NTNU
Innovation and Creativity

Content Distribution in Ad Hoc Networks

Lotte Johnsen

Master of Science in Communication Technology

Submission date:	June 2006
Supervisor:	Øivind Kure, ITEM
Co-supervisor:	Ole-Ingar Bentstuen, Forsvarets Forskningsinstitutt

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Distribution of both Command Control and Information System (C2IS) (e.g. positioning) in addition to multimedia information in ad hoc networks is a challenge. Solutions from wired networks can probably not be used without modifications. In particular, multicast solutions will be different in ad hoc networks than in wired networks, and "simplified multicast flooding" is now the preferred solution for multicast in ad hoc networks. Also, the capacity of an ad hoc network is often not known and will vary dynamically.

Content distribution in mobile ad hoc networks must thus adapt to the ever changing network properties, and also be very flexible in regard to who is receiving the information. Sometimes the content is going to be distributed to all nodes in the network, and sometimes the receiver may be limited to a particular department or predefined group of nodes.

The candidate will look in to how content distribution (i.e. video and C2IS) should be handled in ad hoc networks where there is high mobility of the nodes and also strongly varying network properties between the nodes. Two main areas to be investigated are limitation of multicast distribution and scaling according to the currently available bandwidth.

Assignment given: 16. January 2006
Supervisor: Øivind Kure, ITEM

Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics and Electrical Engineering

Content Distribution in Ad Hoc Networks

Master's Thesis

Lotte Johnsen

Spring 2006

Preface

This report is the result of my Master's thesis at Norwegian University of Science and Technology (NTNU) in Trondheim during the spring semester 2006, and documents the Master's thesis "Content Distribution in Ad Hoc Networks". The work in this thesis was carried out at Norwegian Defence Research Establishment (FFI) as a part of the project "NbF-Grid" and at Department of Telematics at NTNU.

It has been challenging and interesting to investigate mobile ad hoc network technology. At a middleware forum at FFI in May 2006, it was especially motivating to realize how interested the employees were in one of my solutions; layering.

My thanks go to my supervisor, Ingar Bentstuen, and to my academical advisor, Øivind Kure, who both have been helpful and given me valuable advice and comments. I would also like to thank Jostein Sander for help with practical work around testing and Erlend Larsen for help with the multicast plugin. Thanks also to Knut Øvsthus, Bård Myhre, and Anker Johnsen for reading through my report and giving me comments. In the end I would like to thank Vidar Johansen for great support and encouragement.

Kjeller, June 2006

Lotte Johnsen

Contents

List of Figures

List of Tables

Abbreviations

3GPP	3rd Generation Partnership Project
ABAM	Associativity-Based Ad hoc Multicast
ACK	ACKnowledgement
ADU	Application level Data Unit
ALC	Asynchronous Layered Coding
AMRIS	Ad hoc Multicast Routing protocol utilizing Increasing id-numberS
AmRoute	Ad hoc multicast Routing protocol
AODV	Ad-hoc On-demand Distance Vector
BSS	Basic Service Set
CAMP	Core Assisted Mesh Protocol
CBT	Core-Based Tree
CC	Congestion Control
CCI	Congestion Control Information
CF	Classical Flooding
CGI	Common Gateway Interface
CGSR	Clusterhead-Gateway Switch Routing
CLM	ConnectionLess Multicast
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CTS	Clear to Send
DCF	Distributed Coordination Function
DCT	Discrete-Cosine Transform
DIFS	Dcf InterFrame Space

dB	deciBel
DR	Designated Router
DREAM	Distance Routing Effect Algorithm for Mobility
DSCP	DiffServ Code Point
DV	Distance Vector
DVB-H	Digital Video Broadcasting - Handheld
DVD	Digital Video Disc
DVMRP	Distance Vector Multicast Routing Protocol
DVR	Digital Video Recorder
DV/SC	Digital Video/Still Cameras
EIFS	Extended InterFrame Space
FDT	File Delivery Table
FEC	Forward Error Correction
FFI	Norwegian Defence Research Establishment - (Forsvarets forskningsinstitutt)
FGS	Fine-Grained Scalability
FLID-SL	Fair Layered Increase/Decrease with Static Layering
FLUTE	File Delivery over Unidirectional Transport
GeoCast	Geographic Addressing and Routing
GPS	Global Positioning System
GPSR	Greedy Perimeter Stateless Routing
HSR	Hierarchical State Routing
ICMP	Internet Control Message Protocol
ID	IDentification
IEEE	Institute of Electrical and Electronics Engineers
IGMP	Internet Group Management Protocol
INRIA	Institut National de Recherce en Informatique et en Automatique
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
JPEG	Joint Photographic Experts Group
kbit/s	kilobits per second
LAN	Local Area Network
LANMAR	Landmark Ad Hoc Routing Protocol

LAR	Location-Aided Routing
LBM	Location Based Multicast
LCT	Layered Coding Transport building block
LLC	Logical Link Control
LS	Link State
MP2MP	MultiPoint-to-MultiPoint
MAC	Media Access Control
MAN	Metropolitan Area Network
MANET	Mobile Ad Hoc Network
MBMS	Multimedia Broadcast Multicast Service
MBone	Multicast Backbone
MFP	Manet Forwarding Protocol
MOLSR	Multicast OLSR
MOSPF	Multicast Open Shortest Path First
MPEG	Moving Picture Experts Group
MPR	Multi-Point Relay
NTNU	Norwegian University of Science and Technology - (Norges Teknisk-Naturvitenskapelige Universitet)
ODMRP	On Demand Multicast Routing Protocol
OLSR	Optimized Link State Routing Protocol
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
PC	Personal Computer
PCF	Point Coordination Function
PDA	Personal Digital Assistant
PIFS	Pcf InterFrame Space
PIM-DM	Protocol-Independent Multicast Dense Mode
PIM-SM	Protocol-Independent Multicast Sparse Mode
QoS	Quality of Service
RAT	Robust Audio Tool
RERR	Route ERRor
RFC	Request For Comments
RLC	Receiver-driven Layered Congestion control
RREP	Route REPly
RREQ	Route REQuest

RTS	Request To Send
SBT	Source-Based Tree
SEM	Simple Explicit Multicast
SGM	Small Group Multicast
SIFS	Short InterFrame Space
SIP	Session Initiation Protocol
SMF	Simplified Multicast Forwarding
SPEG	Scalable MPEG
STB	Set-Top Box
TCP	Transmission Control Protocol
TSI	Transport Session Identifier
TTL	Time To Live
TUT	Tampere University of Technology
VIC	VIdeo Conferencing tool
UDP	User Datagram Protocol
VLC	Video Lan Client
WAN	Wide Area Network
WEBRC	Wave and Equation Based Rate Control
WIGMP	Wireless IGMP
WLAN	Wireless Local Area Network
WNet	Wireless Network
Xcast	eXplicit multicast
XML	eXtensible Markup Language
ZRP	Zone Routing Protocol

Abstract

Multicast is used to send data to many receivers simultaneously. Multicast protocols developed for wired networks are not suitable for a Mobile Ad Hoc Network (MANET), mainly because the mobile nodes create a changing topology, and the capacity of the nodes and the links are low compared to a wired network. Hence, it is a challenge to distribute information in an Ad Hoc network.

A goal for this Master's thesis work has been to investigate content distribution in mobile ad hoc networks to find possible ways to reduce necessary volume of distribution data. Scaling according to available bandwidth and alternative distribution methods to multicast has been investigated.

A test architecture consisting of different components useful for content distribution has been configured and tested. Essential components have been:

- Multicast OLSR (MOLSR); a multicast plugin for the MANET protocol Optimized Link State Routing Protocol (OLSR), to forward multicast data
- File Delivery over Unidirectional Transport (FLUTE); a protocol that supports sending multicast data in several layers
- Linux Fedora Core 5 operating system including IEEE 802.11b Wireless Local Area Network (WLAN) adapters
- A topology emulator to simulate different topologies

The first tests were performed using three nodes. Different bit rates and packet sizes were tested to find the best throughput. After adding two more nodes to the network it became clear that the multicast forwarding did not work properly. Many of the first tests hade thus given misleading results. After finding an explanation to the multicast forwarding fault, a modification to the test architecture was done; use an older version of Linux Fedora Core. The new test results then showed that it is possible to forward multicast data using FLUTE on a MANET. As expected, the nodes receive a larger amount of the file sent as the number of hops is decreased; up to 100 percent after one hop, whereas up to 80 percent after four hops. Also, test results showed that low data rates give better throughput than high rates. The best throughput was given after resending the FLUTE session several times. The performance of ad hoc networks is less trustworthy than wired / fixed networks. The amount received varies from 0 to 100 percent. The testing of layering using FLUTE did not give any improvements. However, layered content was not available, so all the channels had the same content. Suggestions to possible ways to provide layered content are described.

The concept of layering is still interesting for MANETs since it provides the ability for nodes with low capacity to receive less content than nodes with high capacity in a relatively simple way. Further tests are needed to see the results using layered content. When implementations of codec frameworks become more available, these may be great for scaling in ad hoc networks.

Chapter 1

Introduction

1.1 Background

Content distribution in traditional wired networks has become relatively straightforward. In these networks the topology is static and most wired networks have sufficient capacity to handle a large amount of traffic. In a Mobile Ad Hoc Network (MANET) however, the situation is different. A MANET is an autonomous collection of mobile users that communicate over relatively "slow" wireless links. Since the nodes are mobile, the network topology may change rapidly and unpredictably over time. The network is infrastructureless and decentralized; all network activity must be executed by the nodes themselves.

In general, there are three ways to send data; unicast (to a single receiver), broadcast (to everyone in a certain area), or multicast (to everyone listening to a certain multicast address). Multicast is often used when distributing content to many receivers simultanously. There are a number of well-established, well-working multicast protocols available for wired networks today. In contrast, there are so far no standard multicast protocols for MANETs. Because of the changing topology and low capacity of MANETs the multicast protocols developed for wired networks cannot be applied in MANETs without modifications; special MANET protocols have to be applied to distribute command and control information and other data.

Content distribution in MANETs is important in application areas, such as emergency/rescue operations, disaster relief efforts, and military networks. In these cases it is important for moving nodes to communicate and distribute content to each other, and these nodes cannot be dependent on wired networks, which are immobile and sometimes not even available.

1.2 The Problem

The problem to be addressed in this Master's thesis is: "How to distribute information in an Ad Hoc network where there is a high degree of mobility and nodes have lower and more dynamic capacity than in wired networks?" To get an overview and to seek content distribution solutions for MANETs different multicast protocols are looked at. In addition, an interesting idea of distributing content in several layers is investigated. One file delivery protocol which offers the possibility of different amounts of channels for nodes with various capacity is tested out in an ad hoc network with five nodes. Two sub problems are given a closer look to narrow the problem range:

1. How to limit the distribution in ad hoc networks?
2. How to scale the distribution after available bandwidth for the different receivers?

1.3 Demarcations

Security and Quality of Service (QoS) will not be paid much attention in this thesis; mainly because they could have been several Master's thesises all by themselves. Other subjects not emphasized in this thesis are:

- Internet Protocol version 6 (IPv6)
- Congestion Control Schemes for MANETs
- Other protocols than File Delivery over Unidirectional Transport (FLUTE) giving scalability
- Video and audio streaming (FLUTE does not support realtime streaming)
- Gateway between different networks scenario

1.4 Outline of the Report

This report first includes some background theory on the underlying IEEE 802.11 Wireless Local Area Network (WLAN). It also contains facts about MANETs, multicast, multicast protocols and other distribution alternatives. The idea of limiting content distribution by scaling the network with layering is described. The file delivery protocol FLUTE is included because it offers possibilities for layering and is used in the testing. Architecture, testing and results are parts of the main work. A discussion of the results as well as a conclusion follows at the end of the report. Details from the tests are included in appendices.

Part I

Theory

Chapter 2

802.11 Wireless LAN

This chapter describes basic wireless LAN technology, and especially 802.11b which is used for testing.

The ad hoc networks use wireless LAN as underlying networks. IEEE 802.11 is used in this Master's work and is described in this section. Other types of wireless LANs exist (e.g. HiperLAN [51] and HomeRF (obsolete) [8]), however they are not much used compared to 802.11.

A WLAN is a wireless local area network. Like all networks, wireless networks transmit data over a network medium. The medium for WLANs is a form of electromagnetic radiation. Two media have been mostly used; Infrared light and radio waves. Radio waves fit MANETs best because they can penetrate most obstructions and offer a wider coverage range than infrared light [17]. WLANs communicate information from one point to another without relying on any physical connection. The devices are mobile. Areas may range from a single room to an entire campus. WLANs can be used either to replace wired Local Area Network (LAN)s, or as an extension of the wired LAN infrastructure [71].

Different standards of WLANs exist. Most of them are part of the working group 11 of the Institute of Electrical and Electronics Engineers (IEEE)

LAN/Metropolitan Area Network (MAN) Standards Committee (IEEE 802). The most popular 802.11 standards are the 802.11b and 802.11g, but there are and will come also 802.11 standards with almost all letters from a to z as extension. They support different speeds and have special features. Some are service enhancements and extensions or corrections to previous specifications. Some of the 802.11 standards are shown in table 2.1.

From a user's perspective IEEE 802.11 reminds very much of Ethernet (wired LAN). Sometimes 802.11 is even referred to as "Wireless Ethernet". Core elements from Ethernet, such as the identification of a node as a 48-bit IEEE 802 Media Access Control (MAC) address, and delivering of frames based on the MAC address, is also present in 802.11 networks. However, frame delivering is much more unreliable in wireless networks, and network administrators need to understand the more complex 802.11 framing.

IEEE 802 specifications are focused on the two bottom layers of the Open Systems Interconnection (OSI) model; the Data link layer (which includes both a Logical Link Control (LLC) sublayer and a MAC sublayer) and the physical layer. Details of transmission and reception are captured by the physical layer while the MAC is a set of rules to determine how to access the medium and send data [17]. This is shown i figure 2.1.

There are certainly limitations within the wireless networks; they do not completely replace wired networks. WLANs are mobile which cause several constrains; mainly the available bandwidth is often reduced. However, the limiting factor in networking a small group of wireless nodes is likely to be the cost of WLAN bandwidth to the supporting infrastructure. A wired Ethernet offers easily 10 Gigabytes per second, while wireless networks have to reduce the rate because of slower hardware and the unreliability of the wireless medium. The use of radio waves may suffer from propagation problems such as multipath interference and shadows. In addition, several security issues are much more dangerous for wireless than wired networks, mainly because they are not protected by physical-access control [17]. Security is beyond the

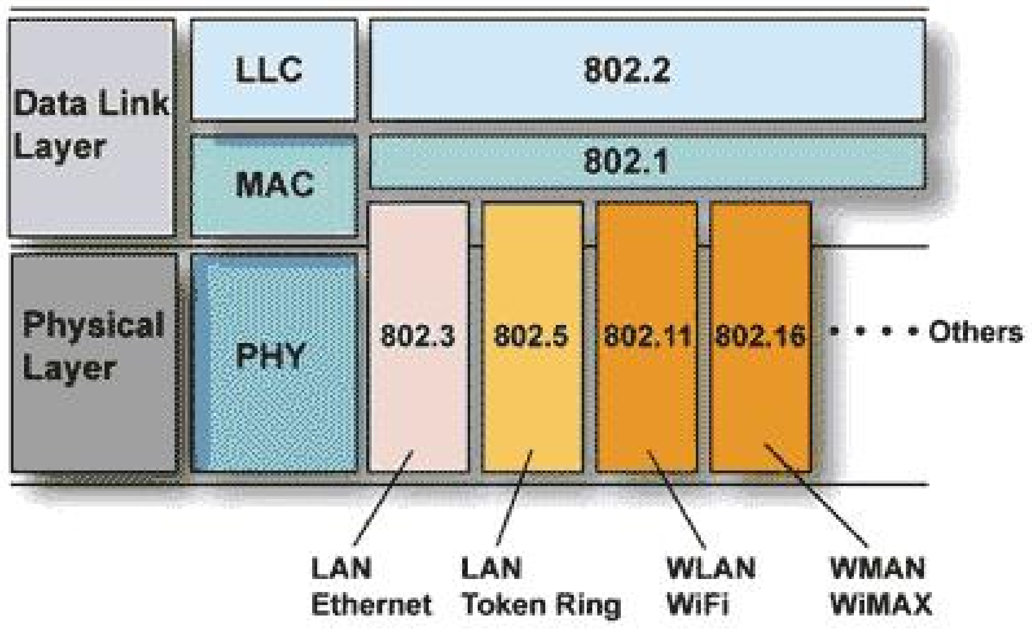

Figure 2.1: The IEEE 802 family and its relation to the two bottom layers of the OSI model [61]

scope of this thesis.

2.1 The hidden node problem

As Ethernet, 802.11 uses a Carrier Sense Multiple Access (CSMA) scheme to control access to the transmission medium. However, collisions waste capacity. Instead of detection, 802.11 uses avoidance (Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA)). There is no centralized controller. Each node in the 802.11 network uses the same method to access and start transmitting to other nodes.

The hidden node problem arises when there are three nodes. All nodes can hear node 2, but node 1 and 3 cannot hear each other (e.g. because of distance). An illustration is shown in figure 2.2. Node 3 is a hidden node for node 1. Since node 1 and 3 cannot hear each other, they may transmit at the same time to node 2. Nodes can only receive one packet at a time. Thus, node 2 will not make sense of anything when receiving packets from nodes 1 and 3 simultaneously. Because the collision is local to node 2, nodes 1 and 3 will not have an indication of the error.

Collisions due to the hidden node problem are hard to detect, mainly because the nodes cannot send and receive at the same time. There are ways to prevent collisions; 802.11 allows nodes to use Request To Send (RTS) and Clear to Send (CTS) signals to clear out an area. For example, if node 1 wants to send data to node 2, it sends an RTS message to node 2. Node 2 answers with a CTS message. This is sent to all nodes in the range of node 2. The CTS message silences the other nodes. Thus, node 1 can send data to node 2 without worrying about collisions. However, when using the RTS/CTS procedure all packets must be positively acknowledged. This produces a fair amount of traffic, and is hence normally used only in high-capacity environments. It is possible to set an RTS threshold for when to use RTS/CTS. For packets larger than the RTS threshold RTS/CTS exchange is used [17].

2.2 Access Control to a Wireless Medium

To control access to a wireless medium a coordination function is needed. There exist two control functions for the 802.11 MAC:

DCF Distributed Coordination Function (DCF) provides the CSMA/CA access mechanism which is similar to the Ethernet access mechanism Carrier Sense Multiple Access with Collision Detection (CSMA/CD). It first checks if the radio link is clear before transmitting. After each packet is sent it uses a random backoff time to avoid collisions. No central control is needed when using DCF. Hence, this can be used in both independent (ad hoc) and infrastructure networks. A possibility for DCF in certain circumstances is to use the RTS/CTS explained in

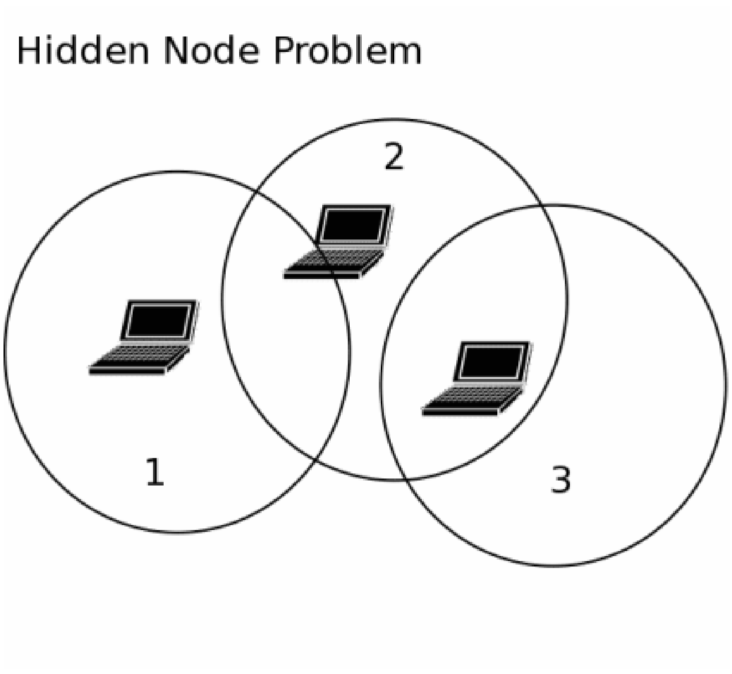

Figure 2.2: The hidden node problem [24]

the previous section to avoid even more collisions.

PCF Point Coordination Function (PCF) provides contention-free services. PCF uses point coordinators which are placed in access points. Hence, PCF is only available in infrastructure networks. Since it is not possible to use PCF in ad hoc networks PCF will not be emphasized in this thesis. The rest of the section will be about DCF.

2.2.1 Interface Spacing

As part of the collision avoidance built into the 802.11 MAC, nodes delay transmission until the medium is idle (see section 2.1). There are four different interframe spaces; Short InterFrame Space (SIFS), Pcf InterFrame Space (PIFS), Dcf InterFrame Space (DIFS) and Extended InterFrame Space (EIFS). SIFS is the shortest space, and high priority packets wait this long before they are resent. Low priority packets wait longer (e.g. by waiting until

EIFS has elapsed) and will thus have less possibility of reaching the target than the high priority packets. An example is shown in figure 2.3.

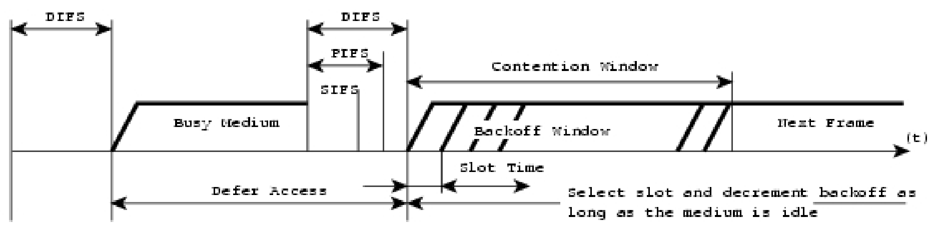

Figure 2.3: 802.11 MAC uses different interframe spaces (e.g. SIFS, PIFS and DIFS) to wait before retrying to send packets out on the medium [22]

2.2.2 More on DCF and Contention-based Access

Most traffic uses DCF. DCF provides a contention-based service. It uses certain rules to access the wireless medium. There are rules for how long to wait before transmitting and retransmitting after the network was idle or busy. Other rules are about special situations, such as error recovery. For example, there are rules for giving acknowledgment; this is the only indication of success. All unicast data must be acknowledged. If errors are detected there will be retransmission after a certain time. Network nodes have two types of retry counters; short and long. The short retry counters are for short packets while long retry counters are for long packets (over the RTS threshold - see section 2.1). They are incremented when a packet transmission fails. They are reset to zero when acknowledgments are received or if broadcast or multicast packets are received.

2.3 Unicast vs. Multicast/Broadcast Handling in 802.11

When sending unicast one can use either Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). TCP includes flow control while UDP does not. Also, 802.11 includes both ACKnowledgement (ACK) and retransmission on the MAC layer for unicast. However, when sending multicast or broadcast 802.11 provides no ACK and retransmission on the MAC layer. Only UDP is used for multicast and broadcast. Hence, the use of multicast or broadcast over 802.11 gives much less reliable network than the use of unicast. Most of the quality of service on the MAC layer mentioned in the previous sections is not possible for multicast and broadcast. The three scenarios are shown in figure 2.4, 2.5 and 2.6.

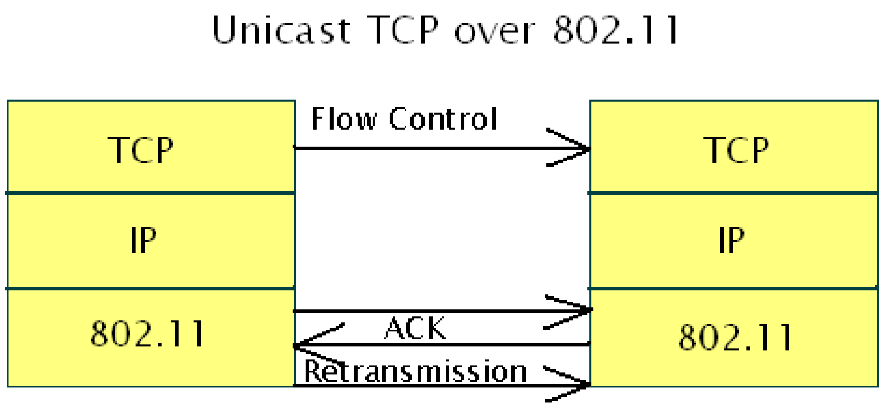

Figure 2.4: Simplified model showing unicast TCP over 802.11 which includes both flow control on the TCP layer and ACK and retransmission on the MAC layer (802.11)

A consequence of no ACK for multicast on the MAC layer is less overhead for multicast than for unicast. For both unicast and multicast the sender waits a DIFS period in addition to a random back-off time before sending a packet. After sending a unicast packet the MAC layer waits a SIFS period for ACK.

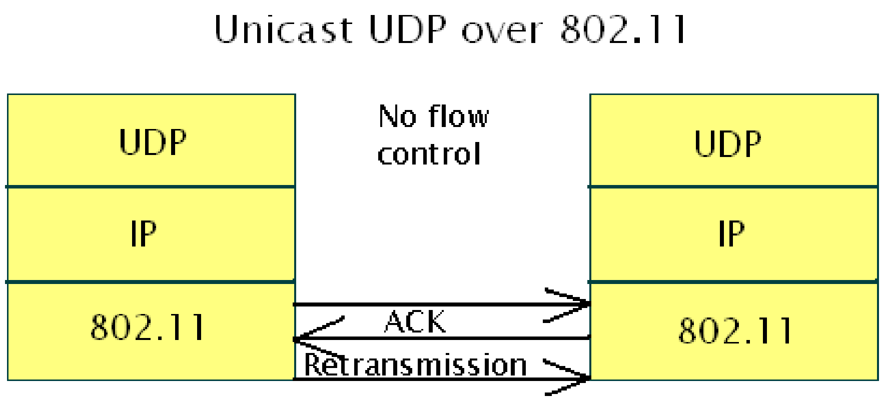

Figure 2.5: Simplified model showing unicast UDP over 802.11 which includes ACK and retransmission on the MAC layer (802.11), but does not include flow control

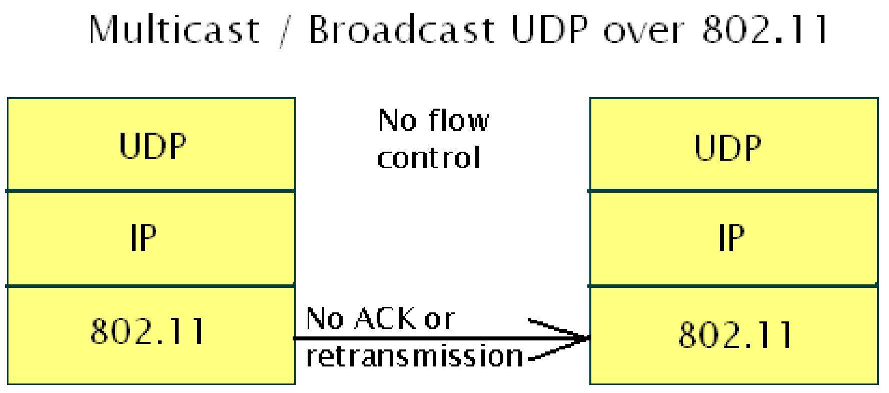

Figure 2.6: Simplified model showing multicast/broadcast UDP over 802.11 which does not include flow control on the UDP layer nor ACK and retransmission on the MAC layer (802.11)

In contrast, no ACK is waited for after sending multicast packets. Hence, the time between each packet sent is less for multicast than for unicast. When data packets become very small or the data rate become large, the amount of time used on waiting DIFS, and eventually SIFS and random back-off time become a larger part of the transmission than for long packets and low bit rates. This means that there is less overhead for large packets or low bit rates. At the same time, if the packets are are larger than 1500 bytes there will be overhead caused by fragmentation on the MAC layer [60].

2.4 IEEE 802.11b

IEEE 802.11b will be described more in detail than the other 802.11 standards because this standard is used in this thesis.

The 802.11b amendment to the original standard was ratified in 1999. The dramatic increase in throughput of 802.11b (compared to the original standard) along with substantial price reductions led to the rapid acceptance of 802.11b as the definitive wireless LAN technology. 802.11b has a maximum raw data rate of 11 Mbit/s and uses the same CSMA/CA media access method defined in the original standard. Due to the CSMA/CA protocol overhead, in practice the maximum 802.11b throughput that an application can achieve is about 5.9 Mbit/s using TCP and 7.1 Mbit/s over using UDP. Although 802.11b cards can operate at 11 Mbit/s, they will scale back to 5.5, then 2, then 1 Mbit/s if signal quality becomes an issue. When using multicast only the deliveries of 1 Mbit/s and 2 Mbit/s are possible [74]. Proprietary extensions to 802.11b have been made. These allow rates up to 44 Mbit/s. However, 802.11g which is backward compatible with 802.11b, has taken over for these proprietary extensions [63].

IEEE 802.11b can operate in two modes; infrastructure mode and ad hoc mode. In infrastructure mode, all traffic passes through a wireless access point. In ad hoc mode the nodes can detect other nodes within a certain range and communicate with each other without the need of these central access points [79]. There are several names for these modes or Basic Service Set (BSS); Independent BSS(IBSS) and Infrastructure BSS (never called IBSS) [17].

802.11 supports the use of multiple channels; frequency bands that can be used simultaneously without any interference with each other. 802.11b supports up to three channels [79].

One drawback for 802.11b is that the frequency range around 2.4 GHz is also used by other equipment, such as microwave ovens. In the US a license is not needed as long as the devices are operated low power. This sets limits on how far and where one can use the network [17].

IEEE standard	*Speed*	*Frequency band*	*Notes*
802.11	1 Mbps & 2 Mbps	2.4 GHz	First standard (1997). Featured both frequency-hopping and direct-sequence modulation techniques. It offered so many options that it was hard to manage interoperability. 802.11 is more a "beta-specification" than a rigid specification.
802.11a	up to 54 Mbps	5 GHz	Second standard (1999), but products not released until late 2000. Because of slow availability of 5 GHz and poor initial product implementations 802.11a did not become as big success as 802.11b. Current 802.11a equipment is improved and is often available for both 802.11a, 802.11b, and 802.11g.
802.11b	5.5 Mbps & 11 Mbps	2.4 GHz	Third standard (1999), but second wave of products. This specification made 802.11 widespread and is still one of the most common 802.11 equipments.
802.11g	up to 54 Mbps	2.4 GHz	Standardized in 2003. Compatible with 802.11b, but higher data rates because of its similarities to 802.11a.
802.11n	up to 540 Mbps	2.4 GHz	Not yet released (release in 2007). Fifty times faster than 802.11b and ten times faster than 802.11a and 802.11g

Table 2.1: Comparison of some of the first 802.11 standards (modified from [17])

Chapter 3

Mobile Ad Hoc Networks

As this thesis faces the problem "How to distribute content in ad hoc networks", knowledge about Mobile ad hoc networks is needed. These kind of networks are described in this chapter.

3.1 What are MANETs

A Mobile Ad Hoc Network (MANET) is an autonomous collection of mobile users that communicate over relatively "slow" wireless links. Since the nodes are mobile, the network topology may change rapidly and unpredictably over time. This network without infrastructure is decentralized; all network activity, including discovering the topology and delivering messages must be executed by the nodes themselves. Hence, routing functionality will have to be incorporated into the mobile nodes [46].

Since the nodes communicate over wireless links, they have to contend with the effects of radio communication in WLAN (see section 2), such as noise, fading, and interference. In addition, the links typically have less bandwidth than a wired network. Each node in a wireless ad hoc network acts both as a host and a router, and the control of the network is distributed among the nodes. The network topology is in general dynamic, because the connectivity among the nodes may vary with time due to node departures, new node arrivals, and the fact that the nodes are mobile [76].

3.2 MANET Routing

Generally routing protocols in MANETs are either based on the Link State (LS) routing-algorithm or on the Distance Vector (DV) routing-algorithm. Common for both of these algorithms is that they try to find the shortest path from the source node to the destination node. The main difference is that in LS based routing a global network topology is maintained in every node of the network. In DV based routing the nodes only maintain information of and exchange information with their adjacent nodes. Keeping track of many other nodes in a MANET may produce overhead, especially when the network is large. Therefore one of the most important issues in MANET design is to come up with schemes that will contribute to reduce routing overhead. Hybrids of LS and DV based routing protocols exist.

Another dimension of MANET routing protocols is difference of being reactive (on-demand) or proactive. A reactive protocol will only try to find a route when something is to be sent. There are no updates given when nothing is sent. Nodes start finding possible routes first when they want to send something. In contrast, a proactive protocol will continuously update the network about the nodes. Hence, there will updated routes to all available nodes independent on packets are sent or not. Proactive protocols are best where the topology is rather stable, whereas reactive protocols are best suited for networks with high degree of mobility.

3.2.1 AODV - Ad Hoc On-demand Distance Vector

The reactive Ad-hoc On-demand Distance Vector (AODV) routing protocol is based on DV-routing. AODV routing establishes a two way connection with the destination node; in case of a link failure the connection is reestablished [55]. Each node saves the following information when it receives a Route REQuest (RREQ) packet: The destination node's address and the last known sequence number of that destination, the source node's address, a hop count (initialized to zero) and a RREQ IDentification (ID). Together the source address and the RREQ ID can be used to detect duplicates; hence they are used to stop RREQ messages from going in loops. For each RREQ message a node receives, it checks whether it has an unexpired route to its destination. If the node has a route and the sequence number is at least as great as the sequence number in the RREQ, the node replies with a Route REPly (RREP) message. The RREP follows the reverse path of the RREQ, and creates a two-way communication link between the source and the destination. Since each route has an associated lifetime, the lifetime is refreshed whenever the route is reused. Both the RREQ and the RREP are needed to create the path. If the source receives more than one RREP it selects the one with the highest sequence number and the lowest hop count.

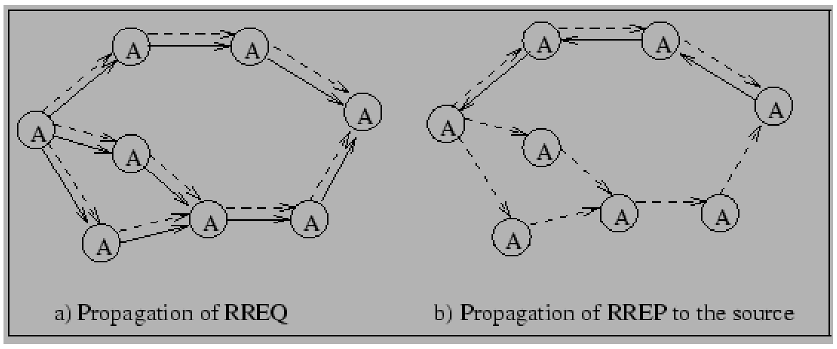

Figure 3.1: AODV route request [21]

AODV contains a number of optimizations that differ from basic DV-routing. The most important is the expanding ring where RREQ propagation is controlled by modifying the Time To Live (TTL) value of the packet, and local repair of broken links. In the local repair optimization, the nodes closest to the link break attempts to repair the link instead of sending a Route ERRor (RERR) message back to the source. The local repair optimization reduces network traffic by avoiding unnecessary route discovery operations in the network. When the source receives a RERR it initializes the route discovery routine again [4]. Figure 3.1 illustrates AODV route request with propagation of RREQ and RREP messages.

3.2.2 OLSR - Optimized Link State Routing Protocol

Optimized Link State Routing Protocol (OLSR) is a proactive LS protocol. It periodically exchanges topology information with other nodes in the network. The protocol uses Multi-Point Relay (MPR)s (forwarding nodes) to reduce the number of "superfluous" broadcast packet retransmissions and also the size of the LS update packets, leading to efficient flooding of control messages in the network [76, 11].

In figure 3.2 the MPR set of node A has been calculated and is set to be node E, F and G. The MPR set contains a minimum set of neighbor nodes that is needed to establish a link with all two-hops nodes. Neighborhood information is periodically exchanged through a HELLO message, which is first used to calculate the MPR set. When node A has calculated its MPR set it broadcasts the set to its neighbors. However, only nodes included in the MPR set need to relay the message. Node A only records other nodes that choose it as one of their MPR nodes and stores them in a list called MPR selectors. As a result nodes that want to communicate with node A must do so through node A MPR nodes and vice versa [4].

The MPR sets help reducing the number of link state updates that have to be sent in a dense network. If the network is sparse every node in the network becomes a MPR node and OLSR is in practice reduced to LS routing.

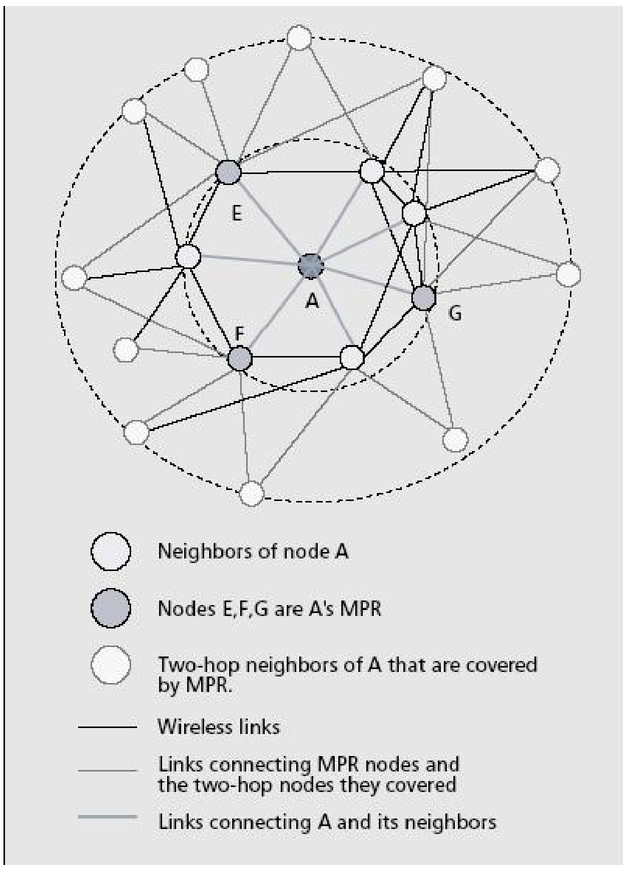
E
A
F
G
Neighbors of node A
Nodes E,F,G are A's MPR
Two-hop neighbors of A that are covered by MPR.
Wireless links
Links connecting MPR nodes and the two-hop nodes they covered
Links connecting A and its neighbors

Figure 3.2: OLSR protocol [76]

Therefore OLSR is best suited in dense networks.

3.2.3 Hierarchical and Geographical Routing Protocols

Flat routing protocols, such as AODV and OLSR, adopt a flat addressing scheme; each node plays an equal role. In contrast, hierarchical routing usually assigns different roles to network nodes. When the size of a MANET increases the flat routing protocols may not be sufficient. Then either a hierarchical or a geographic routing protocol would be a better solution. The hierarchical routing protocols organize the nodes in hierarchies and have smaller routing tables because the nodes only need to keep track of their levels in the hierarchy. Also, in search for destinations the amount of flooding packets is reduced. However, the hierarchical routing protocols may also produce overhead to maintain the hierarchical structure. The geographic routing protocols use the knowledge of node locations to organize the structure of a MANET. They may produce overhead when exchanging coordinates, but all in all they can become more scalable and effective than the flat routing protocols. Examples of hierarchical routing protocols are Clusterhead-Gateway Switch Routing (CGSR), Hierarchical State Routing (HSR), Zone Routing Protocol (ZRP) and Landmark Ad Hoc Routing Protocol (LANMAR). Examples of geographical routing protocols are Location-Aided Routing (LAR), Geographic Addressing and Routing (GeoCast), Distance Routing Effect Algorithm for Mobility (DREAM) and Greedy Perimeter Stateless Routing (GPSR). They are all described in [28]. They will not be described in detail in this thesis because the network considered here is small enough for flat routing protocols to be effective.

3.3 MANET Application Areas

Significant examples of MANETs include establishing survivable, efficient and dynamic communication for emergency/rescue operations, disaster relief efforts, and military networks. Such network scenarios cannot rely on centralized and organized connectivity, and can be conceived as applications of MANETs. However, MANETs are not solely intended for disconnected autonomous operations or scaled scenarios (e.g. hundreds or even thousands of cooperating wireless nodes in a region). They may be used as hybrid infrastructure extensions and in fixed infrastructure operations.

A hybrid infrastructure extension is a dynamic enhancement to a home or campus wireless networking environment. It provides extended service and allows low-cost, low-complexity dynamic adjustments to provide coverage regions and range extensions away from the more fixed infrastructure backbone networks.

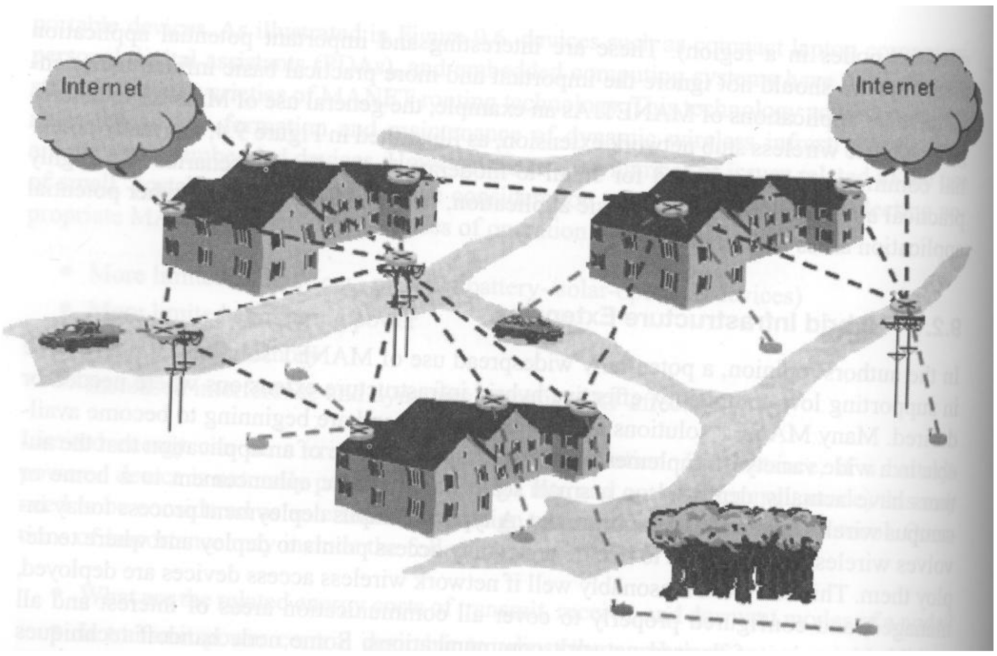

Figure 3.3: Example of hybrid infrastructure extension [4]

Figure 3.3 shows a fictitious, but practical example of a hybrid infrastructure extension. Nodes moving or operating on limited energy may be low-preference routing nodes, thus providing more physical stability to the overall routing grid as well. When allowing certain MANET nodes to be preferred over other nodes in a neighborhood, the more passive MANET nodes may provide range extension and dynamic routing functions on an as-needed basis. This may be appropriate within a campus, community, or a robotic, sensor, or localized business application.

In contrast to the hybrid infrastructure extension there are no fixed access nodes or gateway points to provide confirmation coordination in a fixed infrastructureless operation. Here the participating nodes will have to operate in a peer-to-peer fashion, with appropriate applications and protocols. Examples of esoteric ad hoc applications are ad hoc conferencing, business meeting capabilities and ad hoc homeland defense and disaster relief networks. They will require more distributed forms of auto configuration, service discovery, and management.

There are also other network application areas; cooperatives and sensors. In cooperatives a community of interest (e.g. a small town government, infrastructure-lacking world region, group of interested individuals/club) own and operate a network infrastructure together. These networks could deploy MANET technology to support a self-organizing, adaptive infrastructure. This will be desirable in disadvantaged rural regions and developing countries with lack of resources or an environment not suited for significant fixed-infrastructure developments and services. MANET technology can be used here to help building and operating inexpensive network infrastructure services.

Sensor networks may be more scaled and capable using MANET technology. Commercial, environmental and military applications are all interested in this. MANET technology can support broad applications of self-organizing and distributed sensor networks [4].

Chapter 4

Multicast

Multicasting can be used to efficiently distribute information to many nodes simultaneously. Multicast protocols are well established for wired networks. However, when the nodes become increasingly mobile, the multicast protocols for wired networks cannot be used as they are. They either have to be changed or other multicast protocols have to be used. This chapter includes information about multicast in wired networks and in MANETs.

4.1 What is Multicast

Multicasting is transmission of datagrams to a group of zero or more receivers identified by a single destination address and it is intended for group-oriented computing [31]. While broadcast sends data to all nodes and unicast to certain local addresses, multicast delivers data to a single multicast address which belongs to a multicast group that receivers can join and leave as they want. Internet Protocol (IP) multicast address ranges and uses are shown in table 4.1. The differences between broadcast, unicast and multicast are illustrated in figure 4.1. Multicast can also be termed "selective broadcast", as the receivers of a multicast group are a subset of all receivers. Multicasting can be single-source (point-to-multipoint) or multiple sources (multipoint-to-multipoint). To support multicasting, existing routers must run an appropriate multicast routing software, in addition to traditional unicast routing algorithms [37]. The reliability of delivery of multicast packets is about the same as for unicast packets in wired networks. IP datagrams, for example, are not guaranteed to arrive intact at all members of the destination group or in the original order relative to other datagrams [13].

Range Start Address	*Range End Address*	*Description*
224.0.0.0	224.0.0.255	Reserved for special "well-known" multicast addresses.
224.0.1.0	238.255.255.255	Globally scoped (Internet-wide) multicast addresses.
239.0.0.0	239.255.255.255	Administratively scoped (local) multicast addresses.

Table 4.1: IP Multicast Address Ranges and Uses

4.2 Multicast Applications, Benefits and Challenges

Multicasting is intended for group-oriented computing. Application examples are close collaborations of teams (e.g. rescue patrols, military battalions, scientists, etc) with requirement for audio and video conferencing and sharing of text and images [31].

There are several benefits of using multicast in these group-oriented applications. One benefit is that the receivers' local addresses are transparent to the multicast sender; receivers can change their addresses without notifying the sender. In contrast to multicast, several unicast messages would cause

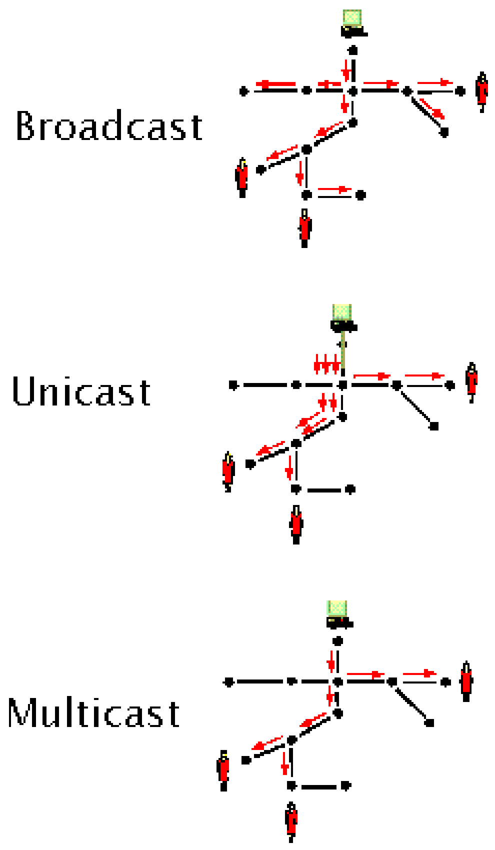

Figure 4.1: Comparison of broadcast, unicast, and multicast technologies. The computer icon represents a sender, and the person icons represent recipients. [52]

much processing at the sender and router. Unicast would also cause delivery delay and bandwidth consumption. Multicast reduces the communication costs when sending to multiple destinations [53].

There are also challenges for multicast in MANETs; for example, the multicast group members are mobile and move around, thus making it hard to use fixed multicast topology. Another key challenge is that transient loops may arrive when reconfiguring multicast trees. To avoid channel overhead the tree reconfiguring schemes have to be simple [31].

4.3 Multicast in fixed/wired networks

In fixed/wired networks, such as the Internet, there are many solutions for multicasting that work well. Examples are multicast variants of routing protocols, Protocol-Independent Multicast Sparse Mode (PIM-SM) [16] and Protocol-Independent Multicast Dense Mode (PIM-DM) [1]. A large amount of other protocols for wired networks exists. They will not be described in this thesis which focuses on multicast protocols for MANETs. However, information about possible solutions can be found in [31, 67].

4.4 Multicast in MANETs

Multicasting in MANETs differs from multicasting in fixed/wired networks. There are no standard and well-working multicast protocols in MANETs. The mobility of nodes in the network can significantly impact the ability of a network to deliver multicast packets successfully to all intended receivers. Broadcasting is often used for communication in MANETs since traditional radios are based on omni-directional antennas. However, problems arise in ad hoc wireless networks multicasting due to mobility of sources, destinations and intermediate nodes in the distribution tree. In addition, there are hidden node problems (see section 2.1) and the presence of multicast group dynamics [37].

4.4.1 Multicast Protocols

It is normal to divide ad hoc multicast protocols into different categories. One classification is if the protocols are proactive (on-demand) or reactive protocols [76, 25] while another classification consists of flat versus hierarchical and geographical protocols (see section 3.2.3). A third way of sorting multicast protocols is dividing them into either meshes, trees or pure flooding. A fourth classification provided by [59] is to divide the ad hoc multicast protocols into six classes as illustrated in figure 4.2:

- Source-Based Tree
- Core-Based Tree
- Multicast Mesh
- Group-Based Forwarding
- Location Based
- Stability-Based Tree

Source-Based Multicast Tree

In Source-Based Tree (SBT) (figure 4.3) a multicast tree is established and maintained for each multicast source node in each multicast group. The number of multicast trees will be the number of groups times the number of sources. This leads to scalability problems when the number of nodes and groups increases. Also, SBT may require prior knowledge of topology information to establish the trees. A positive characteristic, however, is that each multicast packet will be forwarded along the most efficient path. Examples of

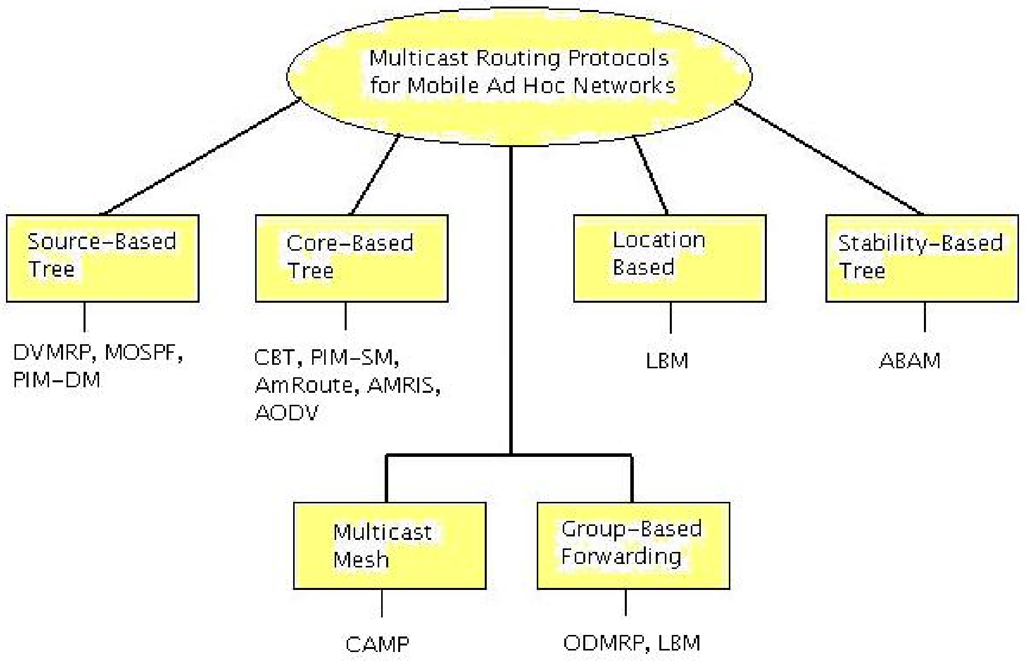

Figure 4.2: Classification of had hoc routing protocols (slightly modified from [59])

SBT protocols are Distance Vector Multicast Routing Protocol (DVMRP), Multicast Open Shortest Path First (MOSPF) and PIM-DM.

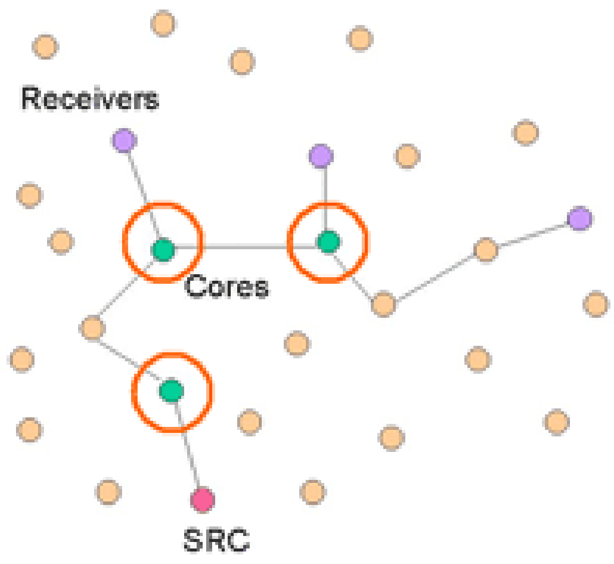

Figure 4.3: Principle of Source-based Tree in an Ad Hoc Wireless Network [37]

Core-Based Multicast Tree

A more scalable routing protocol than SBT is the Core-Based Tree (CBT) protocol. It builds one single tree which is shared among the multicast groups. A special node; the *core node* is selected to establish the shared tree. A core selection algorithm is used to select the core node. The shared tree is either unidirectional or bidirectional. In a unidirectional shared tree the packets are unicasted to the core node (which is the root of the tree) and then distributed along the tree until they reach their destination. However, in bidirectional trees the packets do not have to travel via the core node. They are sent along the branches of the tree to their destination. This bidirectional approach is more efficient in terms of both communication performance and forwarding overhead (number of transmissions needed to forward the multicast packet toward the destination). An illustration of the CBT is found in figure 4.4.

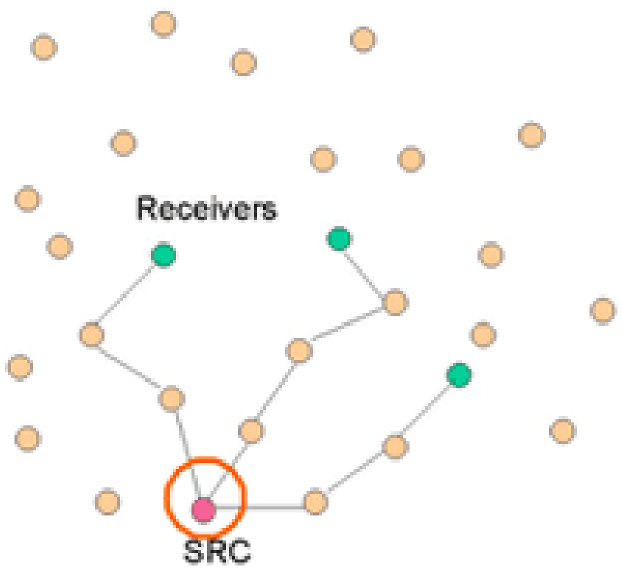

Figure 4.4: The Principle of Core-based Multicast for Ad Hoc Wireless Networks [37]

Three disadvantages of the CBT are:

1. All traffic is concentrated on the shared links. This may lead to congestion at the shared links.
2. Multicast packets tend to be forwarded on less optimal paths because they are forced to transit along the shared links.
3. The core node becomes a single point of failure.

Examples of CBT protocols are CBT [3], PIM-SM [16], Ad hoc multicast Routing protocol (AmRoute) [78], Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS) [75] and AODV [55].

Multicast Mesh

When the rate of link changes increases the tree reconfigurations are not desirable anymore. Multicast Mesh (illustration of a mesh in figure 4.5) provides an alternative; a mesh for each multicast group. The difference between a mesh and a tree is that nodes in mesh can have multiple parents. An advantage of Multicast Mesh is its redundancy; multiple redundant paths avoid frequent mesh reconfiguration. Also, it mimizes the disruption of on-going multicast sessions and reduces protocol overhead. However, the approach leads to unnecessary forwarding of multicast packets along all redundant paths in the mesh. This leads to additional data forwarding overhead. Core Assisted Mesh Protocol (CAMP) is one example of a Multicast Mesh protocol.

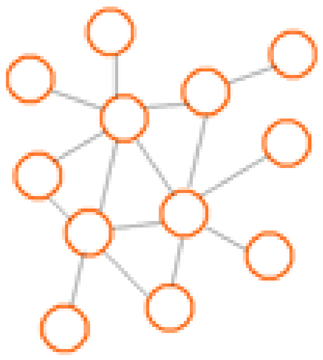

Figure 4.5: A mesh is an interconnection of nodes over multiple paths [37]

Group-Based Forwarding

In Group-Based Forwarding a group of nodes that acts as multicast forwarding nodes for each multicast group. This approach simplifies each node; they do not have to exchange link information with neighbors. Only the forwarding group nodes forward multicast packets. They forward all multicast packets that are not duplicated. Hence, fewer states are kept at each intermediate node and redundant paths are available. Group-based multicast is illustrated in figure 4.6. Example protocols are On Demand Multicast Routing Protocol (ODMRP) Location Based Multicast (LBM).

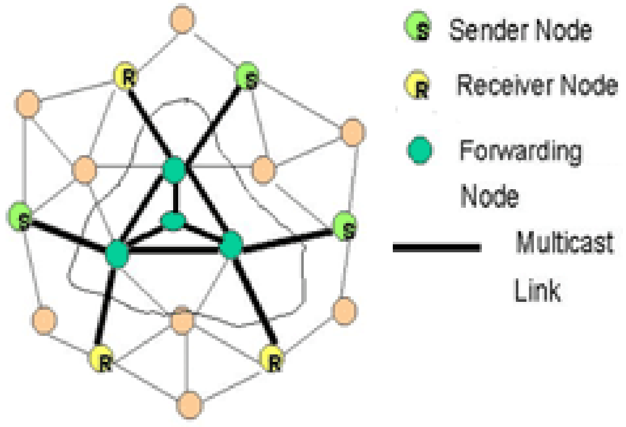

Figure 4.6: Ad Hoc On-Demand Group-Based Multicast [37]

Location Based

Location Based Multicast is closely related to group-based forwarding. In LBM which is an example of location based multicast protocols, a multicast group is defined as all the nodes within a *multicast region*. This region is a rectangle with limited by four corner coordinates. If the source is outside the multicast region the packets have to be forwarded to the multicast region. Nodes in a *forwarding region* will forward the packets they receive. The forwarding region has to encompass the multicast region. In addition it is important to ensure that network connectivity exists between the source node and multicast region.

LBM assumes that every node knows its location. This may be done with Global Positioning System (GPS) technology. Thus, all nodes in the MANET need to install GPS receivers. To find out if a node is in the forwarding zone there are three solutions. One is to simply flood the network. This will produce much control and forwarding overhead. A better solution is to let the source compute a forwarding zone. This forwarding zone can be a rectangular region around the multicast region. All the nodes in the forwarding zone will forward the data. The third solution is to let the distance between the previous node and the destination node decide whether to forward the data or not. The node compares its position with the previous node and the destination location and forwards the data only if it is closer to the destination than the previous node. Solution two and three are shown in figure 4.7 respectively.

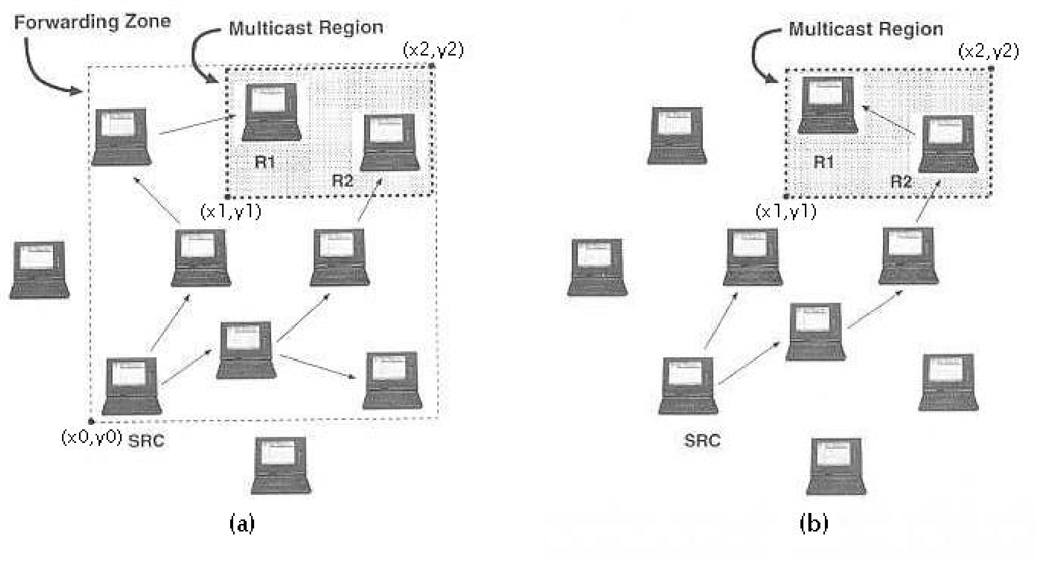

Figure 4.7: LBM forwarding scheme [59]

LBM is not only positive. One negative issue is that it expects all nodes to know their location and that accurate positioning is obtainable. A source needs to know the location of the destination before it can send data. Thus, positional inaccuracy may cause problems. In addition, a node may not receive multicast packets even if it is within the multicast region. One way to come around the inaccuracy is to introduce a *threshold* parameter. The forwarding zone may be expanded by this threshold value to reach the nodes in the gray zone. However, this produces overhead in the forwarding of multicast data.

Stability-Based Tree

Associativity-Based Ad hoc Multicast (ABAM) which is an example of stability-based tree protocols is an on-demand protocol. It takes into account the link and route stabilities and applies *associativity* to ad hoc multicast routing. The concept of association stability is utilized during multicast tree discovery, selection, and reconfiguration. This allows long-lived routes to be selected, thereby reducing the frequency of route reconstructions. There are four components in ABAM:

1. Multicast tree formation:
   This is about making a communication path before the multicast packets are sent. It is done for each multicast session.

2. Handling host membership dynamics:
   This component includes procedures for dynamics handling; it is based on user's demand to stay in a session or not.

3. Handling node mobility:
   This is the core part of the mobility management; it lets a moving node continue its multicast session even if the multicast tree changes.

4. Multicast tree deletion and expiration:
   When a multicast route is no longer necessary this component release the network resources from the session

ABAM is shown by [59] to be robust since the repair of a group can be triggered by a node in the tree or by the migrated node itself. This protocol is also able to handle multicast group dynamics when mobile hosts decide to join or leave an existing group. When using certain scenarios and group sizes (as done in [59]) ABAM has low communication overhead and performs well in throughput [18].

Evaluation of the Protocols

Currently innovations within multicasting in MANETs rely on a variety of concepts, such as associativity, location, group, etc. None of the protocols described above seem to be "The very best Solution". One has to consider tradeoffs in terms of the amount of control overhead incurred and the multicast forward performance in their protocol design [37]. Proper answers to which multicast protocols to use are scenario-dependent.

4.4.2 Multicast Protocol used in this Thesis: MOLSR and Simplified Multicast

Multicast OLSR (MOLSR) is a multicast version of OLSR (see section 3.2.2). It is a proactive protocol which builds a multicast structure in order to route multicast traffic in an ad hoc network. The information MOLSR receives from OLSR are [9]:

- Which nodes are in the group
- Shortest hop count to all group members

There are different implementations of MOLSR [20, 35]. Some are implemented using source-based tree (see section 4.4.1). The implementation used in this Master's thesis [35], however, is implemented using Simplified multicast with MPR. No multicast tree is built.

Simplified Multicast Forwarding (SMF) uses a simplified forwarding multicast mechanism that delivers multicast packets to all MANET multicast receivers within a MANET routing area [25]. There are several algorithms which can be used within SMF for neighbor discovery [25]:

- Classical Flooding (CF)
- Multipoint Relay (MPR)

- Non-Secure Specific MPR (NS-MPR)
- MPR Connected Dominated Set Extension (MPR-CDS)
- Essential Connected Dominating Set (E-CDS)

Classical Flooding (CF) just floods the entire network. MPR limits the flooding with relay nodes which are responsible for forwarding multicast data. All these algorithms are described more in detail in [25]. The SMF implementation used in this Master's thesis builds on the MPR algorithm.

4.5 Alternative solutions to normal multicast

Multicast is not the only solution to handle content distribution to several receivers at the same time. Unicast and broadcast are already mentioned (section 4.1). Other distribution variants which can be useful in special cases are Small Group Multicast, Connectionless Multicast and eXplicit multicast (Xcast). They were all presented around 1998 / 1999. Among these protocols Xcast is the one which is given the most effort in this thesis. That is because Xcast has got the most attention and is still being developed. Small Group Multicast and Connectionless Multicast are just briefly described.

4.5.1 Small Group Multicast

Many multicast schemes support very large multicast groups, however the number of groups is limited. Small Group Multicast (SGM), in contrast, supports a very large number of small multicast groups. SGM senders include a list of destination addresses in the packet headers. SGM-capable routers do not need to store states for various multicast groups. They just parse the SGM headers and use ordinary unicast routing tables to determine where to route the packets. There is no multicast protocol is used in SGM. SGM is not applicable where the multicast groups become large [6].

4.5.2 ConnectionLess Multicast

ConnectionLess Multicast (CLM) is a mechanism for MultiPoint-to-MultiPoint (MP2MP) communication in IP networks. Instead of a group address, a list of member addresses is encoded in the data packets. The traditional Host Group Model [12] for IP multicast requires a globally unique address for each session. To support this model, multicast routing protocols create a state per group in the routers. Like any connection oriented protocol, the Host Group Model suffers from scalability problems in backbone networks where the number of groups to be maintained can be very large. CLM does not have this problem, and additionally has some other advantages. Its limitation lies in the number of members per multicast session, not in the number of sessions. CLM will not replace the traditional multicast model. CLM offers an alternative for MP2MP communication in the cases where traditional multicast becomes problematic. It supports different modes of operation: an end-to-end mode in close conjunction with Session Initiation Protocol (SIP) [19], as well as an interworking mode with PIM-SM and Simple Multicast. Both Internet Protocol version 4 (IPv4) and IPv6 are considered [49, 48].

4.5.3 Xcast

Xcast is a type of multicasting that allows packets to be addressed to multiple destinations [26]. It is best suited for small group communications. A list of destination addresses is explicitly included in each data packet. While traditional multicast lets the group model determine the host's group members, routers using Xcast do not have to keep any multicast session state. Xcast handles each packet by referring to its unicast routing table to determine the next hop for each destination listed in the packet. When branching is needed, the packet is replicated with a subset of the destinations. The sender has to be fully aware of the session members, which it takes as prior information [30, 5]. There are no Request For Comments (RFC)s for Xcast, however the work is still going on for this distribution variant.

The effect of the three distribution models unicast, multicast and Xcast when they are applied to a simple networking scenario is illustrated in figure 4.8.

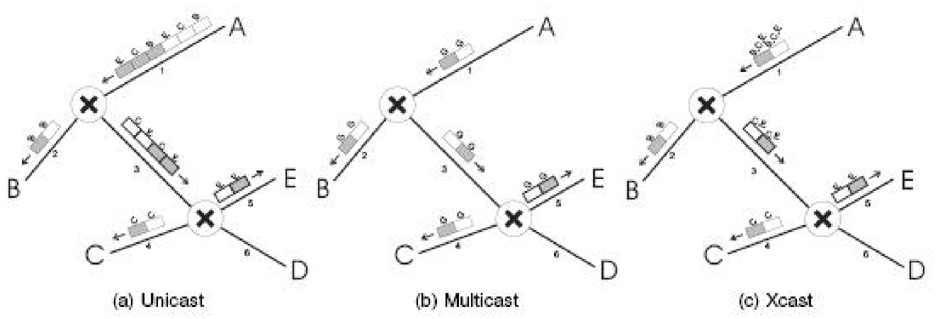

Figure 4.8: Three different distribution models [26]

There are several existing Xcast proposals. One of the most advanced methods is to tunnel Xcast messages as unicast messages between Xcast-enabled hosts and routers. A problem for the Internet is that the adding of support for a new delivery model requires the whole host and router infrastructure to be updated. To avoid this, many protocols use a deployment strategy to introduce the model successively. The tunneling solution for Xcast makes it possible to use Xcast without giving support for this protocol in all hosts and routers. Another solution uses compression devices to examine packets queued at routers and recode unicast traffic as Xcast traffic. The compression devices are added to the outputs and inputs of routers and cause only compressed Xcast traffic to ever travel over a link, and only normal unicast traffic to enter or exit a link. The routers themselves do not need any modification. The compression devices improve the performance over unicasting. However, in all other cases multicast has been proved to be far more superior [26].

Xcast Protocol Stacks and Applications

According to [5] there are two Xcast protocol stacks (software implementations); XCAST4 and XCAST6. These protocol stacks are used for IPv4 and IPv6, respectively. There are also several Xcast applications:

- Multicast Backbone (MBone) Tools
- xcgroup
- ping6x, traceroute6x
- Live CDs

There are implementations of these applications for Linux and FreeBSD. Among the **MBone tools** the well-known VIdeo Conferencing tool (VIC) [72] and Robust Audio Tool (RAT) [43] have been modified to integrate with Xcast. The amount of additional code for these modifications are less than 200 lines for both VIC and RAT. The only drawback is that users of these MBone tools need to specify a long list of IP addresses before conducting an experiment. This is not always trivial because it is difficult to maintain consistent membership information. A solution to this problem is a Common Gateway Interface (CGI) script for httpd (Hyper Text Transfer Protocol Daemon - e.g. a web server) and a client program called "**xcgroup**". The client xcgroup periodically sends a query to the CGI script via http. The CGI script keeps track of all the queries and sends a list of all the clients back to the clients. Hence, the member information stay updated.

Xcast tunneling is already mentioned above. This semi-permeable tunneling lacks a mechanism to check reachability of an overlay Xcast delivery tree. The applications **ping6x** and **traceroute6x** have been developed and modified from ping6 and traceroute6 to meet the demand of reaching this overlay Xcast delivery tree. Operators send probing packets to the destinations specified in the Xcast header. Internet Control Message Protocol (ICMP) [56] messages are sent back in response. By referring to the information from all the ICMP messages it is possible to identify the overlay Xcast delivery tree.

Live CDs are preinstalled CDs with software needed to execute Xcast. These are made to make it easier to spread Xcast-enabled routers in networks. Normally, it is needed to re-compile the computer kernel and install a new library to use XCAST6.

Xcast Extensions

Xcast+ and Simple Explicit Multicast (SEM) are two extensions of Xcast. They are described here:

Xcast is best suited for small networks and personal usage, e.g. chatting with a group of friends or carrying out a small scale business meeting. To be able to use Xcast in a larger scale the extension **Xcast+** is developed. Sources and destinations are associated to a Designated Router (DR). Xcast+ encodes the set of the group members' DRs instead of encoding the group members themselves. When a source wants to send a message to a group, it sends a multicast packet. The DRs receive the multicast packet, convert it to an Xcast packet and forward it to their subnetworks.

Another extension of Xcast is called **SEM**. This approach uses deploying routers at branching points of the Xcast delivery tree to keep track of the tuples of sources and groups and of the next branching router. The probing packets are sent from the source to the DR using Xcast+. Hence, the SEM delivery tree is not affected if there is asymmetry in the unicast paths [5] (A path between two nodes A and B is symmetric if the path is both the shortest path from A to B as well as the shortest path from B to A).

Problems with Xcast

Several problems are still unresolved and needs to be further researched. Among these are:

- Group management should be separated completely from the routing function of Xcast.
- Transport Layer Support, such as congestion control and flow control, should be improved.
- Xcast and broadcast should harmonize better; some research topics focus on how to effectively utilize the broadcast capability of existing access systems, such as satellite or wireless, with Xcast. This topic is the most relevant for this Master's thesis.
- Tunneling with Xcast, especially in the inter-domain and tunneling encapsulation, needs to be further investigated.
- DiffServ Code Point (DSCP) (section 8.2 in [5]) usage needs more investigation for its inter-domain use, such as re-writing and re-mapping. The DSCPs may have to be rewritten as packets cross inter-domain boundaries.
- Where to place the Xcast routers in the network (for example in the core network or at the edge) to achieve less unnecessary traffic, needs to be investigated.

Chapter 5

Scaling

One of the most evident characteristics of MANET nodes is that their capacities vary. Some nodes have the possibility to receive data at high bit rates and with high quality; while other nodes can only receive data with low quality and at low bit rate. This chapter describes different ways to scale the network, so that the different nodes in MANETs have the possibility to receive data according to their capacities. Here, scalability does not mean to support very small or very large multicast groups or networks; instead scaleability means the possibility of supporting low or high capacities.

5.1 Layered coding

In networks with limited capacity, such as ad hoc networks, layered content can be a solution. The layering can be done in two ways, according to the transfer mode; streaming or downloading.

For streaming data, e.g. video, the content layers can have different qualities. The lowest layer will have low quality. All receivers need this layer to see the video content. If the receivers and the network allow it, additional layers can be received. A file without layered coding and a file with layers are shown in figure 5.1.

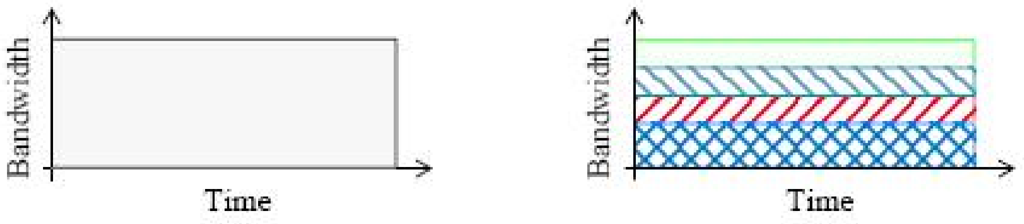

Figure 5.1: Non-layered coding versus layered coding [80]

For downloading a file, e.g. downloading a movie, this can be done at different rates. If possible, the rate can be high. However, if the network capacity is low, the rate can be lower to adjust to the needs.

5.1.1 Codecs

Codec stands for COder/DECoder or COmpression/DECompression. There are codecs for both audio and video. A video codec is a program or an algorithm in a media player which compresses / decompresses media streams. On the sender side the media is compressed and sent out on the network, and on the receiver side the media is decompressed and played. The codecs often have a certain error rate to be able to transfer data in real time. Different codecs use different methods to code and decode the media. Codecs explained in this thesis are Moving Picture Experts Group (MPEG)-1, MPEG-4 and their scalable versions; Scalable MPEG (SPEG) and fine-grained scalability.

MPEG-1

MPEG-1 is a video codec. Almost every computer in the world can play this codec, and very few DVD players do not support it. In terms of technical design, the most significant enhancements in MPEG-1 relative to H.261 were half-pel and bi-predictive motion compensation support [70].

In MPEG video, each frame is broken down into 8x8 pixel blocks, which are converted to corresponding 8x8 blocks of coefficients using the Discrete-Cosine Transform (DCT). Quantization, strategic removal of low order bits from these coefficients, is the primary basis for compression gains in MPEG and very many other similar compression schemes [33].

MPEG-4

MPEG-4 is a codec where each object in the scene is separately compressed. An example is the news on TV. Here the news reader is one object, the table is one object and the bluescreen in the back is one object. The coded part of each object is sent in separate elementary streams. These elementary streams are syncronized together and sent as multiplexed streams. The objects that do not move very much do not need much coding. Hence, it is possible to save effort on these objects.

When it comes to layering, it is possible to to demultiplex the multiplexed stream and code the different elementary streams with various quality. The streams can then be multiplexed again and sent as multiplexed streams with different quality. MPEG-4 was standardized in 2002, however there are few implementations available of doing the demultiplexing explained here. Figure 5.2 shows the elementary streams multiplexed into multiplexed streams.

The MPEG-4 standard is the current state of the art of ITU-T [64] and MPEG [66] standardized compression technology, and is rapidly gaining adoption into a wide variety of applications. It contains a number of significant advances in compression capability, and it has recently been adopted into a number of company products, including for example the PlayStation Portable, iPod, the Nero Digital product suite, the CoreAVC video decoder, Mac OS X v10.4, as well as HD-DVD/Blu-Ray.

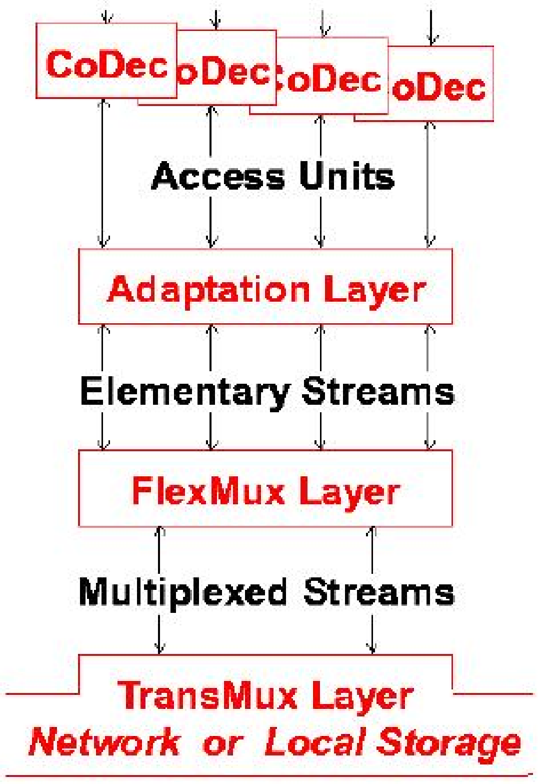

Figure 5.2: Objects in MPEG-4 are multiplexed [27]

SPEG

SPEG transcodes MPEG-1 coefficients to a set of levels, one base level and three enhancement levels. The coefficients from each level are grouped to form layers, four per original MPEG frame, which are the basic Application level Data Unit (ADU)s in SPEG. The steps can be reversed to return SPEG back to the original MPEG. Figure 5.3 shows this pipeline for SPEG. Alternatively, some or or all of the enhancement layer ADUs (from high to low) can be dropped subsituting zero values for the missing data. The effect of such dropping is analogous to having used higher quantization parameters during MPEG encoding, yielding lower bitrate in exchange for less spatial fidelity. SPEG suffices to demonstrate the essential properties of scalable compression, albeit with lower compression efficiency and fewer layers than something like MPEG-4 Finer-Grained Scalability [33].

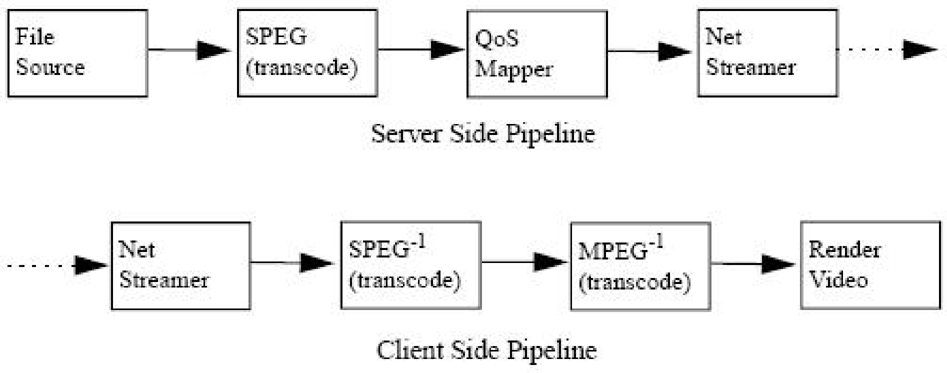

Figure 5.3: One single frame of an SPEG encoded sequence in its four possible steps of quality [32]

Fine-Grained Scalability

MPEG-4 Fine-Grained Scalability (FGS) is a video compression framework that is suitable for streaming applications such as video-on-demand and live TV viewing. This framework has been adopted by MPEG-4 video standard as the core compression tool for streaming applications. MPEG-4 FGS provides an effective mechanism for graceful video quality degradation based on its hierarchical layer structure, which consists of a base layer and one or more enhancement layers [10]. Awide range of bandwidth-variation scenarios are supported. They characterize IP-based networks, in general, and the global Internet, in particular. FGS is also resilient to packet losses, which are common over the Internet [58] and ad hoc networks.

MPEG-4 FGS provides a viable solution to the problems faced by the internet based Video streaming applications e.g. variation in the bandwidth and also the packet loss. Typical platforms for MPEG-4 FGS solutions [58] are

- Personal Computer (PC)s; Notebooks
- Digital TV; Set-Top Box (STB)
- Digital Video Disc (DVD) and Digital Video Recorder (DVR)
- Handhelds; Mobile handsets; Personal Digital Assistant (PDA)s
- Digital Video/Still Cameras (DV/SC)
- Video conferencing and Video phones
- Game consoles; Thin clients

At the moment this thesis is written there are few MPEG-4 FGS implementations. However, in the future this technology has the opportunity to be leading in layered compression.

Progressive JPEG

Joint Photographic Experts Group (JPEG) is not a codec, but is a commonly used standard method of lossy compression for photographic images. The file format which employs this compression is commonly also called JPEG. A simple or "baseline" JPEG file is stored as one top-to-bottom scan of the image. Progressive JPEG divides the file into a series of scans. The first scan shows the image at the equivalent of a very low quality setting, and therefore it takes very little space. Following scans gradually improve the quality. Each scan adds to the data already provided, so that the total storage requirement is roughly the same as for a baseline JPEG image of the same quality as the final scan. Basically, progressive JPEG is just a rearrangement of the same data into a more complicated order [29].

When sending progressive JPEG in low-bandwidth networks, such as ad hoc networks, a possibility could for receivers with low capacity to receive only the first scan with low quality while receivers with higher capacity could receive JPEG images with more details. The difference with a low-quality JPEG image and a high-quality JPEG image is shown in figure 5.4.

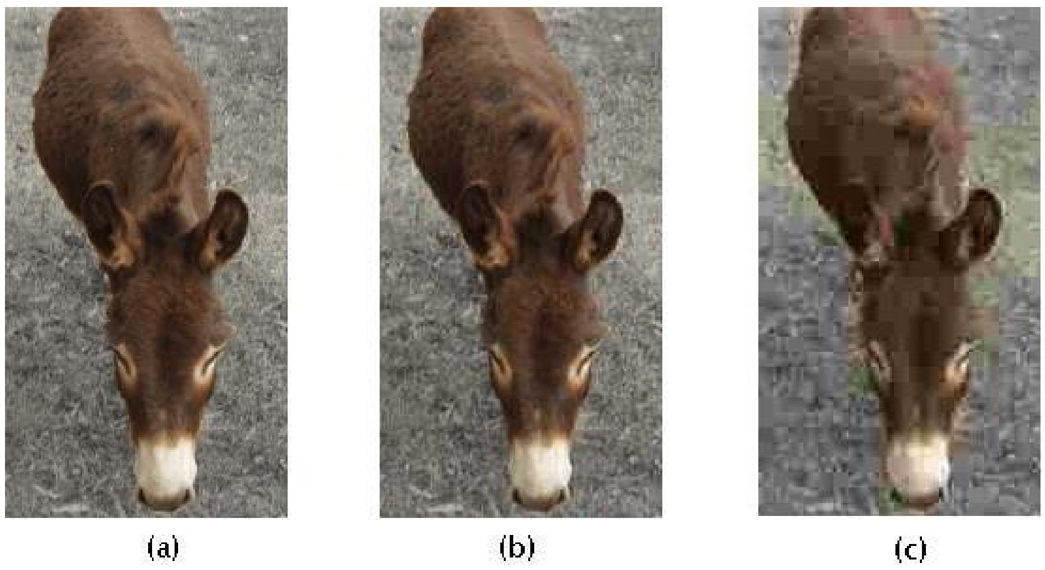

Figure 5.4: The object in (a) is a JPEG image with high resolution (filesize 36 KB) while (b) has lower resolution (filesize 5.7 KB). The defects of a low resolution is first shown if the compression go beyond a threshold which is the case in image (c) (filesize 1.7 KB) [65]

It is possible to convert between baseline and progressive representations of an image without any quality loss. However, specialized software is needed to do this; conversion by decompressing and recompressing is not lossless, due to roundoff errors [29].

5.2 FLUTE

5.2.1 FLUTE information

FLUTE is described here because it is a protocol which offers the possibility of scaling content in multicast networks. Scaling is one of the main aspects in this thesis.

FLUTE is a protocol for unidirectional delivery of files over the Internet and over provisioned and controlled systems (e.g. delivery over wireless broadcast radio systems), and it is especially suited for multicast networks. FLUTE is described in RFC 3926 [50] and builds on Asynchronous Layered Coding (ALC) [38] and Layered Coding Transport building block (LCT) [39]. ALC combines the LCT Building Block, a Congestion Control (CC) Building Block and a Forward Error Correction (FEC) Building Block [38] to provide congestion controlled reliable asynchronous delivery [54]. The FLUTE building blocks (ALC, LCT, CC and FEC) are shown in figure 5.5. The use of CC and FEC is optional.

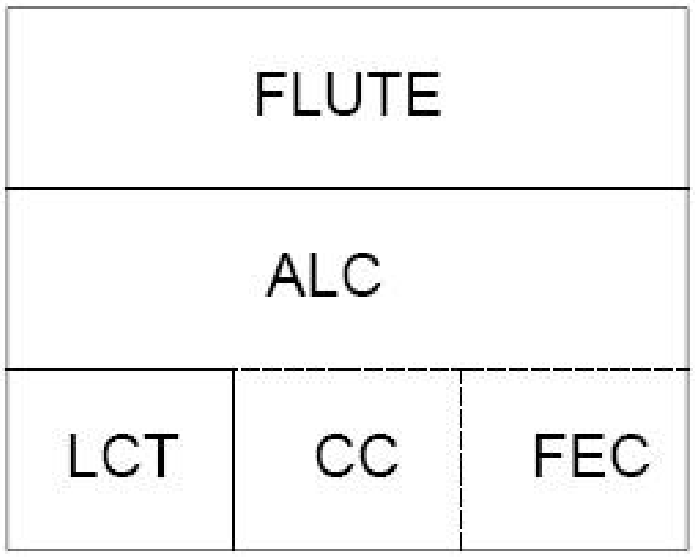

Figure 5.5: FLUTE Building Blocks [54]

For this Master's thesis FLUTE is interesting because it supports multicast at the same time as it supports layering. This makes it possible to scale the richness of a session to the congestion status of the network. Massive scalability is a primary design goal for FLUTE [50]. FLUTE is designed for different types of networks, including LANs, Wide Area Network (WAN)s, Intranets, the Internet, Asymmetric networks, Wireless networks and Satellite networks. In this thesis wireless networks are of particular interest.

FLUTE session

On the top of ALC FLUTE defines a "file delivery session". The concept is inherited from ALC and LCT. It includes details for transport and timing constraints. Properties from ALC/LCT are available as parameters in FLUTE, and receivers can assign them for the received objects. Each session is identified by a Transport Session Identifier (TSI) and a source IP address. The tuple (TSI, source IP address) is unique for each session.

FLUTE uses an File Delivery Table (FDT) instance to describe the all or part of files for the session. The FDT instance is an eXtensible Markup Language (XML) file. FDT instances are always in the lowest layer to ensure that they are received. Layers are described in the next section. An example of four files delivered by a session where all the files are described in one FDT session is shown in figure 5.6. The FDT instance is delivered before the set of files [73]. An FDT-Instance consist of a UDP header, a default LCT Header, LCT Header Extensions, FEC Payload ID, and Encoding Symbol(s) for the FDT Instance.

FLUTE layering

FLUTE inherits layering from ALC. ALC is a protocol instantiation of LCT. The layering makes it possible to send packets in the session to several channels at potentially different rates. The individual receivers can adjust their

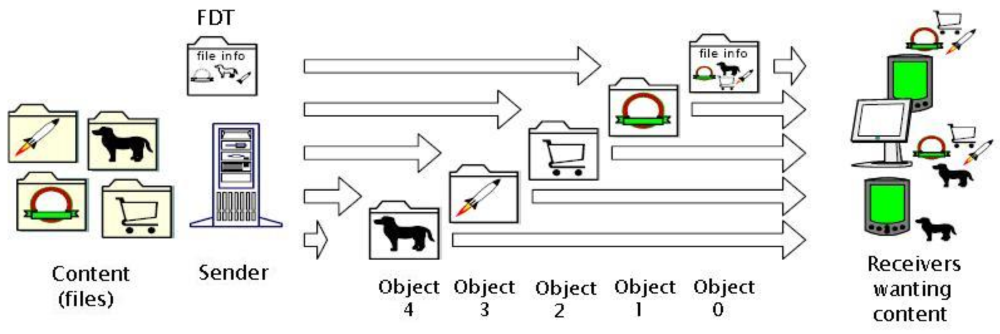

Figure 5.6: FLUTE Delivery with a single FDT Instance [73]

reception rate within a session by adjusting which set of channels they are joined to at each point in the time depending on the available bandwidth between the receiver and the sender. They can do this independent of other receivers [38].

The layering works as follows:
A multiple rate feedback-free congestion control building block is implemented in FLUTE. Congestion control is applied to all packets within a session, and this congestion control is independent from the information of the carried object. The multiple rate congestion control building block specifies in-band Congestion Control Information (CCI) which is carried in the CCI field of the LCT header. It also specifies formats of different lengths; i.e. 32, 64, 96 or 128 bits. The value of C defines the length of the CCI field in the LCT header. The length is a multiple of 32; either 32, 64, 96 or 128 bits as mentioned above. The value of C is the same for all packets sent to a session [38]. The LCT header is shown i figure 5.7. The first five 32 bit words come from the LCT header and the next 2 32-bit words is the FEC Payload ID. The remainder of the packet is the payload. In the MAD/TUT implementation the size of the CCI field is fixed to 32 bits. This could be a limitation. However, the Receiver-driven Layered Congestion control (RLC) congestion control protocol which is used also uses 32 bits, so it does not cause any problems. The FDT packet consists of a UDP header, a LCT

header with extensions, FEC payload ID and encoding symbols for the FDT instance.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   1   | 0 | 0 |1| 1 |0|1|0|0|0|       5       |      128      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Congestion Control Information (CCI, length = 32 bits)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Transport Session Identifier (TSI, length = 32 bits)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Transport Object Identifier (TOI, length = 32 bits)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Sender Current Time                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Source Block Number                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Encoding Symbol ID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Encoding Symbol(s)                        |
|                           ...                                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5.7: ALC header [38]

FLUTE Weaknesses

A weakness of FLUTE which may cause problems is that there may be no mechanism for receivers to effectively reduce their reception rate in wireless networks. The reason for the weakness is that transmission rates allocated to the session may be fixed in wireless networks. The same is true for satellite networks, but satellite networks beyond the scope of this Master's thesis work.

Another potential weakness about FLUTE is the fact that there are no methods for senders to verify the reception success of receivers. Applications that need strict requirements for reliability should not use the FLUTE protocol [15]. However, FLUTE contains FEC; RFC 3452 [40] which brings reliability to the FLUTE protocol. FEC encoding symbols are useful to all receivers for reconstructing content even when the receivers have received different encoding symbols. Furthermore, FEC codes can reduce the need for feedback from receivers to senders to request retransmission of lost packets [40]. Hence, by using FEC mechanisms, objects will be fully received if just sent enough times.

The used implementation of FLUTE also has a weakness; the size of the CCI field in the LCT header is fixed to 32 bits. However, this does not cause any problems because the RLC protocol which the FLUTE implementation is built on also uses only 32 bits. RLC is described in section 5.2.2.

Existing FLUTE Implementations

According to [54] there are five implementations of FLUTE:

1. Tampere University of Technology (TUT)
2. Institut National de Recherce en Informatique et en Automatique (INRIA)
3. University of Bremen
4. Nokia
5. Digital Fountain

The first three implementations (1, 2 and 3) are open source. They are tested against each other. The result is "FLUTE Interoperability Testing Guidelines" [41, 42]. The implementation used in this Master's thesis is the TUT implementation.

FLUTE Applications

FLUTE has become more known and is into commercial standards: It is adopted within 3rd Generation Partnership Project (3GPP) for Multimedia Broadcast Multicast Service (MBMS) file delivery service and is considered by a number of other standards, e.g. Digital Video Broadcasting - Handheld (DVB-H). In addition, FLUTE is used in the commercial world for applications like reliable bulk data transfer with wired feedback channels.

DFBroadcast (Digital Fountain):

- Complete FLUTE-Compliant network transport solution
- Can be integrated into any application or platform
- Capabilities: See DigitalFountain.com
- Set of C-language software libraries designed to support unidirectional data delivery to multiple receivers

MBMS (Multimedia Broadcast/Multicast Service):

- IP datacast (IPDC) type of service
- Offered via existing GSM and UMTS cellular networks
- First phase standards: finalized for UMTS release 6
- Solution for transferring light video and audio clips. Real streaming is also possible via the system. DVB-H is better for heavy duty streaming in a wide area for a large, concentrated audience.
- The first functional mobile terminals supporting MBMS is estimated (in 2004) to be available in 2008. The service is introduced in both networks and terminals.
- FLUTE is selected for MBMS download delivery method.

DVB-H (Digital Video Broadcast, Handheld):

- FLUTE is recommended transport protocol for announcement files in Service Discovery Channel
- FLUTE is proposed transport protocol for push file delivery

5.2.2 RLC - Congestion Control Scheme used by FLUTE

The MAD/TUT implementation of FLUTE builds on the congestion control scheme RLC. This is fully distributed in the meaning that all the congestion control decisions are given to the receivers [57]. Other congestion control protocols are available, e.g. Fair Layered Increase/Decrease with Static Layering (FLID-SL) and Wave and Equation Based Rate Control (WEBRC) and these may have better performance in come cases [44]. However, since RLC is used in FLUTE which is used in this Master's thesis, this protocol is described more in depth.

Without congestion control the network may become unusable because many sources want to make use of the network at the same time. Especially bandwidth demanding applications and content delivered to many receivers can cause congestion. Congestion control for multicast is more challenging than for unicast; mainly because multiple receivers have potentially different bottleneck bandwidths to the source. Secondly, it is hard to find scalable and network-friendly mechanisms to estimate the target rate without causing network congestion.

There are different solutions to the challenges based on multiple receivers. The bottleneck bandwidth problem may be accommodated by either adapt to the slowest receiver or to drop some members, or to select different data rates. RLC uses the latter solution. When it comes to the mechanism to estimate target rate the congestion control algorithm react to congestion control signals, which may come from the source or the receiver and may be either explicit notification or packet losses. Collecting multicast feedback should not be done too slowly nor too aggressively. Instead, it should be based on for example randomization and suppression techniques, election of representatives, or construction of hierarchies for feedback aggregation. Decentralization of functionality is a key when working with scalable services. RLC uses this decentralization and hence performs with high scalability.

RLC mechanisms

RLC build on two basic mechanisms to achieve a target response function; management of multicast membership and layered organization of data.

Multicast routers forward multicast data using Internet Group Management Protocol (IGMP) which is a communication protocol used to manage multicast group membership. Only routers leading to active receivers of a certain multicast group will forward packets to a this group. New receivers send a join IGMP message to the router which will enable forwarding for the group the receiver wants to join. This join phase does not take much time. The leave phase, however, is more time consuming. A receiver that wants to leave a group sends a leave IGMP message to the router which triggers a polling phase. The local router sends a request to the local subnet if any others are still interested in the group. If there are no responses the router sends a prune message for the group to an upstream router to block further forwarding to this group. This may take several seconds and is called leave delay [57]. A newer version (version 3) of IGMP includes support for a "source filtering", that is, the ability for a system to report interest in receiving packets *only* from specific source addresses, or from "all but" specific source addresses, sent to a particular multicast address [7]. This may improve the effectiveness of the group membership management.

The other mechanism; layered organization of data, is the most interesting in this thesis. RLC supports distribution of the same data (possibly, with different quality) to a set of receivers with different, increasing bandwidths. There are different layers, and each receiver can tune its receive bandwidth by joining the appropriate number of layers. The data can be either streaming data (where the different layers become refinement of the information) or reliable data (where the layers have different transfer rate). The application is responsible for defining the organization of the layered data [57].

The RLC algorithm

RLC lets the receivers hop between different layers depending on congestion signals. The following four components are the basics of the RLC algorithm:

1. Layer hopping rule
2. Synchronization points
3. Sender-initiated probes
4. Deaf period

Layer Hopping Rule This component defines how receivers are supposed to react to congestion signals. If a receiver experience no congestion during a specified period of time, the receiver will hop to a higher layer. However, if loss is experienced the receiver will hop to a lower layer. When multiple receivers this is not a very good solution; receivers can compete each other when acting in uncoordinated ways.

Synchronization Points Synchronization points are flagged packets in the data stream which help receivers behind the same bottleneck to synchronize behaviors. Receivers can only make send a join message immediately after a synchronization point. Also, it can only make decisions based on the period from the last synchronization point. A level's synchronization points are a subset of the lower layer's synchronization points. Hence, the lower layers have more chance to increase their subscription level.

Sender-initiated Probes These probes reduce the number of errors when estimating the available bandwidth. The sender initiates a periodic generation of short bursts of packets, followed by an equally long period where no packets are sent. The rate is equal to the upper limit of what is allowed to each layer. If this causes congestion, this is a hint that the subscription level should not be increased.

Deaf Period Normally there will be a delay after IGMP leave messages are sent. By using a deaf period which is slightly longer than the leave delay, the long response time will not be that problematic.

The algorithm If there are n different bandwidth layers, the sender transmit at n different rates. Each rate is constant. Synchronization points are placed proportionally to the bandwidth corresponding to the subscription level. Synchronization points at level i+1 is a subset of the synchronization points at level i. Right before the synchronization points there are sent out bursts proportional to the bandwidths. Figure 5.8 shows how the sequence of packet transmissions from an RLC transmitter and the locations of synchronizations points for five layers.

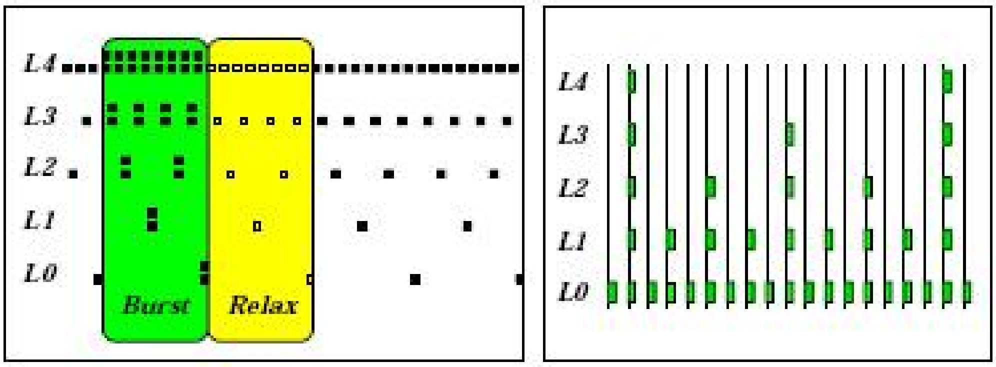

Figure 5.8: Sequence of packet transmissions (left) and location of synchronization points in RLC (right) [57]

At the RLC receiver side there are certain rules that have to be followed when increasing or decreasing subscription level:

Decreasing if there is congestion during normal congestion. However, no decreasing for a certain period of time (the deaf period) after the last decrease.

Increasing at a synchronization point if there has been no congestion since the last synchronization point.

Unchanged otherwise; this includes the case of congestion during a burst or during a deaf period.

The main elements of RLC are the use of synchronization points and the rules for changing subscription levels. There is no feedback sent back to the sender; feedback is implicit in the form of IGMP messages to the local router. The RLC protocol is hence simple and scalable [57]. IGMP is required to be implemented by any host wishing to receive IP multicast. However in a wireless context like MANET, the use of IGMP may lead to inconsistencies. A better protocol for MANETs may be the wireless version of IGMP, namely Wireless IGMP (WIGMP). WIGMP is only concerned with exchanges between hosts and routers to determine group membership. On a multicast router, WIGMP coexists with a multicast routing protocol [34].

Part II

Design and implementation

Chapter 6

MANET Content Distribution Scenarios

To be able to distribute video and control information in mobile ad hoc networks in an effective way, there are certain challenging scenarios especially interesting for this Master's thesis. They are described in this chapter.

6.1 The Scenarios

It is important to limit the distribution in an ad hoc network because the nodes often have less capacity and are more mobile than nodes in a wired network. As opposed to wired networks, there are no well-working standard multicast protocols in MANETs. Also, the ad hoc networks often consists of nodes lacking of multicast support. Hence, multicast protocols cannot be used throughout the network. This section will show some scenarios where a mixture between different distribution mechanisms may be useful.

1. Multicast to near while unicast to one node far away
2. Unicast to faraway multicast groups

3. Gateway on the edge to another network
4. High rates to near nodes and lower rates to faraway nodes
5. Multicast with layered channels

Multicast to near while unicast to one node far away One scenario considered consists of many recipient close to sender while one or a few recipients are far away. In between some nodes do not support multicast; only unicast. To avoid sending (multicast) packets to a huge number of recipients in a wide area (which may produce overhead) the recipient far away get the packets sent as unicast while multicast is used to nearby nodes. This solution is shown in figure 6.1.

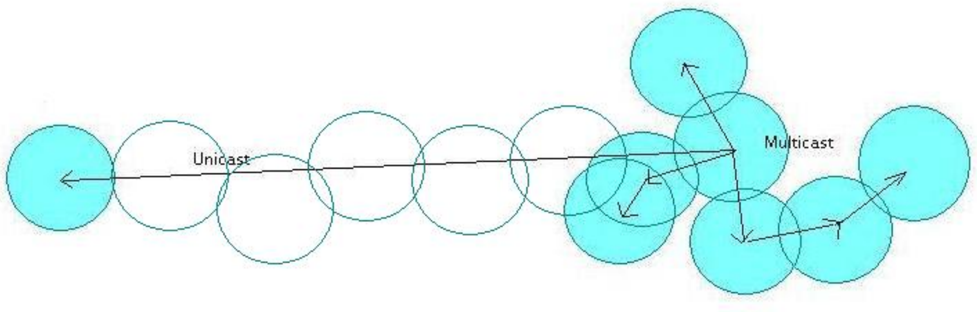

Figure 6.1: Multicast to near while unicast to one node far away

Unicast to faraway multicast groups As in the previous scenario nearby recipients get content spread using multicast. Multicast packets to faraway multicast group are spread as unicast to a group and then further spread as multicast at the destination. The difference from the previous scenario is that multicast group clusters redistribute the content using multicast as shown in figure 6.2.

Gateway on the edge to another network This scenario consists of different types of networks. On the edge of a wireless network there is a gateway into another type of network, e.g. the Internet. Nodes in the wireless network use multicast. The gateway forwards data into the

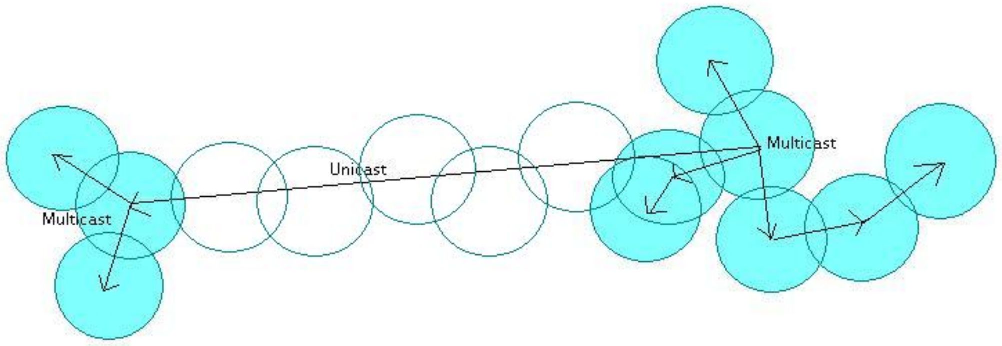

Figure 6.2: Multicast to near while a multicast packet sent far away as unicast packet and spread as multicast at the destination

other network. This is shown in figure 6.3. Some work for OLSR/Open Shortest Path First (OSPF) gateway approaches is shown in [25].

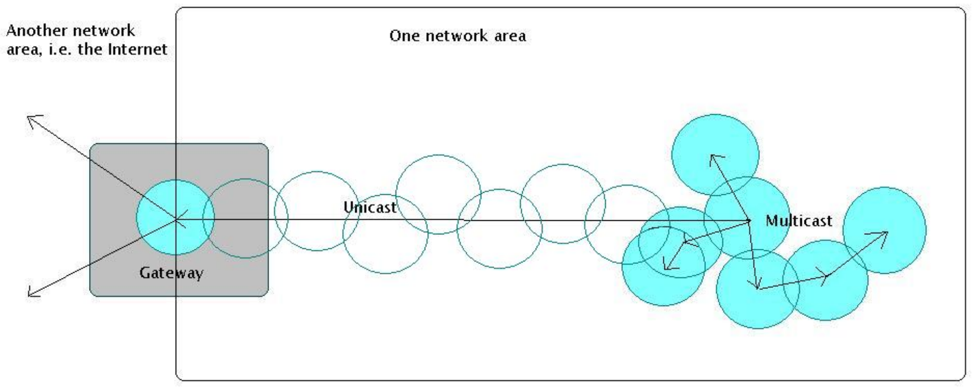

Figure 6.3: Gateway solution on the edge to another network

High rates to near nodes and lower rates to faraway nodes Especially in mobile wireless networks traffic is likely to loose packets on its way to the destination. Faraway nodes may loose more than nearby nodes. Hence, in this scenario multicast traffic to nearby nodes can be sent using a relatively high rate while traffic to faraway nodes should be sent using smaller rates. Figure 6.4 illustrates this.

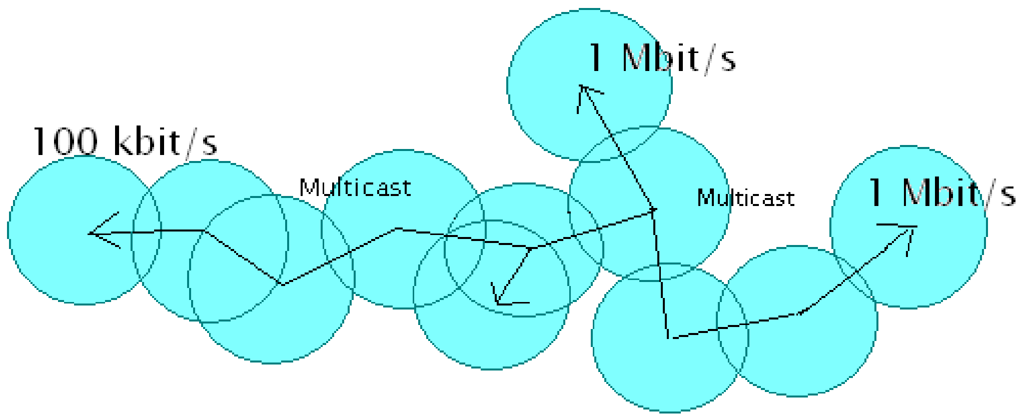

Figure 6.4: Multicast to near nodes using a relatively high rate and to faraway nodes using smaller rates

Multicast with layered channels A similar scenario to the previous one is to send multicast streams with different channels. The most important data which all receivers need to receive is sent on the lowest channel. Additional data is sent layered in other channels. Receivers can set how many channels they want to receive. Nodes with high capacity can set to receive many channels while nodes lacking of resources can set to receive fewer channels. The layered model can be used in two ways; for downloading data with different data rates and for sending different quality of data.

6.2 Evaluations of the scenarios

All the scenarios described in the previous section illustrate different challenges for content distribution in MANETs. The first; "Multicast to near while unicast to one node far away" is about reducing the distribution. It could be solved by setting the TTL for multicast messages to a certain value so that the multicast data is not sent further than necessary. The packet to a node faraway where there are nodes in between that do not support multicast, could be sent simply by using unicast. The next scenario; "Unicast to faraway multicast groups" is similar to the first scenario; some data will have to be sent using unicast whereas other data can be sent using multicast. The difference is that there is a multicast group after the unicast nodes. A solution for distributing multicast packets through an area of unicast nodes is to "tunnel" the multicast packets. Some multicast protocols support tunneling:

[25] has done some work on sending multicast to some nodes and unicast to other nodes using two nonstandard protocols; namely the Manet Forwarding Protocol (MFP) and the Wireless Network (WNet) MANET framework. They both take advantage of the inherent broadcast capability in wireless link networks. Both protocols can forward unicast and multicast packets in MANETs where unicast is a special case of multicast transport with a single receiver. MFP is an on-demand protocol for route discovery, however, existing links are proactively maintained. It uses a broadcast emulation to flood route request packets in the MANET and a directed forwarding algorithm to transport unicast and multicast data, embedded in special MFP packets. WNet on the other side, uses MPR to flood a network; the same as what OLSR does. It is possible for the WNet framework to determine link and node attributes since it is located on the MAC layer. The attributes are used to calculate link qualities which influence the forwarding decision. Hence, the WNet uses a quality-based routing mechanism.

Instead of using normal multicast, Xcast (see section 4.5.3) or its extensions might also be used for these two first scenarios. Only a few nodes within the network has to support Xcast; the rest only unicast is required.

The scenario "Gateway on the edge to another network" could be a Master's thesis on itself. It might look easy to make a gateway which transfer data between different types of networks. However, this is challenging. There is ongoing work with gateways, however it will not be more described nor tested in this thesis.

The two last scenarios are both about scaling according to available bandwidth. The first; "High rates to near nodes and lower rates to faraway nodes" scales the content in different while the last; "Multicast with layered channels" scales the content in different quality. The file delivery protocol FLUTE (see section 5.2) supports content layering in addition to different rates.

Not all scenarios described in this chapter are tested in this Master's thesis work. Based on the theory and the evaluations in this section in addition to available applications and implementations of protocols the following scenarios for multicast distribution in ad hoc networks are tested:

- Multicast to near while unicast to one node far away
- Multicast with layered channels

Chapter 7

Test and Demonstration Architecture

To be able to do the testing several elements are needed. In the theory part different types of technology is described, such as WLAN, MANETs and multicast. In this section the architecture of how the elements are put together is described.

7.1 Comparison to the OSI Model

A list of all elements of the architecture used for testing is shown in appendix A.1. Essential components are:

- MOLSR; a multicast plugin for the MANET protocol OLSR, to forward multicast data
- FLUTE; a protocol that supports sending multicast data in several layers
- Linux Fedora Core 5 operating system
- IEEE 802.11b Wireless Local Area Network (WLAN) adapters

- A topology emulator to simulate different topologies

One comparison to the architecture can be the OSI model [68]. This model divides functions of protocols into seven layers where only neighbor layers communicate; one layer uses the functions of the layer below, and only exports functionality to the layer above. On the top there is an application layer which is the one a user interacts with and on the bottom there is a physical layer. Low layers are transparent to higher layers. For the user, it seems like the applications on two computers communicate directly with each other. Not all layers in the OSI model are normally used. Also, in this comparison not all the seven layers are used. The comparison figure is shown in figure 7.1.

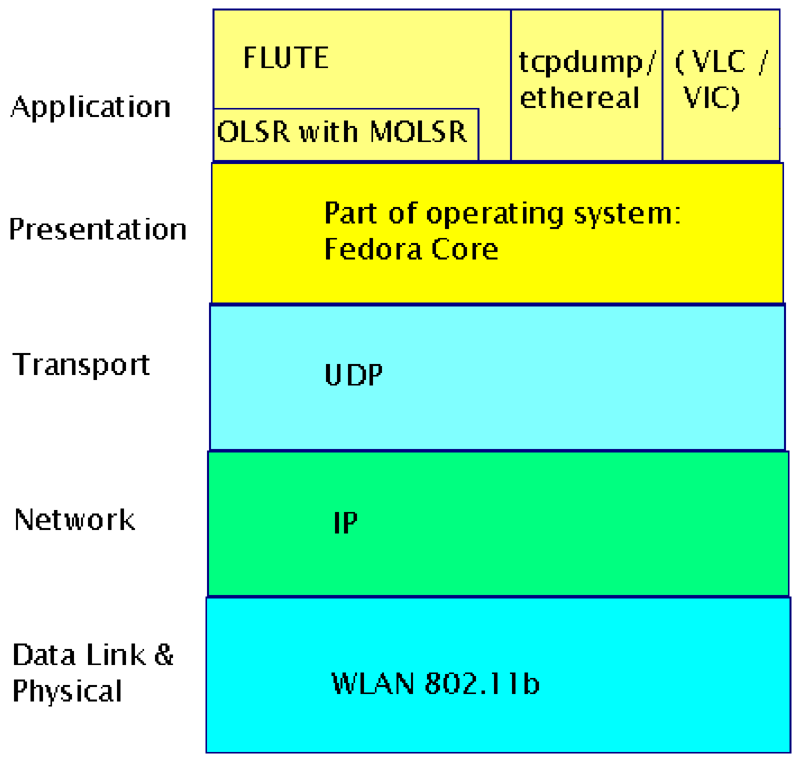

Figure 7.1: Building elements used for testing and their relation to layers in the OSI model

On the top run the different applications, such as FLUTE, tcpdump[69], ethereal, and the OLSR implementation with MOLSR plugin. MOLSR is needed to forward multicast packets that FLUTE sends. Tcpdump and etheral are data tracking applications to see what types of data goes over the interfaces. The video distribution applications VIC and Video Lan Client (VLC) are not used; however, if there was time they could have been modified to make layered content for FLUTE.

At the presentation layer the operating system (Linux Fedora Core) appends a set of application-layer instructions that will be read and executed by the application layer on other computers. Parts of the operating system also deals with presentation issues where a header from this layer is appended to the message. The process repeats through all the layers (transport protocol IP and network protocol UDP) until each layer has appended a header. The headers function as an escort for the message so that it can successfully negotiate the software and hardware in the network and arrive intact at its destination (see figure 7.2). On the bottom there is the WLAN 802.11b which includes functions for physical and data link layer.

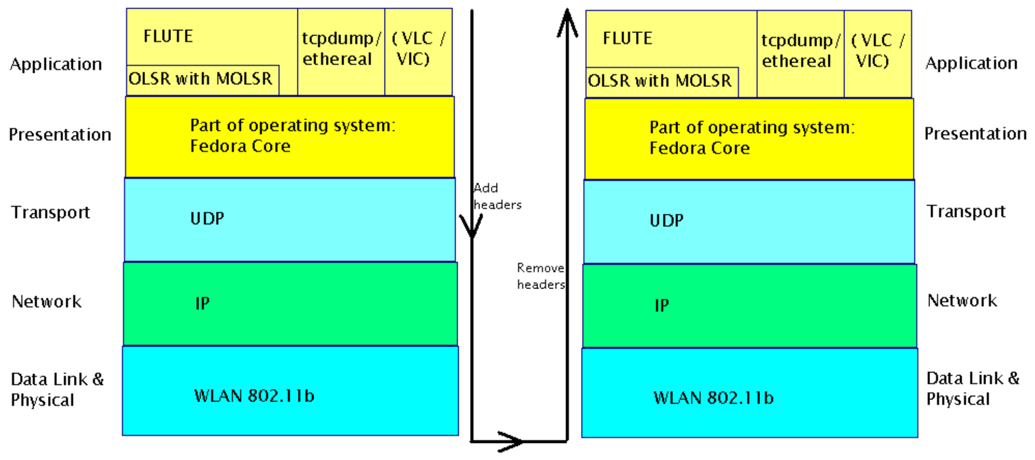

Figure 7.2: Two computers with the same building elements communicating; the sender adding headers for each layer and receiver removing headers.

7.2 Creating different topologies for the computers

The architecture used in the tests consists of five computers with the same applications, operating systems and WLAN adapters. Two different methods are used to create the different topologies for the tests; topology emulator and placing the computers around in a building. The two variants of creating topologies are shown in figure 7.3 and figure 7.4.

The topology emulator inserts cables from each computer. Inside the topology emulator there are attenuators which can create attenuation between the computers; from no attenuation (0 deciBel (dB)) to full attenuation (127 dB). Hence, the topology emulator can create an ad hoc network of any topology; only limited by the number of computers. The topologies are set using topology emulator software on a PC connected to the topology emulator.

When placing computers (laptops) around the building to create different topologies, each laptop has a radio sender and receiver WLAN. The laptops are placed in a way, such that the laptops which are supposed to hear each other are within each other's range; whereas computers that shall not hear each other are out of range of one another.

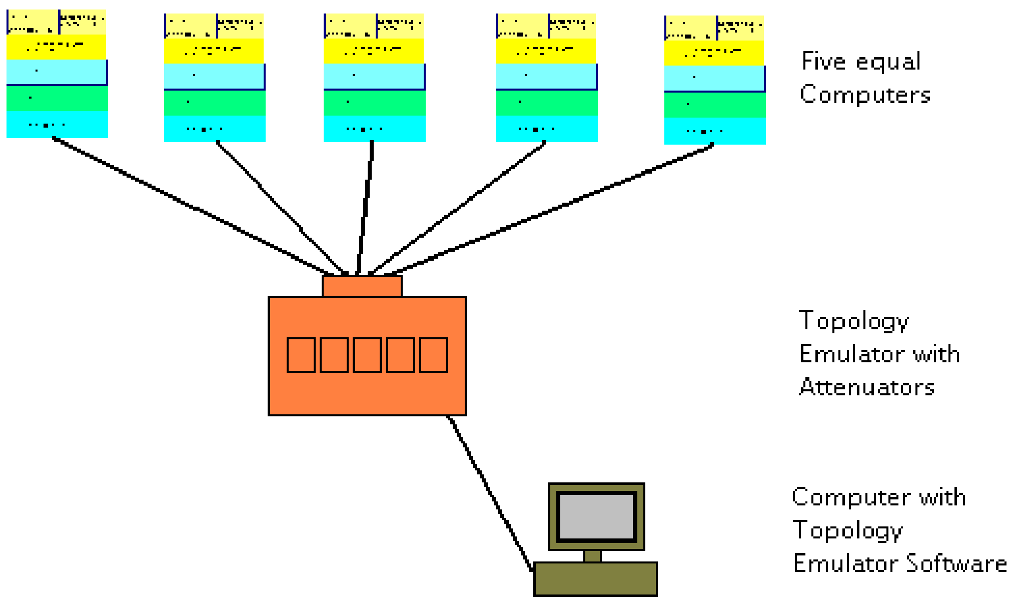
Five equal Computers
Topology Emulator with Attenuators
Computer with Topology Emulator Software

Figure 7.3: Five computers and a topology emulator used to make different topologies

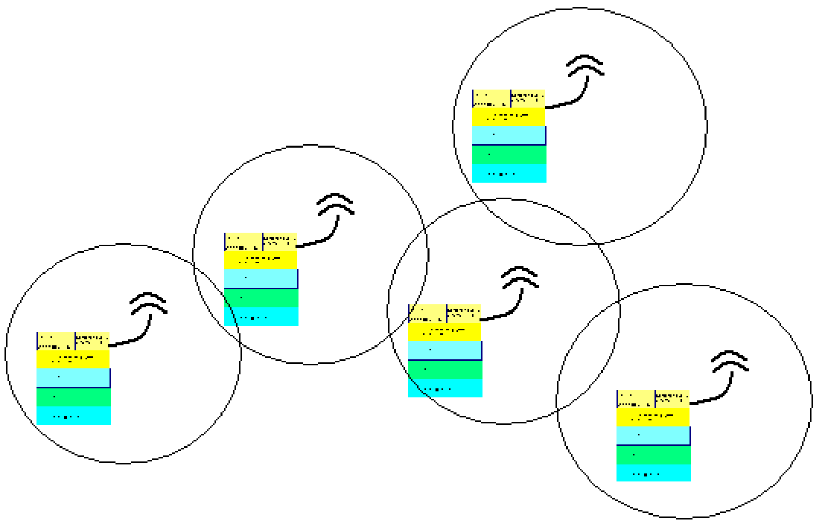

Figure 7.4: Five laptops with radio senders used to make different topologies

Chapter 8

FLUTE testing

This chapter describes the tests done with the test elements of the proposed architecture (7).

8.1 Setup

The test setup consists of in total five computers running Linux Fedora 5 in ad hoc mode and using 802.11b WLAN. They use the lowest data rate; 1Mbps. A topology emulator is used to block signals between the computers to construct different topologies for the ad hoc network. IPv4 is used to transport packets between the computers. IPv6 could also have been used, but it would not give more information than what IPv4 does; hence it is not used. Computer names and corresponding acIP addresses are shown in table 8.1. Details, such as building elements versions, configuration commands and commands for running FLUTE are described in appendix A.

Common for eth1 on all the computers:

- Mode: Ad-Hoc
- Frequency: 2.412 GHz

Computer name	*IP Address*
OLSR1	10.10.20.11
OLSR2	10.10.20.22
OLSR3	10.10.20.33
OLSR4	10.10.20.44
OLSR5	10.10.20.55

Table 8.1: Computers used in the test setup and their IP addresses

- ESSID: "NbFGrid"
- netmask 255.255.255.0
- Interface: eth1

8.2 Running FLUTE

8.2.1 FLUTE options

Most FLUTE options are shown on the MAD-FLUTE homepage [47]. The important options used in this thesis are the following:

General options :

- -S Act as sender, send data; otherwise receive data
- -m IPv4 or IPv6 address for base channel, default: 226.10.40.1 or ff1a::1
- -p Port number for base channel, default: 4001
- -t TSI for the session, default: 0
- -V Print logs to 'str' file, default: print to stdout

- -w Congestion control scheme [0 = Null, 1 = RLC], default: 0; the number of channels, defined by -c option, are used with both schemes and bit rate of each channel is set according to RLC rules

Sender options :

- -c Number of used channels, default: 1
- -r Transmission rate at base channel in kbits/s, default: 250
- -F File or directory to be sent
- -f FDT file (send based on FDT), default: fdt.xml
- -T Time To Live or Hop Limit for the session, default: 1

Receiver options :

- -A Receive files automatically
- -c Maximum number of channels, default: 1
- -F File(s) to be received
- -B Base directory for downloaded files, default: flute-downloads

8.3 FLUTE tests

The tests are presented in table 8.2. They are divided into three categories or groups:

1. Test with topology emulator and three machines (test 1-4)
2. Test with topology emulator and five machines (test 1, 5-9)
3. Test with laptops (test 1 and 10)

The computer names and their IP addresses are shown in table 8.1. Different topologies used are shown in figure 8.3, 8.4, 8.6 and figure 8.7.

8.3.1 Test with topology emulator and three machines

Test no 1: Ping the other machines

The reason for doing this test is that ping is a good indication if it is possible to send data over a link. If ping does not give response over a link, no other data will.

Procedure:

- This test is done before the tests in each category; both unicast and multicast.
- Examples of unicast and multicast ping:
  - Unicast ping to OLSR4 with IP address 10.10.20.44: `ping 10.10.20.44`
  - Multicast ping to multicast group: `ping 224.0.0.1`
- Use the ping command on each machine to find out which of the other machines it can hear.
- Evaluate the response to see if a machine reaches the right machines according to the topology setup.

Expected result: If there is no attenuation between two neighbor machines (set by the topology emulator) there shall be no loss of ping packets. If there are two or more hops between two machines and no attenuation the response shall be redirected; still no loss. If the attenuation value is 275 dB between two machines there shall be no response between these. Multicast ping shall give response from all machines listening to multicast traffic.

Result: Unicast ping works in most cases. Once it gives response from machines the sender should not hear. The topology emulator is restarted; then it works again. Unicast ping is able to reach machines four hops away. Multicast ping, however, does not seem to work properly. Nothing is received.

Evaluation: Unicast ping gives a good indication of which machines hear each other. It works as expected. In contrast, multicast ping does not work in test 2 to test 9. Since FLUTE multicast seems to work in the first tests where there are maximum two hops, there is not paid much attention to this at the beginning.

Test no 2: Sniff between machine 1 and 2

The goal for this test is to check the transmission between two machines. Before using many computers it is good to know if the transmission works between two machines.

a) Check if the FLUTE rate is correct.

Procedure:

- Sending Alien.mpg (large file) from OLSR1 to OLSR2 with rate 250 kilobits per second (kbit/s). One channel is used.
- Receiver is set to listen to the multicast address the sender sends to.
- Tcpdump is used to track the packets going over the interface where the file is sent (OLSR1).
- The number of packets per second is counted
- The number of packets per second is multiplied by the size of each packet (which is in bytes). This is multiplied by 8 to get the answer in bits: How many kilobits per second which is the rate.
- Check if the rate calculated is the same as the rate set in FLUTE.

Expected result: The rate set by FLUTE to be equal to the calculated rate.

Result: The calculation of two different rates (see appendix A.4.1) shows that the rate set by FLUTE is the same as the real rate. E.g. when the rate is set to 250 kbit/s the calculated rate for a period of time is about 245.2 kbit/s. When the rate is set to 50 kbit/s the calculated shows 49.2 kbit/s.

Evaluation: This is acceptable.

b) Test another type of network

Procedure:

- Test setup is the same as in test 2a, but the sender IP address is 10.10.10.161 and the interface eth0 (which is normal Ethernet) is used instead of eth1 (802.11 Wireless LAN).
- Multicast packets are sent to the multicast address 224.1.1.1.
- The amount received on the other machine is tracked and compared to the amount received in test 2a where 802.11 WLAN is used.

Expected result: Better throughput than for 802.11 because this is a wired network and not a wireless ad hoc network.

Result: 100 percent is received and no packets lost.

Evaluation: Wired networks give indeed a better throughput than ad hoc networks; as expected.

c) Test unicast and multicast

Procedure:

- Sending unicast FLUTE from OLSR1 to OLSR2.
- Destination address is the IP address of OLSR2 (instead of the multicast address 224.1.1.1 used in most of the other tests)
- Compare the amounts received using unicast to the amounts received using multicast.

Expected result: Better throughput using unicast than multicast because of better reliability for unicast on the 802.11 MAC layer.

Result: By using unicast 98.25 percent is received and 9 packets lost.

Evaluation: Using unicast gives better throughput than the use of multicast; as expected because of reliability of unicast on the IEEE 802.11 MAC layer (see section 2.3).

d) Test using the multicast plugin MOLSR

Procedure:
- MOLSR is started on both OLSR1 and OLSR2.
- The FLUTE setup is the same as in Test 2 a
- The amount received is compared to the results in the other tests.

Expected result: Since this is only one hop (which this test shows) the result should be equal or better than without using MOLSR.

Result: 66 to 68 percent received. 46 to 49 packets lost.

Evaluation: This is the same result as for not using MOLSR. Thus, the MOLSR does not seem to make better results for one hop.

Test no 3: Test the network capacity with MGEN

This test is done to see maximum transmission rate for unicast and multicast over a WLAN link. MGEN generates packets over the network. Together with tcpdump, trpr and gnuplot this gives a visual picture of the transmission.

a) Unicast (two machines)

Procedure:
- Use MGEN as packet generator on the sender OLSR1.
- Use a script (TcpPlot.sc) on receiver OLSR2 with commands to tcpdump, trpr and gnuplot (see appendix A.4.2) which produces a visual view of the received traffic.
- Test the maximum throughput.
- Evaluate the results.

Expected result: Find a maximum rate to be either 1 Mbit/s, 2 Mbit/s, or 11 Mbit/s.

Result: The maximum throughput is approximately 900 kbit/s.

Evaluation: The throughput of approximately 900 kbit/s shows that this IEEE 802.11b adapter seems to give a throughput of 1 Mbit/s which is one of the 802.11b standards.

b) Multicast (two - three machines)

Procedure:

- Use MGEN as packet generator on the sender OLSR1 (details in appendix A.4.2).
- Use the script (TcpPlot.sc) as in test 2a on receivers OLSR2 and OLSR3 (details in appendix A.4.2).
- Use MOLSR as multicast protocol and send to multicast address 224.1.1.1.
- Look at the throughput for different MGEN input files with different packet lengths and rates.
- Evaluate what gives the best throughput to OLSR2 and OLSR3.

Expected result: Throughput to OLSR2 approximately 900 kbit/s and to OLSR3 a little less.

Result: OLSR2 receives up to a little more than 900 kbit/s. OLSR3 receives only around 0 to 24 kbit/s on most rates. However, when using a packet around 400 the throughput is better. The highest throughput for two hops is 350 kbit/s. For the best throughput of OLSR3, OLSR2 receives up to 800 kbit/s. Gnuplot pictures are shown in figure 8.1 and 8.2. Details are in table A.3, A.4 and A.5.

Evaluation: There is a big difference what is possible to receive after one hop compared to two hops. The best rate and packet length combination found in this test is packet size a little less then 400 bytes and rate 400-500 packets per second. It is surprising that there is such a big difference between one and two hops.

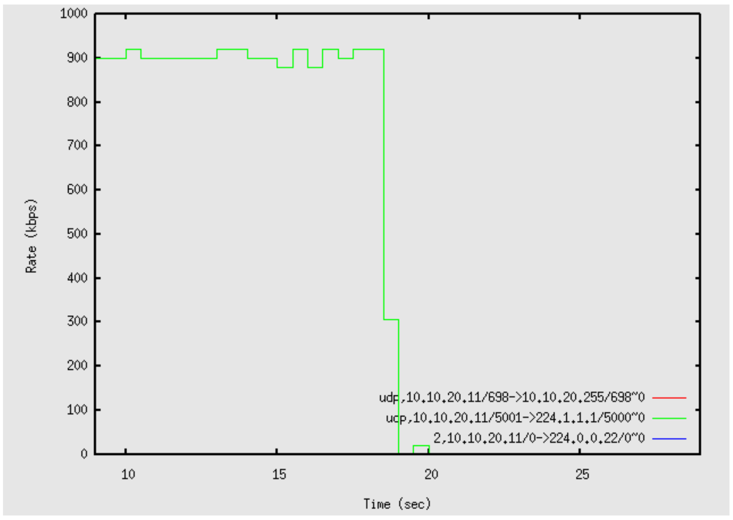

Figure 8.1: The gnuplot picture shows that the data rate for OLSR2 comes above 900 kbit/s

Test no 4: FLUTE with three machines; line topology

The motivation for this test is to check different aspects of content distribution for three computers; such as finding the data rate and packet sizes that produce the best throughput. Also; the test is done to see if MOLSR and attenuation affect the throughput. The topology used in this test is shown in figure 8.3 (a). The topology emulator is used to set the topology. Details are shown in appendix A.4.3.

a) Use FLUTE commands to find the best throughput for multicast

Procedure:

- Send FLUTE multicast data (either small file tamitatest.txt - 38 kilobytes, or large file Alien.mpg - 3.1 megabytes) from OLSR1 to multicast address 224.1.1.1 which OLSR2 and OLSR3 listen to.

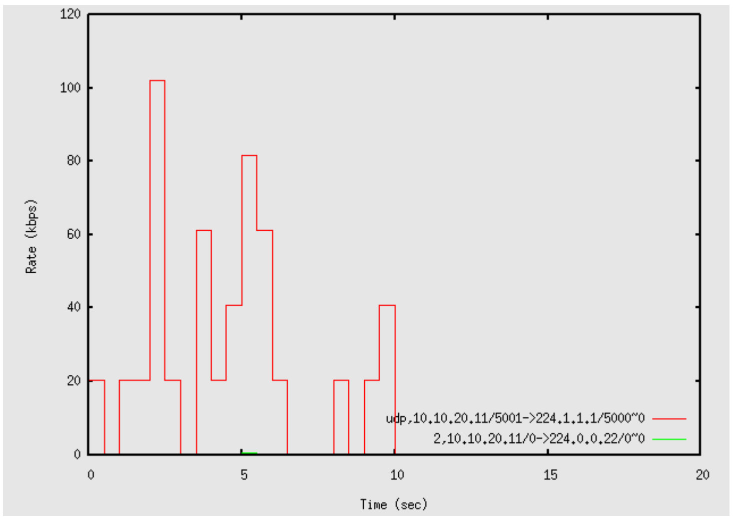

Figure 8.2: The gnuplot picture shows that the data rate for OLSR3 comes up to 100 kbit/s

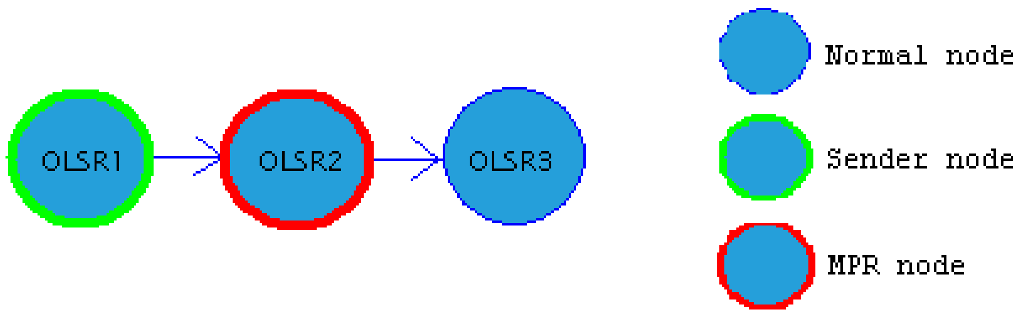

Figure 8.3: Topologies used in testing: Line with three nodes; 1 MPR Hop

- Check different packet sizes (FLUTE command -l) and bit rates (FLUTE command -r) to see what gives the best percent received on the receivers.

Expected result: Packet sizes around 400 kbit/s give the best throughput (from test 2). OLSR3 will receive slightly less than OLSR2. Low bit rates give better results than high bit rates.

Result: The best result is given by the combination of packet size 357 bytes and bit rate 750 bit/s. Then OLSR2 receives more than 99 percent and OLSR3 receives around 20 percent. Many other combinations give zero percent data to OLSR3. Lower rates do not give better throughput than higher rates.

Evaluation: The packet sizes are 20 bytes bigger than the set -l size. That is because of addition of IP and UDP headers. As expected from the results of test 2, the packet sizes around 400 kbit/s give the best throughput. However, it is not expected that OLSR3 receives so little percent; often as little as zero percent when multicast is used. Also unexpectedly, low bit rates do not seem to give better results than high bit rates. It should be easier to transfer the data slow than fast.

b) Test without MOLSR

Procedure:
- Send FLUTE multicast data (large file Alien.mpg - 3.1 megabytes) from OLSR1 to multicast address 224.1.1.1 which OLSR2 and OLSR3 listen to.
- Check if the data comes two hops (to OLSR3) without MOLSR.

Expected result: Multicast data shall not be received by OLSR3 when MOLSR is not running.

Result: The FLUTE application on OLSR3 receives nothing. However, the data come to the eth1 interface of OLSR3 (tracked by tcpdump).

Evaluation: As expected, the data is not received by the FLUTE application two hops away from the sender when MOLSR is not running. On the other side, the data is received by interface of OLSR3, which means that the data is received.

c) Use topology emulator to make an attenuation between OLSR1 and OLSR2

Procedure:

- Run MOLSR on all machines.
- Set the topology emulator to an attenuation of 50 dB and 60 dB.
- Test the throughput with packet size 377 kilobytes and rate 1000 kbit/s.

Expected result: Reduced throughput, however not much.

Result: The throughput between OLSR1 and OLSR2 seems to be affected by 60 dB, but not by 50 dB. Actually, the throughput is better for 50 dB than for 0 dB (test 4b1). In contrast, OLSR3 does not receive anything when the line between OLSR1 and OLSR2 is reduced.

Evaluation: The results do not make very much sense. More tests should be done here to get a more accurate results.

8.3.2 Test with topology emulator and five machines

Test no 5: FLUTE with five machines; line topology, four hops

This test is performed to see how far unicast and multicast data can come. Five nodes is the maximum number of computers available for testing. It is also wanted to find out the effect of channels. The effect of The topology used in this test is shown in figure 8.4 (a). The topology emulator is used to set the topology. In almost all tests the FLUTE commands -l:337 and -r:750 are used because they were shown to be the best combination for throughput in test 4. Complete commands used for sending and receiving FLUTE are written in appendix A.4.4.

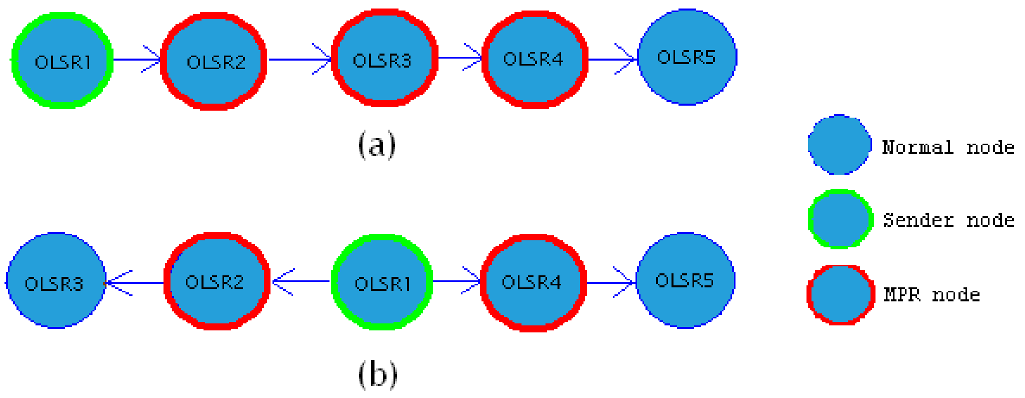

Figure 8.4: Topologies used in testing: (a): Line with five nodes; 3 MPR Hops (b): Line, 2x1 MPR hops

a) Use FLUTE to test throughput for multicast with five nodes in a line

Procedure:
- Run MOLSR on all machines.
- Set the TTL to four or higher to be sure the last node receives the data (e.g. FLUTE command -T:5) .
- Send FLUTE multicast data (either small file tamitatest.txt - 38 kilobytes, or large file Alien.mpg - 3.1 megabytes) from OLSR1 to multicast address 224.1.1.1 which OLSR2, OLSR3, OLSR4 and OLSR5 listen to.
- Check another bit rate (FLUTE command -r) to see if the percent received is affected to the better worse on the receivers.

Expected result: OLSR2 will receive almost 100 percent, while the percentage received by the other nodes in the line will decrease slightly as the number of hops increase. Lower bit rates will lead to better throughput on the faraway nodes.

Result: OLSR2 receives more than 99 percent and OLSR3 receives around 30 percent while OLSR4 and OLSR5 do not receive anything. Sometimes the FLUTE application starts running which means they receive the FLUTE session initiation packets. However, they do not receive any normal FLUTE packets. The session initiation packets have another packet length than the other FLUTE packets. Hence, it is possible to track them especially by using tcpdump. Low bit rate (-r:150) does not give better throughput than the normal rate (-r:750).

Evaluation: Even if the other previous tests show the same, it is not expected that the throughput is better with high than with low rates. OLSR1 receives almost 100 percent, as expected. OLSR2 receives less than expected, but 30 percent is almost ok. What is completely in contrast to the expectation is that OLSR4 and OLSR5 do not receive a single packet (except the initiation packets). Possible reasons why OLSR4 and OLSR5 do not receive normal FLUTE packets are tried to be resolved:

- The TTL function in the FLUTE application does not work
- Synchronizing problems
- OLSR3 does not work as MPR
- Too many OLSR packets which are prioritized over FLUTE packets
- Too large packet size
- OLSR3 becomes a hidden node between OLSR2 and OLSR3, and hence does not manage to forward FLUTE packets
- Something wrong with either the OLSR implementation or the MOLSR plugin

b) Unicast many hops

Procedure:
- Sending unicast FLUTE from OLSR1 to the other machines: First to OLSR3 (2 hops), then OLSR4 (3 hops) and at the

end to OLSR5 (4 hops).

- Destination address is the IP address of OLSR5 (instead of the multicast address 224.1.1.1 used in most of the other tests)
- Compare the amounts received using unicast to the amounts received using multicast.

Expected result: Better throughput using unicast than multicast because of better reliability for unicast on the 802.11 MAC layer. The close machines will receive more than the faraway machines.

Result: OLSR3 receives between 30 and 50 percent. OLSR4 receives around 20 percent. OLSR5 receives around 10 percent. The OLSR application must run to let OLSR4 and OLSR5 to receive anything.

Evaluation: Using unicast gives better throughput than the use of multicast; as expected because of reliability of unicast on the IEEE 802.11 MAC layer (see section 2.3). OLSR4 and OLSR5 receives data using unicast. OLSR4 and OLSR5 receive packets only if the OLSR program is running. Thus, the OLSR program (where the MOLSR plugin is installed) is working for unicast traffic.

c) Four hops with two channels

Procedure:
- Send multicast data on two channels (FLUTE command -c:2) from OLSR1.
- OLSR2 listens to two channels; each one in separate session windows (see appendix A.4.4 for details). OLSR3 , OLSR4 and OLSR5 listen to only one channel.
-

Expected result: The multicast data channel that all the machines are listening to will be sent to all machines. The second channel that only OLSR2 (and OLSR3) are listening to will only come to OLSR2 (and

OLSR3). A little more data will come to channel one (the base channel) than to channel two. It may be easier for OLSR3 and OLSR4 to receive multicast data if they only receive one channel.

Result: OLSR2 receives exactly the same on both channels; almost 100 percent. OLSR3 receives around 21 percent on the one channel. When OLSR3 also receives exactly the same when set to listen to two channels; 21 percent. Also, the number of lost packets is the same on both channels. OLSR4 and OLSR5 do not receive anything; like in test 5a.

Evaluation: In this test it does not seem to be any difference between the base channel and the second channel. It is possible that the reason is that the same data is sent in both channels. If layered content were sent the result might be different. As in test 5a, OLSR4 and OLSR5 (3 and 4 hops from the sender) do not receive any multicast data.

d) Test synchronization between OLSR2 and OLSR3

Procedure:
- Send FLUTE multicast data as in test 5a.
- Track the times for packets on the eth1 interface for OLSR2 and OLSR3
- Count the time difference between each packet for a period of time
- Compare the time differences

Expected result: The time between each packet is sent may be the same all the time. The time difference between the packets are equal in OLSR2 and OLSR4. Hence, this makes synchronization problems. A machine cannot send at the same time as receiving.

Result: The packets are not sent with exactly the same interval. The time difference between each packet is very similar, however not equal in OLSR2 and OLSR3.

Evaluation: This test cannot prove that synchronization is the reason why

OLSR3 does not send multicast packets any further.

e) Same as test 5a, but the machines in another order

Procedure:
- Topology emulator is used to make a new order: OLSR1 - OLSR2 - OLSR4 - OLSR5 - OLSR3.
- Otherwise, same procedure as in test 5a.

Expected result: If OLSR4 works as an MPR the FLUTE multicast data may be received by OLSR5 and OLSR3 which are three and four hops away from the sender.

Result: Same results as in test 5a. Machines three and four hops away do not receive any multicast data.

Evaluation: OLSR3 is not the problem. Other machines work the same way.

Test no 6: FLUTE with five machines; line topology, two hop

The motivation for this test is to find out if the computers work in the same way; if two computers with equal distance from the sender receive the same amount data. The topology used in this test is shown in figure 8.4 (b).

a) One channel

Procedure:
- Use the topology emulator to make the topology shown in figure 8.4.
- Send FLUTE multicast data from the middle node (OLSR1) to the other nodes (one MPR in each direction). This is repeated many times to get trustworthy results.

Expected result: Equal results from the two machines closest to the sender (OLSR2 and OLSR4). Equal results, but less amount received on the two machines two hops from the sender (OLSR3 and OLSR5).

Result: OLSR2 and OLSR4 both receive 98 to 100 percent each time which is both good and very similar. Two hops away from the sender the amount received is more unstable; sometimes they receive around 30 percent and other times they do not even start receiving. The first times the test is run OLSR5 does not receive anything. I does not send out IGMP messages. After restarting OLSR5 it sends out IGMP messages and works fine. OLSR3 gets worse after a restart.

Evaluation: Machines only one hop from the sender seem to work well. They receive both almost hundred percent. Two hops from the sender the machines are more unstable. In an ad hoc network this is understandable, however each machine is not hundred percent stable by itself. IGMP messages seem to be important for receiving FLUTE data.

b) More channels

Procedure:

- Same procedure as in test 6a, however two and three channels are used.
- Track the eth1 interface with tcpdump on all the machines to see what channel(s) they receive from.
- Check IGMP messages to see how many groups (channels) each machine wants to receive.

Expected result: The receivers get data from the number of channels they are set to receive from. If possible, the machines closest to the sender can receive from all available channels while the machines two hops away from the sender can receive from one or maximum two channels. IGMP messages show how many channels each machine want to receive from. OLSR1 sends more data from channel 1 than the other channels.

Result: By looking at the tcpdump messages it seems like all machines receive data from all available channels no matter how many channels they are set to receive from. However, which channels the FLUTE application uses is not visible. Only a few times the receivers send

IGMP messages where they want to receive from two groups. OLSR1 seems to send the same amount of data to each channel. When two channels are used OLSR1 first sends only to channel 1; then every other to channel 1 and channel 2. At the end data is sent only to channel 2. When three channels are used OLSR1 starts the same way as with two channels. After sending from channel 1 and 2 for a little while it starts sending one packet from channel 1, then one packet from channel 2 and then two packets from channel 3 over and over again. After a while only one packet are sent from each channel for a while. Toward the end only packets from channel 1 and channel 2 are sent. At the very end a few short "stop" packets from channel 1 are sent. This way of sending three channels is shown in figure 8.5.

Evaluation: It is not expected that all the machines receive from all channels no matter what they are set to receive. However, it is not visible if the FLUTE application use the information from all the channels or not. According to the FLUTE implementor, the receiver analyzes and stores the data to volatile or non-volatile memory, and rejects duplicate packets. Hence, the amount of received data, is not bigger because the same data is sent in every channel. The receiver probably rejects packets from channels it is not supposed to receive from even if it arrives on the interface. The IGMP group report messages do not seem to be a sign of how many channels each receiver wants.

Test no 7: FLUTE with five machines; two hop mesh topology

The two hop mesh topology is tested because it is an interesting topology which may occur in a MANET. The topology used in this test is shown in figure 8.6 (a).

Procedure:
- Use the topology emulator to make the topology shown in figure 8.6 (a).

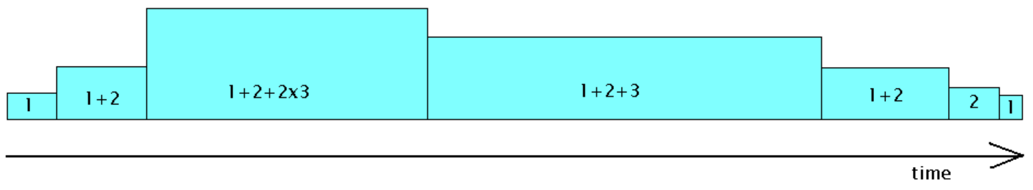

Figure 8.5: How OLSR1 sends three FLUTE channels during a session. Number 1 stands for channel 1, number 2 for channel 2, and number 3 for channel 3. In the third block there are sent two packets of channel 3 for each packet of channel 1 and channel 2.

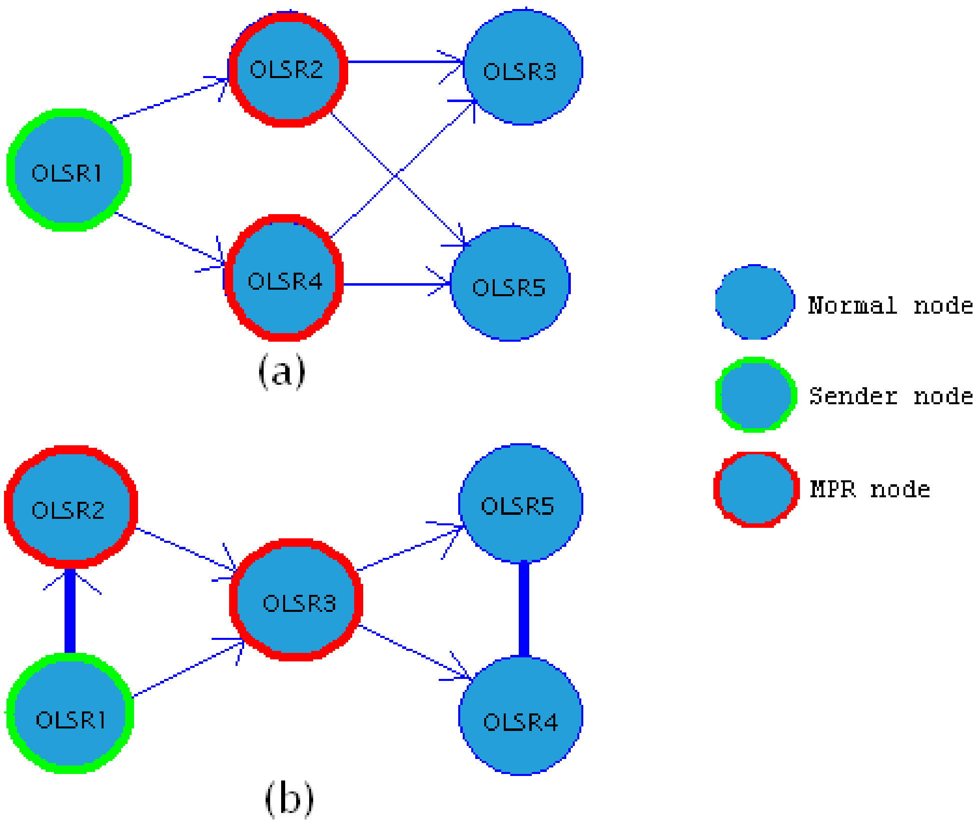

Figure 8.6: Topologies used in testing: (a): 2 Hop Mesh, (b): Bottleneck

- Send multicast FLUTE data from OLSR1. All the other machines listen to the multicast address 224.1.1.1.
- Repeat the multicast sending and track the answers.

Expected result: Better results than for a line topogy (test 5) because shortest path from sender to any receiver is not longer than two hops.

Result: The amount received on all the machines is very high; sometimes all machines receive 100 percent. OLSR5 is the only machine that not always receives 95 to 100 percent; sometimes nothing at all and sometimes around 30 percent.

Evaluation: This topology gives very good results; high percentage received on all the machines. The reasons for good performance seem to be few hops and several machines to forward the data. The few problems with OLSR5 may come from the undefined problems OLSR5 also had in test 6a. In addition, OLSR5 is the only machine which is not a neighbor of OLSR1.

Test no 8: FLUTE with five machines; different capacity

The goal of this test is to see how much attenuation nodes in a MANET tolerate. The topology used in this test is shown in figure 8.6 (b).

a) No attenuation

Procedure:
- Use the topology emulator to make the topology shown in figure 8.6 (b).
- No attenuation is used between the machines that are supposed to hear each other.
- Send FLUTE multicast traffic from OLSR1 to multicast address 224.1.1.1. All the other machines listen to this address.
- Repeat the multicast sending and track the amounts received.

Expected result: OLSR2 and OLSR3 (which are neighbors of OLSR1) receive 100 percent or almost 100 percent. OLSR4 and OLSR5 receive around 20 - 50 percent which is normal for second hop.

Result: OLSR2 and OLSR3 both receive 100 percent seven out of eight times. The reception for OLSR4 and OLSR5 are more variable; OLSR4 receives between 30 and 50 percent while OLSR5 receives between 80 and 100 percent. A few times OLSR4 and OLSR5 do not receive anything, usually because they get an "RLC Warning: Max number of LATE packets received."

Evaluation: The reception for one hop is good and more varying for two hop. The variation for two hops may be because of the data going through a bottleneck (OLSR3). However, the throughput here is acceptable compared to the results from previous tests.

b) Attenuation only from sender OLSR1 and OLSR2 to OLSR3 in the middle; not from OLSR3 to OLSR4 and OLSR5

Procedure:
- Same setup as in test 8a except for an attenuation of 15 dB from OLSR1 and OLSR2 to OLSR3.
- Send FLUTE multicast traffic from OLSR1 to multicast address 224.1.1.1. All the other machines listen to this address.
- Repeat the multicast sending and track the amounts received.

Expected result: OLRS2 still receives 100 percent while the reception for OLSR3 is a little reduced compared to results in test 8a. OLSR4 and OLSR5 receive slightly less than in test 8a.

Result: OLSR2 receives 100 percent and OLSR3 receives 96 to 99 percent, as expected. The FLUTE applications on OLSR4 and OSLR5 start, however they do not receive nothing.

Evaluation: Only the two neighbor nodes of the sender act as expected. Two-hop nodes do not receive anything even if the attenuation is not especially high. They seem to be very sensitive to capacity reduction.

c) Attenuation between OLSR3 and all the other nodes

Procedure:
- Same setup as in test 8a except for an attenuation of 15 dB from OLSR3 to all the other machines.
- Send FLUTE multicast traffic from OLSR1 to multicast address 224.1.1.1. All the other machines listen to this address.
- Repeat the multicast sending and track the amounts received.
- Try different attenuation values (15 dB, 7 dB and 2 dB) to see where OLSR4 and OLSR5 manage to receive data.

Expected result: Same result as expected in test 8b. However, the reception for OLSR4 and OLSR5 should be even worse than in test 8b. Attenuation value of 7 dB may give throughput to OLSR4 and OLSR5.

Result: Each time the test is run both OLSR2 and OLSR3 receive 100 percent while OLSR4 and OLSR5 do not receive anything. Attenuation 15 dB and 7 dB gives no throughput to OLSR4 and OLSR5. When using as little as 2 dB OLSR4 and OLSR5 receive from one to seven percent.

Evaluation: Actually, OLSR3 receives 100 percent here even if there is an attenuation between the sender and OLSR3; this is unexpected as it received a little less when there was no attenuation. OLSR4 and OLSR5 act as in test 8b. Their FLUTE applications start, but they do not receive anything. As little attenuation as 2 dB is needed for OLSR4 and OLSR5 to receive data. This shows that nodes more than one hop away from the receiver are very sensible to reduction in the capacity. Another possibility is that the forwarding protocol is not working properly.

Test no 9: FLUTE with five machines; multicast and unicast

One of the scenarios in section 6.1 includes both multicast and unicast. Not all nodes in a MANET support multicast. Thus, it is interesting to see how FLUTE handles sessions of both unicast and multicast. The topology used in this test is shown in figure 8.7.

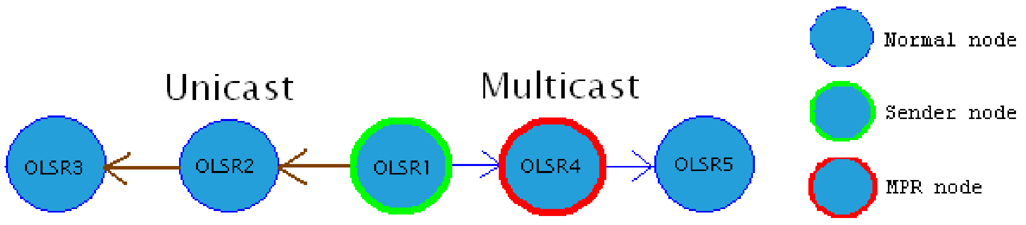

Figure 8.7: Topologies used in testing: Unicast and Multicast at the same time

a)

Procedure:

- Use the topology emulator to make the topology shown in figure 8.7.
- Start two FLUTE sessions on the sender OLSR1; one unicast and one multicast (see appendix A.4.6 for details).
- OLSR2 does not listen to anything. OLSR3 listens for unicast traffic. OLSR3 and OLSR5 listen for multicast traffic.
- Start the unicast and the multicast session at the same time or after each other.
- Try bit rates 250 and 750 kbit/s.

Expected result: The reception for both multicast and unicast the same as for only one session; maybe a little reduced if the multicast and the unicast sessions are sent at the same time. Also, the reception may be if the rates of the two sessions together exceed the total capacity of the eth1 interface of OLSR1.

Result: With rate 750 kbit/s and the two sessions started at the same time the reception is reduced especially for the unicast receiver which gets around 60 percent. Multicast receivers OLSR4 and OLSR5 get 100 percent and around 85 percent, respectively. The tcpdump messages on the eth1 interface of OLSR1 shows that not all packets to bots sessions are sent. First, all the unicast data are sent, then the multicast data. The short stop packets for unicast and multicast are sent at the end.

By reducing the rate to 250 kbit/s unicast receiver improves its reception to 95-100 percent. However, the multicast receivers get less with this rate; OLSR4 95-100 percent and OLSR5 32-42 percent. When the sessions are sent after each other (the second started after the first one is finished) the reception is good for both unicast and multicast receivers. Sending sessions after each other with rate 750 kbit/s gives better results for multicast receivers(OLSR4 99-100 percent and OLSR5 60-88); not for unicast receiver (around 60 percent).

Evaluation: Unicast data prefer a lower rate then the multicast data. When the two sessions are sent at the same time the sender (OLSR1) does not manage to send all the packets in each session. The reception is slightly worse than when sending the sessions after each other. This may be better in future releases of FLUTE. Version 1.5 which may be released soon after this Master's thesis is done, will be thread safe.

Evaluation of the tests so far

The tests so far show that there is obviously something wrong when sending multicast FLUTE data. It should be possible to send multicast data more than two hops. Some possible sources of error are written in the evaluation of test 5a:

1. The TTL function in the FLUTE application does not work

2. Synchronizing problems
3. OLSR3 does not work as MPR
4. Too many OLSR packets which are prioritized over FLUTE packets
5. Too large packet size
6. OLSR3 becomes a hidden node between OLSR2 and OLSR3, and hence does not manage to forward FLUTE packets
7. Something wrong with either the OLSR implementation or the MOLSR plugin

Synchronization is tested in tested in test 5d. Exchange of computers is tested in test 5e. Different packet sizes are tested in test 4a. OLSR3 manages to send OLSR packets. Hence, it does not seem to be a hidden node. The two sources of error left are wrong FLUTE TTL and something wrong with the OLSR implementation or the MOLSR plugin. The implementor of FLUTE writes in an email that there should be nothing wrong with the TTL in the FLUTE implementation.

Ethereal would be a useful program to track more details than tcpdump is able to. Unfortunately, there are problems when trying to install Ethereal on Fedora Core version 5. Hence, other methods must be used to find the source of the multicast error.

By setting the MOLSR plugin in fault mode it prints out eventual errors. It gives an error which comes whenever a buffer is smaller than zero. This error is not displayed on older versions of Fedora (tested on Fedora Core version 2 and 3). Hence, a likely explanation may be that something in Fedora Core version 5 have has changed from earlier versions. By looking at new features for Fedora Core 5 [23] the most probable cause of changes is that a new compiler is included. All the libraries are built using this compiler. Thus, the MOLSR is not able to read information properly from Fedora Core 5.

A possible way to test if the MOLSR plugin works on older versions of Fedora Core is to use laptops with 801.11b adapters running MOLSR. Because of time constraints not all the tests done so far can be done. However, the line topology with five nodes is tested. This is shown in the next section.

8.3.3 Test with laptops

Test no 10: FLUTE with five machines; line topology

The motivation for this test is to see if multicast forwarding works for laptops running OLSR with MOLSR plugin on Fedora Core 2. The topology used in this test the same as used in test 5. It is shown in figure 8.4 (a). The names of the laptops and their IP addresses and MAC addresses are shown in table 8.3.

a) Use FLUTE to test throughput for multicast with five nodes in a line

Procedure:

- Run MOLSR on all machines
- Set the TTL to four or higher to be sure the last node receives the data (e.g. FLUTE command -T:5).
- Place the laptops around in the building to make the same line topology as the topogy emulator made in test 5.
- Send FLUTE multicast data (either small file tamitatest.txt - 38 kilobytes, or large file Alien.mpg - 3.1 megabytes) from OLSR1 to multicast address 224.1.1.1 which OLSR2, OLSR3, OLSR4 and OLSR5 listen to.
- Use Ethereal to track packets going between the machines.

Expected result: OLSR2 will receive almost 100 percent, while the percentage received by the other nodes in the line will decrease slightly as the number of hops increase. Ethereal shows (by looking at MAC addresses) that each laptop receives multicast packets from the previous laptop in the line; not from the sender (OLRS1)

Result: First, the data only goes one hop. After modifying the MOLSR plugin to forward IGMP packets in addition to unicast packets the results become different: OLSR2 receives 74-95 percent, OLSR3 60-90 percent, OLSR4 57-70 percent, and OLSR5 receives 46-65 percent. Ethereal shows that the MAC address of the connection link which has sent the multicast data is the previous machine in the line.

Evaluation: Finally, the multicast forwarding is working; the fourth hop receives up to 65 percent which is considerably better than in the previous tests. Ethereal also shows that the multicast forwarding is working.

b) Four hops with three channels

Procedure:

- Send multicast data on three channels (FLUTE command -c:3) from OLSR1.
- OLSR2 listens to three channels and OLSR3 to two channels. OLSR4 and OLSR5 both listen to one channel.
- Look at tcpdump messages what channels the machines receive from.
- Track percentage received by each laptop.

Expected result: The multicast data channel that all the machines are listening to will be sent to all machines. The second channel that only OLSR2 and OLSR3 are listening to will only come to OLSR2 and OLSR3. Channel three will only come to OLSR2. It may be easier for OLSR3 and OLSR4 to receive multicast data if they only receive one channel.

Result: By looking at the tcpdump messages it seems like all machines receive data from all available channels no matter how many channels they are set to receive from (same as in test 6b). However, which channels the FLUTE application uses is not visible. The percentage received in this test is not better than with only one channel; OLSR2 receives around 86 percent while OLSR3, OLSR4 and OLSR5 receive about 25, 23 and 20 percent, respectively. During much of the receiving time

Evaluation: The results are similar to results in test 4c and 5b; it is not expected that all the machines receive from all channels no matter what they are set to receive. However, it is not visible if the FLUTE application use the information from all the channels or not. According to the FLUTE implementor, the receiver analyzes and stores the data to volatile or non-volatile memory, and rejects duplicate packets. Hence, the amount of received data, is not bigger because the same data is sent in every channel. The receiver probably rejects packets from channels it is not supposed to receive from even if it arrives on the interface. If layered content were sent the result might be different. OLSR2 seems to be a bottleneck since it receives around 86 percent while the other machines receive only 20-25 percent. The reason may be too many programs running at the same time. However, the results are not getting better after a restart. The machines do not always work properly.

c) Repeating the FLUTE session

Procedure:
- Send FLUTE multicast data in the same way as in test 10a. However, repeat the session three times (FLUTE command -n:3)
- Test with two different rates; 250 and 750 kbit/s

Expected result: Repeating the session will improve the total amount received; among other factors because of the FEC function in FLUTE. Lower bit rates will lead to better throughput on the faraway nodes.

Result: When using bit rate 250 kbit/s all the laptops receive over 99 percent. When using bit rate 750 kbit/s the first node receives over 99 percent. The other nodes receive decreasing amounts down to 57-87 percent on the last hop. During the sessions OLSR3 times out several times.

Evaluation: When repeating the sessions all the laptops receive almost hundred percent. This is a very good result. The forward error correction in FLUTE may be one of the reasons why this works so well. In contrast to the previous tests a lower bit rate here gives better throughput than a high bit rate. This is what is expected from the theory. It probably turns out this way here because the multicast forwarding works. In the previous tests multicast forwarding did not work and hence gave strange results. Unfortunately, there is not enough time to test more with the laptops.

Test No	*Test Description*	*Note*	*Category*	*Figure*
1	Ping the other machines	Done in the beginning of each test	All	
2	Sniff between machine 1 and 2	Check the transmission between two machines	One	
3	Test the network capacity with MGEN	Use MGEN to see how much goes through the network	One	
4	FLUTE with three machines; line topology	Use FLUTE to see how much goes through and what FLUTE rate is the best	One	fig 8.3
5	FLUTE with five machines; line topology, five hop	See how much goes through five hop	Two	fig 8.4 (a)
6	FLUTE with five machines; line topology, two hop	See if there are big differences between different machines	Two	fig 8.4 (b)
7	FLUTE with five machines; two hop mesh topology	Test this topology	Two	fig 8.6 (a)
8	FLUTE with five machines; different capacity	See how the capacity affects the throughput	Two	fig 8.6 (b)
9	FLUTE with five machines; multicast and unicast	See if it is possible to send both unicast and multicast at the same time	Two	fig 8.7
10	FLUTE with five machines; line topology	See if more hops work with FLUTE on other machines (using Fedora Core 2)	Three	fig 8.4 (a)

Table 8.2: Overview of the Tests

Computer name	*IP Address*	*MAC Address*
OLSR1	10.10.20.11	00:02:8A:E4:AD:9A
OLSR2	10.10.20.22	00:02:8A:E4:AD:A4
OLSR3	10.10.20.33	00:0E:9B:15:82:6B
OLSR4	10.10.20.44	00:0E:9B:15:82:6E
OLSR5	10.10.20.55	00:0E:9B:15:83:8F

Table 8.3: Laptops used in test 10 and their IP addresses and MAC addresses

Part III

Discussion and conclusion

Chapter 9

Discussion

Some of the possible scenarios have been tested using FLUTE running on a wireless ad hoc network with OLSR including a multicast plugin MOLSR. This section includes a discussion about the obtained results during testing and whether the selected architecture has vitality in mobile ad hoc networks; alternatively if other possible options for content distribution in MANETs are needed.

9.1 About the obtained Results

The tests using FLUTE are described in section 8.3. One of the main problems during testing was the fact that multicast data was not sent more than two hops from the sender. This was not discovered in the beginning because only three computers running the necessary software were available during the first tests. The three computers were used to find the best data rate and packet size for multicast content distribution. For certain rates and packet sizes multicast with FLUTE seemed to work, even though multicast ping gave no response. Since multicast seemed to work with FLUTE the ping issue was left virtually unattended. Further tests were then carried out. In tests using five machines the problem became more clear; there was no multicast data transfered more than two hops. Several possible sources of error where investigated, such as

1. The machines themselves
2. Synchronization problems
3. Too much traffic from OLSR packets
4. OLSR3 becomes a hidden node between OLSR2 and OLSR3, and hence does not manage to forward FLUTE packets
5. Too large packet sizes
6. The OLSR implementation or the MOLSR plugin

To check if the machines themselves were the sources of error the order was changed. Still, multicast data did not travel more than two hops. To find out if there were synchronization problems between the computers, packets where tracked on the computer network interfaces and the times the packets where sent and received where checked against each other. Packets were also tracked to see if the traffic from OLSR was too much for the network; this did not seem to be the case. Since OLSR3 managed to forward OLSR packets, it did not act as a hidden node. Too large packet sizes were also a possible source of error; however smaller packets did not improve the results. The real source of error turned out to be the MOLSR plugin to the OLSR implementation. This was found out by setting the MOLSR plugin to fault mode. It then printed error messages which made it clear that there were problems with multicast forwarding. One possible explanation to the errors is that there is a new compiler included in the new version (version 5) of Linux Fedora Core. All the libraries in the operating system are built using this compiler [23]. All the computers connected to the topology emulator have Linux Fedora Core 5 installed.

By testing an ad hoc network with laptops running version 2 of the operating system Fedora Core the results corresponded more with the expected results. Lower bit rates gave higher throughput and multicast packets were forwarded to all the nodes where the closest node received the most, whereas the amount received decreased slightly as the number of hops increased. More testing is needed to test all possible scenarios.

The layering testing did not work as expected; probably because there was no layering code available. The same data was sent on all channels. VIC or VLC are possible video distribution applications which can be used to modify media to provide layered content. Also, codecs are capable of producing layered content. When MPEG-4 Finer-Grained Scalability frameworks become more available they will be excellent for layering video in networks. So far, SPEG gives some possibilities for layering by offering different levels; one base level and three enhancement layers (see section 5.1.1). MPEG codecs can be used for media streaming.

In addition to the sources of error tested, there are other issues which might have affected the test results. Some of them are:

- Packet sizes and bit rate (these issues were tested, however not when multicast forwarding was working)
- The topology machine is not one hundred percent accurate. Even if there should be a complete block between nodes, there may be some leaking signals.
- Antennas in the room or nearby may disturb the topology setup.
- CSMA/CA protocol overhead may reduce the throughput over UDP [63]. The MAC layer of 802.11 spends time on signaling and transferring data with different bit rates; this reduces the throughput in a link (see section 2.3). The overhead is also dependent on the packet sizes [77]. E.g. packets larger than 1500 bytes are fragmented on the MAC layer.
- Computer capacity; other processes on the computers may impact the

multicast programs

- Noise between the computers; e.g. other networks interfering or people walking in between

The concept of layering is still interesting even if the test architecture has been dominated by problems with software versions not working together and some tests not working as expected. Especially MANETs gain from the possibility of distributing content in several layers. It is an effective way to limit the amount of traffic in the parts of the network where the link or node capacity is low. The FLUTE protocol is supposed to support layering of multicast in ad hoc networks. However, the MAD/TUT implementation of FLUTE has not turned out to be ideal for MANETs as it is not threadsafe and uses RLC with IGMP which are best suited for wired networks. A possible improvement is the support for congestion control scheme for MANETs that may be using WIGMP which is better for wireless networks than IGMP. One improvement that will come already in the next version of the MAD/TUT implementation of FLUTE is the addition of threadsafe sessions. This will for example make it easier to send both unicast and multicast sessions simultaneously.

Through the research, it has become clear that multicast is not the only way to transport data to several receivers; especially not when the network is small and some receivers are close, whereas others are further away with nodes in between only supporting unicast. Among the alternatives of interest is the protocol Xcast and its extensions. Xcast includes several addresses in the header. One advantage is the possibility to tunnel Xcast messages as unicast messages between Xcast-enabled hosts and routers. Xcast is not tested in this Master's thesis and the protocol is not yet standardized, however there is work going on to improve this protocol and its extensions. It will be interesting to follow its development. Also, improvements of multicast protocols and codecs with layered content will become more and more available. The nonstandard multicast protocols MFP and the WNet MANET framework [25] described in section 6.2 also support sending multicast and unicast simultaneously.

Chapter 10

Conclusion and Future Work

10.1 Conclusion

A question asked in the introduction was "How to distribute information in an Ad Hoc network where there is a high degree of mobility and nodes have lower and more dynamic capacity than in wired networks?". This has been investigated through this Master's degree work.

The most convenient way to distribute content to a selective group of nodes is by using multicast. In wired networks there are many well working multicast protocols. In MANETs, however, the frequently changing topology and low capacity result in problems for multicast protocols that are made for wired networks. Different multicast protocols for MANETs have been looked at in this thesis. One solution which has been used for testing is MOLSR with flooding and forwarding MPRs. This protocol is flat and suits best relatively small networks where the degree of mobility is not very high.

Apart from multicast forwarding, layering is an interesting content distribution idea which has a great potential in MANETs. Layering can be used both to limit and scale the distribution in MANETs in the sense that the most important data is sent in a base channel, whereas additional information or quality is sent in additional layers. Receivers can adjust the number of channels they want according to available bandwidth and capacity. The layering can also mean that hosts with high capacity receive data using a higher rate than low capacity hosts. Hence, the sent data is limited to available bandwidth and capacity. The file delivery protocol FLUTE has the capability of sending multicast data in different channels. The hosts decide how many channels they want to receive. By using the a congestion control scheme the number of channels in FLUTE can be adjusted continuously according to available bandwidth.

An architecture consisting of FLUTE combined with MOLSR in an emulated ad hoc network has been tested using five nodes. The first test results showed that multicast forwarding was not working and several tests were done to find out the source of error. An explanation of the error was that the multicast plugin to OLSR did not forward multicast packets due to changes in the new version of the operating system Linux Fedora Core. Laptops with an older version of Linux Fedora Core managed to forward multicast packets; up to 80 percent of the data was received four hops from the sender. Unexpectedly, sending several channels did not improve the amount received. However, layered content was not available. Instead, the same data was sent in all channels. The video distribution applications VIC and VLC can be used to make layered content. In addition, several codecs are capable of making layered content; especially MPEG-4 will probably be important for media distribution in the future when more implementations of the codec become available.

Despite the test problems with software versions not working together and some tests not working as expected, the belief of layering as a method for content distribution in MANETs is still viable. It is a way to limit the amount of traffic in the parts of the network where the link or node capacity is low. FLUTE is one protocol which supports layering of multicast in ad hoc networks. The tested FLUTE implementation is not ideal for MANETs, as it is not threadsafe and uses RLC with IGMP, which are mainly for wired networks. However, improvements will come, for example threadsafe sessions which will make it easier to send both unicast and multicast sessions simultaneously. Other solutions for sending multicast and unicast simultaneously are the use of the nonstandard protocols MFP and the WNet MANET framework; alternatively the use of Xcast.

10.2 Future Work

This thesis has lead to some results as shown above. However, to improve the distribution in MANETs the following is suggested:

Firstly, the MOLSR plugin should be made operating on Fedora Core 5 and other operating system versions to make it possible to forward multicast packets. Generally, it is needed to improve the interoperability to make different software application versions compatible with each other.

FLUTE needs layered content to obtain effects of layering in MANETs. VIC and VLC may be used for this purpose. In addition, codecs, such as MPEG-4, produces layered code. Testing of video streaming using codecs is yet to be tested in MANETs.

An improvement of the FLUTE implementation for ad hoc networks is to include a congestion control scheme for MANETs in FLUTE. Alternatively, bandwidth estimation [36, 14, 45] can be used for deciding the number of channels used in FLUTE layering.

A last improvement could be to test a larger and more mobile network than done in this thesis. A more mobile test network can be obtained e.g. by setting a topology emulator to change attenuation between the machines over time, or by moving laptops with WLAN adapters around while distributing content. If the network becomes sufficiently large hierarchical or geographical multicast protocols may be needed.

Bibliography

[1] A. Adams, J. Nicholas, and W. Siadak. Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised). RFC 3973 (Experimental), January 2005.

[2] Brian Adamson. trpr 2.0b1 user's guide. `http://pf.itd.nrl.navy.mil/protools/trpr.html`.

[3] A. Ballardie. Core Based Trees (CBT) Multicast Routing Architecture. RFC 2201 (Experimental), September 1997.

[4] S. Basagni, M. Cont, S Giordano, and I. Stojmenovic. *Mobile Ad Hoc Networking*. IEEE Press, New Jersey, 2004.

[5] R. Boivie, N. Feldman, IBM, Y. Imai, WIDE / Fujitsu, W. Livens, Colt Telecom, D. Ooms, OneSparrow, O. Paridaens, and Alcatel. Explicit multicast (xcast) basic specification. `http://www.ietf.org/internet-drafts/draft-ooms-xcast-basic-spec-09.txt`, December 2005.

[6] Rick Boivie, Nancy Feldman, and IBM Watson Research Center. Small group multicast ¡draft-boivie-sgm-00.txt¿. `http://www.potaroo.net/ietf/all-ids/draft-boivie-sgm-00.txt`, March 2000. IETF Draft.

[7] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan. Internet Group Management Protocol, Version 3. RFC 3376 (Proposed Standard), October 2002.

[8] CaziTech. Homerf archives. `http://www.cazitech.com/HomeRF_Archives.htm`.

[9] Cho. Optimized multicast. `http://olsrinterop.free.fr/presentations/SAMSUNG-multicast-OLSR.pdf`.

[10] Kihwan Choi, Kwanho Kim, and Massoud Pedram. Energy-aware mpeg-4 fgs streaming. `http://atrak.usc.edu/~massoud/Papers/mpeg4-fgs-dac03.pdf`.

[11] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003.

[12] S. Deering. Multicast routing in a datagram internetwork, 1991. Phd Thesis.

[13] S.E. Deering. Host extensions for IP multicasting. RFC 1112 (Standard), August 1989. Updated by RFC 2236.

[14] Mathieu Déziel and Louise Lamont. Implementation of an ieee 802.11 link available bandwidth algorithm to allow cross-layering, 2005.

[15] Anders EGGEN and Raymond HAAKSETH. A survey of multicast transport protocols for jxta over disadvantaged grids, 2004. FFI/NOTAT/2004/02243.

[16] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. RFC 2362 (Experimental), June 1998.

[17] Matthew S. Gast. *802.11 Wireless Networks*. O'Reilly & Assocates, Inc, 2002.

[18] Mohsen Guizani. Editorial. *Wireless Communications and Mobile Computing*, 1:337–338, 2001.

[19] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP: Session Initiation Protocol. RFC 2543 (Proposed Standard), March 1999. Obsoleted by RFCs 3261, 3262, 3263, 3264, 3265.

[20] Anis Laouti (Hipercom and Inria). Smolsr-molsr. `http://hipercom.inria.fr/SMOLSR-MOLSR/documentation.html`, 2003.

[21] Kenneth Holter. Ad hoc on-demand distance vector (aodv). `http://folk.uio.no/kenneho/studies/essay/node15.html`, 2005.

[22] Cisco Systems Inc. Wireless quality-of-service. `http://www.cisco.com/en/US/products/hw/wireless/ps430/prod_technical_reference09186a0080144498.html`.

[23] Red Hat Inc et al. Fedora core 5 release notes. `http://www.fedora.redhat.com/docs/release-notes/fc5/release-notes-ISO/`.

[24] Humboldt Universitat Informatik. Hidden node problem. `http://sarwiki.informatik.hu-berlin.de/Hidden_Node_Problem`.

[25] INSC. Interoperable networks for secure communications - task 3 (mobility) final report. Draft, May 2006.

[26] Ming-Yee Iu. Explicit multicasting within a unicast infrastructure. `http://csdl.computer.org/comp/proceedings/iscc/2003/1961/00/19610460abs.htm`, 2003.

[27] Frank T. Johnsen. Distribution middleware for multimedia with qos. Presentation for a Middleware forum at FFI, May 2006.

[28] Lotte Johnsen and Markus Vangli. Routing protocols in mobile ad hoc networks, December 2005.

[29] jpeg info@uunet.uu.net. What is progressive jpeg? `http://www.faqs.org/faqs/jpeg-faq/part1/section-11.html`, 1999.

[30] Klara Nahrstedt Kai Chen. Effective location-guided overlay multicast in mobile ad hoc networks. *International Journal of Wireless and Mobile*

Computing (IJWMC), Special Issue on Group Communications in Ad Hoc Networks, Inderscience Publishers, 3:?–?, 2005.

[31] Thomas Knunz. *Multicasting: From Fixed Networks to Ad Hoc Networks*, chapter 23, pages 495–508. John Wiley & Sons, Inc, 2002.

[32] C. Krasic and J. Walpole. Qos scalability for streamed media delivery. Technical Report CSE-99-011, Oregon Graduate Institute of Science & Technology, 1999.

[33] Charles Krasic and Jonathan Walpole. Qualityadaptive media streaming by priority drop. `http://www.cse.ogi.edu/tech-reports/2002/02-015.pdf`, January 2002.

[34] Anis Laouiti, Phillippe Jacquet, Pascale Minet, Laurent Viennot, Thomas Clausen, and Cedric Adjih. Multicast optimized link state routing. `http://hal.inria.fr/docs/00/07/18/65/PDF/RR-4721.pdf`, February 2003.

[35] Erlend Larsen. Ad-hoc olsr-multicast eksperimenter, 2005.

[36] Frank Yong Li, Mariann Hauge, Andreas Hafslund, Øivind Kure, and Pål Spilling. Estimating residual bandwidth in 802.11-based ad hoc networks: An empirical approach, 2004.

[37] Azman Osman Lim. Ad hoc networking. `http://yoshida.kuee.kyoto-u.ac.jp/~aolim/text/adhoc/text/1_1.html`, 2003.

[38] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, and J. Crowcroft. Asynchronous Layered Coding (ALC) Protocol Instantiation. RFC 3450 (Experimental), December 2002.

[39] M. Luby, J. Gemmell, L. Vicisano, L. Rizzo, M. Handley, and J. Crowcroft. Layered Coding Transport (LCT) Building Block. RFC 3451 (Experimental), December 2002.

[40] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, and J. Crowcroft. Forward Error Correction (FEC) Building Block. RFC 3452 (Experimental), December 2002.

[41] H. Mehta, R. Walsh, Nokia, V. Roca, INRIA Rhone-Alpes, J. Peltotalo, S. Peltotalo, and Tampere University of Technology. Flute interoperability testing guidelines draft-mehta-rmt-flute-iop-04. `http://mirror.switch.ch/ftp/mirror/internet-drafts/draft-mehta-rmt-flute-iop-04.txt`, June 2005.

[42] H. Mehta, R. Walsh, Nokia, V. Roca, INRIA Rhone-Alpes, J. Peltotalo, S. Peltotalo, and Tampere University of Technology. Flute interoperability testing guidelines draft-mehta-rmt-flute-iop-05. `http://www.ietf.org/internet-drafts/draft-mehta-rmt-flute-iop-05.txt`, December 2005.

[43] UCL Multimedia. Robust audio tool (rat). `http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/`.

[44] Christoph Neumann and Vincent Roca. Impacts of the startup behavior of multilayered multicast congestion control protocols on the performance of content delivery protocols. `http://doi.ieeecomputersociety.org/10.1109/WCW.2005.11`, 2005.

[45] Martin Normann Nielsen. Bandwidth estimation over 802.11b unicast and multicast traffic. martinnn@ifi.uio.no, 2006.

[46] The National Institute of Standards and Technology. Advanced network technologies division, wireless ad hoc networks. `http://w3.antd.nist.gov/wahn_bkgnd.shtml`.

[47] Tampe University of Technology. Mad/tut project's home page. `http://www.atm.tut.fi/mad/`.

[48] D. Ooms, W. Livens, and Alcatel. Internet-drafts database interface: draft-ooms-cl-multicast-03. `https://datatracker.ietf.org/public/`

idindex.cgi?command=id_detail\&id=4355. Expired.

[49] D. Ooms, W. Livens, and Alcatel. Connectionless multicast. http://archive.dante.net/mbone/refs/draft-ooms-cl-multicast-01.txt, October 1999. Expired.

[50] T. Paila, M. Luby, R. Lehtonen, V. Roca, and R. Walsh. FLUTE - File Delivery over Unidirectional Transport. RFC 3926 (Experimental), October 2004.

[51] PaloWireless. Hiperlan and hiperlan2 resource center. http://www.palowireless.com/hiperlan2/.

[52] Andrew S. Patrick. The human factors of mbone videoconferences: Recommendations for improving sessions and software. http://jcmc.indiana.edu/vol4/issue3/patrick.html, March 1999.

[53] S. Paul. *Multicasting on the Internet and Its Applications.* Kluwer Academic Publishers, 1998.

[54] Jani Peltotalo. File delivery over dvb-h, flute. http://www.tml.tkk.fi/Studies/T-111.590/2005/lectures/peltotalo.pdf, March 2005.

[55] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[56] J. Postel. Internet Control Message Protocol. RFC 792 (Standard), September 1981. Updated by RFC 950.

[57] L Rizzo, L Vicisano, and Jon Crowcroft. The rlc multicast congestion control algorithm. http://info.iet.unipi.it/~luigi/rlc99.ps.gz, 1999.

[58] Ittiam Systems. Mpeg-4 fgs profile. http://www.ittiam.com/pages/products/mpeg4-fgs.htm, 2006.

[59] C-K Toh. *Ad Hoc Mobile Wireless Networks - Protocols and Systems*, chapter 10: Ad Hoc Wireless Multicast Routing, pages 175–213. Printice-Hall, Inc, 2002.

[60] Jean Tourrilhes. Packet frame grouping : Improving ip multimedia performance over csma/ca. `http://www.hpl.hp.com/personal/Jean_Tourrilhes/Papers/Packet.Frame.Grouping.html`, March 1998.

[61] Various. Chinese page about 802.11. `http://www.yesky.com/biz/217863848177172480/20050408/1933027.shtml`.

[62] Various. Gnuplot homepage. `http://www.gnuplot.info/`.

[63] Various. Ieee 802.11 - from wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/IEEE_802.11`.

[64] Various. The itu telecommunication standardization sector (itu-t) homepage. `http://www.itu.int/ITU-T/`.

[65] Various. Jpeg - from wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/JPEG`.

[66] Various. The mpeg home page. `http://www.chiariglione.org/mpeg/`.

[67] Various. Multicast - from wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Multicast`.

[68] Various. Osi model - from wikipedia, the free encyclopedia.

[69] Various. tcpdump / libpcap. `http://www.tcpdump.org/`.

[70] Various. Video codec - from wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Video_codec`.

[71] Various. Wireless lan white papers. `http://www.utdallas.edu/ir/wlans/whitepapers/`.

[72] vic@ee.lbl.gov. vic - video conferencing tool. `http://www-nrg.ee.lbl.gov/vic/`.

[73] Rod Walsh. Advances in mass media delivery to mobiles. `http://mips2004.imag.fr/files/MIPS2004-MMM_tutorial.pdf`, November 2004.

[74] Wei Wang, Soung Chang Liew, and V.O.K. Li. Solutions to performance problems in voip over a 802.11 wireless lan. *Vehicular Technology, IEEE Transactions on*, 54:366– 384, January 2005.

[75] C. Wu and Y. Tay. Amris: A multicast protocol for ad hoc wireless networks, 1999.

[76] M. Gerla X. Hong, K. Xu. Scalable routing protocols for mobile ad hoc networks. University of California, 2002.

[77] Yang Xiao and J. Rosdahl. Throughput and delay limits of ieee 802.11. *Communications Letters, IEEE*, 6:355– 357, August 2002.

[78] Jason Xie, Rajesh R. Talpade, and Anthony Mcauley Mingyan Liu. Amroute: ad hoc multicast routing protocol. *Mobile Networks and Applications*, 7:429–439, 2002.

[79] Changchun Xu, Yanyi Xu, Jingdong Gao, and Jun Zhou. Multi-transceiver based mac protocol for ieee 802.11 ad hoc networks. *IEEE*, 2:804–807, 2005.

[80] Michael Zink. *Scalable Internet Video-on-Demand Systems*. PhD thesis, Technischen Universität Darmstadt, 2003.

Appendix A

FLUTE Test Details

A.1 Test Setup Details

To emulate an ad hoc network network the following elements are used in test setups 1 and 2:

- Five Linux computers with:
  - Operating system: Fedora Core **5**
  - WLAN adapter: 802.11b in Ac Hoc mode
  - IP addresses: 10.10.20.11-55
  - Multicast protocol: MOLSR with SMF
    (Multicast Plugin version 0.0.8 to the UniK implementation version 0.4.9 of OLSR [35])
  - File delivery protocol: FLUTE version 1.4
  - (Video distribution: VLC version 5)
  - Tracking: Tcpdump[69], trpr[2], gnuplot[62]
- Topology emulator

- Files to be sent: Alien.mpg (3127 kilobytes) and tamitatest.txt (37.5 kilobytes)

For test setup 3 a slightly different set of building elements are used:

- Five Linux computers with:
  - Operating system: Fedora Core **2**
  - WLAN adapter: 802.11b in Ad Hoc mode
  - IP addresses: 10.10.20.11-55
  - Multicast protocol: MOLSR with SMF
    (Multicast Plugin version 0.0.8 to the UniK implementation of OLSR [35])
  - Transport protocol: FLUTE version 1.4
  - Tracking: Ethereal
- No topology emulator. Instead, the laptops are carried around in the building to make them hear certain other machines.
- Files to be sent: Alien.mpg (3127 kilobytes) and tamitatest.txt (37.5 kilobytes)

A.2 Congifuration Commands

There are several ways to find out what ip addresses are on the computer, for example:

```
#ip addr
```

or

```
#ifconfig
```

and to see the WLAN settings

```
#iwconfig
```

Check available routes:
```
#route
```

Ping computers to see if the IP addresses work (from all the computers in the network):
```
#ping 10.10.20.44
```

A.2.1 Multicast Address

To be able to use multicast it is necessary to type in this commando:
```
#route add -net 224.0.0.0 netmask 240.0.0.0 dev wlan0
```
This command can be written into the file /etc/rc.d/rc.local to avoid writing it after every boot.

To ping multicast addresses:
```
#ping 224.0.0.1
```

A.3 Commands for running FLUTE

To find the options go to MAD-FLUTE homepage [47].
All commands for FLUTE in this document are started after going to the FLUTE path:
```
#cd /root/flute/flute_mad_fcl_v4.1.1_linux_bin/
```
Text files to transfer are put into a folder files/

Unicast using IPv4:

$$SenderOLSR5:$$

```
#./flute -S -m:10.10.20.44 -p:4000 -t:2 -F:files/test.txt
```

$$ReceiverOLSR4:$$

```
#./flute -A -U -p:4000 -t:2 -s:10.10.20.55
```

Multicast using IPv4:

$$SenderOLSR5:$$

```
#./flute -S -m:224.1.1.1 -p:4000 -t:2 -F:files/test.txt
```

$$ReceiverOLSR4:$$

```
#./flute -m:224.1.1.1 -p:4000 -t:2 -s:10.10.20.55 -F:*.txt
```

Several channels using IPv4:

$$SenderOLSR5:$$

sending 2 channels:

```
#./flute -S -m:224.1.1.1 -p:4000 -t:2
-F:download/olsr3/files/rfc3926.txt -c:2 -V:output-c-sender.txt
```

$$ReceiverOLSR4$$

which wants to receive maximum 2 channels:

```
#./flute -m:224.1.1.1 -p:4000 -t:2 -s:10.10.20.55 -F:*.txt -c:2
```

$ReceiverOLSR3$

which wants to receive maximum 1 channel:

```
#./flute -m:224.1.1.1 -p:4000 -t:2 -s:10.10.20.55 -F:*.txt -c:1
```

A.3.1 Other Commands

Check if files are different :

$OLSR5:$

```
#diff download/olsr4/download/localhost.localdomain/flute flute
```

A.4 FLUTE test data

A.4.1 Test 2

These data belongs to the test described in section 8.3.1. The amount received and the number of lost packets for each test is shown in table A.2.

a1)
As shown in table A.1 in test a1 every sixth second there are 22 packets per second, otherwise there are 21 packets packets per second. Each packet is 1448 bytes. Hence, for one period there are:
((5 x 21 packets/s x 1448 bytes/packet) +

(1 x 22 packets/s x 1448 bytes/packet)) /6 = 30649.333 byte/s.
This multiplied by 8 gives the result in bit/s (there are 8 bits in one byte):
30649.333 byte/s x 8 = 245194.66 bit/s = **245.2 kbit/s**
Hence, the measured rate is very close to the FLUTE rate 250 kbit/s.

a2)
In test a2 every fourth second there are 5 packets per second, otherwise there are 4 packets per second. Each packet is 1448 bytes (times 8 to get bits). Hence, for one period there are:
((3 x 4 packets/s x 1448 bytes/packet) +
(1 x 5 packets/s x 1448 bytes/packet)) 8/4 = 49232 bit/s = **49.2 kbit/s.**
Also here, the measured rate is very close to the FLUTE rate 50 kbit/s.

A.4.2 Test 3

Sender and receiver commands and files

- Sender: `./mgen input lotte-example.mgn`
- Receiver(s): `/root/mgen/MGEN/TcpPlot.sc`

Script TcpPlot.sc for use on the receivers when sender generates packets with MGEN:

```
tcpdump -l -x ip -i eth1 src host 10.10.20.11 |
./trpr real window 0.5 auto X | gnuplot -noraise -persist
```

Example file lotte-example.mgn for use on the sender together with MGEN:

```
0.0 ON 1 UDP SRC 5001 DST 10.10.20.22/5001 PERIODIC [250 1250]
```

The first number is the start time for packet generation. The next number (1) is the name of the signal sent (if more signals they are named 2, 3, 4 and so on). The next means that the packets are sent using UDP from source port 5001 to destination IP address 10.10.20.22 port 5001. The packets are sent periodic; 250 packets per second, and each packet with size 1250 bytes.

A.4.3 Test 4

Sender command:

```
./flute -S -m:224.1.1.1 -p:4000 -t:2 -c:1 -T:2
-F:files/tamitatest.txt -w:1 -r:250
```

Receiver command:

```
./flute -A -m:224.1.1.1 -p:4000 -t:2
-s:10.10.20.11 -B:test4 -w:1
```

A.4.4 Test 5

a)
Sender command:

```
./flute -S -m:224.1.1.1 -p:4000 -t:2 -c:1 -T:4
-F:files/Alien.mpg -l:337 -r:750 -w:1
```

Receiver command:

```
./flute -A -m:224.1.1.1 -p:4000 -t:2
-s:10.10.20.11 -B:test5 -w:1
```

b)
Sender command:

```
./flute -S -U -m:10.10.20.55 -p:4000 -t:2
-f:fdt_tsi2.xml -r:750
```

Receiver command:

```
./flute -A -U -p:4000 -t:2
-s:10.10.20.11 -B:test5 -w:1
```

c)
Sender command:

```
./flute -S -m:224.1.1.1 -p:4000 -t:2 -c:2 -T:4
-F:files/Alien.mpg -l:337 -r:250 -w:1
```

Receiver command:

Channel 1:

```
./flute -A -m:224.1.1.1 -p:4000 -t:2
-c:1 -s:10.10.20.11 -B:test5/ch1 -w:1
```

Channel 2:

```
./flute -A -m:224.1.1.2 -p:4000 -t:2
-c:1 -s:10.10.20.11 -B:test5/ch2 -w:1
```

A.4.5 Test 6

b) Sender command:

```
./flute -S -m:224.1.1.1 -p:4000 -t:2 -c:2 -T:4
-F:files/tamitatest.txt -l:357 -r:350 -w:1
```

Receiver command:

```
./flute -A -m:224.1.1.1 -p:4000 -t:2
-c:2 -s:10.10.20.11 -B:test5/ch1 -w:1
```

A.4.6 Test 9

Sender unicast command:

```
./flute -S -U -m:10.10.20.33 -p:4000 -t:2
-f:fdt_tsi2.xml -r:750 -w:1
```

Sender multicast command:

```
./flute -S -m:224.1.1.1 -p:4000 -t:2 -T:4
-F:files/tamitatest.txt -l:357 -r:750 -w:1
```

Receiver unicast command:

```
./flute -A -U -p:4000 -t:2
-s:10.10.20.11 -B:test9/unicast -w:1
```

Receiver multicast command:

```
./flute -A -m:224.1.1.1 -p:4000 -t:2
-s:10.10.20.11 -B:test9/multicast -w:1
```

A.4.7 Test 10

a)
Sender command:

```
./flute -S -m:224.1.1.1 -p:4000 -t:2 -c:1 -T:4
-F:files/Alien.mpg -l:337 -r:750 -w:1
```

Receiver command:

```
./flute -A -m:224.1.1.1 -p:4000 -t:2
-s:10.10.20.11 -B:test5 -w:1
```

Test No	*Second*	*number of packets*	*Second*	*number of packets*	*Second*	*number of packets*
a1	09	21	26	21	42	21
a1	11	21	27	21	43	21
a1	12	22	28	21	44	21
a1	13	21	29	21	45	21
a1	14	21	30	22	46	21
a1	15	21	31	21	47	22
a1	16	21	32	21	48	21
a1	17	21	33	21	49	21
a1	18	22	34	21	50	21
a1	19	21	35	22	51	21
a1	20	21	36	21	52	21
a1	21	21	37	21	53	22
a1	22	21	38	21	54	21
a1	23	21	39	21	55	21
a1	24	22	40	21		
a1	25	21	41	22		
a2	21	4	30	5	39	5
a2	22	5	31	4	40	4
a2	23	4	32	4	41	4
a2	24	4	33	4	42	4
a2	25	4	34	5	43	5
a2	26	5	35	4	44	4
a2	27	4	36	4	45	4
a2	28	4	37	4	46	4
a2	29	4	38	4	47	5

Table A.1: Packets per second

Test No	*Computer name*	*Percent re-ceived*	*Number of lost pack-ets*	*Note*
2a1	OLSR2	65.69	45	
2a2	OLSR2	66.34	105	
2b	OLSR2	100.00	0	
2c	OLSR2	98.35	9	
2d1	OLSR2	67.70	46	
2d2	OLSR2	66.06	49	

Table A.2: Percentage received and number of packets lost for the machines in test 2

Test No	*Packets per second*	*Packet size [bytes]*	*Receiving computer name*	*kbit/s received*	*Note*	*Figure*
3a	250	1250	OLSR2	900	unicast	
3b1-1	250	1250	OLSR2	900	multicast from this and forward	
3b1-1	250	1250	OLSR3	20-100		
3b1-2	250	1250	OLSR2	900		
3b1-2	250	1250	OLSR3	0-40	a few leaps	
3b1-3	250	1250	OLSR2	900		
3b1-3	250	1250	OLSR3	0-102	a few leaps	
3b1-4	250	1250	OLSR2	900		
3b1-4	250	1250	OLSR3	0-41, 81	mostly around 0-41, one leap to 81	
3b1-5	250	1250	OLSR2	900		
3b1-5	250	1250	OLSR3	0-40, 82	mostly around 0-40, one leap to 82	

Table A.3: Percentage received and number of packets lost for the machines in test 3 (table one out of three)

Test No	*Packets per second*	*Packet size [bytes]*	*Receiving computer name*	*kbit/s received*	*Note*	*Figure*
3b2-1	22	1448	OLSR2	100-200	worse than for higher rates	
3b2-1	22	1448	OLSR3	0-0.6	a few leaps	
3b2-2	22	1448	OLSR2	75-190		
3b2-2	22	1448	OLSR3	0, 24	One leap to 24, otherwise 0	
3b2-3	22	1448	OLSR2	100-220		
3b2-3	22	1448	OLSR3	0, 24	Two leaps to 24 (at beginning and end)	
3b2-4	22	1448	OLSR2	100-220		
3b2-4	22	1448	OLSR3	0, 24	One leap to 24 at the end	
3b3-1	77	1448	OLSR2	600		
3b3-1	77	1448	OLSR3	0, 24	Two leaps to 24 (at beginning and end)	

Table A.4: Percentage received and number of packets lost for the machines in test 3 (table two out of three)

Test No	*Packets per second*	*Packet size [bytes]*	*Receiving computer name*	*kbit/s received*	*Note*	*Figure*
3b4-1	300	400	OLSR2	800		
3b4-1	300	400	OLSR3	275		
3b4-2	300	400	OLSR2	800		
3b4-2	300	400	OLSR3	275		
3b5-1	400	400	OLSR2	800		
3b5-1	400	400	OLSR3	0-240, 350	Onle leap to 350: highest throughput for two hops	
3b6-1	500	380	OLSR2	800		
3b6-1	500	380	OLSR3	310	best throughput overall for two hops	
3b6-1	500	480	OLSR2	800		
3b6-1	500	480	OLSR3	220		
3b6-1	500	780	OLSR2	900		
3b6-1	500	380	OLSR3	40-130, 155	A few leaps to 155, otherwise 40-130	

Table A.5: Percentage received and number of packets lost for the machines in test 3 (table three out of three)

Test No	*-r [bit/s]*	*-l [bytes]*	*Receiving computer name*	*Percent received*	*Number of lost packets*	*Note*
4a1	250	1428	OLSR2	52.06	27	
4a1	250	1428	OLSR3	0	-	starts running, but receives nothing
4a2	250	714	OLSR2	85.13	2	
4a2	250	714	OLSR3	3.72	17	better than 4a1
4a3	357	714	OLSR2	100	0	good
4a3	357	714	OLSR3	20.45	32	even better than 4a2
4a4-1	357	714	OLSR2	99.61	9	large file in test 4a4-1 and forwards
4a4-1	357	714	OLSR3	12.54	404	more lost packets because of large file
4a4-2	357	714	OLSR2	99.50	10	
4a4-2	357	714	OLSR3	12.44	277	
4a4-3	357	714	OLSR2	99.53	10	
4a4-3	357	714	OLSR3	12.54	344	
4a5-1	357	176	OLSR2	99.71	11	
4a5-1	357	176	OLSR3	0	227	gets RLC Warning: Max number of late packets received
4a5-2	357	176	OLSR2	99.65	11	
4a5-2	357	176	OLSR3	0	640	gets RLC Warning: Max number of late packets received

Table A.6: Percentage received and number of packets lost for the machines in test 4 with different bit rates and packet sizes (table one out of three)

Test No	*-r [bit/s]*	*-l [bytes]*	*Receiving computer name*	*Percent re-ceived*	*Number of lost pack-ets*	*Note*
4a6-1	357	500	OLSR2	99.67	5	large file in test 4a4-1 and forwards
4a6-1	357	500	OLSR3	17.19	364	more lost packets because of large file
4a6-2	357	500	OLSR2	99.61	5	
4a6-2	357	500	OLSR3	16.17	349	
4a7-1	357	750	OLSR2	99.64	5	
4a7-1	357	750	OLSR3	18.59	427	
4a7-2	357	750	OLSR2	99.60	5	
4a7-2	357	750	OLSR3	20.60	251	The highest amount received for OLSR3
4a8-1	357	1000	OLSR2	77.97	95	Worse than in the other tests. Understandable, since the rate is 1000 kbit/s and max throughput to OLSR2 is only approximately 900 kbit/s.
4a8-1	357	1000	OLSR3	17.37	314	
4a8-2	357	1000	OLSR2	78.12	102	
4a8-2	357	1000	OLSR3	17.92	467	

Table A.7: Percentage received and number of packets lost for the machines in test 4 with different bit rates and packet sizes (table two out of three)

Test No	*-r [bit/s]*	*-l [bytes]*	*Receiving computer name*	*Percent re-ceived*	*Number of lost pack-ets*	*Note*
4b1-1	357	1000	OLSR2	0	107	without MOLSR
4b1-1	357	1000	OLSR3	0	339	without MOLSR
4b1-2	357	1000	OLSR2	78.21	100	without MOLSR
4b1-2	357	1000	OLSR3	0	378	without MOLSR, OLSR3 receives nothing
4b1-3	357	1000	OLSR2	78.41	102	without MOLSR
4b1-3	357	1000	OLSR3	0	401	without MOLSR, OLSR3 receives nothing
4c1-1	357	250	OLSR2	0	-	topology emulator: OLSR1-OLSR2: Attenuation 60 dB
4c1-1	357	250	OLSR3	0	-	
4c2-1	357	250	OLSR2	96.24	22	topology emulator: OLSR1-OLSR2: Attenuation 50 dB
4c2-1	357	250	OLSR3	0	-	receives nothing
4c2-2	357	250	OLSR2	96.34	22	topology emulator: OLSR1-OLSR2: Attenuation 50 dB
4c2-2	357	250	OLSR3	0	-	receives nothing

Table A.8: Percentage received and number of packets lost for the machines in test 4 with different bit rates and packet sizes (table three out of three)