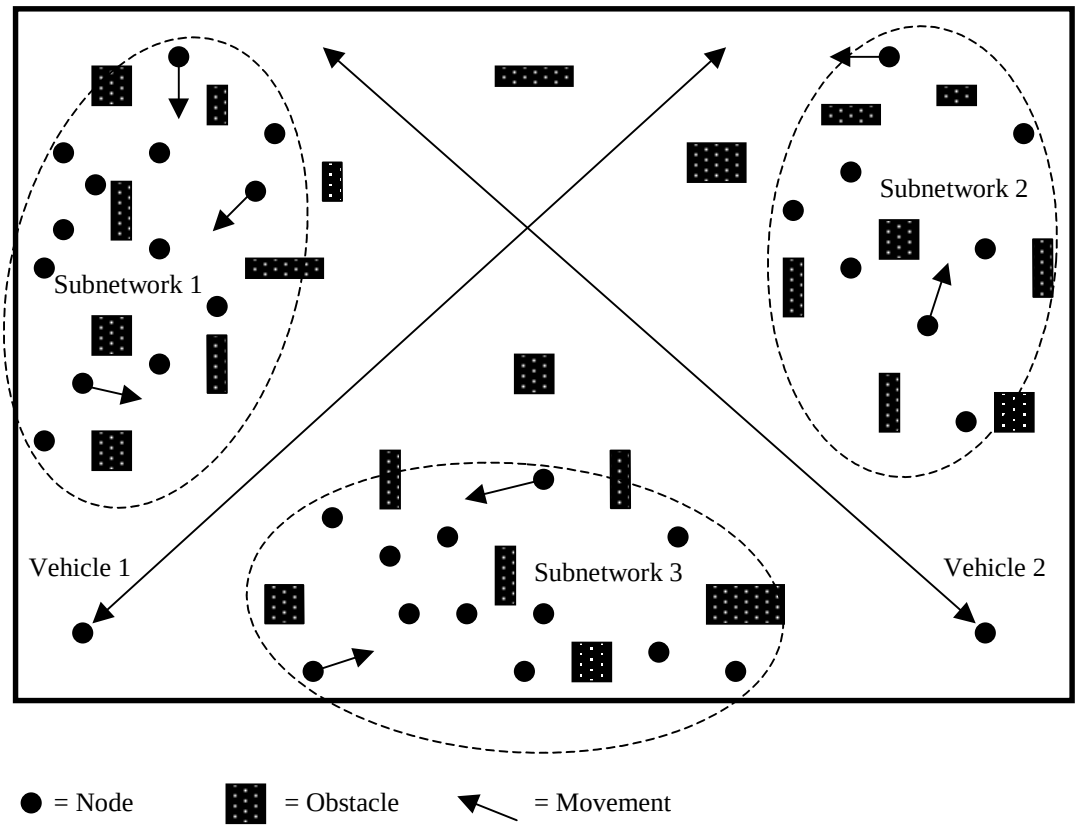


Figure 25: Disaster area scenario.

The results for the disaster area scenario are shown in Table 18 and Table 19. The calculated mobility factor for this scenario is quite high. Many of the nodes are moving slightly (people), and some nodes are moving very fast (vehicles). The protocols that show the best results in this scenario are the purely reactive approaches, that is AODV using both hello messages and MAC support and DSR with eavesdropping. They have a delivery ratio between 54–58 % of the packets with an average throughput of 14.1 – 14.8 Kbit/s. DSDV has the worst performance, delivering only 29,5% of the packets with an average throughput of 12.42 Kbit/s. A delivery ratio of only 58 % for the best protocol may sound like a terrible ratio, but because of network partitioning, it is hard for any protocol to successfully deliver more packets than that.

DSDV show the lowest delay, but this is due to the low delivery ratio. All versions of AODV keep an average delay between 0.89 – 1.05 seconds, while the DSR versions lie between 1.19 – 1.25 seconds. Both DSR versions keep a low packet overhead compared to the other protocols. The byte overhead is however twice as large for DSR compared to AODV.

As expected, DSDV uses the shortest paths with an average of 3.42 hops. The other protocols has an average hopcount of 5.0-5.2 hops.

Table 18: Disaster area simulation results.

	DSDV	**DSR – 1**	**DSR – 2**	**AODV – 1**	**AODV – 2**	**AODV – 3**
Mobility factor	1.154619	1.154619	1.154619	1.154619	1.154619	1.154619
Received	29.5%	58.0%	54.50%	48.0%	52.2%	54.0%
Throughput	12.42 Kbit/s	14.79 Kbit/s	14.43 Kbit/s	13.07 Kbit/s	13.49 Kbit/s	14.09 Kbit/s
Sent	29616	29616	29616	29616	29616	29616
Average delay	0.196 s	1.252 s	1.187s	0.888 s	1.052 s	0.988 s
Dropped	20867	12440	13464	15382	14161	13635
Received packets	8749	17176	16152	14234	15455	15981
Packet overhead	41402	27048	30692	61557	50686	77311
Byte overhead	6497476	4880992	5137148	2426164	2426164	3096228
Average hopcount	3.42 hops	5.01 hops	5.16 hops	5.23 hops	5.24 hops	5.26 hops

DSDV drops a lot of packets because of route failures. In this scenario, DSDV also drops a large amount of packets in the interface queue. One large cause for drops in the other protocols, lies in the sendbuffer. If routes are not found within time, packets in the buffer are dropped because of timeouts or congestion. Another large cause for packet drops is route failures.

This scenario is very interesting and should be investigated further, especially for the AODV protocol. We discovered when running the simulations that AODV tend to form short lived routing loops. Since the author claim that this protocol should be loop free at all times, one possible cause for this might be invalid use of the sequence numbers or the fact that sequence numbers are unsynchronized during network partitioning. Very few packets were dropped because of this (less than 5). Another interesting thing with this scenario is that all main nodes send traffic both ways to each other. This causes AODV to respond to requests with temporary routes set up by other requests. Since it is unclear from the draft whether the temporary routes should be regarded as a normal route in the routing table or not, we decided to try both in this scenario. By only allowing RREPs to use the temporary routes, and by not allowing a node to respond to a RREQ with a temporary route, the routing loops could be avoided. However, the performance of the protocol decreased. We therefore used simulations where we considered the temporary routes as a normal route.

Table 19: Packet drops in disaster area.

Cause for drop	**DSDV**	**DSR – 1**	**DSR - 2**	**AODV – 1**	**AODV - 2**	**AODV - 3**
Sendbuffer		10033	10277	9836	10693	10941
Route failures	15853	1990	2960	5047	3046	2470
ARP	479	35	64	292	241	64
Interface queue	4528	3	159	34	20	44
End of simulation	7	1	9	2	1	116

5.6.5 Summary realistic scenarios

The realistic scenarios show how the protocols behave in certain situations. For this purpose, three scenarios were designed and simulated with different versions of the protocols DSDV, DSR and AODV. DSR show the best performance results overall. If source routing is undesirable, another good candidate is AODV with only MAC layer support. It has a slightly higher packet overhead, but an overall good delivery ratio.

5.7 Observations

The protocols that we have worked with most are AODV and DSR, thus we have observed some important differences for these protocols, that in some way affect the performance of the protocols.

5.7.1 Ability to find routes

AODV

To illustrate AODVs ability to find routes, we use a simple scenario (see Figure 26). The scenario consists of four nodes A, B, C and D connected to each other.

Figure 26: Simple example scenario.

Node A needs a route to node D. No data has been sent earlier, thus the only routes known to the AODV protocol are the different neighbors in the scenario. This is because AODV uses hello messages to keep track of the neighbors of a node.

A RREQ is generated by node A and broadcasted to its neighbors. The RREQ propagates through the network until it reaches node C. Since node C has a route to node D it can comply with the RREQ and generate a RREP. During the propagation of the RREQ, a reverse path is set up to node A.

The RREP is unicasted back to A. It uses the temporarily backward route set up by the RREQ. During the propagation of the reply, a forward route is set up to node D by all intermediate nodes, resulting in the routing table shown in Table 20, thus all temporary routes are updated to active routes.

Table 20: Routing tables for AODV after a route discovery process.

Node A		**Node B**		**Node C**		**Node D**	
Destination	**Nexthop**	**Destination**	**Nexthop**	**Destination**	**Nexthop**	**Destination**	**Nexthop**
B	B	A	A	B	B	C	C
D	B	C	C	D	D		
		D	C	A	B		

After the route discovery process, only B and C have routes to all other nodes in the scenario. Node A does not know that there is a route to node C, nor does node D know of the route to A and B. Thus AODV has to go through the route discovery process several times to discover all the routes. This leads to more control traffic and higher delays. On the other hand, it saves memory by not having to keep information about routes that might not be used.

DSR
The ability to find routes differs slightly between the DSR and AODV protocol. DSR can make use of the source route carried in each packet header to discover routes to nodes by which the packet has traveled through. If new routes are found, DSR can take advantage of this without having to go through a new route discovery phase, thus reducing the number of control packets and decreasing the delay. The memory usage of a node is instead increased by having to store more routes.

We consider the same scenario as for the AODV protocol (Figure 26). Node A needs a route to node D. No data has been sent earlier, thus DSR has no knowledge of any routes. Since no routes are known, the RREQ propagates until it reaches node D. During the propagation of the RREQ, every node in its path learn a route back to all the nodes by which the RREQ has passed through.

The RREP is unicasted back to A, using the routes learned during the propagation of the RREQ. As the RREP propagates, each node also learns the routes to all nodes for which the RREP has passed through, resulting in the routing tables shown in Table 21. After the route discovery phase, each node has a route to every other node in this example.

Table 21: Routing caches for DSR, after a route discovery process.

Node A		**Node B**		**Node C**		**Node D**	
Destination	**Path**	**Destination**	**Path**	**Destination**	**Path**	**Destination**	**Path**
B	B	A	A	A	B-A	A	C-B-A
C	B-C	C	C	B	B	B	C-B
D	B-C-D	D	C-D	D	D	C	C

5.7.2 Temporary backward routes

When a RREQ propagates in search for a route in AODV, a temporary backward route is set up towards the requesting node. A RREP might use this backward route. This route is normally set to expire after 3 seconds if it is not used. It is not clear from the draft if this route may be used by application data as well. Using this route for data affects the behavior of the protocol in some way and it is not described how the protocol should handle this. One of the problems with this temporary route is with the way it is set up. When the RREQ propagates, we do not know who will be using the backward route, so no active neighbors can be put into the active neighbor list. If during this time, a link failure or an expired route entry somewhere breaks the route, the rest of the nodes in the route are not informed of this and therefore have redundant routes in their tables. When the route is used again, the application data will come to a node in the route that does not have a valid entry for the destination and will therefore be dropped while a triggered RREP tries to inform the nodes in the route that it is broken. This problem will become more apparent in a network with a higher load.

Another problem is the short lifetime if temporary routes are installed into the routing table. Lets say node A sets up a temporary route to D. Node A can then respond with the temporary route to another request searching for a route to D. Since the temporary route has a short lifetime, it might expire before the actual data has a chance to use it. This results in unnecessary control overhead and packet drops.

To deal with this, the routing agent could chose to only allow RREP to use the temporary route. Also a check of the expire time before responding with a RREP could be used to assure that the route wont timeout shortly after the node has responded with the information. A question about this was directed to the author of the protocol with the response that this check should not be done, instead the protocol should rely on the sequence numbers to provide the freshness of a route.

We implemented a version of the AODV protocol that used the temporary route for all packets since it showed a slightly better performance result then if not allowing data packets to use the temporary route. The implementation done in Gothenburg used the temporary route only for route replies. So we have a small difference in our implementations.

5.7.3 Buffers

The use of send buffers in the routing protocol also affects the performance of the routing protocol; especially the size of the buffers and the time a packet is allowed to stay there before it is dropped. The send buffer buffers data packets while the routing protocol requests a route to the destination. In our simulations, AODV used a buffer capable of holding 64 packets and allowing them to stay in the buffer for maximum 8 seconds. Allowing the packets to stay in the buffer for longer period of times will of course increase the amount of successfully routed packets, but also increase the average delay. Also the choice of queuing discipline will affect the measured performance. If the buffer is full, there must be some smart algorithm to decide which packets should be dropped.

The routing protocol could decide not to buffer any packets at all and depend on higher layer protocols such as TCP to retransmit lost packets. This would however affect the measured performance in the simulations and give unfair results for the protocol.

5.8 Discussion

If we compare the work that we have done against the work that was done by the CMU Monarch project [3], many similarities can be seen. Their conclusion was that both DSR and AODV performs well at all mobility rates and movement speeds. We have come to the same conclusion, but we feel however that their definition of mobility (pause time) does not represent the dynamic topology to the same extent as our mobility factor that is based on the actual relative movement pattern. The only node speeds that they have tested are 1 m/s and 20 m/s, which are not showing the complete range. Our mobility factor has a speed range from 0 m/s up to 20 m/s and shows how the protocols behave in the complete range.

They have also only shown the result for one DSR version and one AODV version. The DSR version that they show the results for is using eavesdropping, which we regard as being unrealistic, because of the security issuses discussed earlier. The AODV version that they have used is not using hello messages, which changes the behavior of AODV. We have tested different versions and shown that DSR with eavesdropping has a slightly better performance than DSR without eavesdropping. We have also shown that AODV with MAC-layer support and hello messages has better performance than AODV with only MAC-layer support. Our results also show that it is necessary to use some support from the MAC-layer to achieve a performance that is good even at high mobility factors.

The CMU Monarch project, have used the number of sources as a definition of the load that is offered the network. Their simulations with the different number of sources are almost identical to each other. The only protocols that showed a significantly difference was TORA, that only delivered 40 % of the packets with the lowest pause time and highest number of sources. Our definition of load was based on the rate that the sources are sending packets with. Even DSR that had the best results in the CMU paper fails to deliver a large fraction of the packets when the rate is increased. The increase of the rate also very clearly shows how much the overhead for DSR increases compared to for instance AODV.

5.9 Classification

Why is there any need for classification of routing protocols? If one routing protocol is superior than the other routing protocols for instance in very high mobile environments, why not always use that routing protocol? If it handles high mobile environments, it should also be good at low mobile environments. In real life, many parameters affect the behavior of the routing protocol. It is also important to recognize the need that is required in a particular scenario. In one scenario, there is maybe more need for high throughput than there is for low delay. In another scenario there is maybe more need for low delay etc.

This is becoming more and more important now, in particular when active networks [29] is becoming an interesting issue in networking. Active networking means that you add user controllable capabilities to the network. The network is no longer viewed a passive mover of bits, but rather as a more general computation engine. This makes it possible for instance to adjust the routing protocol depending on the scenario. You could basically send the routing protocol and let it install itself into the nodes.

5.9.1 Mobile networks

As the simulation results show, the mobility of the network greatly affects the performance of the protocols. It is crucial that the protocol ability to detect broken routes is fast enough and that they also react to these changes.

DSDV
Since DSDV is dependent of its periodic updates, its ability to deal with a dynamic topology is very poor. It has a poor ability to fast detect broken links and takes time to converge. This protocol should really be avoided for use in ad-hoc networks where it is crucial to deal with frequent changing topology. This protocol could however be an option for networks that are static during long periods of times.

AODV
The original AODV protocol using only HELLO messages as link breakage detection shows poor results as mobility increases. This protocol needs better link breakage detection. Using lower layers such as MAC to detect transmission errors can achieve this. If this is used, the protocol actually shows a very good performance. This protocol is a definite choice for highly mobile networks.

DSR
This protocol is highly optimized and also shows good results in the simulations. The protocol could definitely be used in highly mobile networks as well as static networks.

5.9.2 Size of networks

When talking about the size of a network, it is not only the number of nodes in the network that is of interest. The area that the nodes are spread out over is also interesting. This basically decides the connectivity of the network. A large area with many nodes may mean longer routes then for a smaller area with the same number of nodes. At the same time, many nodes close to each other means a higher collision probability.

DSDV
This protocol does not scale well. Its use of periodic broadcasts limits the protocol to small networks. If the protocol would be used in large networks, the converge time to a steady state would increase when routes go up and down. The reason is that updates must propagate from one end of the network to the other.

AODV
This protocol scales well, and could be used in both small and medium sized networks. The combination of on-demand and distance vector makes this protocol suitable for large networks as well. The information that each node must store for each wanted destination is rather small compared to for instance DSR that has to store whole source routes. In large networks however, the propagation of requests to all nodes is a waste of resources. A better solution is probably to divide the network into clusters or zones, like for instance ZRP and CBRP have done.

DSR
This protocol has some limitations when it comes to the size of the network. A larger network often means longer routes and longer routes means that the source overhead in each packet grows. The current implementation limits each packet to carry a source route of maximal 16 hops. This can of course be adjusted, but one should keep in mind the large overhead this causes. One could imagine a network with 20 nodes connected in a straight line. Then this implementation would not manage to route to all nodes. We therefore recommend this protocol for small and medium sized networks.

5.9.3 Network scenarios

Conference
For low mobility scenarios, like the conference scenario that we did simulations on, DSR is the best protocol to use if the hopcount is small (fewer than 5 hops). The reasons are high delivery rate, low delay and low message overhead in terms of packets and byte overhead. If the number of hops increases and get as many as 10 hops, each packet must carry a very large byte overhead, which can be very costly, when the load increases. Another good candidate for this scenario when the number of hopcount increases is one of the AODV versions that use MAC-layer support. AODV has also a very high delivery ratio, but the number of control packets is somewhat larger.

Event coverage
In average mobile scenarios, like the event coverage scenario, where nodes tends to cluster and almost all communication is within the clusters, DSR is by all means the best protocol. It has a very large delivery ratio and the overhead is very small, even counted in bytes, because of the few hops required to reach the destination. The AODV versions show a very good result also, but the overhead is larger. So for these type sort of networks DSR it the best protocol.

Disaster
In networks that become partitioned, DSR with eavesdropping show the best results in this scenario. It has a high delivery ratio, high throughput, a delay around 1.2 seconds and low packet overhead. It also uses only 5 hops in average to reach a destination. This protocol is therefore recommended in this type of scenario. Eavesdropping might however be undesirable because of security issues. One other candidate for this scenario is AODV with MAC support. It has almost as high delivery ratio as DSR and also a lower delay. The packet overhead is twice as high but the byte overhead is smaller.

5.10 Improvements

The simulations have shown that DSR with and without eavesdropping and the AODV versions that use link-layer support has the overall best result in almost all simulations. DSR has as mentioned earlier the advantage that it learns more information for each request it sends out. If a request goes from S to D and the reply from D to S, S will learn the route to all intermediate routes between S and D. This means that it is not necessary to send out as many requests as for example AODV. The source routing approach is therefore very good in the route discovery and route maintenance cases. However, source routing is not desirable to use for data packets. First of all, it adds a lot of overhead. Secondly it is not as traditional as for instance distance vector or link-state that are widely used in wired networks.

Our proposal is therefore to implement a protocol that is a combination of source routing and distance vector. Source routing should be used in route discovery and route maintenance phases. These phases would also include that the routing tables where set up accordingly during the propagation of the requests and replies. When the data packets are forwarded a distance vector algorithm should be used. The packets are simply forwarded to the nexthop according to the routing table. This in combination with that the protocol stores several routes for each destination would probably mean a protocol with a performance that is even better than the protocols that have been simulated in this master thesis.

6 Implementation study

The implementation study that was conducted at Ericsson Mobile Data Design in Gothenburg [28] has implemented the AODV protocol. The goal was to deliver a working routing protocol as specified in the original AODV draft [19].

6.1 Design

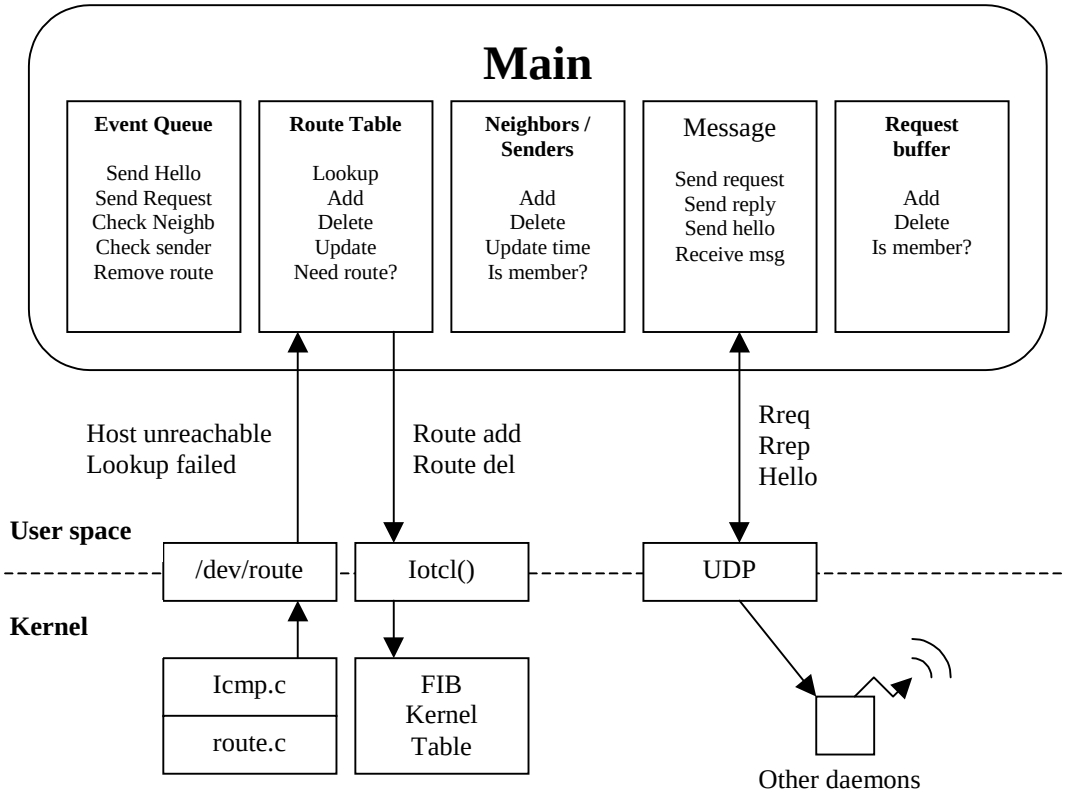


Figure 27: Overview of AODV daemon.

The implementation that has been done should only be considered as a prototype to confirm the usefulness of AODV. It is implemented as a daemon in user space. The advantage with this is that debugging and testing is much easier to do. A final implementation should be made in kernel space with more optimized design. Thus abandon the modular design and optimize the code to go really fast.

Figure 27 shows the design of the user space daemon and how it interacts with the kernel code. The different modules will be explained in the following chapters.

6.1.1 Main

Main ties together all other modules of the user space daemon. It is also in main that the AODV specific code lies.

6.1.2 Event queue

Queue for events that are supposed to be performed at certain times. These events include:

- Periodic hello messages
- Send / retransmit route requests
- Timeouts of route entries
- Hello timeout

6.1.3 Route table

This module is an interface against the submodules Daemon Table and Kernel Table. It takes care of these two, so when you want to lookup or change a route from Main, you only have to do one call to this module instead of two to the both submodules. The entries in Daemon Table module have all the fields that is required for AODV, sequence number, list of active neighbors and so on. The operations supported are Lookup, Add and Delete. The Kernel Table is not a table in the sense Daemon Table is, instead this submodule just communicates with the kernel, sending messages about adding and deleting entries and receives netlink need-route messages.

6.1.4 Neighbors / senders

Keeps track of the neighbors of a node. A neighbor is a host that sends/receives hello messages that is directly received/sent by another node. This demands that the links are bi-directional. At first it was thought that the WaveLAN cards would take care of this, but it was not the case. The signal strength and the range was different between the hosts. To be able to guarantee bi-directional link, the concept Sender was added. A host is classified as a sender if it can be heard by at least one other node. When sending hello messages a node sends the list of current Senders and if the receiver of the Hello messages is in this list, it is a Neighbor and can start act as one

6.1.5 Request buffer

This buffer prevents the network of being flooded by multiple request for the same address. This buffer stores already processed requests.

6.1.6 Message

Handles the different types of messages that the daemon can send and receive. These messages are:

- Hello
- Route Request
- Route Reply.

6.2 Setup

The computers used for this implementation study was:

- 2 stationary computers with Lucent WaveLAN ISA cards.
- 3 laptops with Lucent WaveLAN PCMCIA cards.

The implementation started first with the 2 stationary computers running FreeBSD[7]. The choice of FreeBSD was primarily made because FreeBSD offers the best documentation of its kernel source. It was later discovered when the laptops arrived that FreeBSD was incompatible with the Lucent WaveLAN PCMIA cards in the laptops. Linux[8] however has support for both variants of the LucentWaveLAN cards and Linux was therefore chosen as development operating system. It is not very hard to port the code to FreeBSD in the future if so desired.

[7] FreeBSD 2.2.6
[8] Linux Red Hat 5.1

6.3 Testing

The purpose of testing the implementation is to verify that the implementation works correctly and to see if the performance is suitable for real life applications.

6.3.1 Correctness

To verify that the implementation of AODV behaved correctly, computers were placed in different scenarios to test the different parts of the protocol. The following tests where made:

- Bi-directional links and neighbors: Tests that two nodes can send hello messages to each other and registry the counterpart as a neighbor.
- Unidirectional links and neighbors: It tests the same as above, but with the difference that we only have an uni-directional link, so only one of the nodes can hear the other and registry him as a neighbor.
- Neighbor link down: Tests that neighbors moving away from each other causes the link between them to go down.
- Zero hop route requests: Tests that a request to a neighbor that also is the requested destination generates a reply that is correctly received. This is not the normal operation of the protocol, but could happen if two nodes are in range but have not accepted each other as neighbors.
- Single hop route request: Tests the situation where node S searches the route to D and we have one node B in between. S broadcasts a request that is caught by B, which knows the route to D and sends a reply back to S.
- Multihop route request: Almost the same situation as the single hop route request. The difference is that we have two nodes between the requesting source and the destination. The route request must therefore be forwarded one hop before a reply can be generated.
- Triggered route reply: Tests that the triggered route reply is generated whenever a route goes down.

6.3.2 Performance

No actual performance tests were done. The results for so few nodes would be misleading. Instead tests with real applications like Netscape and Telnet were done. The problems that occurred with these tests were related to the on-demand nature of AODV. Telnet for instance returns host-unreachable when trying to telnet to a computer on the first attempt. The second attempt however is successful. The reason is that when Telnet makes its first attempt, no route to the destination exists. This will result in an error message from the kernel to Telnet, at the same time as a new request is sent to the neighbors. This request will eventually find a route to the destination and it is installed in the routing table. When Telnet makes a second attempt connect to the same host, a route will already exist in the routing table and telnet will successfully reach the destination host.

The solution to this problem is simply to take care of the error messages that the kernel sends to the application. The error message should be buffered and if a route is not found in a certain amount of time, the error message should be sent to the application, but if a route is actually found the error message can be discarded.

6.4 Problems / Limitations

Problems that occurred during the implementation include:

- FreeBSD incompatibilities: As mentioned before, lack of functional drivers to FreeBSD forced the implementation study to be done under Linux.
- Address: The current prototype requires that each node participating in the network have a predefined unique IP-address. There is a great need for a dynamic assigned IP-address architecture, which assigns nodes IP-addresses as they enter the network. IPv6 holds such functionality. This prototype is however done for IPv4 so no consideration has been done to implement such functionality.
- Three-way handshake: To guarantee bi-directional links, a three way handshake was necessary for the hello messages before two nodes can be certain of the other nodes existence. The handshake uses piggybacking. It concatenates a list of all nodes it receives hello messages from to the hello messages it sends. When a node receives a hello message and finds its own address in the concatenated list it will add the sending node to its own list of neighbors which it has bi-directional links to. Three hello

messages instead of one wastes bandwidth, but could also in a worst case scenario mean that it could take as long as two full hello intervals before two nodes in range of each other accepts the bi-directional links.

- Temporary routes: When a request is broadcasted from a node, it will propagate through the network and at the same time install a temporary route back to the source. The problem with this temporary route is first of all that the request is unaware of if the route ever will be used and secondly who will use it. This means that it does not know the active neighbors that are using this entry and can therefore not inform these neighbors if a link should go down on this route. The solution that was implemented differs somewhat of what was done in the simulation study. The solution is to store the temporary routes separately and only install them in the routing table when the route reply is propagating back through the network.

6.5 Improvements

Possible features that could be added to this prototype:

- Link layer hellos: The addition of link layer feedback from 802.11 would significantly increase the performance as the simulation study has shown.
- Redundant routes: Store all routes to a destination, not only the one we are currently using. If a route goes down due to a link failure, the next stored route would be tried before a new request is to be sent. This saves a lot of overhead a and makes the delay somewhat smaller.
- Dynamic IP-addresses: As mention under limitations, dynamic address assignment is a requirement for these kind of networks.
- Multicast: Multicast groups within a ad-hoc network could be added. The latest AODV draft has support for this.

6.6 Implementation conclusions

The implementation of the AODV protocol has shown that it is possible to get these protocols to work in real-life. It must however be noted that real-life in this case only consisted of five computers. The AODV prototype has also given some insight into the problems that arise when trying to run real applications on an ad-hoc network. Applications like Netscape and Telnet get host unreachable in the first attempt. The second attempt finds the route successfully. This has to do with the on-demand feature of the AODV protocol.

7 Conclusions

7.1 Results

The simulations have shown that there certainly is a need for a special ad-hoc routing protocol when the mobility increases. It is however necessary to have some sort of feedback from the link-layer protocol like IEEE MAC 802.11 when links go up and down or for neighbor discovery. To only be dependent on periodic messages at the IP-level will result in a very high degree of packet losses even when mobility increases a little. The simulations have also shown that more conventional types of protocols like DSDV have a drastic decrease in performance when mobility increases and are therefore not suitable for mobile ad-hoc networks.

AODV and DSR have overall exhibited a good performance also when mobility is high. DSR is however based on source routing, which means that the byte overhead in each packet can affect the total byte overhead in the network quite drastically when the offered load to the network and the size of the network increases. In these situations, a hop-by-hop based routing protocol like AODV is more desirable. One advantage with the source routing approach is however that in its route discovery operation it learns more routes. Source routing is however not desirable in ordinary forwarding of data packets because of the large byte overhead. A combination of AODV and DSR could therefore be a solution with even better performance than AODV and DSR.

Another key aspect when evaluating these protocols is to test them in realistic scenarios. We have tested them in three types of scenarios. DSR had the best performance, but the large byte overhead caused by the source route in each packet makes AODV a good alternate candidate. It has almost as good performance.

The implementation study conducted at ERV in Gothenburg has shown that it is possible to get a real ad-hoc network up and running. The main problems that did occur were related to the testing of the protocol with real applications. When a route was needed by the application and the route did not exist in the routing table, the kernel informed the applications of a connection error before giving the routing protocol enough time to find a route.

7.2 Further studies

Ad-hoc networking is a rather hot concept in computer communications. This means that there is much research going on and many issues that remains to be solved. Due to limited time, we have only focused on the routing protocols. However there are many issues that could be subject to further studies.

First of all, the simulator environment could be improved. These are just some of the improvements that could be made:

- More routing protocols, for instance TORA, ZRP and CBRP.
- Measurement of computing complexity.

Secondly, there are many issues related to ad-hoc networks that could be subject to further studies:

- Simulations which take unidirectional links into consideration.
- Some sort of analysis of whether many small control messages are more costly to send in terms of resources than fewer large control messages.
- Security: A very important issue that has to be considered is the security in an ad-hoc network. Routing protocols are prime targets for impersonation attacks. Because ad-hoc networks are formed without centralized control, security must be handled in a distributed fashion. This will probably mean that IP-

Sec [14] authentication headers will be deployed, as well as the necessary key management to distribute keys to the members of the ad-hoc network.

- Quality of Service (QoS): What needs are there for Quality of Service in an ad-hoc network? This is related to what the networks actually will be used for.
- Hand-over of real-time traffic between nodes. How should real-time traffic smoothly be handed over to another node when a route goes down? Should flooding be used before a route is found?
- Multicast: We have only looked at unicast routing. Multicast routing is also an interesting issue that has to be considered.
- Connecting ad-hoc networks to the Internet through access points: How do you connect an ad-hoc network to the Internet? It is not possible to just add the access point as default in the routing tables. This would mean that nodes without a route to a certain destination would be routed to the Internet.
- Mobile IP: Integration of mobile IP into ad-hoc networks.
- Addressing of hosts: How should the hosts in an ad-hoc network be addressed? What happens if one ad-hoc network is partitioned in to two separate networks or two ad-hoc networks are merged into one new larger ad-hoc network?

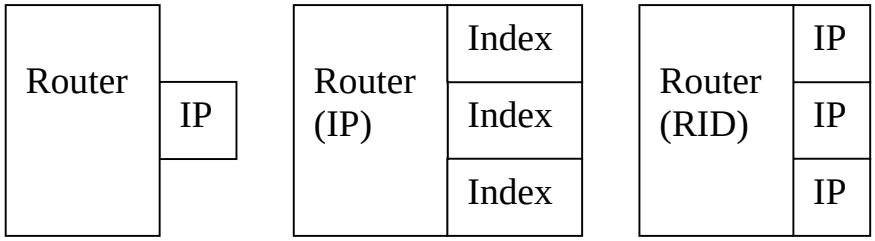


Figure 28: Different router identification approaches. From left to right: 3a, 3b, 3c.

There are basically three types of identifiers to consider (Figure 28):

1) Interface identifiers: Interfaces are present on both routers and hosts. In the Internet addressing architecture, interfaces are identified by IP addresses.
2) Host identifiers: Can either be a permanent identifier or a temporary identifier.
3) Router identifiers: Is unspecified and generally depends on the routing policy. There are three approaches to MANET router identification and addressing currently being considered:
   a) Single IP address (AODV): Identifies a MANET node (both the router and the host) with a single IP address. This approach leaves several issues open:
      - How to support sets or subnets of hosts attached to a MANET router?
      - How to support the use of multiple wireless interfaces?
   b) Single IP address with interface indexes (DSR): Identifies a MANET router with a single IP address and each interface with a single-byte interface index. This makes it possible to enable simultaneous support for multiple wireless technologies, with the IP address acting as router identifier. The problem with this scheme is that it is not IP in the classical sense where interfaces are identified by IP addresses. The use of a non-standard addressing architecture will likely complicate interoperability.
   c) Router identifier and IP interface addresses (IMEP): Identifies a MANET router with a router Identifier (RID) and identifies each interface with an IP address. This approach can support sets or subsets of attached hosts and simultaneous use of multiple wireless technologies.

Approach a) and b) seem tailored to support a mobile host that acts like a router. Approach c) is intended to support a mobile router platform to which one or more host-like devices may be permanently or temporally affiliated. These approaches say nothing about how IP addresses are assigned to interfaces (on hosts or routers), or what the RID is and how it is assigned. This is a separate problem, although one which is related to routing.

8 References

[1] Dimitri Bertsekas and Robert Gallager, *"Data Networks - 2nd ed"*. Prentice Hall, New Jersey, ISBN 0-13-200916-1.

[2] Bommaiah, McAuley and Talpade. AMRoute, "Adhoc Multicast Routing Protocol", Internet draft, draft-talpade-manet-amroute-00.txt, August 1998. Work in progress.

[3] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu and Jorjeta Jetcheva, "A performance Comparison of Multi-hop Wireless Ad Hoc Network Routing Protocols". *Mobicom'98, Dallas Texas, 25–30 October, 1998*.

[4] Josh Broch, David B. Johnsson, David A. Maltz, "The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks". Internet Draft, draft-ietf-manet-dsr-00.txt, March 1998. Work in progress.

[5] Scott Corson and Joseph Macker, " Mobile Ad Hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations". Internet-Draft, draft-ietf-manet-issues-01.txt, March 1998. Work in progress.

[6] M.Scott Corson, S. Papademetriou, Philip Papadopolous, Vincent D. Park and Amir Qayyum, "An Internet MANET Encapsulation Protocol (IMEP) Specification". Internet draft, draft-ietf-manet-imep-spec01.txt, August 1998. Work in progress.

[7] Kevin Fall and Kannan Varadhan, "ns notes and documentation". The VINT project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, May 1998. Work in progress.

[8] Zygmunt J. Haas and Marc R. Pearlman, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks", Internet draft, draft-ietf-manet-zone-zrp-01.txt, August 1998. Work in progress.

[9] IEEE Computer Society LAN MAN Standards Committee, *"Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications"*, IEEE Std 802.11-1997. The Institute of Electrical and Electronics Engineers, New York.

[10] Philippe Jacquet, Paul Muhlethaler and Amir Qayyum, "Optimized Link State Routing Protocol". Internet draft, draft-ietf-manet-olsr-00.txt, November 1998. Work in progress.

[11] Mingliang Jiang, Jinyang Li and Yong Chiang Tay, "Cluster Based Routing Protocol (CBRP) Functional specification". Internet draft, draft-ietf-manet-cbrp-spec-00.txt, August 1998. Work in progress.

[12] David B. Johnson and David A.Maltz, "Dynamic source routing in ad hoc wireless networks". In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153-181. Kluwer Academic Publishers.

[13] David B. Johnson and David A. Maltz, "Protocols for adaptive wireless and mobile computing". *In IEEE Personal Communications*, 3(1), February 1996.

[14] Stephen Kent and Randall Atkinson, "Security Architecture for the Internet Protocol", Internet draft, draft-ietf-ipsec-arch-sec-07.txt, July 1998. Work in progress.

[15] Mobile Ad-hoc Networks (MANET). URL: http://www.ietf.org/html.charters/manet-charter.html. (1998-11-29). Work in progress.

[16] Vincent D. Park and M. Scott Corson, “Temporally-Ordered Routing Algorithm (TORA) Version 1: Functional specification”. Internet draft, draft-ietf-manet-tora-spec-01.txt, August 1998. Work in progress.

[17] Vincent D. Park and M. Scott Corson, “A performance comparison of the Temporally-Ordered Routing Algorithm and Ideal Link-state routing”. In *Proceedings of IEEE Symposium on Computers and Communication ’98*, June 1998.

[18] Charles E. Perkins, *“Mobility support, Mobile IP and Wireless Channel Support for ns-2”, presentation slides*. URL: http://www.svrloc.org/~charliep/mobins2/, (1998-11-29). Work in progress.

[19] Charles E. Perkins, “Ad Hoc On Demand Distance Vector (AODV) Routing”. Internet draft, draft-ietf-manet-aodv-01.txt, August 1998. Work in progress.

[20] Charles E. Perkins, “Ad Hoc On Demand Distance Vector (AODV) Routing”. Internet draft, draft-ietf-manet-aodv-02.txt, November 1998. Work in progress.

[21] Charles E. Perkins, “Mobile Ad Hoc Networking Terminology”. Internet draft, draft-ietf-manet-term-00.txt, October 1997. Work in progress.

[22] Charles E. Perkins and Pravin Bhagwat, “Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers”. In *Proceedings of the* SIGCOM '94 *Conference on Communications Architecture, protocols and Applications*, pages234-244, August 1994. A revised version of the paper is available from http://www.cs.umd.edu/projects/mcml/papers/Sigcomm94.ps. (1998-11-29)

[23] Larry L. Peterson and Bruce S. Davie, *“Computer Networks - A Systems Approach”*. San Francisco, Morgan Kaufmann Publishers Inc. ISBN 1-55860-368-9.

[24] Plummer, D., "An Ethernet Address Resolution Protocol - or - Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", RFC-826, MIT, November 1982.

[25] Theodore S. Rappaport, *“Wireless Communications: Principles and Practice”*. New Jersey, Prentice Hall. ISBN 0-13-375536-3.

[26] Raghupathy Sivakumar, Prasun Sinha and Vaduvur Bharghavan, “Core Extraction Distributed Ad hoc Routing (CEDAR) Specification”, Internet draft, draft-ietf-manet-cedar-spec-00.txt, October 1998. Work in progress.

[27] Martha Steenstrup, *“Routing in communication networks”*. New Jersey, Prentice Hall. ISBN 0-13-010752-2.

[28] Jerry Svedlund and Johan Köpman, *“Routing protocols in wireless ad-hoc networks - An implementation study”*. Uppsala: Uppsala University. Master thesis.

[29] D. L. Tennenhouse and D. Wetherall, “Towards an active network architecture”. In *Multimedia Computing and Networking 96*, San Jose, CA, Jan 1996.

[30] The CMU Monarch Project. “The CMU Monarch Projects Wireless and Mobility Extensions to *ns”*. URL: http://www.monarch.cs.cmu.edu/. (1998-11-29). Work in progress.

Appendix A -Terminology

This appendix contains some terminology [21] that is related to ad-hoc networks.

A.1 General terms

Bandwidth: Total link capacity of a link to carry information (typically bits).

Channel: The physical medium is divided into logical channel, allowing possibly shared uses of the medium. Channels may be made available by subdividing the medium into distinct time slots, distinct spectral bands, or decorrelated coding sequences.

Convergence: The process of approaching a state of equilibrium in which all nodes in the network agree on a consistent state about the topology of the network.

Flooding: The process of delivering data or control messages to every node within the any data network.

Host: Any node that is not a router.

Interface: A nodes attachment to a link.

Link: A communication facility or medium over which nodes can communicate at the link layer.

Loop free: A path taken by a packet never transits the same intermediate node twice before arrival at the destination.

MAC-layer address: An address (sometimes called the link address) associated with the link interface of a node on a physical link.

Next hop: A neighbor, which has been designated to forward packets along the way to a particular destination.

Neighbor: A node that is within transmitter range from another node on the same channel.

Node: A device that implements IP.

Node ID: Unique identifier that identifies a particular node.

Router: A node that forwards IP packets not explicitly addressed to itself. In case of ad-hoc networks, all nodes are at least unicast routers.

Routing table: The table where the routing protocols keep routing information for various destinations. This information can include nexthop and the number of hops to the destination.

Scalability: A protocol is scalable if it is applicable to large as well as small populations.

Source route: A route from the source to the destination made available by the source.

Throughput: The amount of data from a source to a destination processed by the protocol for which throughput is to be measured for instance, IP, TCP, or the MAC protocol.

A.2 Ad-hoc related terms

Ad-hoc: "For this special or temporary purpose" or "a special case without generic support".

AODV: Ad-Hoc On-Demand Distance Vector. Routing protocol for wireless ad-hoc networks.

Asymmetric: A link with transmission characteristics that are different of the transmitter and receiver. For instance, the range of one transmitter may be much higher than the range of another transmitter on the same medium. The transmission between the two hosts will therefore not work equally well in both directions. See also symmetric.

Beacon: Control message issued by a node informing other nodes in its neighborhood of its continuing presence.

Bi-directional: see symmetric.

CBRP: Cluster Based Routing Protocol. Routing protocol for wireless ad-hoc networks.

Cluster: A group of nodes typically in range of each other, where one of the nodes is elected as the cluster head. The cluster head ID identifies the cluster. Each node in the network knows its corresponding cluster head(s) and therefore knows which cluster(s) it belongs to.

DSDV: Dynamic Sequenced Distance Vector. Routing protocol for wireless Ad Hoc networks.

DSR: Dynamic Source Routing. Routing protocol for wireless Ad Hoc networks.

Proactive: Tries to maintain the routing map for the whole network all the time. See also reactive.

Reactive: Calculates route only upon receiving a specific request. See also proactive

RREQ: Routing Request. A message used by AODV for the purpose of discovering new routes to a destination node.

RREP: Route Reply. A message used by AODV to reply to route requests.

Symmetric: Transmission between two hosts works equally well in both directions. See also asymmetric.

TORA: Temporally Ordered Routing Algorithm. Routing protocol for wireless ad-hoc networks.

Unidirectional: see asymmetric.

ZRP: Zone Routing Protocol. Routing protocol for wireless ad-hoc networks.

Appendix B -AODV implementation for ns

This appendix contains a little more details about the implementation of AODV that we did for ns. The implementation of AODV is done according to the draft [19] released in August 1998.

B.1 Message formats

AODV have four different messages that it uses for route discovery and route maintenance. All messages are sent using UDP.

B.1.1 Route Request – RREQ

The format of the route request message is shown in Figure 29.

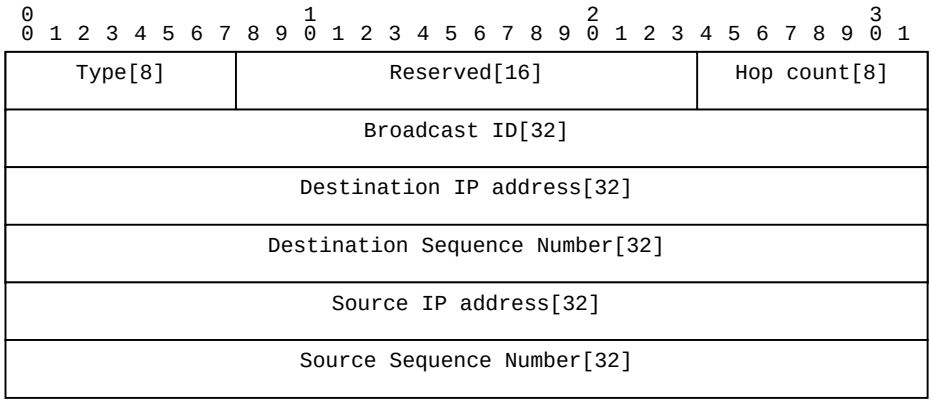


Figure 29: Route request format.

- **Type**: Type of message.
- **Reserved**: Reserved for future use. Currently sent as 0 and ignored on reception.
- **Hop count**: Number of hops from the source IP address to the node handling the request.
- **Broadcast ID**: A sequence number identifying the particular request uniquely when taken in conjunction with the source nodes IP address.
- **Destination IP address**: IP address of the destination for which a route is required.
- **Destination sequence number**: The last sequence number received in the past by the source for any route towards the destination.
- **Source IP address**: IP address of the node that originated the request.
- **Source sequence number**: Current sequence number for route information generated by the source of the route request.

B.1.2 Route Reply - RREP

The format of the route reply message is shown in Figure 30.

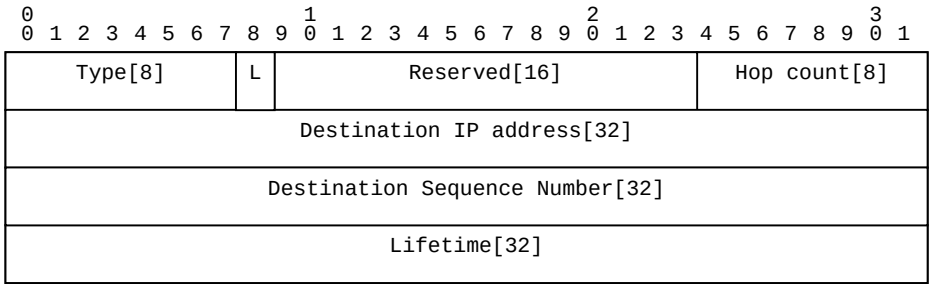


Figure 30: Route reply format.

- **Type**: Type of message.
- **L**: If the L-bit is set the message is a hello message and contains a list of the nodes neighbors.
- **Reserved**: Reserved for future use. Currently sent as 0 and ignored on reception.
- **Hop count**: Number of hops from the source IP address to the destination IP address.
- **Destination IP address**: IP address of the destination for which a route is supplied.
- **Destination sequence number**: The destination sequence number associated to the route.
- **Lifetime**: Time for which nodes receiving the Reply consider the route to be valid.

B.1.3 Hello

Hello messages are a special case of Route reply messages. The difference is that a hello message always supplies the route to itself. This means that the hop count field is set to 0, the destination address set to the nodes IP address and the destination sequence number set to the nodes latest sequence number.

B.1.4 Link failure

Link failure messages are also special Route reply messages, but in this case the destination reflects the route that has broken. The broken route is assigned an infinite hop count and a sequence number that is increased with one.

B.2 Design

Figure 31 shows how AODV was designed when implemented for ns.

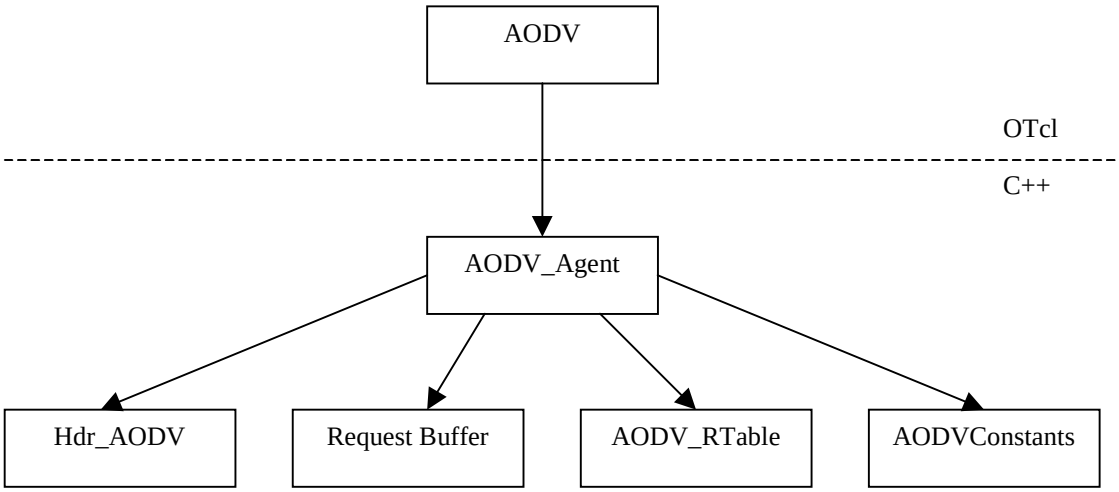


Figure 31: AODV design of implementation for simulator.

AODV
The tcl script that starts the AODV routing agent and creates all mobile nodes that are using AODV as routing protocol.

AODV_Agent
Implements all AODV specific parts. Handles RREQ, RREP, Hello and Triggered RREP. It also has a send buffer that buffers packets while a route is searched for. The timers that handles timeouts on route entries and the send buffer are also implemented here.

Hdr_AODV
Defines the message format for all messages that AODV uses.

Request Buffer
Implements the request buffer that prevents a node to process the same RREQ multiple times.

AODV_RTable
The routing table that AODV uses. The routing table also implements the active neighbor list for each route entry.

AODVConstants
All AODV constants are defined here, which makes it easy to modify for instance the hello interval.

B.3 Important routines

B.3.1 Sending RREQ

RREQ will only be sent by the source nodes (no intermediate nodes sends RREQs), if there does not exist any route for the destination.

```
 1.   if ( no route exists ){
 2.     check request buffer for requests already sent for destination
 3.     if ( no request sent already ){
 4.       create a RREQ packet
 5.       add (dest addr, broadcast ID) to request buffer
 6.       locally broadcast RREQ
 7.       set timer for RREP_WAIT_TIME before rebroadcasting RREQ
 8.       increment broadcast ID
 9.     } else {
10.       buffer packet from stream or discard, according to need
11.     }
12.   }
```

B.3.2 Receiving RREQ

When a node receives a RREQ, it must first of all decide if it already has processed the RREQ. The RREQ is discarded if it has been processed. Otherwise the source address and the broadcast ID from RREQ will be buffered to prevent it from being processed again.

```
 1.   if ( (source addr, broadcast ID) in request buffer ){
 2.     discard request -- already heard and processed
 3.   } else {
 4.     add (source addr, broadcast ID) to request buffer
 5.   }
```

The next step is to create or update the route entry in the routing table. This route can be used by the RREP when a route is found.

```
 1.   if ( no route to source addr ){
 2.     create a route entry for source addr
 3.   } elseif ( source seqno in RREQ > source seqno in route entry ){
 4.     update route entry for source addr
 5.   } elseif ((source seqno in RREQ = source seqno in route entry) AND
 6.              ( hop count in RREQ < hop count in route entry )){
 7.     update route entry for source addr:
 8.   }
```

Then, the node must check if it knows the route to the wanted destination. If the node knows the route it will unicast a RREP to the source. Otherwise it will forward the RREQ.

```
 1.   if ( you are destination of RREQ ){
 2.     create a RREP packet:
 3.     unicast RREP to source of request
 4.   } elseif (( have route to destination ) AND
 5.     (destination seqno in route entry >= destination seqno in RREQ)){
 6.         create a RREP packet:
 7.         unicast RREP to source of request
 8.   } else {
 9.     forward RREQ
10.   }
```

B.3.3 Forwarding RREQ

When a node receiving a RREQ that it has not processed yet does not have a route, it will forward the RREQ.

```
1.   create a RREQ packet:
2.    copy all fields from received RREQ into new packet
3.     increment hop count field
4.   locally broadcast new RREQ packet
5.   discard received RREQ
```

B.3.4 Forwarding RREP

When a node receives a RREP that is not addressed for the node, it will set up the forward route by updating the routing table and forward the RREP back to the requesting source. This part is however not explicit specified in the AODV draft.

```
 1.   if ( route to requested destination does not exist ){
 2.     create a route entry for requested destination
 3.   }elseif(destination seqno in RREP >
 4.   destination seqno in route entry){
 5.         update route entry for requested destination
 6.   } elseif ((destination seqno in RREP =
 7.            destination seqno in route entry )
 8.            AND ( hop count in RREP < hop count in entry )){
 9.         update route entry for requested destination
10.   }
11.   if (route to requesting source exists){
12.     forward RREP to requesting source
13.   }
```

B.3.5 Receiving RREP

When the originating source receives the RREP it will update the routing table.

```
 1.   if (route to destination does not exist){
 2.     create a route entry for destination
 3.   } elseif (destination seqno in RREP >
 4.             destination seqno in route entry){
 5.         update route entry for destination
 5.   } elseif (( destination seqno in RREP =
 6.               destination seqno in route entry ) AND
 7.             (hop count in RREP < hop count in entry )){
 8.         update route entry for destination
 9.  } else {
10.     discard RREP
11.   }
```

B.3.6 Hello handling

Each node periodically broadcasts a hello message to all neighbors. When a node receives a hello message it knows that the sending node is a neighbor and will update the routing table.

```
1.   if (route entry for HELLO source exists){
2.     update route entry
3.     if (destination seqno in HELLO >
4.         destination seqno in route entry){
5.       update destination seqno in route entry
6.     }
7.   } else {
8.     create route entry for HELLO source:
9.   }
```

B.3.7 Forwarding packets

AODV uses a active neighbor list to keep track of which neighbors that are using a particular route. These lists are used when sending triggered route replies. The neighbor lists are updated every time a packet is forwarded.

```
1.   if (route entry to destination exists){
2.    if (neighbor who forwarded packet to you !=
3.        active neighbor for route){
4.         add neighbor to active neighbor list for route entry
5.     }
6.   }
```

B.3.8 Sending Triggered RREP

Link breakages are detected by either the link layer which notifies the routing agent or by using hello messages. If a node has not received hello messages from a node for a certain amount of time it will assume that the link is down. Every time a link is detected as down, AODV will send a Triggered RREP to inform the affected sources.

```
1.   for (each address in the active neighbor list for a route entry){
2.     create a link failure notice packet
3.     unicast to active neighbor
4.   }
```

B.3.9 Receiving Triggered RREP

Every time a Triggered RREP is received informing about a broken link, the affected route entry must be deleted and neighbors using this entry must be informed.

```
1.   if (have active neighbors for broken route){
2.     send Triggered RREP
3.   }
4.   delete route entry for broken route
```

Appendix C -Simulator screenshots

This appendix shows some screenshots of Network animator and Ad-hockey. Network animator is the visualization tool for ns and Ad-hockey is the visualization for the mobility extension developed by the CMU Monarch project.

C.1 Network animator

Figure 32 shows a screenshot of Network animator. The scenario contains 19 nodes in a wired network. Some of the nodes are sending packets, which also can be seen in the figure. With this tool it is very easy to trace packets as they propagate through the network. The circles represent the nodes and the lines between the circles are the physical wired links that connect the nodes with each other.

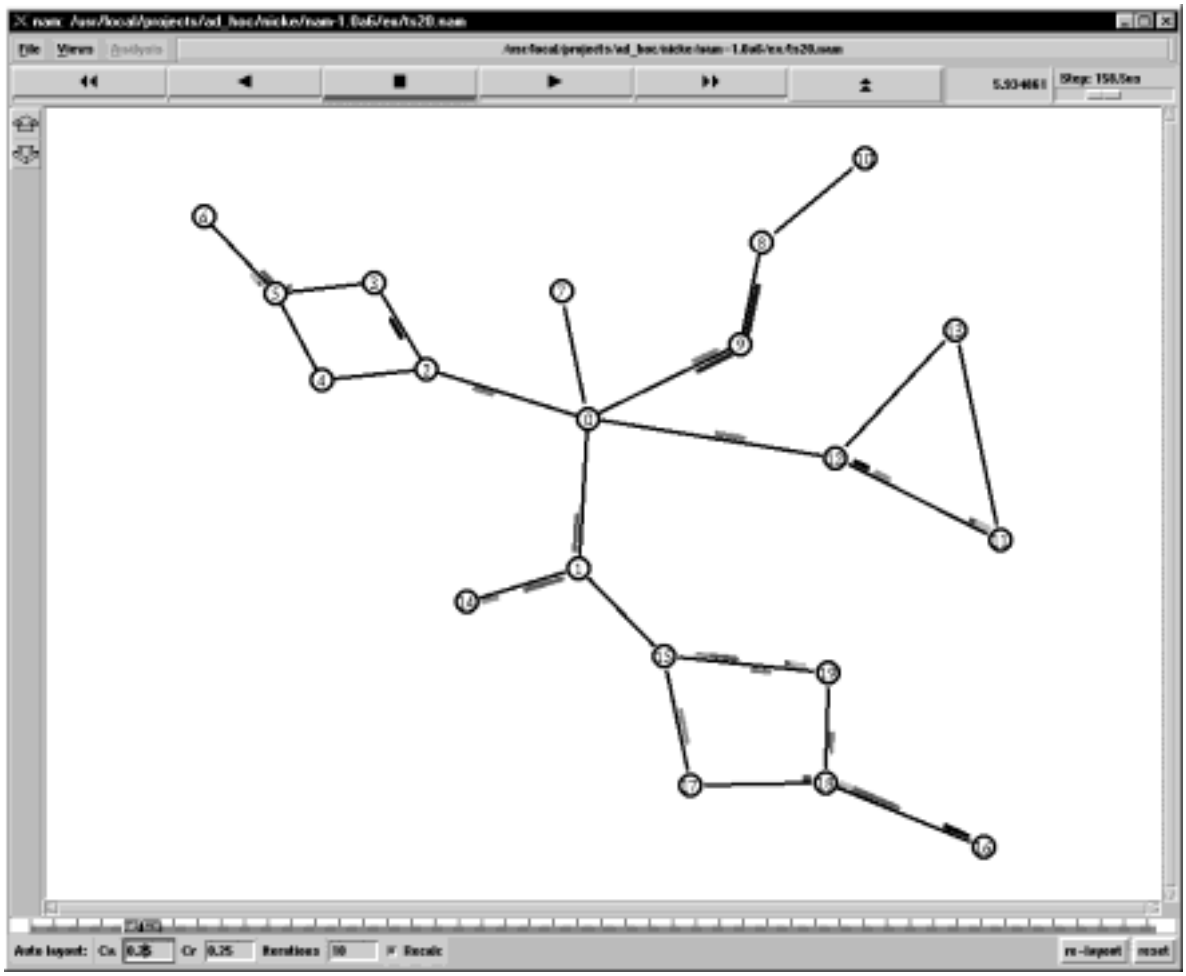


Figure 32: Screenshot – Network animator.

C.2 Ad-hockey

The screenshots of Ad-hockey that can be seen in Figure 33, Figure 34 and Figure 35 shows the playback trace of the realistic scenarios that with did simulations on. The big white rectangle in the middle is the movement area. The different colored circles are the nodes. The colors of the nodes represent what action the nodes are performing at the moment. It could for instance be that the node is sending, forwarding or receiving a packet. The lines between the nodes do not represent wired links. The lines are actually packets that propagate from node to node. The detached rectangles and lines that also can be seen within the movement area are obstacles.

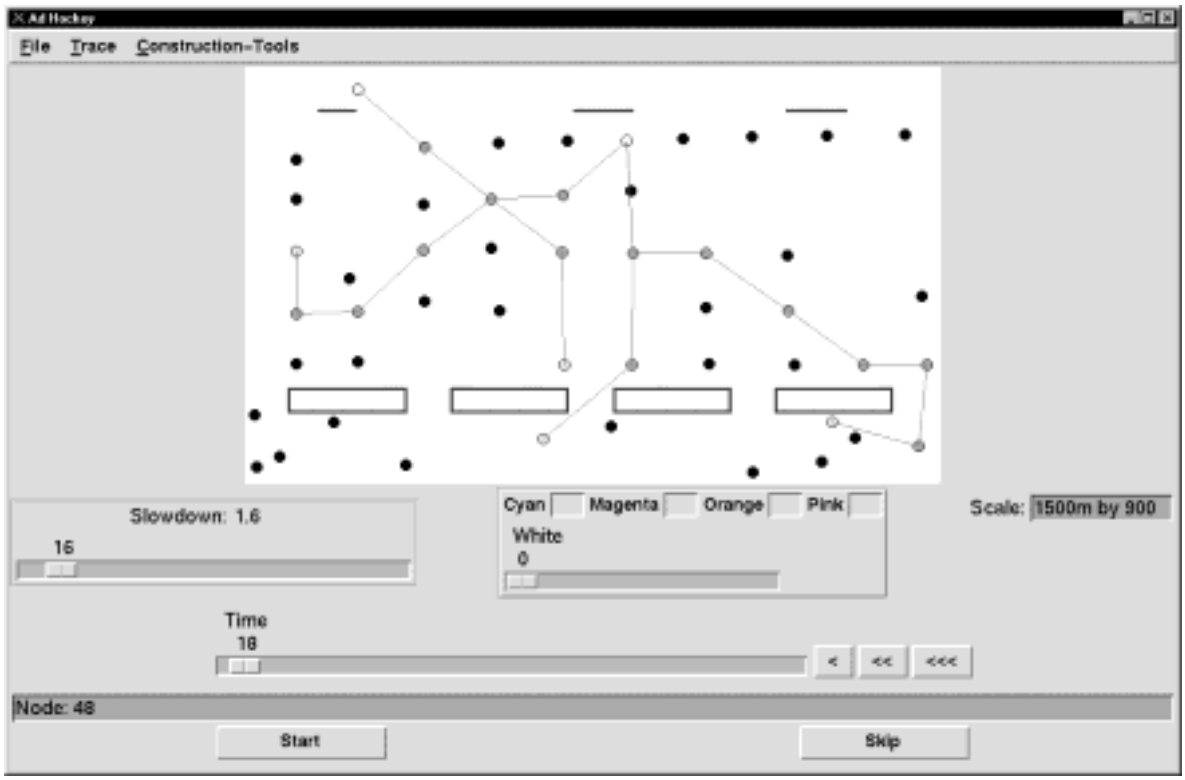


Figure 33: Screenshot – Ad-hockey – Conference scenario.

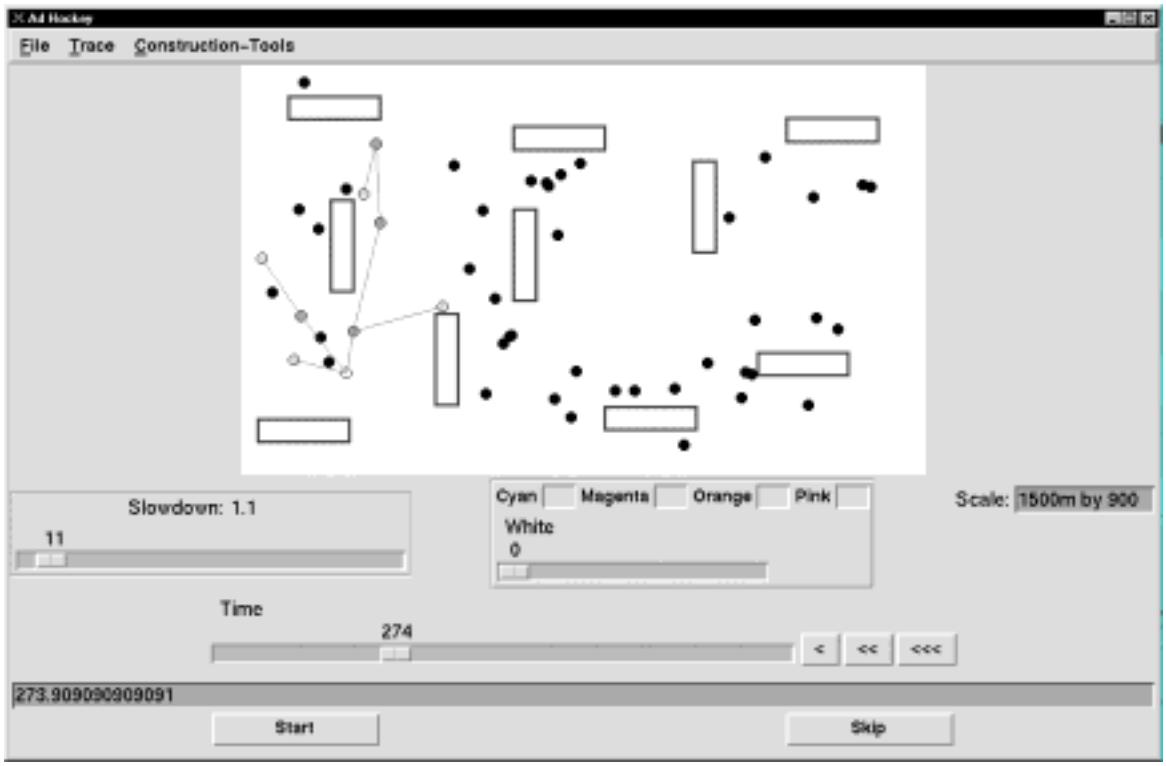


Figure 34: Screenshot – Ad-hockey – Event coverage scenario.

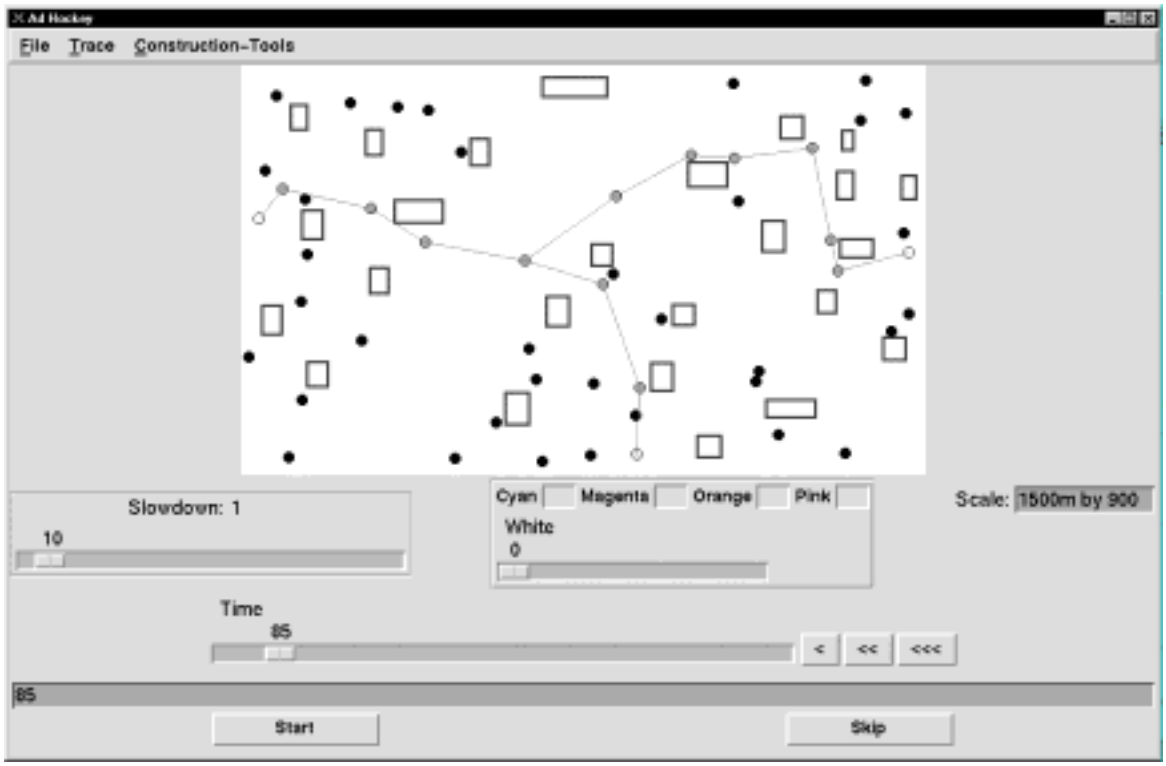


Figure 35: Screenshot – Ad-hockey – Disaster area.