Quality of Service (QoS) in mobile ad hoc networks

By

Morten Kronstad Vinje

Formatert: Engelsk (Storbritannia)

Agder University College
Faculty of Engineering and Science

Grimstad

Formatert: Engelsk (Storbritannia)

July 2006

I Abstract

Mobile ad hoc networks inherits several of the issues that ordinary wireless networks have and include several new ones. With internet becoming a multi media enviroment, there are many new requirements that must be satisfied, especially when it comes to real time service. Quality of servce protocol for MANETs has been a topic of interest. In this paper I propose a QoS –aware bandwidth protocol based on available bandwidth calculation. My proposal is based on a few existing proposals today and the theory around these. I then show an implementation available bandwidth calculation and routing functions. This implementation has not been completely sucsessful. This paper should give a view on what is necessary to achieve quality of service in a MANET.

II Preface

This master thesis is the end report for my Master of Science in Information and Communication Technology education at HiA. The thesis was written during the spring and summer of 2006.

The thesis was initiated by professor Andreas Hafslund who graduated at the University Graduate Center in Kjeller (UniK) and works for Thales. Andres Hafslund is an expert in various communication techniques, and has been the contact person to Thales. Professor Andres Prinz has been my supervisor at Hia. Andreas Prinz was appointed as a professor for Systems Development at HIA in 2003. He studied mathematics and computer science at the Humboldt-University in Berlin, Germany, and received his M.Sc. in mathematics (1988) and Ph.D. (1990) in computer science. Prinz has worked in several projects dealing with the development of modern telecommunication systems using advanced technology and has therefore great knowledge within the field of my thesis.

Andreas Hafslund and UniK has been a great resource to me in this thesis. Especially in the start when I performed parts of my studies at UniK. UniK is a graduate educational institution for master and doctoral students, primarily from UiO and NTNU, but also for continuing education students from commerce and industry. UniK offers courses and supervision on behalf of the students' home institution, and the students receive university degrees from their respective institutions.

I also want to thank HiA and Andreas Prinz for a professional behavior and valuable feedback when needed, and Andreas Tønnesen for previous work on the oslrd protocol.

I would also like to thank the following persons for their efforts and understanding manner through my research: Sissel Andreassen and Stein Bergsmark

III Table of contents

VI Abbreviations

ABR	-	Associatively-based routing
ABW	-	Available bandwidth
ACK	-	Acknowledgement
Ad-hoc	-	Latin phrase which means “for this purpose”
AIMD	-	Additive Increase, Multiplicative Decrease
ANSN	-	Advertised Neighbour Sequence Number
AODV	-	The Ad hoc On-demand Distance Vector
ARQ	-	Automatic repeat request
CA	-	Collision Avoidance
CE	-	Congestion experienced
CSMA	-	Carrier sense multiple access
CSMA/CA	-	Carrier sense multiple access with collision avoidance
CTS	-	Clear to send
DIFS	-	Distributed coordination function interframe space
DLAR	-	Dynamic load-aware routing
ECN	-	Explicit congestion notification
EIFS	-	Extended interframe space
ETX	-	Expected transmission
GSM	-	Global System for Mobile Communication
IEEE	-	Institute of Electrical and Electronics Engineers or IEEE
IP	-	Internet protocol
LQ	-	Link quality
MAC	-	Media Access Control
MANET	-	Mobile Ad-hoc network
MID	-	Multiple Interface Declaration
MPR	-	Multipoint Relay
MSC	-	Mobile switching centre
NAV	-	Network allocation vector
NLQ	-	Neighbour link quality
NP	-	Nondeterministic polynomial
OLSR	-	Optimized Link State Routing protocol
PDA	-	Personal Data Assistance
QMPR	-	QoS Multipoint Relay
QOLSR	-	QoS OLSR
QoS	-	Quality of service
RREP	-	Route reply message
RREQ	-	Route request message
RERR	-	Route Error Message Format
RTS	-	Ready to send
SSR	-	Signal stability routing
SWAN	-	Stateless Wireless Ad Hoc Networks
TC	-	Topology control
TC-message	-	Topology control messages
TCP	-	Transmission Control Protocol (TCP)

TOS	-	Type-of-Service
TTL	-	Time to live
UDP	-	User Datagram Protocol (UDP)

1. Introduction

1.1 Background

In the past few years wireless computing has been introduced to the commercial market and has proven to be a rapidly emerging technology. As this technology still is a relatively new approach to computer traffic, and as the option of wireless connections has moved into our houses, offices, and into public areas, there are several issues that must be worked out. Customers expect to have the same quality of services and capabilities using wireless networks that they get from wired networks.

The Ad hoc network concept was originally developed for military use. Finding a use for this technology in the commercial setting and solving the many issues related to this technology has become one of the most active development fields in the wireless community.

Wireless ad-hoc networks have several desirable qualities for military and commercial use. These networks do not need any infrastructure, can be established almost anywhere, and are dynamic in nature. Because of the dynamic nature, they change over time as the nodes move around, joining or leaving the network. Mobile Ad hoc network (MANET) is an especially difficult task to explore because the technology is in an early stage and the commercial market is making new demands.

The nodes that make up a network at any given time communicate with and through each other. In this way every node can establish a connection to every other node that is included in the MANET. Examples of nodes can be personal devices like, our mobile phones, Laptops, Personal Data Assistants (PDA's), etc. Smaller and simpler devices also use wireless ad-hoc networking, like wireless headsets, hands free, etc.

The MANET research community has not yet developed a complete standard to use on the routing, quality of service (QoS) or how to best estimate available bandwidth. That is why it is important to study the different theories and combine them to find the best possible overall solution to make the MANET a fully working wireless network alternative.

1.2 Problem statement

The problem statement below is translated from Norwegian.

A mobile ad-hoc network (MANET) is an autonomous system of mobile nodes. The nodes work as routers for each other. In this way all communication in the network will go through one or more hops through nodes. The network has no common infrastructure or centralized access point. The nodes use wireless communication, and can move freely. This means that the topology in the network will be random and dynamic, resulting in great demands on the routing protocol.

Future service and applications in mobile communication, must work adaptively to the mobile terminals capacity and the underlying characteristics of the network. This results in requirements to the quality of service (QoS), which is delivered through the network. OLSR is an ad-hoc routing protocol that is open source and based on optimized link state routing. The actual implementation of OLSR that will be used in the project comes from www.olsr.org. The thesis is a closer study of the interaction between QoS and routing protocols in an ad-hoc network and involves three parts:

1. The proposal of a model for QoS, based on theoretic studies of other proposals. The two proposals of most interest is:
    - ETX: OLSRD link quality extension (www.olsr.org)
    - QOLSR (http://qolsr.lri.fr/)

   OLSR should be extended to include QoS routing functionality that is proposed by the student. The QoS metric that should be used is available bandwidth, measured on the local IP layer.

2. A thorough analysis should be done of the chosen model for QoS routing. This should be done by looking at pros and cons of the chosen functionality and the related implementation that is done.

3. The main task is to implement in C under Linux the extended functionality to OLSR. This could be done as a plug-in to accomplish QoS routing. The task should be concluded by a proof of concept test shat shows that the implementation actually works

1.3 Limitations

- Because of the many problems related to MANET, this thesis focuses on the problem at hand and does not include the effect of my implementation on the total picture, e.g. what effects are to security etc.
- Because of the hardware necessary for testing and that the available bandwidth algorithm only works one some wireless cards a real life test impossible
- Because of the time scale of the master thesis some of the theories that are discussed in this paper are not implemented.

The SWAN protocol theory is the basis of my thesis as this is a working QoS-aware protocol. After thorough review of the theory and implementation, and testing of its functions I found that many of its functionalities was more complex than foreseen. Especially the approximate bandwidth estimation and the QoS flow control within the network. Because of this the master thesis has been concentrated around calculating available bandwidth, message handling and routing based available bandwidth.

1.4 Purpose

The purpose of this thesis is to contribute to the solving the many problems that MANETs faces. If MANET is to be commercially widespread and the attractive qualities used, the research community must resolve and test as many aspects of the problems as possible. Some of the most prominent issues to handle are discussed in this paper. The implementation and implementation suggestions in this paper are meant to contribute to the further development of MANETs.

1.5 Motivation

My main motivation in this master thesis was that I always have had an interest of how things work. I had mostly theoretical experience routing protocols and low-level networks, and wanted to get more experience in the programming part. This way MANET has been a very good area of research. I have also had the possibility to test my theories, and have achieved a deep understanding of my first program of this magnitude.

Much of the earlier work on protocols for MANET network is targeted on finding feasible routes to the destination without considering current network traffic or application requirements. From this work it can be concluded that the probability of the network becoming overloaded is high. This might be acceptable for data transfer, but as real-time applications are becoming an important part of the traffic, a QoS-aware routing protocol that incorporates an admission control scheme and feedback scheme to meet the requirements of real-time application, seems to be a good solution.

In [11], it is shown that a similar approach to the Ad hoc On-demand Distance Vector (AODV) network has the result of increasing packet delivery ratio greatly, while packet delay and energy dissipation decrease significantly, and the overall end-to-end throughput is not impacted, compared with routing protocols that do not provide QoS. I believe my work can contribute to similar results in the OLSR environment.

1.6 Chapter Overview

This paper will start off with an introduction of the basic knowledge needed to understand how a wireless network operates, and some of the general challenges and constraints that should be solved to meet the costumer’s demands, in chapter 2. Chapter 2 also reviews some of the most prominent solution to routing and includes a detailed discussion of OLSR. Chapter 3 handles the specific quality of service area. This chapter shows which issues must be solved discusses existing protocol solutions for QoS, some of the topics of chapter 3 will be discussed I chapter 4 in an implementation view. Chapter 4 discusses the functionalities that I want to implement. Chapter 5 Test the implementations done and discusses the results. In chapter 6 I suggest what work that should be done to make my protocol work properly and last but not least chapter 7 ends my paper with a conclusion.

2. Literature study

In this part of the paper I will give an introduction of the basic knowledge needed to understand how a wireless network operates. The theories in this part are essential to understanding later topics that are related to my Diploma Thesis.

2.1 The wireless network

There are two basic types of wireless networks that are of interest; the cellular concept and the Ad hoc concept. The cellular concept is basically the same as is used in cellular phone technology (GSM), and is a highly researched area. But there are still problems to be solved, especially concerning sending heavy loads of data traffic in the network. The cellular concept is mentioned to give the reader an understanding of the difference between networks that need infrastructure and the networks that don't need infrastructure. The Ad hoc network concept as originally developed for military use. The Ad hoc network will be discussed thoroughly in this paper together with the mobility issue that influences all types of wireless networks and protocol types.

2.1.1 Cellular concept

The cellular concept replaces the old fashioned high powered transmitter (large cell) with many low power transmitters (small cells). Each cell has one base station that provides coverage of a small portion of the total service area. Each station (cell) is allocated a set of channels from the total channel pool for the whole service area. Each neighbouring cell is assigned a different set of channels. The result is systems where interference is minimized

By limiting the coverage area to within the boundaries of a cell, the same group of channels may be used to cover different cells that are separated from another by distances large enough to keep interference levels within tolerable limits. This is called frequency re-use or frequency planning and is done by clustering cells together. The cell cluster is a group of cells that together represents all the channels that the network (service) has available. The cluster size is typical 3, 7 or 12. *Figure 1* illustrates the concept of cellular frequency re-use with seven cells, where cells labeled with the same letter use the same group of channels. The figure is a conceptual and simplistic model of the radio coverage for each base station. The hexagon radio coverage is actually determined from complex field measurements and propagation prediction models. It is important that the distance between cells with the same channel sets (group) is large enough to keep the interference level down. It is proven that the capacity of the network is proportional with the number of times the set is repeated in the network. This gives the network the characteristic property that when the number of cells in the clusters decreases, the capacity of the network increases. The capacity is also determined by the total bandwidth of the channels and cell size. As the cell size decreases the capacity per quadrate (measurement) increases.

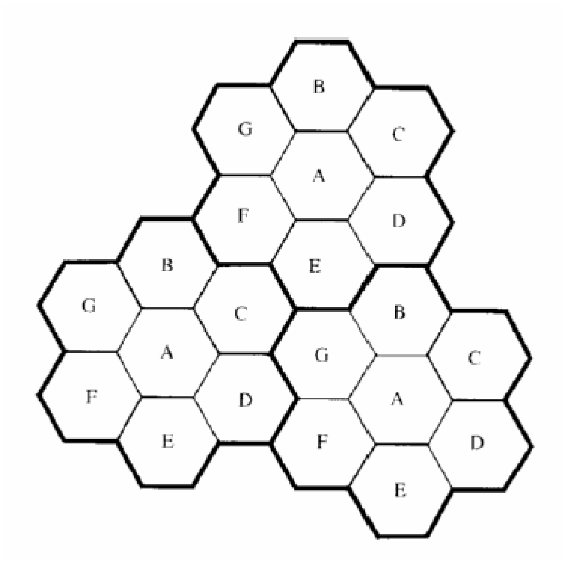


Figure 1: Cellular network

In a cellular network, the backbone network performs all networking operations, and mobile devices only communicate directly with one of the base stations of which they are within reach. The cellular concept has proven to be reliable and has good coverage. On the downside, it is dependent on a strong infrastructure to be able to perform, which is expensive and not a good solution in all areas.

2.1.2 Ad hoc concept

Research within the Ad hoc technology has been ongoing for nearly 30 years. Work within this field has rapidly expanded due to recent technology breakthrough in wireless devices and battery capacity. The network community has become more interested as the wireless devices have become inexpensive and widely available.

Unlike cellular networks, the ad hoc concept is basically several communication devices (nodes) that communicate with each other with no infrastructure or pre-determined organization of available links, as *Figure 2* illustrates. This is done by giving each individual node the responsibility of dynamically discovering which other nodes they can communicate with directly. In multi hop ad hoc networks some nodes might be out of reach from each other, and do not have the possibility of communicating directly with each other. In order to deliver packages through the network, each node is required to redirect the packages from the originator to their destination. The way the redirections are performed, is decided by the routing protocols and will be discussed later in this section.

Having no supporting infrastructure means that all network intelligence must be situated inside the mobile devices that make up the network. The nodes in an ad hoc network act as both hosts and routers. Because the network topology can change quickly and unpredictably as a result of nodes leaving and entering network and link breakage, it should be self-configuring and very adaptable to changes.

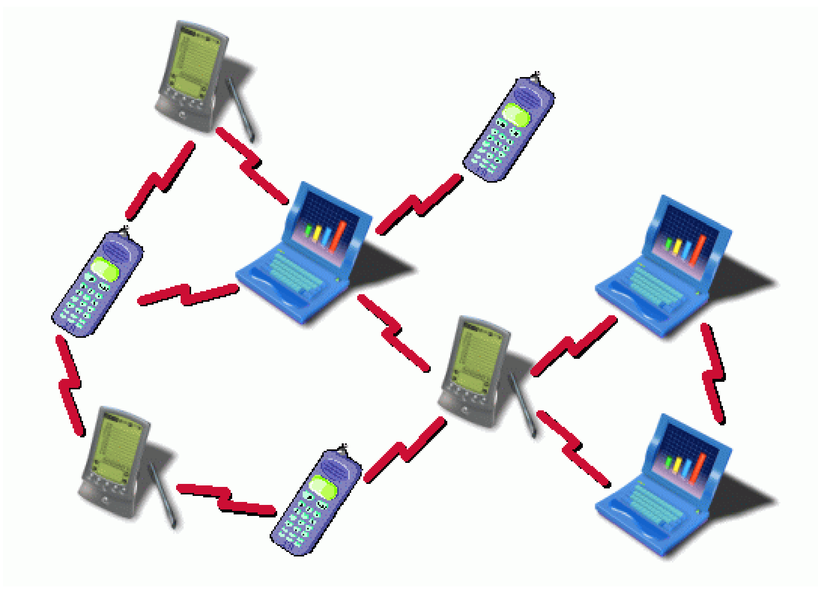


Figure 2: Ad hoc network

The Ad hoc network is not designed to be used in large networks where it is important to always be reachable, as in 2G and 3G cellular systems. Its main strengths are rather robustness of the network, the seamless connectively with devices in the neighbourhood and the fact that the network only consists of the participating devices. Listed below are several applications that can benefit from an Ad-hoc network.

Conference:
In the modern conference it is almost unthinkable that the members do not have a laptop, notebook or PDA available. It would be nice for them to be able to seamlessly and with no configuration to exchange information, use a local printer or connect to the Internet via a wireless Internet gateway. This way every one can immediately download the current presentation, browse through it on their laptop, print it on the local printer or e-mail it to colleagues. When the ad hoc network is used to support this application, there is no need for infrastructure and it avoids unnecessary overhead.

Rescue operations:
There are many situations where there is no infrastructure present, but where it is necessary to establish a network fast. Situations like nature disasters, wars and hunger crisis in underdeveloped countries, are examples of this. Ad-hoc networks are an obvious solution, since such networks can be deployed very quickly and have no need of any existing infrastructure.

Home networks:
Today, many households have several computers in different rooms. Most people would like to connect these to each other. Even if there are several possible ways of connecting them together, the ad hoc network is a very easy and elegant solution. In the future more devices in

the household (washing machine, refrigerator, heating, etc) will be network controlled. Some of these devices are portable and some stationary. The ad hoc network would bee an ideal solution to make devices communicate with each other and become remotely controllable in an environment with changing topology.

Personal area networks (PAN):
The PAN connects several devices on a person. A good example of this is a soldier. A modern soldier has a video camera on his weapon and video information is broadcasted directly to his goggles. He has sensors that register how many bullets there are left in the clip and the total amount available. He has sensors that warn him about toxics in the air and other useful information. These devices can communicate with each other and thus give the soldier unique situational awareness. This information can be sent to the troop commander in a new ad hoc network between soldiers. This technology can be transferred to the commercial market and has several areas of application.

Sensor Dust:
It is difficult to monitor terrain and dangerous environmental conditions. It could be done by employing several tiny and cheap sensors (nodes), forming an ad hoc network. The ad hoc network gives the possibility of remotely coordinating the sensor, instead of endangering personnel. The robustness of the network is another beneficial quality of the ad hoc network. If one sensor fails it will not affect the information flow from the remaining sensors. An example of an ideal place to install this technology is in the exploration of the planet Mars. The network can be established with stationary and mobile sensors that can be controlled from Earth.

Games:
This is an example of a completely commercial aspect of the ad hoc network. Customers can now play with the people accidentally within the neighbourhood. This is a great way to pass time in public areas as in trains, train stations or airports.

Military:
It was the U.S Department of Defense that sponsored the first research of ad hoc networks to enable packet switching technology to operate without the restriction of a fixed wired infrastructure. They did this with good reasons; the robustness of the network is without comparison. To cripple the network an enemy must destroy a large percentage of the active nodes (command posts) and even then the information flow will continue in parts of the network. It is virtually impossible to destroy an ad hoc network completely. Since one of the most important tasks in a military campaign is to keep command lines open, it is understandable that the military establishment has put a lot of money into developing ad hoc networks.

2.1.3 Mobility

To be able to understand the basics of mobility, it is important first to understand the different levels of granularity. The three levels of mobility are:

1. Macro-mobility: This is the movement of a device through a global network. It should be possible to move through such a network without the existing communication breaking. Macro mobility is usually solved with Mobile IP

2. Micro-mobility: This is the movement of a device in one single administrative domain of the global network. For most access networks this is the lowest form of mobility. Cellular networks are an example of this.
3. Ad-hoc mobility: This is the lowest form of mobility and handles only the movement of devices within a single ad hoc network.

Protocols have different solutions on how to handle mobility in each level, and most protocols only handle only one level of mobility. They let other protocols handle mobility over the other levels. Ad-hoc mobility handling is discussed more thoroughly in chapter 2.3.4.

2.2 Issues in cellular networks

Within a cellular network the mobility of devices are handled pretty straight forward, but is still an issue that are studied. When a device enters a cell and requests to establish a link, it is assigned a channel, either fixed or dynamic. In a fixed channel assignment strategy the device will be allocated a channel from a fixed set of channels in the cell. If all of the channels are occupied the request will be denied and the subscriber does not receive any service.

In a dynamic channel assignment strategy, channels are not allocated to different cells permanently. The channel allocation is done at a higher level in the hierarchy; the base station sends a request to the *mobile switching centre* (MSC). The MSC allocates the channel to the cell following an algorithm that take into consideration the cost function (likelihood of further blocking within cell, etc.). If the device moves into another cell with the communication link still open, a handoff between the cells occurs. This is generally done by the MSC which automatically transfers the link to a new channel belonging to the new base station (cell).

The processing of handoff is an important task in any cellular radio system and many strategies exist, but this subject is out of scope for this paper and will not be discussed further

2.3 Issues in ad hoc networks

There are several issues within ad hoc networks that make them very complicated to integrate with the existing global internet. Generally the most prominent problems are the identification of mobile terminals and the correct routing of packets from and to each terminal while they are moving. The problems are addressed below.

2.3.1 Routing

Routing is one of the most complicated problems to solve as ad hoc networks have a seamless connectivity to other devices in its neighbourhood. Because of multi hop routing no default route is available. Every node acts as a router and forwards each other's packets to enable information sharing between mobile nodes. There are several ad hoc protocols proposed to solve this problem. This subject is the main part of this paper, and will be thoroughly discussed.

2.3.2 Security

Obviously a wireless link is much more vulnerable than a wired link. The science of cracking the encryption and eavesdropping on radio links has gone on since the first encryption of radio links was established. One of the turning points for the allied forces during World War II was when USA cracked the submarine message encryption system for the German submarines. The German submarines had become a large problem for the allied forces, and now the USA had control over the location and the mission of each submarine in the German navy through surveillance of their radio transmissions.

The fundamental security mechanisms in practically every network are still based on cryptographic keys. Even though this technology has been vastly improved, the example above shows how difficult it is, and the importance of, making a wireless link secure. In an ad hoc network there is increased possibility of eavesdropping, spoofing and denial-of-service attacks due to lack of physical security.

Furthermore, a malicious user can attach to the network. Such a user can load the available network resources, like wireless link and batteries of other users, and disturb normal network operation. All proposed routing protocols place complete trust on the devices that make up the ad hoc network and they are therefore very vulnerable to malicious users. The user can insert spurious information into routing packets and cause routing loops, long time-outs and advertisements of false or old routing table updates. Security has several unsolved issues that are important to solve to make the ad hoc network into a good solution.

2.3.3 Quality of Service (QoS)

QoS is an important issue within ad hoc networks. For ad hoc networks to become more successful within the commercial market, the gap between wired and wireless QoS must be closed. This is a difficult task for the developers, because the topology of an ad hoc network will constantly change. Reserving resources and sustaining a certain quality of service, while the network condition constantly changes, is very challenging. There is a large interest within the wireless community to improve the QoS in ad hoc networks and there are many approaches to do this.

This paper discusses on the SWAN protocol as it is agreed through the community that this is the best approach to QoS today and shows what aspects a working QoS routing protocol should include. But there are some problems with this protocol as discussed in section 3.4. When it comes to QoS, scalability is a major problem. Scalability can be defined as whatever the network can provide of acceptable level of service to packets, even in the presence of a large number of nodes in the network.

In section 3 of this paper I give a detailed walk through of the many aspects of QoS. Other implementation approaches to the QoS problem will also be discussed.

2.3.4 Mobility in ad-hoc networks

The complicated part of handling mobility in an ad hoc network is to support the ad hoc mobility. The macro mobility is normally solved with *mobile IP*. This section gives a brief walkthrough of the most important issues, desirable qualities and metrics for ad hoc mobility. Since there is no infrastructure, there is no way of guaranteeing the overview over all nodes and their mobility in the network. It is the routing protocol that decides how detailed the network will register the movement of nodes. This section of the paper explores some of the characteristics and issues and lists some desirable qualities for MANET routing and quantitative metrics that can be used to assess the performance of the routing protocol. The lists in this section are taken from [14], and some of the text has been altered slightly.

The MANET exists of multiple nodes that move independent of each other and organize themselves arbitrarily. Thus the network's wireless topology may change rapidly and unpredictably. Such a network may operate in standalone fashion, or may be connected to the larger Internet.
The MANET inherits the traditional problems of wireless communication and ad hoc networks. Listed below are some more characteristics with adjacent issues for MANET networks.

Dynamic topologies:
Because nodes can move arbitrarily, the network topology can change, frequently and unpredictably and consist of both bidirectional and unidirectional. This results in route changes, frequent network partitions and possible packet losses.

Bandwidth constrained:
Wireless links have significantly lower capacity than their hardwire counterparts. One effect of this is that congestion is typically the norm rather than the exception and this problem will only increase as applications demand more bandwidth

Variation in link and node capability:
Each node is equipped with at least one radio interface that all have varying transmission/receiving capabilities and operate across different frequency bands. This may result in possible asymmetric links.

Energy constrained operation:
Each node in the MANET network has limited power supply and processing power is limited. This in turn limits services and services that can be supported by each node. This becomes a bigger issue in mobile ad hoc network, as each node is acting as both an end system and a router at the same time. Additional energy is required to forward packets from other nodes.

In the process of trying to make a route in an MANET network, we must first know what the desirable qualities are. The following is a list of desirable qualitative properties of MANET routing protocols.

1. Distributed operation: This is an essential property, but it should be stated nonetheless.

2. Loop freedom: Desirable to avoid problems such as worst-case phenomena, e.g. a small fraction of packets spinning around in the network for arbitrary time periods.
3. Demand based operation: Instead of assuming a uniform traffic distribution within the network, let the routing algorithm adapt to the traffic pattern on a demand or need basis.
4. Proactive operation: The flip side of demand based operation. In certain context, the additional latency demand based operation incurs may be unacceptable. If bandwidth and energy resources permit, proactive operations are desirable in these contexts.
5. Without some form of network-level or link-level security, a MANET routing protocol is vulnerable to many forms of attack. This problem has been discussed earlier in this paper.
6. “Sleep” period operation: As a result of energy conservation or some other need to be inactive, nodes of a MANET may stop transmitting and/or receiving for arbitrary time periods. A routing protocol should be able accommodate such sleep periods without overly adverse consequences. This property may require close coupling with the link-layer protocol thought a standardized interface.
7. Unidirectional link support: Bidirectional links are typically assumed in the design of routing algorithms, and many algorithms are incapable of functioning properly over unidirectional links. In situation where a pair of unidirectional links (in opposite directions) form the only bidirectional connection between two ad hoc regions, the ability to make use of them is valuable.

The following is a list of quantitative metrics that can be used to assess the performance of any routing protocol. This is important if at a later moment I shall compare the results in my Diploma Thesis with other approaches. I will not discuss these metrics in detail.

1. End-to-end data throughput and delay: Statically measure of data routing performance.
2. Route acquisition time: Time required to establish route when requested
3. Percentage out-of-order delivery: An external measure of connectionless routing performance
4. Efficiency: Internal measure of policy performance effectiveness.

As proven in this section there are many problems that remain unsolved in MANET networks and many interesting research topics remain to be explored.

2.4 Protocol techniques for MANET

The development of a protocol for mobile ad hoc networks is very complicated, and it is important that the protocol is able to adjust and adapt to topology changes without imposing too high demands on available resources of the network. Wireless links have several limitations, not only in terms of bandwidth. Portable devices have limited capacity (battery power, available memory, and computing power) that further complicates the protocol design. Several protocols for ad hoc networks have been developed. The protocols can perform well under certain situations that they are designed to solve, but they fail completely in other situations that can occur in the network. There has not yet been developed a protocol that can handle the complete complexity of an ad hoc network. Ad hoc routing protocols can roughly be subdivided into five categories.

1. Flooding;
2. Proactive routing protocols;
3. Reactive routing protocols;
4. Hybrid routing protocols;
5. Protocols using knowledge of physical position of the devices that makes up the ad hoc network.

The following section gives a brief description of each category.

2.4.1 Flooding

Flooding is the simplest routing solution. When a node wants to send packets from one node to another, it will simply broadcast the packets to its neighbours. The neighbouring node that receives the packets will rebroadcast them to all its neighbours and the process continues throughout the network. The destination node will eventually receive the packets, as will the whole network. Flooding produces a lot of overhead in the network and consumes a large part of the limited available resources.

The advantage is that it is easy to implement, requires minimal computing power from ad hoc devices, and needs no knowledge of the network's topology. This type of routing solution can work well in small networks and networks where many of the nodes can have use of the data received. It should be avoided in large ad hoc networks.

2.4.2 Proactive routing protocols

In proactive routing protocols (also called table-driven protocols), all nodes know the topology of the network. This is done by use of adjusted versions of the classical distance vector and link state algorithms. Because all nodes in the network must be updated on changes, control packages must be sent to their neighbours regularly to inform about possible topology changes.

The disadvantages of this approach are that the control packages consume a large amount of bandwidth, and every node must store large amounts of information, as they must have knowledge of the complete network topology. The memory requirements and number of control packets will increase as the number of nodes in the network increases. The advantage of these protocols is that they are able to quickly find a new route in the network.

Proactive protocols are best in networks where the nodes are close together and where only few hops are necessary to connect two nodes. Optimized link state routing (OLSR) is an example of a proactive routing protocol for ad hoc networks that is discussed in section 2.6.2 in this paper.

2.4.3 Reactive routing protocol

In reactive routing protocols (also called source-driven protocols), the nodes only store the routing information that is necessary at the moment. This is done by using the query/reply principle. If the node that transmits the data does not know the route to the destination node, then the transmitting node must try to discover a route for the data flow. This is done by flooding a route-request (RREQ) into the network. How the nodes react on the (RREQ) and calculate a route depends on which reacting routing protocol is in use. The flooding principle is only used to find a route, hence the data packages are sent more efficiently.

The advantage of a reactive protocol is that only the actively participating nodes in the network are kept in memory, and only these nodes have an influence on the network. Compared to the proactive protocols, the reactive protocols will get a delay when establishing a new route between nodes because of the establishment of a route with RREQ flooding. If the devices in the network are very mobile, the control overhead can become as large as in the case of proactive protocols.

Reactive protocols are a good choice when a network becomes large and has a large hop-diameter. Ad hoc on-demand distance vector routing (AODV) is an example of a reactive protocol for ad hoc networks and is discussed in later sections 2.5.1 of this paper.

2.4.4 Hybrid routing protocols

Hybrid routing protocols were designed to exploit the benefits of both reactive and proactive routing protocols. A hybrid routing protocol has the option of using proactively determined routes to nodes that are close by, and reactively compute routes to nodes further away. There are several methods of achieving a hybrid protocol, but this paper will not explore this further.

2.4.5 Protocols that make use of the known physical location

These protocols make use of a system that can determine the exact location of the devices (nodes). This can be done with help of Global Position System. When the position of each node is known, the broadcasting of RREQ can be accomplished much more efficiently. It is also possible to flood data packets towards the destination node.

2.5 Two common protocol solutions for MANET

Below is an overview of AODV and OLSR, which are two of the most used protocols in MANET.

2.5.1 AODV

The AODV protocol is a straightforward reactive routing protocol. In AODV, on receiving a query, the transit nodes “learn” the path to the source (backward learning) and enter the route in the forwarding table. When the destination node eventually receives the query it also has received the path to the source node. This path can be used to make a response which permits the establishment of a full duplex path. As explained earlier in this paper the network is flooded when a new path is to be found. To reduce overhead during this phase, AODV has altered the classic flooding technique slightly. The difference is that AODV drops the query packet if it encounters a node which already has a route to the destination. When the path has been established the link will only be maintained as long as the source uses it. If a link failure occurs it will be reported through the intermediate nodes to the source node. This will result in another query-response procedure in order to find a new route and establish a new link.

Figure 3 illustrates an AODV route lookup session. Node A wishes to initiate traffic to node J for which it has no route. Node A broadcasts a RREQ which is flooded to all nodes in the network. When this request is forwarded to J from H, J generates a RREP. This RREP is then unicasted back to A using the cached entries in nodes H, G and D.

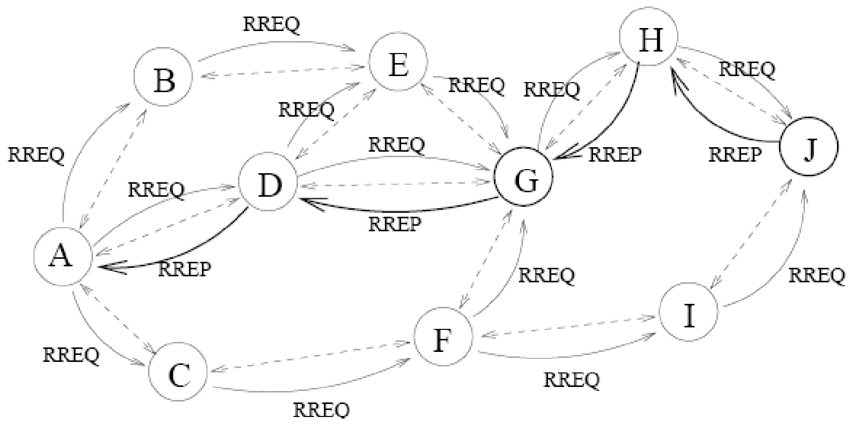


Figure 3: AODV route lookup session as shown in [15]

AODV defines three types of control messages for route maintenance.

- RREQ - A *route request* message is transmitted by a node requiring a route to a node. As an optimization AODV uses an *expanding ring* technique when flooding these messages. Every RREQ carries a *time to live* (TTL) value that states for how many hops this message should be forwarded. This value is set to a predefined value at the first transmission and increased at retransmissions. Retransmissions occur if no replies are received. Data packets waiting to be transmitted (i.e. the packets that initiated the RREQ) *should* be buffered locally and transmitted by a FIFO principal when a route is set.

- RREP - A *route reply* message is unicasted back to the originator of a RREQ if the receiver is either the node using the requested address, or it has a valid route to the requested address. The reason one can unicast the message back, is that every route forwarding a RREQ caches a route back to the originator.
- RERR - Nodes monitor the link status of next hops in active routes. When a link breakage in an active route is detected, a RERR message is used to notify other nodes of the loss of the link. In order to enable this reporting mechanism, each node keeps a "precursor list", containing the IP address for each its neighbours that are likely to use it as a next hop towards each destination.

Figure 3 illustrates an AODV route lookup session. Node A wishes to initiate traffic to node J for which it has no route. Node A broadcasts a RREQ which is flooded to all nodes in the network. When this request is forwarded to J from H, J generates a RREP. This RREP is then unicasted back to A using the cached entries in nodes H, G and D.

The advantage of AODV is that it creates no extra traffic for communication along existing links. Also, distance vector routing is simple, and doesn't require much memory or calculation. However AODV requires more time to establish a connection, and the initial communication to establish a route is heavier than some other approaches.

2.5.2 OLSR

The Optimized Link State Routing (OLSR) is described in RFC3626 [16]. OLSR is a proactive, link-state routing protocol, employing periodic message exchanges to update topological information in each node in the ad hoc network. That topological information is flooded to all nodes in the network, providing immediately available routes, where the control traffic overhead is constantly low. Below is a description of the parts of OLSR of most interest for my thesis.

OLSR defines three basic types of control messages.

- HELLO messages are transmitted to all neighbours. These messages are used for neighbour sensing and MPR calculation. The HELLO message is shown in table[1]
- Topology Control (TC) messages are the link state signalling done by OLSR. This messaging is optimized in several ways using MPRs. The TC message is shown in table [2]
- Multiple Interface Declaration (MID) messages are transmitted by nodes running OLSR on more than one interface. These messages list all IP addresses used by a node. This message will not be discussed in detail in this paper.

There are three basic elements in OLSR that build up the routing protocol; a mechanism for neighbour sensing, a mechanism for efficient flooding of control traffic, and a specification of how to select and diffuse sufficient topological information in the network in order to provide the optimal route. The elements are described below.

Neighbour sensing:
In OLSR this is done by every node in the network. The mechanism periodically emits a hello message to its neighbouring nodes. This hello message contains the emitting node's own address, a list of known neighbouring nodes and the status of the link to each neighbour. The regularly received hello messages allow each node to detect changes in its neighbourhood and two-hop neighbourhood. The topology information in the hello messages is maintained in each node for routing purposes. Figure 3 shows a classic hello message as given in [16]. The hello message is of great importance in this thesis, because the available bandwidth estimation must be included into the message to flood it through the two-hop neighbourhood.

```
        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
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |          Reserved             |     Htime     |  Willingness  |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |   Link Code   |   Reserved    |       Link Message Size       |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                  Neighbour Interface Address                  |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                  Neighbour Interface Address                  |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       :                             .  .  .                           :
       :                                                               :
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |   Link Code   |   Reserved    |       Link Message Size       |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                  Neighbour Interface Address                  |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                  Neighbour Interface Address                  |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       :                                                               :
       :                                       :
   (etc.)
```

Table 1: A hello message

Below is a description of each field ([16]).

Reserved:
This field must be set to "0000000000000" to be in compliance with reference [].

Htime:
This field specifies the HELLO emission interval used by the node on this particular interface, i.e., the time before the transmission of the next HELLO. The HELLO emission interval is represented by its mantissa (four highest bits of Htime field) and by its exponent (four lowest bits of Htime field). In other words:

HELLO emission interval=C*(1+a/16)*2^b [in seconds]

where a is the integer represented by the four highest bits of Htime field and b the integer represented by the four lowest bits of Htime field. The proposed value of the scaling factor C is specified in section 18 of reference [16].

Willingness:
This field specifies the willingness of a node to carry and forward traffic for other nodes.
A node with willingness set to WILL_NEVER must never be selected as MPR by any node.
A node with willingness set to WILL_ALWAYS must always be selected as MPR. By default, a node should advertise a willingness of WILL_DEFAULT. Implementation of willingness often set to an integer between 0 and 7. Willingness is set to 0 for WILL_NEVER and 7 for WILL_ALWAYS. WILL_DEAFAULT is set by the configure file.

Link code:
This field specifies information about the link between the interface of the sender and the following list of neighbour interfaces. It also specifies information about the status of the neighbour. Link codes, not known by a node, are silently discarded.

Link message size:
The size of the link message, counted in bytes and measured from the beginning of the "Link Code" field and until the next "Link Code" field (or - if there are no more link types – the end of the message).

Neighbour Interface Address:
The address of an interface of a neighbour node.

<u>Efficient flooding of control traffic:</u>
The hello message only sends topology information to its neighbours. The OLSR network can become large and needs a way to flood the whole network with the network topology information as its changes. OLSR handles this problem with a control message, which is flooded into the entire network in an “efficient” (each node only receives the message once) way. OLSR has optimized a simple flooding strategy, based on the idea that all nodes relay (forward) the message if it is the first time they receive the message. The optimised strategy uses multipoint relays (MPRs) to flood the networks. The idea of MPR is to minimize the flooding broadcast packets in the network by reducing duplicate retransmission in the same region. Each node in the network independently selects a set of MPRs. The MPRs set is chosen by each node selects its MPR set from among its 1-hop symmetric neighbours. This set is selected such that it covers (in terms of radio range) all symmetric strict 2-hop nodes. The information required to perform this calculation is acquired through the periodic exchange of HELLO messages. The MPR set is re-calculated when a change in 1-hop or 2-hop neighbours sets with bi-directional link is detected. This is done in the same way as in the original OLSR implementation.

Only the MPRs nodes have the responsibility to relay the control messages sent from the original node. This is illustrated in figure 4 which shows the optimized OLSR technique where the center node is the start point of the flooding. Careful selection of MPRs (the filled nodes) may greatly reduce duplicate retransmissions.

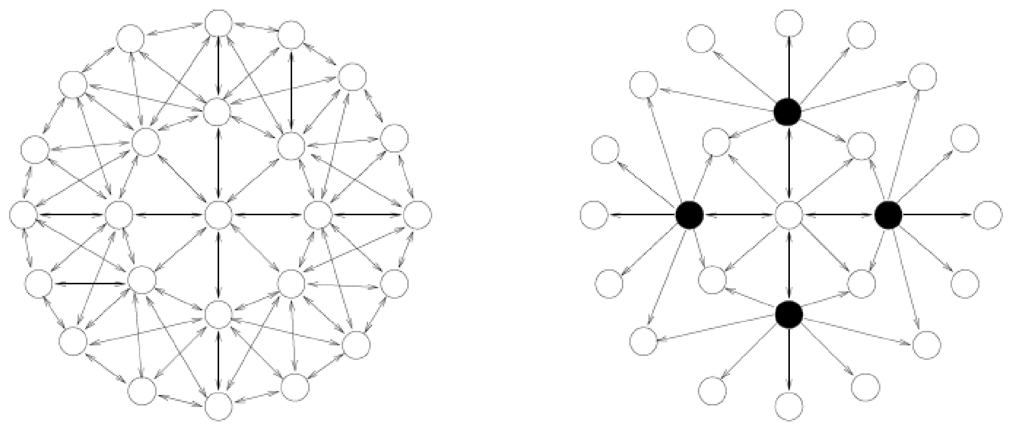


Figure 4:
To the left: Flooding a packet in a wireless multihop.
To the right: Flooding a packet in a wireless multi-hop network from the centre node using MPRs (black). The arrows show *all* transmissions.

Optimal route

The control message sent by the optimized OLSR flooding technique is called a topology control message (TC message) and is sent periodically through the network. These messages contain the address of the node generating the TC message, as well as the addresses of all the messages the MPR selector of that node. The TC message announces reachability to all its MPR nodes, since all nodes in the network have selected a set of MPRs nodes, reachability will be announced through the network.

The result is that all nodes will receive a partial topology graph of the network, made up by all reachable nodes and the set of links between a node and its MPR selectors. With this information available it is possible to use a shortest path algorithm to compute the optimal path from a node to any reachable node in the network.

Below, *table 2* shows a classic TC message as proposed in reference [16] RFC-3626 and below is the description of each field. The TC message is also of great importance for this thesis, because it must be changed to include an available bandwidth field to flood this through the whole network.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              ANSN             |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Advertised Neighbour Main Address               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|               Advertised Neighbour Main Address               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Table 2: A TC-message

Advertised Neighbour Sequence Number (ANSN):
A sequence number is associated with the advertised neighbour set. Every time a node detects a change in its advertised neighbour set, it increments this sequence number ("Wraparound" is handled as described in section 19 in reference [16]). This number is sent in this ANSN field of the TC message to keep track of the most recent information. When a node receives a TC message, it can decide on the basis of this Advertised Neighbour Sequence Number, whether or not the received information about the advertised neighbours of the originator node is more recent than what it already has.

Advertised Neighbour Main Address:
This field contains the main address of a neighbour node. All main addresses of the advertised neighbours of the Originator node are put in the TC message. If the maximum allowed message size (as imposed by the network) is reached while there are still advertised neighbour addresses which have not been inserted into the TC message, more TC messages will be generated until the entire advertised neighbour set has been sent. Extra main addresses of neighbour nodes may be included, if redundancy is desired.

Reserved:
This field is reserved, and MUST be set to "0000000000000000 for compliance with reference [16].

The routing table calculation is an important part of OLSR. Each node maintains a routing table witch makes it possible to route packets to other nodes in the network. All nodes that receive a TC-message will parse and store some of the connected pairs of a form [last-hop, node] where “nodes” are the address found in the TC message list. The database that represents the routing table is built by tracking the connecting pairs in a descending order.

The procedure uses [last-hop, destination] pairs in the topology table to get a complete route from source to destination. To get the optimal path, the forwarding nodes only pick the connected pairs on the minimal path. The route entries in the routing table consist of destination address, next-hop address, and estimation distance to destination. All known route destinations are recorded in the table. The links that are broken or partially known are not entered in the table. If any tables (neighbour table and topology table) are changed because of changes in the topology, the routing table must be recalculated to update the route information about each known destination in the network. It is possible to alter this routing calculation by choosing other routing techniques as QoS –aware routing, as discussed in section 3 of this paper.

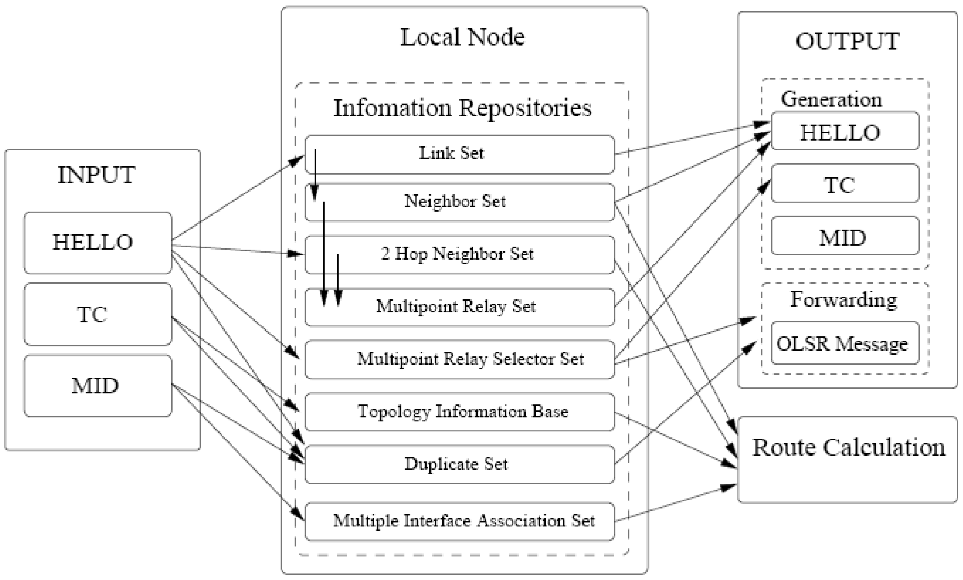


Figure 5: Overview of information repositories in OLSR, from reference [15]

Figure 5 displays an overview of the information repositories in OLSR and their relations to message processing, message generation and route calculation. The following text quoted form [15], to keep the exactness of the of the text definition.

"Received HELLO messages trigger updates in the link set which again triggers updates in the neighbour set, which then again triggers recalculation of the MPR set. The 2 hop neighbour set is also updated based on received HELLO messages again triggering a recalculation of the MPR set. Finally the MPR selector set is updated according to information received in HELLO messages. Received TC messages triggers updates in the topology set while the MID set is updated upon receiving MID messages. All received messages will also be registered in the duplicate set if not already registered. When generating HELLO messages, the link set, neighbour set and MPR set is queried. When generating TC messages, the MPR selector set is queried. When forwarding control traffic, the MPR selector set and the duplicate set is used. Finally, route calculation is based on information retrieved from the neighbour set, the 2 hop neighbour set, the TC set(the Topology Information Base) and the MID set."

Most of the information stored in each set is very similar to what the messages contain. Of this reason I shoes not to specify each set of the OLSR protocol. In section 4 I show a proposed extension implementation to OLSR and this section include some of the sets that are important to this paper and how they are altered.

The advantage of this approach is that connections are made quickly. The disadvantage is that communication to discover networks occurs continuously. Because the program is fairly large and complex, continuous calculation and memory burdens may be too heavy for small computers.

2.6 Basic available bandwidth algorithms

There are several ways of estimating available bandwidth in an ad hoc network. In this paper I will consider the *"listen" bandwidth estimation* and *"hello" bandwidth elimination.* As bandwidth estimation is the key to support QoS it is important that these estimations are well thought out and proven to work. In reference [11] the conclusion is that the *"hello" bandwidth estimation* is the better option but only barely. Still the conclusion of this thesis when it comes to picking an available bandwidth estimation algorithm is to use the listen bandwidth estimation algorithm. The reason for this is that it is used by many QoS protocols and in this case is simpler to implement in the existing protocol.

2.6.1 Listen bandwidth estimation

Reference [11] proposes a distributed bandwidth reservation mechanism where every node in the network must measure the link bandwidth. The available bandwidth is calculated by the nodes by using the network allocation vector (NAV) to find the ratio between free time and overall time. This is a very straightforward technique, but the problem is that in practice the IEEE 802.11 equipment does not grant access to the NAV. The solution to this problem is to use the fact that every time a packet is sent, it will keep the sender node busy for a period of time. With calculations based on various IEEE 802.11 standards, it is possible to calculate how long the sender or receiver was busy for each transmitted frame.

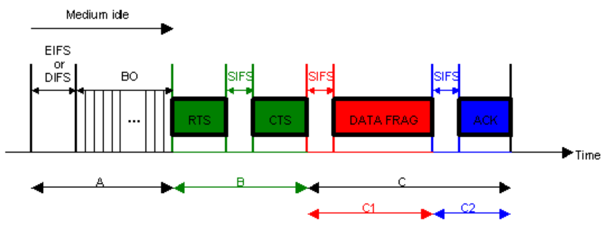


Figure 6: IEEE frame exchange sequence

The general IEEE 802.11 frame exchange sequence is detailed in *Figure 6*. Interval A only affects busy transmit time and the medium will be sensed idle. If DIFS (Distributed coordination function interframe space), EIFS (Extended interframe space) or BO (back off) is interrupted, interval A will be restarted. Interval B exists only for unicast frames larger than a certain threshold. The C2 interval, only exists in unicast frames. If based on the standard (a, b or g), physical layer, basic service set characteristics, modulation rate and frame characteristics, the duration of each frame exchange sequence is added up to get the busy time. The total busy time is calculated to be the sum of busy received and busy transmitted time and the percentage of busy time is averaged over an interval of configurable duration. For a deeper understanding of the IEEE 802.11 protocol I show to article [2].

The listen bandwidth estimation technique has three imprecision's in this algorithm. If interval A is interrupted, a new DIFS or (EIFS) will be sent when the interval is resumed. The interval A is invisible to the driver code and therefore the repeated DIFS (or EIFS) cannot be considered in the busy time calculation. The second source of imprecision occurs when there are no frames sent, but the medium is still considered idle. The reason for this situation is that the energy level is below a certain threshold, which results in that no physical sensing is visible in the driver. The last source affects received frames. For received frames, the receiver is aware of presence of retries. But it only knows there was a number of retries but does not know exactly how many. However, some of the retries may be decoded as received frames, in which case the retriever's busy time will take into account these retries. For more detailed information about the calculations read reference [11] and [12].

There is one more backside of the listening technique. When a route breaks, the host cannot release the bandwidth immediately. This is because it does not know how much bandwidth each node in the broken route consumes. It has a great influence on the accuracy of bandwidth estimation when a route is broken

2.6.2 Hello bandwidth estimation

The hello band-width estimation is based on that the sender's current bandwidth usage and the senders one hop neighbours current bandwidth usage are piggybacked on the standard hello message. Each node bases its calculation on the information from the hello message and frequency reuse pattern. When we study the underlying IEEE 802.11 MAC the nodes are allowed to access the wireless channel when the media is free. Furthermore, the normal interference distance is twice the transmitting range. With the help of the frequency reuse pattern we can simplify the bandwidth calculation problem, and each node can approximate its residual bandwidth information based on information from hosts within two-hops (the interference range).

When using an OLSR protocol we can simply change the hello message to include the consumed bandwidth. Now the nodes are able to get the bandwidth consumed from its two-hop neighbours. The ability to get the bandwidth upper bound within the two-hop circle varies with the topology and traffic status. The raw channel bandwidth is the soft upper bandwidth of the total bandwidth. This soft upper bandwidth is accurate enough to use in the estimation to approximate the bandwidth usage.

Once a node knows the bandwidth consumption of its one-hop and two-hop neighbours, the available bandwidth estimation becomes simple. The residual bandwidth is calculated by taking the raw channel bandwidth minus the overall consumed bandwidth, divided by a weight factor. The weight factor in necessary due to IEE 802.11 MAC's nature and some overhead required by the routing protocol in the ready-to-send (RTS), clear-to-send (CTS), and acknowledgment (ACK) packets consume bandwidth. The back off scheme cannot fully use the entire bandwidth, and packets can collide, resulting in packets retransmissions. Furthermore, the routing protocol needs some overhead to maintain or discover the route.

3 Quality of Service

QoS is usually defined as a set of service requirements that need to be met by the network while transporting a packet stream from a source to a destination. The network is expected to guarantee a set of measured prespecified service attributes to users in terms of end-to-end performance, such as delay, bandwidth, probability of packet loss, and delay variance (jitter). Power consumption and service coverage area are two other QoS attributes that are more specific to MANET. QoS metrics can be concave or additive. Bandwidth is concave in the sense that end-to-end bandwidth is the minimum of all the links along the path. Delay and delay jitter are additive, so that the end-to end delay (jitter) is the accumulation of all delays (jitter) of the links along the path.

QoS metrics could be defined in terms of the parameters or a set of parameters. In this case only the end-to-end minimum available bandwidth will be used as a parameter. Differentiated services (diff-serv) and integrated services (int-serv) are two classical ways of achieving QoS in internett today and is the basis of QoS methods. Some of these theories integrated into the SWAN protocol (see [10]) and is mentioned in the QOLSR specifications (see [25]).

3.1 General QoS techniques in use

Both of the services mentioned below are commonly implemented into routers of wired networks to improve the QoS and there for of interest. Especially is the resource reservation technique of great influence for several MANET solutions. The SWAN protocol mentioned in section 3.3 has similar solutions.

3.1.1 Integrated services (int-serv)

Int-serv identifies three main categories of service concerning the integration: the traditionally best-effort services, real-time services and controlled link-sharing services.

- Best-effort services are those we currently experience on the internet. They are characterized by absence of any QoS specifications. The network provides the quality that it actually can contribute. Examples of best-effort traffic are FTP, mail and FAX.
- Real-time services are services that have very critical requirements in terms of end-to-end delay, probability of loss and bandwidth. They usually require a guarantee from the network.
- Controlled link-sharing is a service that might be requested by network operators when they wish to share a specific link among a number of traffic classes. Network operators may set some sharing policies on the link utilization among these traffic classes; specifically some percentage of bandwidth may be assigned to each traffic class.

The int-serv QoS solution uses the resource reservation protocol (RSVP) to flood messages through the network, and reserves resources for every flow at every router hop from source to destination. Every router along the path must maintain soft states information. Int-serv requires a lot of signalling, therefore the overhead is a concern when the network scale increases.

3.1.2 Differentiated services (diff-serv)

Diff-serv is a light weight alternative to int-serv. The concept of diff-serv is to differentiate the user data from control and management information. A field in the header of the Internet Protocol (IP) Data Unit was designed for these purposes: the Type-of-Service (TOS) field. The octet dedicated to this field indicates the specific treatment that the packet expects to receive from the network.

The TOS bits are divided up as follows:

- 3 bits dedicated to priority of the datagram
- 3 bits define the type of service (TOS) which correspond to QoS expected by the IP datagram
- 2 bits are reserved for future use.

Diff-serv does not maintain the state of each and every flow as Int-serv does, but rather discriminates the packets according to their priority. The edge routers classify the traffic type, while the individual routers that forward the data will decide the fate of the packets according to local policies of the packet types. Diff-serv is easier to maintain, more scaleable and has less signalling than int-serv.

3.2 Problems related to QoS in MANET

Because of the resource limitations and dynamic nature of MANET networks, it is especially important to be able to provide QoS. However the characteristics of these networks make QoS support a very complex process. QoS support in MANET includes issues at the application layer, transport layer, network layer, MAC layer and physical layer of the network infrastructure. In Mobile multihop wireless networks there are several unique issues and difficulties that do not apply to the traditionally wired internet infrastructure. Below I have listed the most important issues.

3.2.1 Unpredictable link properties

Wireless media is very unpredictable and packet collisions are an unavoidable consequence of wireless networks. Signal propagation faces difficulties such as fading, interference, and multipath cancellation. These properties of the wireless network make measurements such as bandwidth and delay of the link unpredictable.

3.2.2 Node mobility

Movement of nodes in the ad hoc network creates a dynamic network topology. Links will be dynamically formed when two nodes moves into transmission range of each other and are torn down when they move out of transmission range. Node mobility makes measurements in the network even harder and measurements as bandwidth is essential for QoS.

3.2.3 Limited battery life

There is limited power of the devices that establish the nodes in the ad hoc network due to limited battery life time. QoS should consider residual battery power and rate of battery consumption corresponding to resource utilization. The technique used in QoS provisioning should be power aware and power efficient.

3.2.4 Hidden and exposed terminal problem

In a MAC layer with traditionally carrier sense multiple access (CSMA) protocol, multihop packet relaying introduces the “hidden terminal” problems. The hidden terminal problem happens when signal of two nodes, say A and B, that are out of reach of each other’s transmission range, collide at a common receiver, say node C. With the same nodal configuration, an exposed terminal problem will result from a scenario where node B attempts to transmit data (to someone other than A or C) while node C is transmitting to node A. In such a case, node B is exposed to the transmission range of node C and thus defers its transmission even though it would not interfere with the reception at node A. Carrier sense multiple access with collision avoidance (CSMA/CA) reduces the effect of hidden terminal problem, but there is no solution for the exposed terminal problem today. Hidden and exposed terminal problem is not only a QoS problem, but is a recurring problem through the aspect of the MANET network.

3.2.5 Route maintenance

The dynamic nature of the network topology and the changing behaviour of the communication medium make the precise maintenance of network state information very difficult. Because of this, the routing algorithms in MANET must operate on imprecise information. Since the nodes can join and leave the ad hoc network environment as they please, the established routing path may be broken at any time even during the process of data transfer. Thus, the need arises of routing paths with minimal overhead and delay. Since the QoS-aware routing would require reservation of resources at the routers (nodes), the problem of a heavily changing topology network might become cumbersome, as reservation maintenance with updates along the routing path must be done.

3.2.6 Security

Without adequate security, unauthorized access and usage may violate QoS negotiations. The nature of broadcasts in wireless networks potentially results in more security exposure. The physical medium of communication is inherently insecure, so it is important to design aware routing algorithms for MANET.

Because of the difficult properties of mobile wireless networks there has been a suggestion of using soft QoS. The definition of Soft QoS is that after a connection setup, there may exist transient periods of time when QoS specifications is not honoured. However we can quantify the level of QoS satisfaction by the fraction of total disruption time over the total connection time. This ratio should not be higher than a threshold. SWAN uses this technique and is discussed later in this paper. QoS adaptation can be done in several layers. The physical layer should take care of changes in transmission quality, for example by adaptively increasing or decreasing the transmission power. Similarly, the link layer should react to the changes in link error rate, including the use of automatic repeat request (ARQ). A more sophisticated technique involves an adaptive error correction mechanism that increases or decreases the amount of error correction coding in response to changes in transmission quality of desired QoS. As the link layer takes care of the variable bit error rate, the main effect observed by network layer will be a change in effective throughput (bandwidth) and delay. Again the SWAN protocol is a good example of these statements.

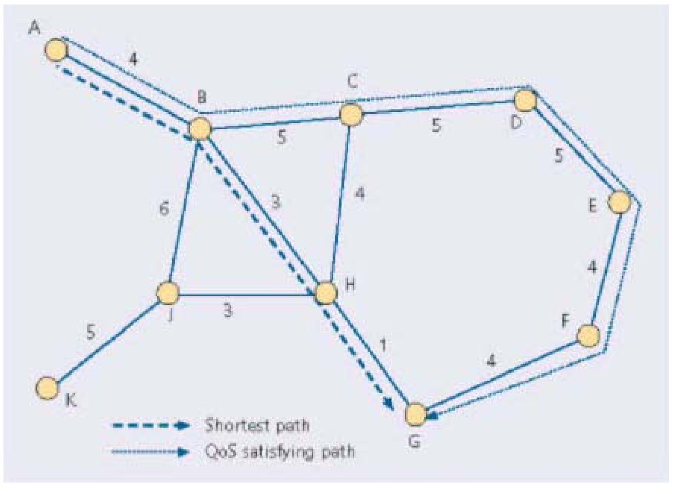


Figure 7: QoS-aware routing in ad hoc network

Constraint-based routing protocols use metrics other than the shortest path to find a suitable and feasible route. Associatively-based routing (ABR) and signal stability routing (SSR) take into account the node's signal strength and location stability so that the path chosen is more likely to be long-lived. Dynamic load-aware routing (DLAR) consider the load of intermediate node as the primary route selection metrics. In *figure 7* is an example of how a

QoS-aware routing chooses its route in an ad hoc network. To limit the amount of flooding (routing) messages we should integrate QoS in flooding-based route discovery as explained later.

3.3 SWAN (Stateless Wireless Ad Hoc Networks)

Even though the final implementation of my routing protocol did not include all the functionalities of SWAN, SWAN has been an important starting point for research for my thesis. It shows some functionality that could be included into the implementations at a later time, and is therefore of great interest for my thesis.

SWAN is a stateless network model which uses distributed control algorithms to deliver differentiation in mobile wireless ad hoc network in a simple, scalable and robust manner. The architecture of SWAN is designed to handle real-time UDP traffic, and best effort UDP and TCP traffic without management of per-flow state information. SWAN uses a source-based admission of real-time traffic. SWAN uses sender-based admission control for real-time UDP traffic and explicit congestion notification (ECN) to dynamically regulate admitted real-time sessions in the face of network dynamics brought on by mobility or traffic overload conditions.

The basic idea of SWAN is that the network rate is performed locally at every node in the network. The rate control is designed to restrict best effort traffic to give the necessary bandwidth to support real time traffic. The total traffic of best effort and real-time traffic is kept under a certain “threshold rate”, which is calculated to be under the saturation level of the wireless channel with a good margin. Below is a description of the basic mechanisms to support rate regulation in SWAN

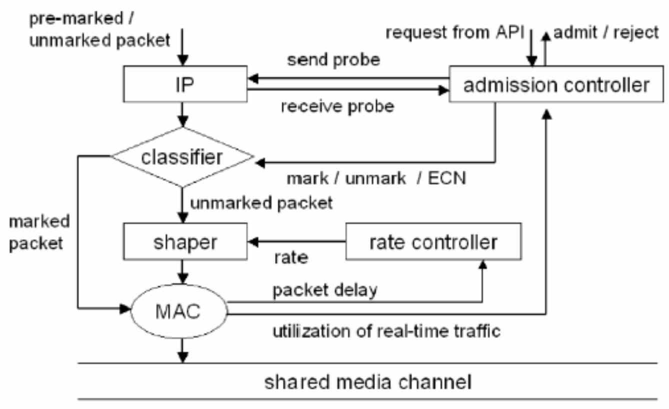


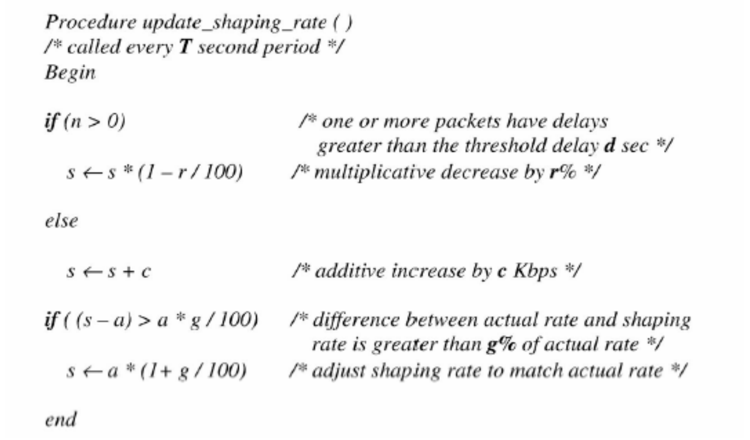
Figure 8: SWAN architecture as shown in [10]

.

SWAN has several important mechanisms used to support rate regulation of best effort, as illustrated in *figure 7*. The classifier and shaper operate between the IP and best effort MAC layers. The classifier’s task is to filter out the best effort packets which will be processed in the shaper. The shaper’s task is to delay the best effort packets in conformance with the rate calculated by the rate controller. The decision of admitting a real-time session is taken by the admission controller. Below is a more detailed discussion of the rate controller and admission controller.

3.3.1 Local Rate control of best effort traffic

SWAN uses an AIMD rate control algorithm based on feedback from the MAC. The information that is of interest from the MAC is the packet delay measured by the MAC layer. The packet delay is the time it takes to send the packet between the transmitter and next-hop receiver including the total deferred time (including possible collision resolution) plus the time to fully acknowledge the packet. This is measured at the source node by subtracting the time that a packet is passed to the MAC layer (from upper layer) from the time an ACK packet is received from the next-hop receiver. The AIMD algorithm is shown in *figure 8*.



```
Procedure update_shaping_rate ( )
/* called every T second period */
Begin

if (n > 0)                  /* one or more packets have delays
                               greater than the threshold delay d sec */
   s ← s * (1 – r / 100)    /* multiplicative decrease by r% */

else

   s ← s + c                /* additive increase by c Kbps */

if ( (s – a) > a * g / 100)  /* difference between actual rate and shaping
                               rate is greater than g% of actual rate */
   s ← a * (1+ g / 100)     /* adjust shaping rate to match actual rate */

end
```

Figure 9: AIMD algorithm as shown in [10]

The general procedure is that each mobile host increases its transmission rate (c kbs) until one or more packets exceed the threshold for delay (discovered by the rate controller). If the threshold is exceeded, the rate controller reduces the rate. The threshold delay is based on the real-time delay requirement of applications in wireless networks

3.3.2 Source-based admission of real-time traffic

A mobile host is able to listen to all packets sent within their radio transmission range. In each node in the ad hoc network, the admission controller measures the resource availability by calculating a weighted moving average of the real-time traffic. The approach that SWAN uses is to admit real-time traffic up to a conservative threshold rate. The available bandwidth of the shared media is calculated to be the difference between the conservative threshold and the current measured real-time traffic. Any remaining unutilized bandwidth may be used by best effort traffic. The Source based admission technique sends a probe to the receiver. The probe is a UDP control packet that stores the “bottleneck bandwidth” that represents the end-to-end bandwidth availability. The receiver returns the “bottleneck bandwidth” result in another probe back to the source. The admission control tests if the sessions required bandwidth exceeds the end-to-end bandwidth availability and then accepts or declines the session.

3.3.3 Dynamic regulation of real-time traffic

Because the real-time flows admitted along a certain path can be dynamically rerouted, and because nodes are unaware of flow rerouting due to mobility, resource conflicts can arise and persist. False admission is the result of multiple source nodes simultaneously initiating admission control at the same instance and sharing common nodes between source destination pairs.

To solve this problem SWAN uses dynamic regulation of real-time traffic. This is done with help of an ECN-based regulation of real-time session. ECN-based regulation is conducted by each node detecting violation (congestion/overload conditions) using the periodic traffic measurement (as discussed earlier) When a node detects a violation, it starts marking ECN bits in the IP header of the real-time packets. The destination node monitors the ECN bits and notifies the source using a regulate message. When the source node receives a regulate message, it initiates reestablishment of its real-time flow, with the same process as if it had been a new session. If a node is congested it will mark all packets with CE (Congestion experienced). Then all sessions traversing this node would be forced to re-establish their real-time service at the same instance. Instead of re-establishing all sessions at the same instance, SWAN proposes that all sessions wait a random time before initiating a reestablishment. This is important if the sessions are not to experience false admission.

3.4 OLSRD QoS-aware based on link quality extension

OLSRD has been of paramount importance for this thesis, as this shows a working QoS aware routing protocol over an OLSR platform. The implementation of my protocol has proven to be easier because of the implementation example of this extension. Most of the text below is edited from reference [17].

OLSRD has been of paramount importance for this thesis, as this shows a working QoS aware routing protocol over an OLSR platform. The implementation of my protocol has proven to be easier because of the implementation example of this extension

Release 0.4.8 of olsrd offers an experimental implementation of an ETX-like metric. When calculating a routing table for us, pure RFC-compliant OLSR simply minimizes the number of hops between ourselves and the other nodes in the MANET, even if this means that a route via a single very bad link will be preferred to a route via two excellent links, although the latter would probably have been the better choice.

The proposal specifies that the protocol must be able to distinguish between good links and bad links. This is possible by calculating the packet loss from HELLO messages received from the neighbour of a node.

If 3 out of 10 packets that are sent from a neighbour node to a "our" node are lost, then packet loss will be 30 %, and chance of a successful packet transmission will be 70%. This chance of success on link between neighbour and us is called "Link quality" for a successful packet transmission from this neighbour to ourselves is 7/10 = 0.7 = 70%. This probability is what we call the *Link Quality*. So the Link Quality says how good a given link between a neighbour and us is in the direction from the neighbour to us. It does so by saying how likely it is that a packet that we send is successfully received by our neighbour.

However, it is also important to know the quality of the link in the opposite direction, i.e. how many of the packets that we send out are received by each of our neighbours. This is our neighbour's idea of the link quality. This quality is defined as "neighbours Link Quality". This might seem a bit confusing at that is why I include a quote from [17].

"They represent the probability that a packet that our neighbour sends actually makes it to us (Link Quality) and that a packet that we send actually makes it to our neighbour (Neighbour Link Quality")

The probability of a successful packet roundtrip is an important metric in this proposal. This is the probability that we successfully send a packet to our neighbour, and, upon receiving it, our neighbour successfully replies with a response packet. This round trip is calculated by multiplication, NLQ x LQ where NLQ is Neighbour Link Quality and LQ is Link Quality

It is no possible to answer the question of how many transmission attempts it will typically take to get a packet from us to a neighbour or from the neighbour to us. It is 1 / (NLQ x LQ) this value is called Expected Transmission Count (ETX). Note that this number is valid for both directions of the link, as in both cases we have to look at the probability for a successful packet round trip. ETX is a QoS metric and routing based on ETX is handled in a similar way as proposed in section 4.

When the proposal calculate total ETX in a route it adds all the ETX values between the source and the destination nodes and calculates the best routes from these calculations.

4 Implementations for QoS into OLSRD

QOLSR is a proactive QoS routing protocol for mobile ad hoc networks. The protocol inherits the stability of a link state algorithm and has the advantage of having optimal routes, in terms of multiple-metrics, immediately available when needed, due to its proactive nature. QOLSR provides end-to-end QoS requirements and minimizes flooding of control traffic by using only selected nodes, called MPRs, to retransmit control messages. This technique significantly reduces the number of retransmissions required to flood a message to all nodes in the network. QOLSR requires only partial link state and their QoS information to be flooded in order to provide optimal routes under QoS constraints as those in whole network topology. All nodes, selected as QMPRs, must declare the link QoS information to their QMPR selectors, using TC messages.

4.1 Approach

My approach to the implementation of quality of service aware OLSR protocol is based on available bandwidth calculations. First of all there are many approaches on how to handle the many problems in the MANETs, and several of the solutions have been evaluated in some protocols, but not in others. That is why I stress the importance to have a broad view and look at several solutions to the different issues at hand, and also because these solutions mention protocols which should have many of the same positive effects on the network as in an OLSR protocol.

In this thesis I build my study on earlier implementations of QoS-aware routing protocols as SWAN, and the link quality extension to olsrd, and show a partial solution to the QoS problem. They solve similar problems that I have faced in my implementation. QOLSR proposes extensions that should be added to OLSR to make it a QoS-aware routing protocol, and has had a great influence on my work.

The functionalities that I have chosen to implement, based on my studies, are theories that are well known in the community. The effects that these proposed functionality changes to the OLSR protocol has in the MANET have been well documented, but not tested. That is why I believe it is important to implement the different approaches suggested, to be able to compare these to the theories that have already been established and other protocol solutions. My implementation is a start to being able to have a fully functional QoS-aware protocol based on available bandwidth calculation, and therefore a contribution to the goal of making a MANET a better network solution.

In this section I show a study of which functionality I think should be implemented and the functionalities actually implemented by my extensions to olsrd. This can be viewed as my implementation theory for this master thesis.

My proposed solution may simplified be described as a combination between the functionalities proposed in QOLSR and SWAN implemented into olsrd.
In this section I break down the most important changes to the functionalities that should be implemented to include QoS aware routing to OLSR.

4.2 Implementing available bandwidth into the protocol

After studying the available bandwidth problem for some time, through theory and in the SWAN protocol, it was clear that I had to start changing the driver of the wireless card. After some testing, I found out that this was out of the scope for this thesis. However, the available bandwidth algorithm is an essential part of including QoS based on available bandwidth. After discussing this problem with my supervisor, Andreas Hafslund, we found an alternative solution. The solution proposed in reference [27] shows an implementation of an available bandwidth algorithm and other metrics that can be of use. The proposal discusses what the greatest problems were in implementing these metrics. The program is an open source and in my implementation I will transfer the necessary code to include the listen bandwidth estimation into my implementation. Below is a discussion of the implementation of the listen bandwidth estimation proposed in [27].

The additional constraints of unpredictable link quality and the fact that it can significantly vary over time, is the reason for many new solutions proposing a cross-layer solution. The solution proposed in reference [27] is a cross-layer solution. The solution allows the protocols above the link layer to get information about the current state of the link. Cross-layer design greatly increases the complexity of the design of the network system as shown in [17], but for the wireless technology this should be an improvement, as shown in [3], [4].

In the implementation of [17], it has been required do the implementation above the chipset or firmware, so they can eventually be ported to any IEEE 802.11 software driver that runs on a network card. The metrices for QoS calculations becomes more accurate the deeper one has access to the MAC. For most IEEE 802.11 chipsets (chip dependent) the MAC functionalities are implemented inside the chipset and are the deepest access possible. Linux provides access to the MAC through a driver. The driver that has been altered in this proposal to include the information from MAC needed to calculate available bandwidth, is based on Atheros 5212 chipset cards. These cards have been chosen because of the access to an open source Linux driver (see 19), MAC functionalities in the driver and the support of IEEE 80211a b and g. It is worth mentioning that one of the main issues in the work on achieving the necessary information from the MAC, is that the manufacturers of the network cards often do not let the driver bee open source and are reluctant to letting outsiders get an insight into the driver.

The proposed method of calculating the listen estimation bandwidth calculation follows the method explained in 2.6.1. To install and use the wireless network driver, read see [26] .

The modified driver reports its output through the Linux “proc”file system. Each node in the QoS-aware OLSR based on bandwidth, should read the output of the modified driver to update its available bandwidth. The modified driver also reports several other medium metrics: idle time, busy, transmit and receive time in addition to the MAC layer delay. The modified driver also calculates and reports the link metrics: available bandwidth given short frames (132 bytes), available bandwidth given large frames (2346 bytes) and the available bandwidth given average frame size. The modified driver reports the modulation rate currently used and the average frame size for the different types of frames on that link. The frames are unicast, receive data multicast, receive management unicast and receive management multicast .For my implementation the available bandwidth, given average available bandwidth is the value of interest.

4.3 Implemented functionalities

The QOLSR project shows a detailed theoretical approach on how to implement a QoS-aware protocol. This project has been important for my thesis, and I have followed the methodic outlined in reference [25] throughout the implementation. Because of this it is very hard to distinguish between the QOLSR and this part of my thesis. I would suggest that the reader goes through [25] for an even deeper and more complete understanding of the whole protocol.

4.3.1 Neighbour sensing

The neighbour sensing is handled by the original OLSR protocol as previously discussed. But it is important to specify that all links must be checked in both directions to be valid. This is done by the HELLO message which is received by all one hop neighbours, but not relayed to further nodes. This hello message also includes the QoS matrices. My thesis proposal suggests including the available bandwidth metric. A HELLO message is shown in the following table and the different fields are described bellow:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Reserved             |     Htime     |  Willingness  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Link Code   |   Reserved    |       Link Message Size       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Neighbour Address                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           QoS fields value bandwidth                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           QoS fields value link quality                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Neighbour Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           QoS fields values bandwidth                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           QoS fields value link quality                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                             .  .  .                           :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Link Code   |   Reserved    |       Link Message Size       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Neighbour Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            QoS fields values bandwidth                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             QoS fields link quality                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Neighbour Address                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
etc.
```

Table 3: QoS Hello message

The description of the fields that are not described in the original HELLO message in *Table 1* is as follows:

Available Bandwidth:
The suggested configuration of this field is 24-bit number, measured in Kbits/second. The first 16 highest bits are the integer part and the last 8 lowest bits represent the decimal part. The last 8-bit number, measured in milliseconds uses the last 8 bit of the 32 bit field. The delay on link is represented by its mantissa (four highest bits of Delay field) and by its exponent (four lowest bits of Delay field). The implementations in this thesis do not use the delay QoS metric, and because of this I propose to use the whole field for the available bandwidth measurement. When the whole 32 bit field is used for the available bandwidth value, the implementation of converting the field from the HELLO messages to storage in the tables becomes less complicated. As the float field also is stores at 32 bit.

Link quality:
The remaining 32-bits represent the other QoS parameters that should be exchanged between neighbour nodes to get the final value on the link. In the implementation of this thesis this is the link quality field.

From the hello message a receiving node records a set of neighbours in a table. The table is shown below. An overview of all sets in OLSR and how they interact is shown in *figure 5*. The implementation changes done by this thesis to the standard OLSR protocol does not change the interaction of messages in OLSR, only the contents of the sets.

Neighbour address
Status
Willingness
Available bandwidth
Link quality
Time

Table 4: Neighbour set

- The neighbour address field in the table represents the address of the neighbour
- The status field represents the status of the link between the present node an that neighbour (e. g. symmetric (2 way link) or asymmetric link)
- The willingness field is an integer between 0 and 7 which specifies the node's willingness to carry traffic on behalf of other nodes
- The available bandwidth field represents the available bandwidth on the link between the present node and that neighbour.
- The link quality field represents the link quality on the link between present node and that neighbour.
- The time field represents the time at which this record expires and must be removed

4.3.2 Quality of service multipoint relay selection

Each node of the network selects independently its own set of QMPRs. This set is calculated to contain a subset of the 1-hop neighbours which provides maximum bandwidth and minimum delay from each 2-hop neighbour to the given node. The QMPR set needs not to be optimal, however it SHOULD be small enough to minimize the number of generated TC messages in the network. The information required to perform this calculation is acquired through the periodic exchange of HELLO messages. QMPRs of a given node are declared in the subsequent HELLOs transmitted by this node, so that the information reaches the QMPRs themselves. The QMPR set is re-calculated when a change in 1-hop or 2-hop neighbour sets with bi-directional link is detected; or a change is detected in their QoS conditions. Reference [20, 21] gives an analysis and examples of QMPR selection algorithms.

4.3.3 TC message

Topology Control extension messages are sent by each QMPR node in the network at regular intervals to declare its QMPR selector set and QoS conditions. The information flooded through the network by these TC messages gives most of the information needed to calculate the routing table. *Table 5* shows how I have altered the original TC message to the QoS aware TC message used in the implementation.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              ANSN             |           Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             QoS Multipoint Relay Selector Address             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             QoS fields value bandwidth                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             QoS fields value link quality                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             QoS Multipoint Relay Selector Address             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             QoS fields values bandwidth                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             QoS fields link quality                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
etc.
```

Table 5: QoS TC message

The description of the fields not discussed earlier in T*able3* is shown below.

- 32 bit float variable in the bandwidth field.
- I use the already existing Link quality field in OLSRD as the last QoS field.

Each node maintains a routing table which allows it to route packets for the other destinations in the network with optimal metrics respecting QoS constraints. The nodes which receive a TC message parse and store some of the connected pairs of form:

Node
Source
Available bandwidth
Link quality

Table5: TC QoS route information

- The node field represents the addresses found in the TC message list.
- The source field represents the QMPR selector of "node"
- The available bandwidth field represents the available bandwidth on the link between “node” and “source”
- The link quality field represents the link quality between "node" and "source".

If the route is only partially known or broken it will not be stored.

4.3.4 Two-hop neighbours

As only bandwidth and link quality values on links to 2-hop neighbours are used for QoS multipoint relay selection, a node records a set of "2-hop tuples" that is shown in the table below:

Address
2-hop address
2-hop available bandwidth
2-hop link quality
Time

Table 6: 2-hop set

This set describes a symmetric, MPR or QMPR links between its neighbours and the 2-hop neighbourhood.

- The address field represent the address of a neighbour
- The 2-hop address is the address of a 2-hop neighbour to the present node
- The 2-hop available bandwidth designates the available bandwidth on the link between address and the 2-hop address.
- 2-hop link quality is the link quality on the link between the address and the2-hop address
- Time specifies the time at which a tuple expires and must be removed.

The route proposed to the 2-hop address is the widest route based on available bandwidth.

4.3.5 Processing of messages

Upon receiving a HELLO message, the node should update the neighbour information corresponding to the sender node. It provides the same algorithm described in OLSR [2] for

HELLO processing. Moreover, it adds and updates bandwidth and link quality values in neighbour set, 2-hop neighbour, MPR Selector set and QMPR selector set.

TC messages are generated by QMPRs retransmitted by MPRs in order to diffuse the messages in the entire network. The tuples in the topology set are recorded with the topology information that is exchanged through TC extension messages, following the same algorithm for TC message processing as described in reference [25]. Moreover, it adds and updates available bandwidth and link quality metrics values in the topology set.

4.3.6 Updating tables

QOLSR applies the same heuristic used for the selection of MPRs in OLSR. MPR selection procedure is detailed in [16]. After selecting the MPRs among the neighbours, the link status of the corresponding 1-hop neighbours is changed from a symmetric link to an MPR link in the neighbour table. The MPR set is re-calculated when:

- A change in the neighbourhood is detected, i.e. either a symmetric link with a neighbour is failed, or a new neighbour with a symmetric link is added; or
- A change is detected in the 2-hop neighbourhood such that a symmetric link is either detected or broken between a 2-hop neighbour and a neighbour.

The QMPR set is re-calculated when:

- A change in the neighbourhood is detected, i.e. either a symmetric link with a neighbour is failed, or a new neighbour with a symmetric link is added; or
- A change is detected in the 2-hop neighbourhood such that a symmetric link is either detected or broken between a 2-hop neighbour and a neighbour; or
- A change of bandwidth or link quality on links of 1-hop or 2-hop neighbourhood is detected. Therefore, a node measures the percentage change of previous bandwidth or link quality if the percentage change of more than 10% is detected.
- The bandwidth or delay percentage changes exceed bandwidth threshold or link quality threshold.

4.3.7 Routing table Calculation

It is important to define the different routing technique

- A path bandwidth is the minimum bandwidth value on intermediate arcs.
- A widest path between two nodes is the path having maximum path bandwidth between those two nodes.
- A shortest path between two nodes is normally defined as path the least hops or having minimum path delay between those two nodes. But In this thesis the link quality is used as the variable in the shortest path. All the node link quality in a path is added up and the path with the best total link quality is chosen.
- A shortest-widest path is the widest path, and with shortest delay when there is more than one widest.

My proposal for implementation of a route calculation is a widest path algorithm. The needed implementation for a widest path algorithm is discussed below:

AVL tree:

To look up the available nodes in the routing process an AVL tree is used. This works well in the implementation of OLSR even though AVL trees have been receiving some critique of being a better approach theory than in practice, because of their high implementation complexity to keep them balanced compared to other alternatives.

The AVL tree is named after its two inventors, G.M. Adelson-Velsky and E.M. Landis, An AVL tree is a self-balancing binary search tree. In an AVL tree the heights of the two child sub trees of any node differ by at the most one level. Accordingly it is also called height-balanced. Lookup, insertion, and deletion all take O(log *n*) time in both the average and worst cases. Additions and deletions may require the tree to be rebalanced by one or more tree rotations.

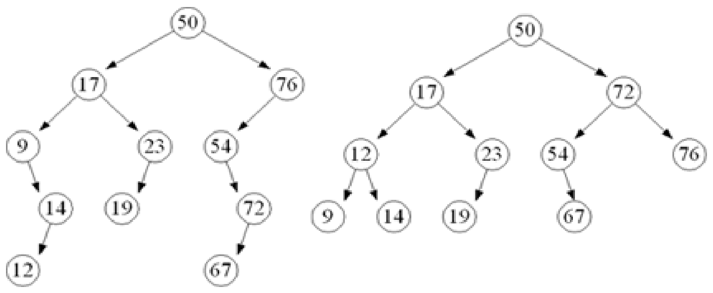


Figure 10: Comparison between normal binary (to the left) tree and an AVL tree (to the right) from []

The balance factor of a node is the height of its right subtree minus the height of its left subtree. A node with balance factor 1, 0, or -1 is considered balanced. A node with any other balance factor is considered unbalanced and requires rebalancing the tree. The balance factor is either stored directly at each node or computed from the heights of the subtrees.

Dijkstra's algorithm:

The Dijkstra's algorithm was originally discovered by Prof Dr Edsger Wybe Dijkstra and was intended to solve the single-source shortest path problem for a directed graph with a non-negative edge weight.

To simplify the description, one might think of it as if the vertices of the graph represent cities and edge weights represent driving distances between pairs of cities connected by a direct road. Dijkstra's algorithm can be used to find the shortest route between two cites. It is easy to compare this problem with the problem of finding routes between nodes as vertices parameter and the available bandwidth as the edge parameter. The problem I face is that in my implementation I need the route with the best bottleneck available bandwidth value, as described in section 2.7.1 That's why I must alter the algorithm to choose the path with the least available bandwidth.

The input of the algorithm consists of a weighted directed graph *G* and a source vertex *s* in *G*. I will denote *V* the set of all vertices in the graph *G*. Each edge of the graph is an ordered pair of vertices (*u*,*v*) representing a connection from vertex *u* to vertex *v*. The set of all edges is denoted *E*. Weights of edges are given by a weight function *w*: $E \rightarrow [0, \infty)$; therefore *w*(*u*,*v*) is the non-negative cost of moving directly from vertex *u* to vertex *v*. The cost of an edge in my implementation is available bandwidth. The cost of a path between two vertices is normally the sum of costs of the edges in that path, but in my implementation the cost is the edge with the least available bandwidth. For a given pair of vertices *s* and *t* in *V*, the algorithm finds the path from *s* to *t* with lowest cost (i.e. the shortest path). The algorithm can also be used for finding costs of shortest paths from a single vertex *s* to all other vertices in the graph, which can also be of use.

The algorithm works by keeping for each vertex *v* the cost *d*[*v*] of the shortest path found so far between *s* and *v*. Initially, this value is 0 for the source vertex *s* (*d*[*s*]=0), and infinity for all other vertices, representing the fact that we do not know any path leading to those vertices (*d*[*v*]=∞ for every *v* in *V*, except *s*). When the algorithm finishes, *d*[*v*] will be the cost of the shortest path from *s* to *v* — or infinity, if no such path exists.

The basic operation of Dijkstra's algorithm is edge relaxation: if there is an edge from *u* to *v*, then the shortest known path from *s* to *u* (*d*[*u*]) can be extended to a path from *s* to *v* by adding edge (*u*,*v*) at the end. This path will have length *d*[*u*]+*w*(*u*,*v*). If this is less than the current *d*[*v*], we can replace the current value of *d*[*v*] with the new value. Edge relaxation is applied until all values *d*[*v*] represent the cost of the shortest path from *s* to *v*. The algorithm is organized so that each edge (*u*,*v*) is relaxed only once, when *d*[*u*] has reached its final value.

Below is the pseudo code of the algorithm. The pseudo code shows the original version and the altered one. In the following algorithm, u := Extract_Max(Q) searches for the vertex *u* in the vertex set *Q* that has the best *d*[*u*] value. That vertex is removed from the set *Q* and returned to the user. The variable x keeps record of the best edge outgoing on each vertic.

```
    function Dijkstra(G, w, s)
       for each vertex v in V[G]                       // Initializations
             d[v] := infinity
             previous[v] := undefined
       d[s] := 0
       S := empty set
       Q := V[G]
       while Q is not an empty set                     // The algorithm itself
//           u := Extract_Min(Q)                       //old version
             u := Extract_Max(Q)                       // new version
             S := S union {u}
             X := 0                                    //new version
            for each edge (u,v) outgoing from u //only if within threshold
                        //new version
                   if (d[u] < w(u,v) && w(u,v) > x)
                          d[v] := d[u]
                          x := w(u,v)
                          Q := Q union {v}
                          previous[v] := u

                    Else if w(u,v) < d[u] && d(u,v) > x)
                          d[v] := w(u,v)
```

```
20                        x:= w(u,v)
21                        Q := Q union {v}
22                        Previous[v] := u

//               Old relax verion instead of line 13 and down
//                  if d[u] + w(u,v) < d[v]
//                        d[v] := d[u] + w(u,v)
//                       Q := Q union {v}
//                       previous[v] := u
```

Figure 11: Edited Dijkstra algorithm

My proposed edited Dijkstra algorithm in Figure 11 should return the routes with the best bottleneck value for available bandwidth. Vertex should be looked at as nodes and edges as available bandwidth on link.

A problem discovered after implementation and some testing, was that the available bandwidth values on the outgoing links for a node would all be the same, as they all showed the available bandwidth for that node. This was not a surprise in itself, but will reduce the efficiency of the Dijkstra algorithm. There are two ways to solve this problem. The first is to set the available bandwidth value of a link between a node and its neighbour to the neighbour's available bandwidth value or NULL if there is no neighbour. The other is to multiply the neighbour's available bandwidth with our bandwidth. Both of these solutions would give each link out from a node individual values from which they could calculate the route..

A more generic problem would be to find all the shortest paths between *s* and *t* (there might be several different ones of the same length). Then, instead of storing only a single node in each entry we would store all nodes satisfying the relaxation condition. For example, if both *r* and *s* connect to *t* and both of them lie on different shortest paths through *t* (because the edge cost is the same in both cases), then we would add both *r* and *s* to previous[t].

When the algorithm completes, data structure will actually describe a graph that is a subset of the original graph with some edges removed. Its key property will be that if the algorithm was run with some starting node, then every path from that node to any other node in the new graph will be the shortest path between those nodes in the original graph, and all paths of that length from the original graph will be present in the new graph. Then to actually find all these short paths between two given nodes, we would use path finding algorithm on the new graph, such as depth-first search.

The original thought was to include a Dijkstra algorithm that would include both available bandwidth and link quality as parameter for the routing. However this part of the implementation proved to be more complicated than expected. Some of the details around the problem I discovered in the routing process is described under *section 5*.

4.4 Output implementation

Most outputs that the original olsrd printed out have been edited. belw shows an example of a typical output from my extension implementation. All the available bandwidth (ABW) extensions have been included. To show the available bandwidth estimations for neighbours, links, topology and routing (Dijkstra). For an in-depth explanation of each field, see Appendix A.

```
--- 16:02:54.09 ---------------------------------------------------- LINKS

IP address      hyst   LQ     lost   total  NLQ     ETX   BW
10.0.0.2        0.000  1.000  0      10     1.000   1.00  100.000  -0.008
10.0.0.3        0.000  1.000  0      10     1.000   1.00  100.000  -0.008

--- 16:02:54.09 ------------------------------------------------ NEIGHBORS

IP address      LQ     NLQ    ABW      SYM   MPR   MPRS  will
10.0.0.2        1.000  1.000  100.000  YES   NO    NO    3
10.0.0.3        1.000  1.000  100.000  YES   NO    NO    3

--- 16:02:54.09 ------------------------------------------------- TOPOLOGY

Source IP addr  Dest IP addr   LQ     ILQ    ETX    ABW
10.0.0.2        10.0.0.1       1.000  1.000  1.00   99.000
10.0.0.2        10.0.0.3       1.000  1.000  1.00   99.000
10.0.0.3        10.0.0.1       1.000  1.000  1.00   75.000
10.0.0.3        10.0.0.2       1.000  1.000  1.00   75.000
```

4.5 QOLSR suggestions, not implemented

There is one problem in routing that must be defined. The text below is quoted from [25] to make sure the problem is exactly defined:

"*A problem is called NP (nondeterministic polynomial) if its solution (if one exists) can be guessed and verified in polynomial time; i.e. nondeterministic means where no particular rule is followed to make the guess. If a problem is NP and all other NP problems are polynomial-time reducible to it, the problem is NP-complete. Thus, finding an efficient algorithm for any NP-complete problem*
implies that an efficient algorithm can be found for all such problems, since any problem belonging to this class can be recast into any other member of the class. It is not known whether any polynomial-time algorithms will ever be found for NP-complete problems, and determining whether these problems are tractable or intractable, remains one of the most important questions in theoretical computer science. When an NP-complete problem must be solved, one approach is to use a polynomial algorithm to approximate the solution; the answer thus obtained will not necessarily be optimal but will be reasonably close".

The issue of finding a routing path in a MANET with addictive (loss probability, etc) and multiplicative metrics (delay, delay jitter, hop-count) is NP-Complete if the total metrics exceeds 1. The below text is quoted from [25] and describes this problem:

"There is no efficient polynomial-time algorithm that can surely find a feasible path that simultaneously satisfies both constraints. Including a single metric, the best path can be easily defined. Otherwise, including multiple metrics, the best path with all parameters at their optimal values may not exist. For example, a path with both maximum bandwidth and minimum delay may not necessarily exist. Thus, the precedence among the metrics in order to define the best path must be decided"

In my thesis, bandwidth is considered to be the only metric for calculating path, but the use of other metrics should be considered in the future and are proposed in QOLSR.

The proposed use of path algorithms in [25] for routing purposes is shown below:

Best-effort flows:
A shortest-widest path algorithm is suggested as the best solution for these flows. The algorithm finds the path with maximum bandwidth (widest path) and when there is more than one widest path the path with shortest delay is used.

In my proposal it might seem natural to use the information of link-quality for use as a shortest path in my algorithm. I argue that when the widest path is calculated, the probability of another path having exactly the same result is at best very slim, and therefore the extra calculation and storing space necessary for finding the shortest-widest path is not worth the effort. This is so especially when kept in mind that the devices often used in MANETs often have little battery power and storing space. The reason for my argumentation is that the wireless medium is very random and unpredictable and that the calculation of the available bandwidth is stored with high precision (32 bit float variable). Another reason that has to be mentioned is that even though the calculation is stored with high precision, the actual calculation is not precise. The result of this is that, even if the case of two paths having the same available bandwidth path value, it would only be for a short time (until next update) and the values would in fact not be the same because of the inaccuracy.

The only way I can see the shortest-widest path would be of any use in this case, is if it were altered. The altered version that I would suggest would be to use a percentage version, where the available bandwidth for a path is within a percentage (an example could be 10%) from the widest path. It would use the path with the lowest delay or, since my implementation is not using delay as a metric, it would be exchanged with link quality. This suggestion would complicate the route calculation and storing space even more as section shows 4.3.6. Since I could not find enough documentation to back up my argument or to predict the consequences, and my thesis left no time for testing, I decided not to explore this further, but still only implement the widest path algorithm based on this argumentation and that in my task description only mention the use of available bandwidth as a metric .

QoS flows:
The proposed solution in reference [25] for QoS flows that need to satisfy a number of QoS constraints like bandwidth > Threshold_bandwidth and/or delay<Threshold_delay, etc., is that the efficient scalable heuristic, based on Lagrangian relaxation, is applied. This heuristic can treat two, three and four-metrics and provides a polynomial solution with a good approximation bound that yields high performance in finding feasible paths. This algorithm is not explored in further because of the time span of the thesis. But the routing algorithm should be implemented at a later time.

4.6 Implementing a test function

I have included an extension to the existing olsrd_switch program. This function creates a file "/etc/connection.dat" where all clients and their respective available bandwidths are recorded. When the link sets are recalculated in olsrd, the routing process looks up the available bandwidth value of the file instead of the dynamic calculation from the network card driver.

4.7 Effects of the implementation

Incorporateing an admission control scheme and feedback scheme to meet the requirements of real-time application, seems to be a good solution. Increasing packet delivery ratio greatly, while packet delay and energy dissipation decrease significantly, and the overall end-to-end throughput is not impacted, compared with routing protocols that do not provide QoS.

Including available bandwidth routing will decrease the link breakage and decrease the overall message overhead by message in the network and enables admission control schemes and feedback schemes to be implemented.

However on the sown side the communication to discover networks occurs continuously and the message size increased more. Because the program is fairly large and complex, continuous calculation and memory burdens may be too heavy for small computers and the battery life decrease

4.8 General about the implementation .

When the implementation was first started, I tested the plug-in interface that was implemented into olsrd. I found that when using this plug-in interface I did not get the access I needed for my implementation. For this reason I decided to add my implementation as an extension to qolsrd.

All of my changes to olsrd have been done inside the /src folder. Most of the changes have been done to the files that start with qos_, but there are several changes done in other files. Most changes have been commented with "MKV". My implementation has only considered linux and IP-v4 compatibility

Several software tools are needed for developing projects like olsrd. Fortunately all these tools are made freely available through the GNU project[24]. The tools used for development are the GNU compiler collection (gcc), the GNU emacs editor, the GNU make utility, the GNU debugger(gdb), the GNU profiler(gprof), valgrind and memproof.

For details about installing driver, installing olsrd and the recommended configuration file I show to Appendixes. Modified MADWIFI driver for interlayer interaction driver can be downloaded at [26] and the original olsrd program can be downloaded at [28]
.

5 Test of implementation

5.1 How to use the test program

The olsr_switch is a traffic router that accepts multiple olsrd instances to connect and communicate over TCP via the loopback interface. Olsr_switch works on the two data sets; clients and links. When receiving traffic from an olsrd client, the switch will forward this traffic to all other clients witch have a valid link. For a closer look on how to use olsrd_switch I refer to Appendix D
My available test program has been implemented into the olsrd_switch programme and works like this:

The way to change the available bandwidth values for a node in the file is shown by the command below. IP address represent the IP address for the client node.

abw <ipadress> <available bandwidth>

An example of the update is

Abw 10.0.0.1

5.2 How to test

My master thesis requires me to test the concept of my thesis. My implementation has three major parts that must be tested

5.2.1 The available bandwidth algorithm

Remember that this code should not be tested if you have not read through text included in [26], as this code only works on some network cards and may have unintended consequences on other network cards. That is why I have included a QoS setting 5 in the configure file. This allows the program to run only with the olsrd_switch configured available bandwidth settings.

I quote from reference [27] to prove this concept and have tried similar tests on my network to ensure that this concept and implementation works.

"One of the two nodes was sending ten "pings" every second, each echo request and echo reply being 65507 data bytes long. "

"The percentage busy receive and transmit time reported by the driver is a little bit higher than the theoretical computed values (16.6% compared to 18% for receive, and 31.1% compared to 32% for transmit) for several reasons. Firstly, the value reported by the driver is rounded up to the next integer. Secondly, the driver considers the retries in the calculations (see section 3 of this paper for details). In the theoretical calculations, we assume no

collisions and no retries. If we were comparing results taken in a congested environment, many retries would occur and the theoretical values would be less precise than the value reported by the
driver. In this example, the idle time is 50%, therefore fewer collisions and retries occurs. The last reason why the reported value is higher is that the modified driver also considers the medium busy time spent by management frames. Since there are few of them and since they are small, ignoring them in the theoretical calculation has no major impact.

*Another observation on the GUI is the available bandwidth. For example, if we look at the link metrics of the destination node, the available bandwidth for small frames is 1.771Mbps, 10.687Mbps for large frames, and 8.703Mbps for frames of 1510 bytes. The average frame size to this destination is 1510 bytes. To explain how the available bandwidth values are obtained, consider the example of small frames (132 bytes including IP and IEEE 802.11 frame overhead). Using the same formulas as above, the frame exchange sequence time of each frame of length 132 bytes is 298µs. Because the idle time is 50%, 0.5*106/298µs = 1677 such frames could be sent every second if all the idle time was used for that purpose and if no additional collision or retry occurs. Since each of those frame carry 132 bytes, the available bandwidth is 1677frames/sec * 132bytes = 1771Kbps. Similar calculations are made for large and average size frame size."*

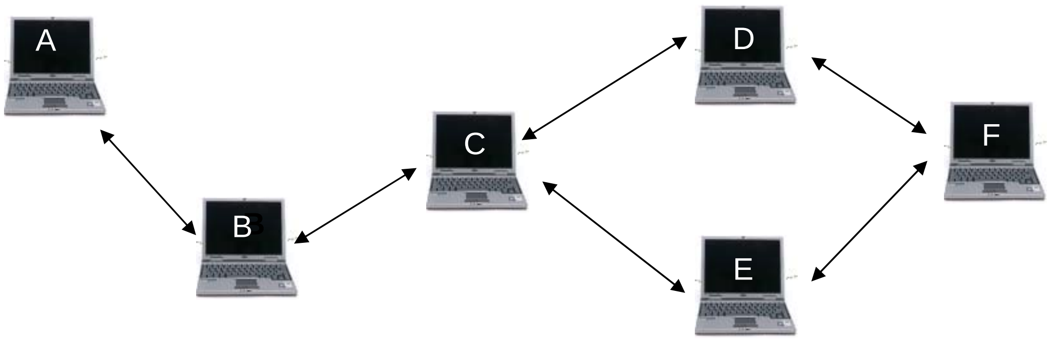
5.2.2 Message flow

To be able to achieve a correct routing calculation based on available bandwidth, it is important that the available bandwidth calculation of each node is correctly flooded through the network. There are several ways to control this. First of all the output of debug level 2 shows every node's topology, link neighbour and routing information. Message flow has been tested on the network before I implemented the routing protocol. I have tested the message flow on large and small network and ensured that the information shown in the debug level 2 outputs is correct.

Furthermore, I have included my TC messages and hello message to the –dispin and –dispout commands. How to use these command I show to man page for olsrd in appendix B These commands show what messages are sent out from and in to a node. All my nodes show that the correct messages are sent through the network.

5.2.3 Route calculation

To prove that the routing calculation works as it should, I have constructed a simple scenario, as shown in *Figure 12*. When traffic is sent from A to F, the routing protocol must chose to send traffic through D or E. The routing protocol is supposed to choose the node with the best available bandwidth. In the test I initially set node D to 10 mbs and node E to 15mbs. If the protocol works properly, it should choose to send the traffic through node E. Then I change the available bandwidth to let node D become 20 mbs. This is more than the 10% change that must happen to force the node to update the available bandwidth calculation and flood the network with the new topology information. Now the routing protocol should pick the route through node D instead. There should also be several tests in larger networks to ensure that the routing protocol works properly. This will be done when time allows.



Figur 12: route test

When I initially tested my protocol, late in my master thesis, I found that I had not carried out my implementation correctly in accordance with my proposal in chapter 4. This was a surprise to me at this time, and I had little time to correct the problem. When I tried to correct this problem, a different problem occurred. I got a never ending loop when printing my routing. It was obvious to me that something had gone wrong in the implementation of the Dijkstra's algorithm. The algorithm is complicated, and several corrections were tried to correct the problem. The implementation of olsrd has about 40000 lines of code which have been time extremely time consuming to go through and understand. Several changes have been done to make my implementation a successful extension. When my critical error occurred, I could not find an easy solution with the time frame that was left for my thesis. For this reason I am sad to conclude that the implementing of the routing part of my protocol has not been successful.

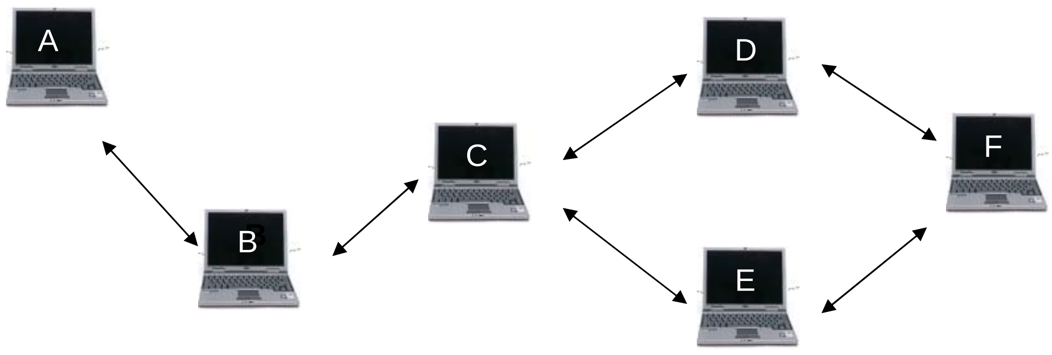
The strange thing about this result, is that the error does not occur to all the nodes, and that it, based on the experiences I have had with the testing, does not always occur.

For a more thorough analysis and solution for this problem I need time to retest several areas of my code. I must continue to test all the implementation of my routing protocol and all information that my routing protocol bases its calculations on. This is in reality all my code. Because of the complexity of this code, this will be a time consuming process, but in no way impossible. Hopefully I will be able correct this error at a later time. As my experience and understanding of the olsrd protocol has greatly increased through this thesis.

5.2.4 QMPR calculation

I have created a simple scenario to show if a node chooses the correct QMPRs nodes. The test is very similar to the route test. First of all it is important that all the nodes have been set to the same willingness and that the nodes have been set to only chose one QMPRs node in the configuration file. In this way I ensure that a node does not force itself to be chosen as a QMPR and prohibit the possibility of ending up with two nodes as QMPRs. If node F in the figure below is to choose a QMPRS from D and E, it must chose the node with the best link.

If D has an available bandwidth 10 Mbs and E has a available bandwidth of 15 Mbs, F should choose E as its QMPRS.



Figur 13: QMPRs test

This test suffers from the same error as previously defined. The QMPRS selection should also be tested in a larger network when the time allows.

6 Work to be done

As this master thesis only call for a test of concept, there is still a lot of testing outstanding, specifically regarding the effect of the protocol on the network. This testing is very complicated because of the random and unpredictable behaviour of the MANET. To confirm a successful implementation, there should be a test on the behaviour of the MANET in a small network and compare this to the behaviour in a large network. The behaviour of the network is very dependent on the size. In section 2.3.2 there is a list of matrices that would be natural to use for testing purposes .

A test of the implementation with different parameters to the configuration file, is also necessary. A small change to the parameters can have a large effect on the behaviour of the MANET, and the conclusion of the effectiveness of the MANET can change. It is important to find the best configuration of the parameters. This is an ongoing discussion in the MANET science environment and was out of scope for this master thesis.

There should also be a test that compares the QoS-aware OLSR, based on available bandwidth calculation, with a similar AODV implementation and other routing protocols. The contest between the protocols and in which situations the different protocols are best, is ongoing and the result will change as the different protocols improve.

When implementation has gone through the tests above and the necessary changes have been made, a real life test should be done. The difference between a simulation and a real life situation has often times proven to be significant.

The implementation suggested for this paper includes only the message handling and routing of a best effort flow. To include a complete QoS-aware routing protocol different flows should be defined and processed in different ways. The SWAN protocol shows how this is done in an AODV MANET. Many of the functionalities proposed in SWAN should be included into my implementation. The architecture in an OLSR environment, including these functionalities would look very similar to the SWAN proposal as shown in *Figure 8* except for the send and receive probes shown in the figure. These probes would not be needed and their functionality would be handled by the TC and HELLO messages in the OLSR MANET.

There are some issues in the implementation that must be handled before the implementation could be called a success. These issues a probably minor, but the complexity of the program and the timescale of the thesis have not allowed for sufficient time to handle them in this thesis.

7 Conclusion

This master thesis has shown that there are many solutions to routing in MANETs and that these networks have many problems outstanding that must be solved before a MANET can be called an adequate solution. However, because of the many attractive properties of the MANET there is no reason not to try to solve the problems.

I propose my own solution for QoS-aware OLSR based on available bandwidth. My proposal bases itself on earlier proposals and theory for solving the many issues in MANETs. The implementation of the extension to oslrd has not been successful. The routing has shown to be too complex to handle for me in the timescale of this thesis. The reason for this is related to my inexperience of implementing programs of this size, the complex nature of the available bandwidth calculation and the amount of the theory that had to be reviewed in this thesis.

The calculation of the available bandwidth estimation has proven to be complex. Even though my solution is not completely my own, I have shown how it should be solved and implemented this into my solution. In the flooding of the available bandwidth through the MANET, my implementation has been a success. By altering the existing messages in the OLSR protocol, I have proven that each node stores the available bandwidth of all the other nodes in the MANET. I have reviewed how the implementations should be done and chosen an extension solution from the plug-in possibility. My extension has not changed the functionality of the old implementation of olsrd, but is an extension option to it.

Even if the testing had been successful, it would have been an overstatement to call the implementation a success. The protocol has only been rudimentary tested and has not been tested in a real world environment. The olsrd implementation has already been used in real life situations, and I think that with some more time for implementations and testing my implementation will prove to be an improvement to olsrd.

I hope my implementation will be a start for further study and implementations, and contribute to solve some of the problems that MANET's faces.

References:

[1] I.Chlamtac, M. Conti and J. J-N. LIU, "Mobile Ad Hoc Network: Imperatives and challenges", Elsevier Ad Hoc Networks, Vol. 1, No 1, July 2003

[2] B.P. CROW, I. Widjaja, J.G.Kim and P.T, Sakai, "IEEE 802.11 Wireless Local Area Networks", IEEE Communications Magazine, Sept. 1997.

[3] S.Xu and T. Saasawi, #Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless Ad hoc Networks?", IEER Communication Magazine, sept. 1997.

[4] X, Hong, K. Xu and M.Geria "scalable Routing Protocols for Mobile Ad Hoc Networks", IEE Network, July/August 2002, pp 11-21

[5] C.E. Perkins and E. Belding.Royer, "the Ad Hoc ON-Demand Distance Vector Protocol", Chapter 6 in Ad Hoc Networking by C.E Perkins (ed), Addison-Wesley 2001

[6] P.Jacquet, P.Muhlethaler, T.Clausen, A.Laouiti, A.Qayyum and L.Viennot, " Optimized Link State Routing Protocol for Ad Hoc networks", IEE INIMIC conference, 2001.

[7] T.H Clausen, G.Hansen, L Christensen and G.Behrmann "The Optimized Link State Routing Protocol, Evalution through Expriments and Simulations" The 4th International Symposium on Wireless Personal Mobile Communications (WPMC) Seopt.2001.

[8] P.Mohapatra, J.Li,C. Gui "QoS in Mobile Ad Hoc Networks", IEEE Wireless Communications, June 2003

[9] Q.ni, LRomdhani, T. Tutletti and I Aad, "A survey of QoS Enhancements for IEEE 802.11 Wireless LAN", journal of wireless Communication and Mobile Computing Wiley, 2004, vol.4, No.5, pp547.566

[10] G.S. ahn, A.T.Campell, A.Veras an L-H Sun, "Supporting Service Differentiation for real-Time and Best-Effort Traffic in Stateless Wireless Ad Hoc Networks (SWAN)", IEE T. Mobile Computing, Vol.q, No.3, Jul-S. 2002ept

[11] Lei Chen, Wendi B Heinzalman, "QoS.Aware Routing Based on Bandwidth Estimation for Mobile Ad Hoc Networks" Vol 23, No 3, Mars 2005.

[12] Mathieu Deziel, Louise Lamont "Implementation of an IEE 802.11 Link Available Bandwidth Algorithm to allow Cross-Layering", 2004

[13] Charles E. Perkins "Ad Hoc Networking"

[14] RFC 2501

[15] Andreas Tønnesen, "Implementing and extending the Optimized Link State Routing Protocol", 2004

[16] RFC-3626, "Optimized Link State Routing Protocol (OLSR)

[17] http://www.olsr.org/docs/README-Link-Quality.html

[18] V. Kawadia. P. R. Kumar, "A Cautionary Perspective on Cross Layer Design", IEEE Wireless Communication Magazine. July 9, 2003, Revised June 24, 2004.
[19] MADWIFI, "Multiband Atheros Driver for WiFi", [Source Code], Available: http://sourceforge.net/projects/madwifi.

[20] H. Badis, A. Munaretto, K. Al Agha, and G. Pujolle. "Optimal Path Selection in a Link State QoS Routing Protocol". IEEE VTC2004-spring, May 2004.

[21] H. Badis and K. Al Agha. " QOLSR, QoS routing for Ad Hoc Wireless Networks Using OLSR". European Transactions on Telecommunications, Vol. 15, No. 4, pp 427-442, 2005.

[22] http://en.wikipedia.org/wiki/Dijkstra_algorithm

[23] http://www.crc.ca/en/html/manetsensor/home/software/software

[24] Richard M. Stallman et al. *GNU project*. http://www.gnu.org.

[25] Hakim Badis, Khaldoun A. Agha, "Quality of Service for Ad hoc Optimized Link State Routing Protocol (QOLSR)", 2006

[26] http://www.crc.ca/en/html/manetsensor/home/software/software

[27] M. Déziel, L. Lamont, "Implementation of an IEEE 802.11 Link Available Bandwidth Algorithm to allow Cross-Layering", WiMob 2005 Wireless and Mobile Computing, Networking and Communications, Montreal, Canada, August 22-24, 2005.

[28] http://www.olsr.org

APPENDIX A

My extension to the 0.4.8 olsrd program has introduced some changes in the debug output. Below is an overview of the outputs on debug level 2, which is the recommended default for the available bandwidth extensions.

```
--- 14:28:56.80 ---------------------------------------------------- LINKS

IP address       hyst   LQ     lost   total  NLQ    ETX   ABW
192.168.0.1      0.000  1.000  0      10     1.000  1.00  130.356
```

This table contains the links to our neighbours. It contains the following columns.

IP address - the IP address of the interface via which we have contact to the neighbor.

- *hyst* - the current hysteresis value for this link.
- *LQ* - the quality of the link determined at our end.
- *lost* - the number of lost packets among the *n* packets most recently sent by our neighbour via this link. *n* is the link quality window size.
- *total* - the total number of packets received up to now. This value starts at 0 immediately after a link has come to life and then counts each packet. It is capped at the link quality window size.
- *NLQ* - this is our neighbour's view of the link quality. Previously we have called this the Neighbour Link Quality. This value is extracted from LQ HELLO messages received from our neighbours. NB: If a neighbour stops sending packets completely, we do not have any means of updating this value. However, in this case the LQ value will decrease and the link thus be detected as becoming worse.
- *ETX* - this is the ETX for this link, i.e. 1 / (NLQ x LQ).
- *ABW This is the* available bandwidth calculated at the present node

```
--- 14:28:56.80 ------------------------------------------------ NEIGHBORS

IP address       LQ     NLQ    SYM   MPR   MPRS  will  ABW
10.0.0.6         1.000  1.000  YES   YES   NO    6     125
```

This table contains a list of all our neighbours. It is closely related to the link table in that we are connected to a neighbour via one or more links. The table has the following columns.

- *IP address* - the main IP address of the neighbour.
- *LQ* and *NLQ* - the LQ and NLQ values of the best link that we have with this neighbour. (In multi-interface configurations we can have more than one link with a neighbour.)
- *SYM* - this states whether the link to this neighbour is considered symmetric by olsrd's link detection mechanism.

- *MPR* (multi-point relay) - this indicates whether we have selected this neighbour to act as an MPR for us.
- *MPRS* (multi-point relay selector) - this indicates whether the neighbour node has selected us to act as an MPR for it.
- ABW the best available bandwidth value of the best link we have with this neighbour

```
--- 14:28:56.80 ------------------------------------------------- TOPOLOGY

Source IP addr   Dest IP addr     LQ     ILQ    ETX    ABW
10.0.0.6         192.168.0.2      1.000  1.000  1.00    125.000
10.0.0.6         10.0.0.5         1.000  1.000  1.00    75.000
```

This table displays the topology information that olsrd has gathered from LQ TC messages. It states which nodes in the network report links to which other nodes and which quality these links have. It is olsrd's view of the world beyond its immediate neighbour nodes, i.e. its view of the nodes that it cannot reach directly. This table has the following columns.

- *Source IP addr* - the node that reports a link.
- *Dest IP addr* - the node to which the source node reports the link.
- *LQ* (link quality) - the quality of the link as determined by the source node. For the source node this is the Link Quality. For the destination node this is the Neighbour Link Quality.
- *ILQ* (inverse link quality) - the quality of the link as determined by the destination node. For the source node this is the Neighbour Link Quality. For the destination node this is the Link Quality. We just did not want to name it "NLQ", as we use NLQ only for the link quality reported by our neighbours. But functionally this is equivalent to the NLQ we know from the link and neighbour tables.
- *ETX* - the ETX value for this link, calculated by ETX = 1 / (ILQ x LQ)
- *ABW* the available bandwidth determined by the source node.

```
--- 14:28:56.80 ------------------------------------------------- DIJKSTRA

10.0.0.6:1.00:125.000 (one-hop)
10.0.0.5:2.00:75 <- 10.0.0.6:1.00:75.000 (one-hop)
```

This table displays the best routes that olsrd could find for each destination that it knows about. The leftmost IP address given on each line is the destination of a route. The remaining IP addresses in a line specify the nodes on the route between ourselves and the destination address. Moving from the destination address to the right, address by address, moves us closer from the destination to ourselves, hop by hop.

In the above case we see routes to two nodes, 10.0.0.6 and 10.0.0.5. In the first line, there are not any intermediate nodes between us and the destination, the destination address is the only IP address in this line. In the second line we have one intermediate node, 10.0.0.6. So, the second line describes a route to 10.0.0.5 via 10.0.0.6.

The number after the first colon following an IP address in the table, is the total ETX of the route up to this IP address, i.e. the sum of the ETX values of all hops between us and this IP address. The number after the second colon, is the bottleneck available bandwidth between us and this IP address.

In the above example the first line represents a path with an ETX value of 1.00 to 10.0.0.6. As we have seen in the neighbour table above, 10.0.0.6 is our neighbour, so the route to it consists only of a single hop, which has an ETX of 1.00 and ABW of 125.

In the second line, 10.0.0.5 is not a neighbour of ours. However, 10.0.0.6 is, and from the topology table above we can tell that 10.0.0.6 reports a link to 10.0.0.5. So, we can reach 10.0.0.5 via 10.0.0.6. This is what this line says. Remember that each line represents a route by first giving the IP address of the destination (10.0.0.5) and that moving to the right means moving towards ourselves until one of our (one-hop) neighbours is reached. If we move from 10.0.0.5 to the right, we find 10.0.0.6, which is our (one-hop) neighbour. So we have a route.

If we would like to know which path a packet that we send to 10.0.0.5 takes, we have to read the line backwards. We then see that the packet first travels to our (one-hop) neighbour 10.0.0.6 via a link that has an available bandwidth of 125 and ETX of 1.00 (which we can confirm by looking at the neighbour table above). From there, it is forwarded to 10.0.0.5 via another link that has available bandwidth of 75 and an ETX of 1.00, resulting in a total ETX of 1.00 + 1.00 = 2.00 and available bandwidth of 75 which is the number that follows 10.0.0.5. Remember that the ETX value given for an IP address is the cumulative ETX for the complete route up to this IP address. But the available bandwidth is the least available bandwidth that is recorded. for the complete route up to this IP address.

If olsrd is able to find a route between us and the destination, the last IP address in the line is one of our neighbours. In this case, "(one-hop)" is appended to the line to illustrate that the last IP address is one of our (one-hop) neighbours. However, let us assume that we havejust switched on olsrd. In this case, it does not know about all links in the network, yet, as it has not received QoS TC messages from all nodes. So, it may know that a node exists (as it has already received QoS TC messages from it) but it does not necessarily know how to reach it (as it may not have received QoS TC messages from nodes between it and ourselves, yet). In this case the last IP address is the last node that is reachable from the destination and the line ends with the word "FAILED".

The same is true for neighbours to which we do not have a symmetric link. We know that they are there, but we do not have a link to them, hence olsrd cannot find a route, which results in "FAILED".

APPENDIX B

olsrd

Section: Maintenance Commands (8)
Updated: Jun 2004

NAME

olsrd - Optimized Link State Routing protocol daemon

SYNOPSIS

olsrd [**-i interface1 [interface2 ...]**] [**-f configfile**] [**-d debuglevel**] [**-ipv6**] [**-ipc**] [**-dispin**] [**-dispout**] [**-bcast broadcastaddress**] [**-delgw**] [**-hint HELLO interval**] [**-tcint TC interval**] [**-midint MID interval**] [**-hnaint HNA interval**] [**-tos TOS value**] [**-T scheduler poll rate**]

DESCRIPTION

olsrd is an implementation of the Optimized Link State Routing protocol for Mobile Ad-Hoc networks(MANET). The protocol is described in RFC3626. It is designed to be run as a standalone server process - but as it is still in an experimental stage most users will prefer running it with some debug output which is directed to STDOUT.

This manual page only lists the command line arguments. For details of the configuration file see the comments included in **/etc/olsrd.conf.** Note that none of these options need to be set at the command line - all these options and others can be set in the configuration file.

The homepage of olsrd is **http://www.olsr.org**

OPTIONS

-i *interface1 ... interfaceN*
    This option specifies on what network interfaces olsrd should run. These interfaces cannot be aliased interfaces such as eth0:1.

-f *configfile*
    This option overrides the default configuration file path used by olsrd - **/etc/olsrd.conf**

-d *debuglevel*
    This option specifies the amount of debug information olsrd should write to STDOUT. If set to 0 olsrd will run in the background.

-ipv6
    This option instructs olsrd to use the Internet Protocol version 6. The default is version 4.

-ipc

This option allows the GUI front-end created fro olsrd to connect to olsrd at runtime.

-dispin

This option, when set, causes olsrd to display all incoming packet data on STDOUT. When using IPv4 the data is displayed in decimal format, when using IPv6 the data is displayed in hexadecimal format.

-dispout

This option, when set, causes olsrd to display all outgoing packet data on STDOUT. When using IPv4 the data is displayed in decimal format, when using IPv6 the data is displayed in hexadecimal format.

-delgw

If this option is set olsrd will remove any default routes set prior to adding an Internet route based on OLSR routing.

-bcast *broadcastaddress*

This option specifies what IPv4 broadcastaddress to use for OLSR control traffic. The only value that currently makes sense when setting broadcast address mannually is **255.255.255.255.** The default action is to use the broadcastaddres that the network interface is preconfigured with(per interface).

-hint *seconds*

This value sets the interval on which **HELLO** messages should be generated. The value is a floating point number representing seconds.

-tcint *seconds*

This value sets the interval on which **TC** messages should be generated. The value is a floating point number representing seconds.

-midint *seconds*

This value sets the interval on which **MID** messages should be generated. The value is a floating point number representing seconds.

-hnaint *seconds*

This value sets the interval on which **HNA** messages should be generated. The value is a floating point number representing seconds.

-tos *TOS-value*

This option sets the **type of service** value that should be set in the OLSR control traffic packet IP headers.

-T *seconds*

This option sets the polling intervall of the scheduler. The default is 0.1 seconds. This option should only be considered if running with really low emission intervals.

FILES

/etc/olsrd.conf

SEE ALSO

olsrd.conf(8).

patibility

=============

is driver requires the following support in your kernel:

o Wireless Extensions versions 14 or later (version 16 preferred)

o Sysctl support
o Crypto API support (AES support is used if present, otherwise the
AES-CCMP cipher module falls back to a private implementation)

Testing has been done with kernels from 2.4.2x to 2.6.10. Early 2.4 kernels may require patches; e.g. for crypto support and/or updated wireless extensions.

APPENDIX C

olsrd.conf

Section: File Formats (5)
Updated: Dec 2004
Index Return to Main Contents

NAME

olsrd.conf **-** configuration file for **olsrd(8)**

DESCRIPTION

The file *olsrd.conf* Which is located in */etc* by default, contains run-time configuration for the Optimized Link State Routing daemon **olsrd(8).** Olsrd can however be set to read an alternative configuration file at startup using the **-f** command line argument. A configuration file parser/generator **olsrd_cfgparser(2)** can be built as both a standalone executable or a dynamically linked library from the olsrd sources. The DLL version can be used to easily create tools for generating/parsing olsrd configuration files.

The configuration file consists of comments, single options and option blocks.

COMMENTS

Comments are everything following a **#** in a line. This data is discarded. Commenting out options is an easy way to make olsrd use the default value for that option.

SINGLE OPTIONS

Single options are single lines options that consists of a keword and a user supplied value. Note that a comment can follow such a option on the same line. Valid single options are:

DebugLevel [0-9]
Controls the amount of debug output olsrd sends to stdout. If set to 0, olsrd will detatch from the current process and run in the background. A value of 9 yields a maximum of debug output. Defaults to **0**.

IpVersion [4|6]
Olsrd supports both IP version 4 and 6. This option controls what IP version olsrd is to use. Defaults to **4**.

AllowNoInt [yes|no]
Olsrd supports dynamic configuration of network interfaces. This means that interfaces on which olsrd runs, can be reconfigured and olsrd will update itself with no need to be restarted. Olsrd also supports removal and addittion of interfaces in run-time. This option specifies if olsrd should keep running if no network interfaces are available. Defaults to **yes**.

TosValue [0-16]
This value controls the type of service value to set in the IP header of OLSR control traffic. Defaults to **16**.

Willingness [0-7]
Nodes participating in a OLSR routed network will announce their willingness to act as relays for OLSR control traffic for their neighbors. This option specifies a fixed willingness value to be announced by the local node. 4 is a neutral option here, while 0 specifies that this node will *never* act as a relay, and 7 specifies that this node will *always* act as such a relay. If this option is not set in the configuration file, then olsrd will try to retrieve information about the system power and **dynamically** update willingness according to this info. If no such info can be retrieved willingness is set to **4**.

UseHysteresis [yes|no]
If set to yes hysteresis will be used as explained in section 14 of RFC3626.

HystScaling [0.01-0.99]
Sets the scaling value used by the hysteresis algorithm. This must be a positive floating point value smaller than 1.0. Consult RFC3626 for details. The default value is **0.5.**

HystThrHigh [HystThrLow-0.99]
This option sets the upper threshold for accepting a link in hysteresis calculation. The value must be higher than the one set as the lower threshold. Defaults to **0.8.**

HystThrLow [0.01-HystThrHigh]
This option sets the lower threshold for setting a link to asymmetric using hysteresis. The value must be lower than the one set as the upper threshold. Defaults to **0.3.**

Pollrate [0.0-]
This option sets the interval, in seconds, that the olsrd event scheduler should be set to poll. A setting of 0.2 will set olsrd to poll for events every 0.2 seconds. Defaults to **0.1**.

TcRedundancy [0|1|2]
This value controls the TC redundancy used by the local node in TC message generation. To enable a more robust understanding of the topology, nodes can be set to announce more than just their MPR selector set in TC messages. If set to 0 the advertised link set of the node is limited to the MPR selectors. If set to 1 the advertised link set of the node is the union of its MPR set and its MPR selector set. Finally, if set to 2 the advertised link set of the node is the full symmetric neighbor set of the node. Defaults to **0.**

MprCoverage [1-]

This value decides how many MPRs a node should attempt to select for every two hop neighbor. Defaults to **1** , and any other setting will severly reduce the optimization introduced by the MPR secheme!

LinkQualityLevel [1-2]

This setting decides the Link Quality scheme to use. If set to 0 link quality is not regarded and olsrd runs in "RFC3626 mode". If set to 1 link quality is used when calculating MPRs. If set to 2 routes will also be calculated based on distributed link quality information. Note that a setting of 1 or 2 **breaks RFC3626 compability!** This option should therefore only be set to 1 or 2 if such a setting is used by all other nodes in the network.

ClearScreen [yes|no]

If set to yes and olsrd is running with a debuglevel >0 the terminal to which output is sent(STDOUT) is cleared prior to writing updated tables. This makes it easier to follow changes in real-time by eye. If STDOUT is not a terminal(eg. it is a file), then no action is taken upon writing tables to STDOUT.

OPTION BLOCKS

Option blocks are configuration options that holds a body of sub-options encapsulated in curled braces(**{}**). Valid options are:

IpcConnect {[sub-options]}

Olsrd can allow processes to make a TCP connection to itself on which data regarding the topology will be transmitted. This is typically used by GUI applications to provide a user-friendly front-end to olsrd. This option block controls thees kind of connections.

MaxConnections [0-5] This option specifies how many connections that can exist simoultneously. Multiple connections have not been tested, and does not work! This option should only be used to control wheter or not processes can connect to olsrd by setting it either to 0, which will tell olsrd not to allow any connections, or by setting it to a positive value. Defaults to 0.

Host [IPv4 address]

This option specifies a single host that is allowed to connect to olsrd. By default only the loopback address(127.0.0.1) is set to be allowed. So if you want to be able to connect from another host you should add it here. This option can be repeated to add multiple hosts.

Net [IPv4 netaddress] [IPv4 netmask]

Here you can specify an entire netrange of IP addresses which olsrd will allow TCP connections from. This option can be repeated to add multiple networks.

Hna4 {[sub-options]}

Hosts in a OLSR routed network can announce connecitivty to external networks using HNA messages. This optionblock is used to set the IPv4 nteworks to be announced by this host.

[IPv4 netaddress] [IPv4 netmask]

Specifies a IPv4 network to announce in HNA messages. Multiple entries can be added. To announce Internet connectivity set **0.0.0.0 0.0.0.0**

Hna6 {[sub-options]}

Hosts in a OLSR routed network can announce connecitivty to external networks using HNA messages. This optionblock is used to set the IPv6 nteworks to be announced by this host.

[IPv6 netaddress] [0-48]

Specifies a IPv6 network to announce in HNA messages. The second value is the prefix-length of the network address. Multiple entries can be added. To announce Internet connectivity set **:: 0**

LoadPlugin "[plugin-name]" {[sub-options]}

Specifies a plugin that olsrd is to load at startup.

PlParam [key] [value]

Sends a pair of parameters to the plugin at initialization. Consult individual plugin documentation to find the possible parameters.

Interface "[device-name1]" "[device-name2]" ... {[sub-options]}

This optionblock specifies one or more network interfaces on which olsrd should run. Atleast one network interface block must be specified for olsrd to run! Various parameters can be specified on individual interfaces or groups of interfaces. This optionblock can be repeated to add multiple interface configurations.

Ip4Broadcast [IPv4 address]

Forces the given IPv4 broadcast address to be used as destination address for all outgoing OLSR traffic on the interface. In reallity only the address **255.255.255.255** makes sense to set here. If this option is not set the broadcast address that the interface is configured with will be used. This address will also be updated in run-time if a change is detected.

Ip6AddrType [site-local|global]

This option sets what IPv6 address type is to be used in interface address detection. Defaults to site-local.

Ip6MulticastSite [IPv6 address]

Sets the destionation of outgoing OLSR traffic on this interface to use the specified IPv6 multicast address as destination if the site-local address type is set on this interface.

Ip6MulticastGlobal [IPv6 address]

Sets the destionation of outgoing OLSR traffic on this interface to use the specified IPv6 multicast address as destination if the global address type is set on this interface.

HelloInterval [0.0-]

Sets the interval on which HELLO messages will be generated and transmitted on this interface.

HelloValidityTime [0.0-]

Sets the validity time to be announced in HELLO messages generated by this host on this interface. This value must be larger than than the HELLO generation interval to make any sense. Defaults to 3 * the generation interval.

TcInterval [0.0-]

Sets the interval on which TC messages will be generated and transmitted on this interface.

TcValidityTime [0.0-]

Sets the validity time to be announced in TC messages generated by this host on this interface. This value must be larger than than the TC generation interval to make any sense. Defaults to 3 * the generation interval.

MidInterval [0.0-]

Sets the interval on which MID messages will be generated and transmitted on this interface.

MidValidityTime [0.0-]

Sets the validity time to be announced in MID messages generated by this host on this interface. This value must be larger than than the MID generation interval to make any sense. Defaults to 3 * the generation interval.

HnaInterval [0.0-]

Sets the interval on which HNA messages will be generated and transmitted on this interface.

HnaValidityTime [0.0-]

Sets the validity time to be announced in HNA messages generated by this host on this interface. This value must be larger than than the HNA generation interval to make any sense. Defaults to 3 * the generation interval.

Weight [0-]

When multiple links exist between hosts the weight of the interface is used to determine the link to route by. Normally the weight is automatically calculated by olsrd based on the characteristics of the interface, but here you can specify a fixed value. Olsrd will choose links with the lowest value.

MISC

The homepage of olsrd is **http://www.olsr.org**

FILES

/etc/olsrd.conf

APPENDIX D

Olsr_Switch network simulation

$Id: README-Olsr-Switch.html,v 1.3 2005/06/04 22:28:27 kattemat Exp $

Summary

This document gives a brief introduction to the olsr.org OLSR daemon network simulation extentions and tools. The network simulation tool olsr_switch provides olsrd processes running in a special *host-emultion* mode with access to a virtual network. Multiple such processes can run on a host. This network can be freely manipulated by the user allowing for simulation of dynamic large-scale networks.

Only IPv4 is supported for now.

Background

olsr_switch was inspired by the uml_switch used to communicate between UML instances, which I have used for olsrd testing. Also I remember seeing that the guys at LRI working on the qolsr project has done something similar.

The target users for this kind of things are probably mainly developers and researchers. But it might come in handy for others as well.

Description

olsr.org

The olsr.org OLSR daemon is an implementation of the Optimized Link State Routing protocol a IP routing protocol for mobile ad-hoc networks. Multiple interesting extensions are available for the implementation. Read more at olsr.org.

olsr_switch

The application, called olsr_switch, is really a traffic router that will allow multiple olsrd instances to connect and communicate over TCP via the loopback interface. Connecting to olsr_switch on remote hosts will be possible, but it is not implemented at the current time.

Basically olsr_switch works on two datasets - clients and links. A client is a connected olsrd process and there are two uni-directional link between all nodes(A->B and B->A) that can be manipulated independently. When receiving traffic from a olsrd client the switch will forward this traffic to all other clients to which it has a valid link. Links can be set to three different "modes" based on the *quality* set on the link:

- **Open**(quality 100) - all traffic will be forwarded over this link.

- **Closed**(quality 0) **-** no traffic will be forwarded over this link.
- **Lossy**(quality 1-99) **-** in this mode the quality constraint is used as the chance in percentage, for traffic to be forwarded over the link. So if quality is set to 25 there will in average be 75% packet loss.

The application provides the user with a prompt/shell where various commands for topology manipulation or data retrieval are available. A help command is provided for the users convenience. This help command will also provide the user with specific help on all available commands. More on this later.

olsrd host-emulation clients

The communication interface against olsr_switch can naturally not be 100% transparent for the olsrd process which normally runs on UDP/698 on "real" network interfaces. Olsrd itself had to be modified to set up a virtual interface connecting to olsr_switch. This virtual interface is transparent to all overlying functionality. This means that that olsrd host-emulation can run in both RFC and LQ mode, and that plugins can be loaded normally etc. But no routes are added by the olsrd process. Later on route information might be signaled from the olsrd instance to olsr_switch so that a centralized route database is available to the user. As of now, you can watch the olsrd debug output or easier yet, run one node with the httpinfo plugin to get the route/topology/link information.

Currently olsrd cannot run on both real and host-emulation interfaces, but this might change in the future if I get convinced that it is useful. No routes can be added by the host-emulating olsrd instances, so

Building and running

As of yet, olsr_switch builds and runs on GNU/Linux and FreeBSD systems.

Building

The olsr.org olsrd source code and pre-compiled packages can be downloaded from the olsr.org download section. But as of now you need the CVS version if you want the olsr_switch code. To check out the current snapshot do:

```
cvs -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/olsrd login

cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/olsrd co olsrd-
current
```

Now enter the olsrd source directory and do:

```
olsrd-current# make
```

to build olsrd itself, and:

```
olsrd-current# make switch
```

to build olsr_switch. If no error messages occurred, you should now have the two executables *olsrd* and *olsr_switch* available in the current directory.

Running

Before starting any olsrd host-emu clients the switch must be started. This is done using the command *olsr_switch* and the application will initiate the communication socket and provide the user with a prompt for input:

```
olsrd-current# ./olsr_switch
olsrd host-switch daemon version 0.1 starting
Initiating socket TCP port 10150
OHS command interpreter reading from STDIN
>
```

olsrd_switch has a buildt-in command for starting and stopping local olsrd instances and we will learn more about this later. But olsrd can also be ran in host-emulation mode from the command line. To start olsrd in host-emulation mode do:

```
./olsrd -hemu IP-ADDRESS
```

Here *IP-ADDRESS* will be the IP address that the process will set as its main address in the emulation mode. This address has no connection to the real IP-stack and can be chosen freely. Of cause, no two instances can run using the same IP.
It might be wise to have a separate olsrd configuration file for host-emulation. In that case issue:

```
./olsrd -f FILENAME -hemu IP-ADDRESS
```

As of now there is no emulated destination address used. All traffic will be passed on regardless of the network address used. Olsrd communicates using broadcast/multicast so that multiple "overlapping" networks might exist.

Command-line interface

Blah blah blah

help

Type 'help cmd' for help on a specific command

```
> help
Olsrd host switch version 0.1
Available commands:
        help - Help on shell commands
        exit - Exits olsr host switch
        log - Displays or sets log bits
        list - List all connected clients or links
        link - Manipulate links
```

list

The *list* command is used for client and link listing. The help page states:

```
> help list
Usage: list <clients|links>
Description:
This command will list all the clients or all the links registered
by olsr_switch. By default clients are listed.
```

link

The *link* command is used for manipulating links. Here is the help page for this command:

```
> help link
Usage: link <bi> [srcIP|*] [dstIP|*] [0-100]
Description:
This command is used for manipulating olsr links. The link quality
is a number between 0-100 representing the chance in percentage
for a packet to be forwarded
on the link.
To make the link between 10.0.0.1 and 10.0.0.2 have 50% packetloss do:
 link 10.0.0.1 10.0.0.2 50
Note that this will only effect the unidirectional link
10.0.0.1 -> 10.0.0.2. To make the changes affect traffic
in both directions do:
 link bi 10.0.0.1 10.0.0.2 50
To completely block a link do:
 link 10.0.0.1 10.0.0.2 0
To make all traffic pass(delete the entry) do:
 link 10.0.0.1 10.0.0.2 100
Note that "bi" can be used in all these examples.
Wildcard source and/or destinations are also supported.
To block all traffic from a node do:
link 10.0.0.1 * 0
To set 50% packetloss on all links to 10.0.0.2 do:
 link * 10.0.0.2 50
To delete all links do:
 link * * 100
Wildcards can also be used in combination with 'bi'.
To list all manipulated links use 'list links'.
```

olsrd

The *olsrd* command is used for manipulating links. Here is the help page for this command:

```
> help olsrd
Usage: olsrd [start|stop|show|setb|seta] [IP|path|args]
Description:
This command is used for managing local olsrd instances from within
olsr_switch.
The command can be configured in runtime using the setb and seta sub-
commands.
To show the current olsrd command-configuration do:
 olsrd show
To set the olsrd binary path do:
 olsrd setb /full/path/to/olsrd
To start a olsrd instance with a IP address of 10.0.0.1, do:
 olsrd start 10.0.0.1
To stop that same instance do:
 olsrd stop 10.0.0.1
```

others

Two other commands are available:

- **exit -** terminates olsr_switch.
- **log -** sets the log level.

Read the help pages for details.

Running olsrd in host-emulation mode

Now for a real world example of connecting olsrd instances. Let's assume we have a modified configuration file, */etc/olsrd.emu.conf* that we use for host-emu instances. We'll choose the 10.0.0.0/24 IP address space for our clients.
First start the olsr_switch in one terminal:

```
./olsr_switch
```

Now start the olsrd instances in other terminal(s). If you want to follow olsrd operation you should use a debug value > 0. To easen CPU usage and terminal count you can start multiple instances that will run in the background using the *-d 0* option. In this example we'll run our olsrd instances in the foreground using debug level 1. Start them off(in separate terminals) using:

```
./olsrd -f /etc/olsrd.emu.conf -hemu 10.0.0.x -d 1
```

where x is 1-8(we'll run 8 instances). The screen terminal multiplexer application is *highly* recomended for making your life easier if working on multiple terminals.

Here is the output form the 10.0.0.1 instance of olsrd after some time:

```
       *** olsr.org - 0.4.10-pre (May 30 2005) ***

--- 20:53:26.58 ---------------------------------------------------- LINKS

IP address       hyst   LQ     lost   total  NLQ    ETX
10.0.0.8         0.000  1.000  0      10     1.000  1.00
10.0.0.7         0.000  1.000  0      10     1.000  1.00
10.0.0.6         0.000  1.000  0      10     1.000  1.00
10.0.0.5         0.000  1.000  0      10     1.000  1.00
10.0.0.4         0.000  1.000  0      10     1.000  1.00
10.0.0.3         0.000  1.000  0      10     1.000  1.00
10.0.0.2         0.000  1.000  0      10     1.000  1.00

--- 20:53:26.58 ------------------------------------------------ NEIGHBORS
```

```
IP address        LQ     NLQ     SYM   MPR   MPRS  will
10.0.0.2          1.000  1.000   YES   NO    NO    3
10.0.0.3          1.000  1.000   YES   NO    NO    3
10.0.0.4          1.000  1.000   YES   NO    NO    3
10.0.0.5          1.000  1.000   YES   NO    NO    3
10.0.0.6          1.000  1.000   YES   NO    NO    3
10.0.0.7          1.000  1.000   YES   NO    NO    3
10.0.0.8          1.000  1.000   YES   NO    NO    3

--- 20:53:26.58 ------------------------------------------------- TOPOLOGY

Source IP addr    Dest IP addr     LQ     ILQ    ETX
```

We now have our own virtual network! Notice that you can start olsrd instances from olsr_switch using the *olsrd* command. The equvivalent of the above command line statement would be:

```
olsrd start 10.0.0.x
```

Given that the olsrd command is configured properly (see olsrd show, setb and seta).

At our switch prompt the command *list* yields the following output:

```
All connected clients:
        10.0.0.8 - Rx: 647 Tx: 89 LinkCnt: 0
        10.0.0.7 - Rx: 752 Tx: 105 LinkCnt: 0
        10.0.0.6 - Rx: 790 Tx: 120 LinkCnt: 0
        10.0.0.5 - Rx: 809 Tx: 112 LinkCnt: 0
        10.0.0.4 - Rx: 811 Tx: 125 LinkCnt: 0
        10.0.0.3 - Rx: 804 Tx: 138 LinkCnt: 0
        10.0.0.2 - Rx: 805 Tx: 140 LinkCnt: 0
        10.0.0.1 - Rx: 829 Tx: 119 LinkCnt: 0
```

Hey - everything is running a-ok!

Manipulating links

So lets create some link trouble. This introduction has become too long already, so we'll introduce two simple example conditions:

- o We want 10.0.0.1 only to have a link to 10.0.0.2 and no one else.
- o We want 10.0.0.8 only to have 25% chance of getting direct traffic trough to 10.0.0.2,3,4

Here's what we need to do:

```
> link bi 10.0.0.1 * 0
Setting bidirectional link(s) 10.0.0.1 <=> 10.0.0.8 quality 0
Setting bidirectional link(s) 10.0.0.1 <=> 10.0.0.7 quality 0
Setting bidirectional link(s) 10.0.0.1 <=> 10.0.0.6 quality 0
Setting bidirectional link(s) 10.0.0.1 <=> 10.0.0.5 quality 0
Setting bidirectional link(s) 10.0.0.1 <=> 10.0.0.4 quality 0
Setting bidirectional link(s) 10.0.0.1 <=> 10.0.0.3 quality 0
Setting bidirectional link(s) 10.0.0.1 <=> 10.0.0.2 quality 0


> link bi 10.0.0.1 10.0.0.2 100
Removing bidirectional link(s) 10.0.0.1 <=> 10.0.0.2 quality 100

> list links
All configured links:
        10.0.0.8 => 10.0.0.1 Quality: 0
        10.0.0.7 => 10.0.0.1 Quality: 0
        10.0.0.6 => 10.0.0.1 Quality: 0
        10.0.0.5 => 10.0.0.1 Quality: 0
        10.0.0.4 => 10.0.0.1 Quality: 0
        10.0.0.3 => 10.0.0.1 Quality: 0
        10.0.0.1 => 10.0.0.3 Quality: 0
        10.0.0.1 => 10.0.0.4 Quality: 0
        10.0.0.1 => 10.0.0.5 Quality: 0
        10.0.0.1 => 10.0.0.6 Quality: 0
        10.0.0.1 => 10.0.0.7 Quality: 0
        10.0.0.1 => 10.0.0.8 Quality: 0
```

Now our first condition is met. First all bidirectional links from 10.0.0.1 was blocked and then the bidirectional link to 10.0.0.2 was opened. Now 10.0.0.1 can only see 10.0.0.2 as a neighbor. Note that only manipulated links are listed when issuing 'list links'. olsrd at 10.0.0.1 now shows:

```
       *** olsr.org - 0.4.10-pre (May 30 2005) ***

--- 21:17:46.06 ---------------------------------------------------- LINKS

IP address       hyst   LQ     lost   total  NLQ    ETX
10.0.0.2         0.000  1.000  0      10     1.000  1.00

--- 21:17:46.06 ------------------------------------------------ NEIGHBORS

IP address       LQ     NLQ    SYM    MPR    MPRS  will
10.0.0.2         1.000  1.000  YES    YES    NO    3

--- 21:17:46.06 ------------------------------------------------- TOPOLOGY

Source IP addr   Dest IP addr     LQ     ILQ    ETX
10.0.0.2         10.0.0.1         1.000  1.000  1.00
10.0.0.2         10.0.0.3         1.000  1.000  1.00
10.0.0.2         10.0.0.4         1.000  1.000  1.00
10.0.0.2         10.0.0.5         1.000  1.000  1.00
10.0.0.2         10.0.0.6         1.000  1.000  1.00
10.0.0.2         10.0.0.7         1.000  1.000  1.00
10.0.0.2         10.0.0.8         1.000  1.000  1.00
```

works like a charm. Now let's make sure we can meet condition two:

```
> link bi 10.0.0.8 10.0.0.2 25
Setting bidirectional link(s) 10.0.0.8 <=> 10.0.0.2 quality 25

> link bi 10.0.0.8 10.0.0.3 25
Setting bidirectional link(s) 10.0.0.8 <=> 10.0.0.3 quality 25
```

```
> link bi 10.0.0.8 10.0.0.4 25
Setting bidirectional link(s) 10.0.0.8 <=> 10.0.0.4 quality 25

> list links
All configured links:
        10.0.0.8 => 10.0.0.4 Quality: 25
        10.0.0.8 => 10.0.0.3 Quality: 25
        10.0.0.8 => 10.0.0.2 Quality: 25
        10.0.0.8 => 10.0.0.1 Quality: 0
        10.0.0.7 => 10.0.0.1 Quality: 0
        10.0.0.6 => 10.0.0.1 Quality: 0
        10.0.0.5 => 10.0.0.1 Quality: 0
        10.0.0.4 => 10.0.0.8 Quality: 25
        10.0.0.4 => 10.0.0.1 Quality: 0
        10.0.0.3 => 10.0.0.8 Quality: 25
        10.0.0.3 => 10.0.0.1 Quality: 0
        10.0.0.2 => 10.0.0.8 Quality: 25
        10.0.0.1 => 10.0.0.3 Quality: 0
        10.0.0.1 => 10.0.0.4 Quality: 0
        10.0.0.1 => 10.0.0.5 Quality: 0
        10.0.0.1 => 10.0.0.6 Quality: 0
        10.0.0.1 => 10.0.0.7 Quality: 0
        10.0.0.1 => 10.0.0.8 Quality: 0
```

Now for a look at olsrd 10.0.0.8s output:

```
       *** olsr.org - 0.4.10-pre (May 30 2005) ***

--- 21:23:00.35 ---------------------------------------------------- LINKS

IP address       hyst   LQ     lost   total  NLQ    ETX
10.0.0.7         0.000  1.000  0      10     1.000  1.00
10.0.0.5         0.000  1.000  0      10     1.000  1.00
10.0.0.6         0.000  1.000  0      10     1.000  1.00
10.0.0.2         0.000  0.800  2      10     0.498  2.51
10.0.0.3         0.000  0.600  4      10     0.498  3.35
10.0.0.4         0.000  0.900  1      10     0.800  1.39

--- 21:23:00.35 ------------------------------------------------ NEIGHBORS

IP address       LQ     NLQ    SYM   MPR   MPRS  will
10.0.0.2         0.800  0.498  YES   YES   NO    3
10.0.0.3         0.600  0.498  NO    NO    NO    3
10.0.0.4         0.900  0.800  YES   NO    NO    3
10.0.0.5         1.000  1.000  YES   NO    NO    3
10.0.0.6         1.000  1.000  YES   YES   NO    3
10.0.0.7         1.000  1.000  YES   YES   NO    3

--- 21:23:00.35 ------------------------------------------------- TOPOLOGY

Source IP addr   Dest IP addr     LQ     ILQ    ETX
10.0.0.2         10.0.0.1         1.000  1.000  1.00
10.0.0.2         10.0.0.3         1.000  1.000  1.00
10.0.0.2         10.0.0.4         1.000  1.000  1.00
10.0.0.2         10.0.0.5         1.000  1.000  1.00
10.0.0.2         10.0.0.6         1.000  1.000  1.00
10.0.0.2         10.0.0.7         1.000  1.000  1.00
10.0.0.2         10.0.0.8         0.898  0.498  2.24
10.0.0.5         10.0.0.4         1.000  1.000  1.00
```

```
10.0.0.6         10.0.0.8         1.000  1.000  1.00
10.0.0.7         10.0.0.2         1.000  1.000  1.00
10.0.0.7         10.0.0.3         1.000  1.000  1.00
10.0.0.7         10.0.0.8         1.000  1.000  1.00
```

Yes, there most certainly are some very weak links here.

Well, lets leave it at that :-) The command line interface is meant to be used by applications as well as humans, so if somebody wants to create a GUI front-end that should not be to much work.

Performance

Regarding CPU load I have not done any real testing, but I did try seeing how far I could get on my 1.3Ghz/512MB-RAM desktop system running LQ olsrd instances in the background initiated from olsrd_switch. When reaching a certain amount (15+) the cPU load is very high for neighbor detection, but as soon as links stabelize the CPU is almost idle again. I have ran with 30+ nodes with no problem. But do not start to many instances at the same time.

Note that, this was only using a idle network(no topology changes except new nodes joining). But as soon as olsrd instances can connect from other hosts one can distribute the load. Also the application will be subject to various future optimizations.

Network load measurement tools will also be on the to-do list.