



DEGREE PROJECT IN COMMUNICATION NETWORKS, SECOND LEVEL
STOCKHOLM, SWEDEN 2014

Reactive Networking using Dynamic Link Exchange Protocol

KIM NILSSON

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING

Abstract

This master thesis studies the possibilities of using a radio-router protocol in order to increase the quality of service in dynamic tactical network environments. We cover three radio-router protocols with emphasis on Dynamic Link Exchange Protocol (DLEP).

Many applications, such as voice and video communication, have bandwidth and latency requirements which need to be fulfilled in order to provide a sufficient level of quality. This poses a problem in tactical network environments where links are typically dynamic and both bandwidth and latency can vary. A radio-router protocol can alleviate this problem and also improve the routing in a network by allowing routers to take part of link-layer information.

By using a radio link emulator (RLE) developed by Saab we are able to simulate dynamic network environments. We have performed two experiments by combining the RLE and an implementation of a subset of the DLEP specification draft. Both experiments simulate typical military network scenarios and allow us to analyse the effects of utilizing link-layer feedback.

Our results show that by using DLEP it is possible to provide better quality of service in highly dynamic conditions. We also show that DLEP can influence Optimized Link State Routing (OLSR) by making OLSR aware of changes in the network topology. This leads to a reduced network convergence time with only a small increase in OLSR overhead.

Keywords: DLEP, OLSR, Radio-router Protocol, Reactive Routing, QoS, Link-feedback, Tactical Networking, Dynamic Networking

Acknowledgments

I would like to thank Dr. Anders Gunnar at Saab for all his advice and for the opportunity to do this thesis. Furthermore, I couldn't have done this without the help of Dr. Jayedur Rashid. Both Jayedur and Anders always had time for my questions - thanks for that.

Finally, I would like to thank Ljubica Pajević for her excellent feedback and guidance.

Contents

1	Introduction	1
1.1	Background	1
1.2	Objective	1
1.3	Delimitations	1
1.4	Sustainable development	2
1.5	Methodology	2
1.6	Ethical aspects	3
1.7	Structure	3
2	Literature Review	4
2.1	Radio-Router Protocols	4
2.1.1	Point-to-point Protocol over Ethernet	4
2.1.2	Radio-Router Control Protocol	6
2.1.3	Dynamic Link Exchange Protocol	7
2.2	Utilizing the Link Feedback	9
2.2.1	Routing	9
2.2.2	Quality of Service	10
2.3	Discussion	11
3	Capabilities of Dynamic Link Exchange Protocol	12
3.1	Destination Update Message	14
3.2	Destination Up and Destination Down Messages	15
3.3	Feedback Mechanisms	15
3.3.1	Dynamic Traffic Shaping	16
3.3.2	Reactive Routing	18
4	Testing Environment	20
4.1	Software	20
4.1.1	VirtualBox	20
4.1.2	Radio Link Emulator	20
4.1.3	Optimized Link State Routing	20
4.1.4	iperf	21
4.1.5	iptables	21
4.1.6	tcpdump	21
4.1.7	Linux Traffic Control	21
4.1.8	Dynamic Link Exchange Protocol	22
4.2	Hardware	25
5	Experiments	26
5.1	Scenario 1 - Quality of Service	26
5.1.1	Description	26
5.1.2	Execution	27
5.1.3	Results and Discussion	29
5.2	Scenario 2 - Dynamic Routing	33
5.2.1	Description	33
5.2.2	Execution	33
5.2.3	Results and Discussion	34
6	Discussion	38
7	Future work	39

List of Figures

1	PPPoE neighbour setup [1, p. 72].	5
2	Average OSPF overhead in the nodes of a 10-node network in bits per second [2].	6
3	Setup using R2CP [1].	7
4	Setup using DLEP [3].	8
5	End-to-end availability measured by using ICMP pings [2].	10
6	Excerpt of a dynamic shaper configuration file.	17
7	An example of a <i>traffic control</i> setup.	18
8	Interaction of the traffic control subsystem with the IP stack [4].	22
9	A DLEP message with nested data items.	23
10	An overview of the DLEP server interaction with different subsystems.	24
11	An overview of the radio link emulator and how DLEP is integrated.	25
12	The topology of this scenario.	26
13	Network topology	27
14	Emulated link bandwidth and stream cutoff rates.	28
15	The configuration file used for the dynamic traffic shaper. .	28
16	Radio link utilization. Without DLEP and traffic shaping. .	29
17	Latency between <i>Client A</i> and <i>Server A</i> and incoming packet jitter.	30
18	Amount of dropped packets versus time.	31
19	Radio link utilization. Configuration with DLEP and the dynamic traffic shaper.	31
20	Latency between <i>Client A</i> and <i>Server A</i> and incoming packet jitter.	32
21	Amount of dropped packets.	32
22	Showing one possible router from A to B.	33
23	Network topology	34
24	Overhead and packet loss using 2.5 and 5.0 seconds timings.	35
25	Overhead and packet loss using 5 and 10 seconds timings. .	36
26	Overhead and packet loss using 10.0 and 20.0 seconds timings.	37

List of Tables

1	DLEP Messages	12
2	DLEP Data Items	13
3	Destination Update Data Items	14
4	Destination Up Data Items	15
5	Traffic generated by the four applications.	28
6	Link status	34
7	OLSR configuration settings.	35

1 Introduction

1.1 Background

Military tactical communication networks are usually heterogeneous, often mobile, networks. They are comprised of many different networking techniques such as radio and satellite links, as well as wired ethernet. Routing IP packets and providing quality of service is a difficult problem in wired, static networks and at the tactical edge the environment makes it even more challenging. In this environment there are intermittent radio links which suffers from both varying bandwidth and latency. These highly dynamic conditions makes it difficult to perform well-informed routing decisions and to maintain an acceptable quality of service [3]. Tactical military environments are characterized by short time scale decisions and provide a critical communication infrastructure for soldiers. This creates high requirements on the underlying information structure under harsh conditions.

However, there are ways to improve the situation. Some radio devices are able to expose valuable link-layer information [5]. Providing this information to the network routers allows them to perform more intelligent routing decisions and to respond to network changes, both in terms of the network topology and the quality of its links [2]. In addition to allowing routers to perform more well informed routing decisions, it is possible to provide a higher quality of service by performing packet prioritization as well as data rate shaping based on current properties of the link such as latency or bandwidth.

Unfortunately, link-layer information is usually not provided by the means of some standard interface, rather it is often tightly coupled with built-in hardware and proprietary solutions [1] [5]. In recent years there has been a lot of work done in order to develop a standardized way, a radio-router protocol, that allows routers to take advantage of the available link-layer information that the radio provides [6] [7] [8].

1.2 Objective

The aim of this master thesis project is to investigate how it would be possible to utilize link status information in tactical routers. For example, one might use this information in order to perform more intelligent routing and traffic shaping.

- Is it possible to provide a required level of quality of service for time critical tasks such as voice communication, in a highly dynamic network environment by utilizing link state information?
- Can the use of radio feedback improve the end-to-end availability in a dynamic network environment?

1.3 Delimitations

There are a number of different radio-router protocols and it is clear that some are more mature than others. This master thesis primarily focuses on Dynamic Link Exchange Protocol (DLEP). DLEP is a young radio-router protocol but does however have an active community and is being actively

developed. In addition to DLEP, we also cover two other radio-router protocols: Point-to-point over Ethernet (PPPoE) and Radio-Router Control Protocol (R2CP) but not as thoroughly as we do DLEP.

The link-state information that is provided by any radio-router protocol must be utilized and influence some other system, for example a routing protocol. The mechanism that is utilizing the link-feedback is not necessarily part of the radio-router protocol itself and must be realised by some other system. We used Optimized Link State Routing (OLSR) as our routing protocol. OLSR is a proactive routing protocol optimized for large and highly dense networks, refer to section 4.1.3 for more information about OLSR. There are many other routing protocols that might be just as interesting, especially routing protocols that are specialized towards ad-hoc scenarios. Ad-hoc environments are typically dynamic and are likely to have much to gain by using a radio-router protocol.

By having the logic which utilizes the network feedback in the network devices, i.e. routers, it becomes transparent for the applications using the underlying network. This makes it possible to use existing software, without any modifications, and benefit from any network performance gains. In contrast to controlling the network flow in the network devices it is possible to keep the logic in the application layer. This would allow for more fine-grained traffic shaping that can be implemented in the clients. Application layer feedback mechanisms are not covered by this thesis.

1.4 Sustainable development

The findings published in this thesis does not have any effect on the sustainable development of our society.

1.5 Methodology

This master thesis can be separated into four parts. Step one was to study the task at hand. This was achieved by reviewing literature which led to a deep understanding of the problem and gave insight to what has previously been done. It also gave indications on which mechanisms to use in order to improve network performance. By comparing different radio-router protocols with each other DLEP was chosen as the candidate to use in our test environment as it is the most prominent one out of the three protocols we studied.

Step two consisted of implementing a subset of the DLEP protocol specification according to version 5 of the DLEP Internet-Draft. DLEP is a feature rich protocol and by only implementing the necessary parts we were able to spend more time on analysing our results.

The third part of this thesis was to design the tests and set up the test environments. We performed two different test scenarios in order to properly isolate the effect of each mechanism.

Finally, we collected all the data and analyzed the results. It is worth mentioning that all the tests were run in a virtual environment. The reason for this is that military radio equipment is expensive and support for the protocols studied in this thesis is scarce.

1.6 Ethical aspects

This master thesis is at its core a technical report about radio communication protocols. The goal is that the technology discussed can be used by soldiers in order to have a more reliable communication system.

The work has been performed in cooperation with Saab which is, amongst other things, an arms manufacturer. Saab is based in Sweden and acts under Swedish law. It is worth mentioning that this report is published in the public domain.

1.7 Structure

Chapter 2 of this thesis presents a literature review which provides an introduction to what a radio-router protocol is and gives an overview of three different protocols. It also shows some of the previous work that has been done in this area and explains how one can utilize the information provided by the aforementioned protocols in order to improve network performance.

Chapter 3 presents a more detailed study of DLEP, dissecting some of its messages and explaining how these can be utilized.

In Chapter 4, we describe the software and hardware used in our experimental environments. A brief introduction to each software tool as well as version numbers are given in order to make it possible to accurately reproduce our experiments.

Chapter 5 describes each experiment in general, followed by a more detailed description of how the experiment was executed, the results of the experiment and a discussion section.

Chapter 6 presents a discussion of our findings, followed by Chapter 7, which provides a few examples of interesting topics that could be further studied.

2 Literature Review

In this chapter we give an introduction of radio-router protocols. We will present a comparison between three different protocols, highlighting their respective strengths and weaknesses. We conclude this chapter with a section about different feedback mechanisms that radio-router protocols support, in order to increase network performance.

2.1 Radio-Router Protocols

Routers are a key component of packet switched networking, allowing several networks to be connected in order to create an internetwork. At the tactical edge, networks are often connected over the air, through a radio link. Radio links often offer lower bandwidth and stability than, for example, an ethernet connection does. This makes radio links connecting networks a critical point in the network design.

A radio-router protocol is a protocol that allows the radio to communicate with, typically, a router. The information coming from the radio could in theory be sent to any device, not necessarily a router. This communication channel provides a way for the radio to keep the router informed of any radio link changes.

Radio-router protocols are usually comprised of the following three major functions [3]:

- Available link-metric: There must be some information available to the radio device about the current state of its links.
- Link metric transportation: The protocol must supply the means for transporting the link-metric from the radio to the router.
- Feedback mechanism: The router must have some system in place to utilize the link information.

The three protocols mentioned in this article provide a shared set of measurements. These are: link latency, current data rate, maximum data rate and relative link quality.

The following subsections describes in detail three different radio-router protocols.

2.1.1 Point-to-point Protocol over Ethernet

Point-to-point protocol over Ethernet (PPPoE) was amongst the first protocols that were able to provide link-layer information to routers so that a router was able to perform, for example, more intelligent routing decisions [1], [8]. PPPoE is not a radio-router protocol per se, rather it is a point-to-point protocol. It has been extended through various RFCs to support both a credit based flow control mechanism, granting better support for links with variable bandwidth as this allows to limit the data rate preventing buffer overflows, as well as added support for link-metrics [9].

PPPoE uses *Active Discovery Quality* packets (PADQ) to transport link state information such as relative link quality, latency, current data rate and maximum data rate [9]. The following figure shows how a typical PPPoE setup would look like.

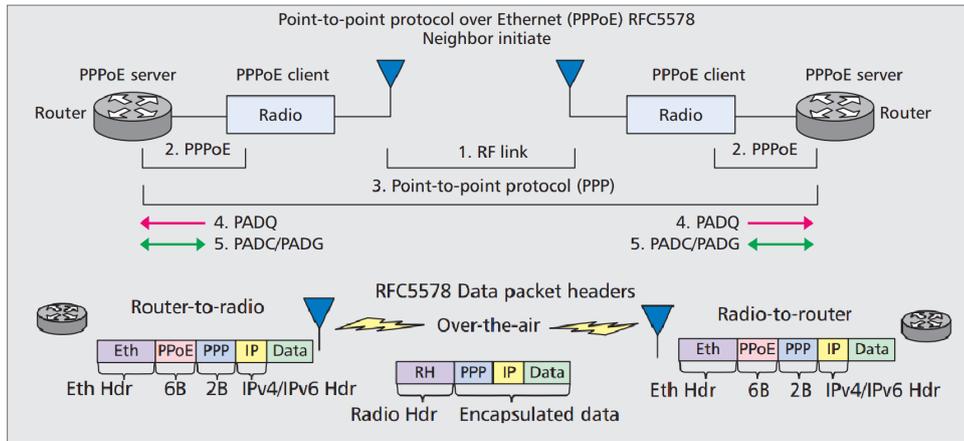


Figure 1: PPPoE neighbour setup [1, p. 72].

Once a radio link has been established between two radio devices, an event is triggered. This event will cause the PPPoE client running in each radio to initiate a PPPoE session with any neighbouring router. Each radio will then probe the radio link at a predetermined interval and send link feedback to the corresponding router over the established PPPoE session in PADQ packets [1].

The router is free to utilize this link state information in order to perform more intelligent routing decisions, e.g. dynamically calculate link costs. Exactly how this would be done is not specified by PPPoE. In addition to acting as a transport layer for link-metrics, PPPoE also allows the router to request credits from the radio. These credits are then used to supply the radio with more data. This provides a way of controlling the data flow to the radio [9].

There are some disadvantages in using PPPoE. Since it is a point-to-point protocol, it will be very inefficient in networks where the underlying network layer has broadcast capabilities, e.g. radio equipment with broadcast support. In order to support broadcast and multicast transmissions, each packet has to be replicated over several point-to-point connections. This causes the same data to be sent over the air several times.

The effect of not being able to send broadcast packets can have a major impact on the network performance. An example of this is demonstrated by Figure 2 which shows the overhead caused by a routing protocol called Open Shortest Path First (OSPF). The experiment was executed using three different radio-router protocols, in the article referred to as radio-router interfaces (*R2RI*), as well as once without any radio-router protocol in order to obtain a control data set. By utilizing the broadcast capabilities of the underlying ethernet network OSPF can reduce the amount of data that has to be sent over the network. This is not possible by using PPPoE due to its point-to-point nature. The overhead in Figure 2 is comprised of *hello*, *link-state advertisement* and *database description* packets which are part of OSPF. The experiment was executed in a 10-node network using two different OSPF hello-timers. This parameter controls how often a OSPF will generate *hello* messages which are part of OSPF's neighbour detection system.

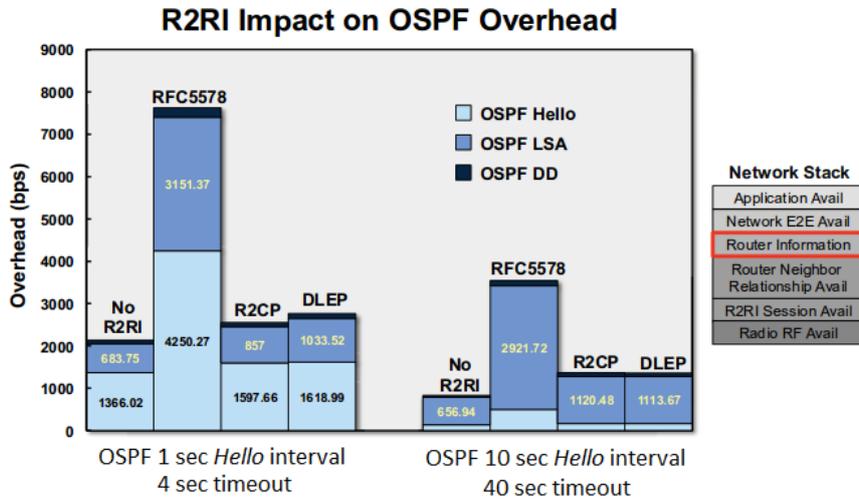


Figure 2: Average OSPF overhead in the nodes of a 10-node network in bits per second [2].

It is clear that PPPoE (denoted as RFC5578 in Figure 2) causes much more overhead to be sent, since it does not have native support for multicasting as compared to the other two radio-router protocols presented in the figure, R2CP and DLEP. The overhead will grow larger as more nodes are added to the network [2]. By comparing the left hand side, which shows OSPF running with a *hello* interval timer of 1 second, to the right hand side which shows OSPF running with a 10 second timer, we can see how the overhead generated by the *hello* messages grows rapidly, being about three times as high as when no radio-router interface is used in both cases.

Another disadvantage of PPPoE is that there must be a point-to-point session established before any data can be sent. This session causes additional overhead data being sent over the air (caused by the PPP header), even in unicast situations, and it can be troublesome to keep the session alive over intermittent links [2]. The need for a point-to-point connection also means that there must be devices running PPPoE on both ends of a connection.

2.1.2 Radio-Router Control Protocol

Radio-router Control Protocol (R2CP) was developed as a response to PPPoE. R2CP does not suffer from the same issues as PPPoE does when it comes to broadcasting, since it differs fundamentally in its design. Opposed to PPPoE, R2CP is a pure radio-router protocol, which induces no overhead over the radio link.

R2CP is described in an RFC draft and provides a way of communicating link state information from a radio device to a router by using UDP as its control channel [7]. Figure 3 shows a typical R2CP setup.

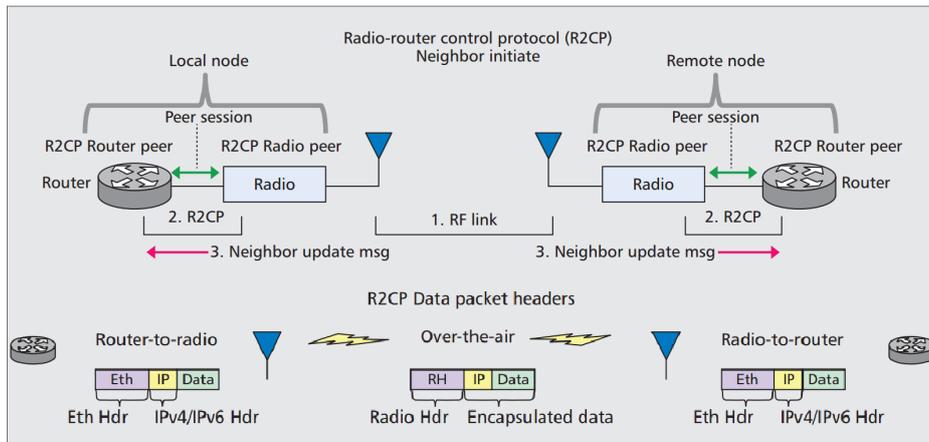


Figure 3: Setup using R2CP [1].

As can be seen by looking at the topology and data flow of an R2CP setup, there is no extra data sent over the air as the radio-router protocol is only acting locally between the radio and a locally connected router. There is no need for a point-to-point connection as in the case of PPPoE. This makes R2CP more flexible and easy to use since it does not depend on the same technique being used in both ends of a connection [1].

As soon as the radio is turned on, it tries to establish a session with any locally connected routers. It does so by sending out a *modem initiate message* to either a predefined IP address and port number or to the network layers broadcast address. Once a session has been established, it is kept alive by using heartbeat messages. The radio can use this session to update the router with information regarding new links, as well as to continuously send link state information such as latency, relative link quality and current data rate to the router. The router uses this information to dynamically calculate OSPF link costs according to a formula specified by the R2CP specification. The link cost formula is based on four weighted values provided by the radio. These values are latency, data rate, relative link quality and a resource factor. The resource factor can be used to specify, for example, how much power is left in a device. This is useful for small, battery powered radio devices [7].

R2CP is a unidirectional protocol, which causes some limitations in its usability. There is no way for the router to send requests to the radio, rather the router must always rely on the information provided. An effect of this is that R2CP provides no control-flow mechanism. The router is expected to control the data rate only by using the provided current data rate metric.

2.1.3 Dynamic Link Exchange Protocol

Dynamic Link Exchange Protocol (DLEP) is similar to R2CP in its design. DLEP is currently specified as an Internet-Draft by the IETF. Since its first appearance in 2010 it has gone through some changes and the latest draft version when writing this is version 5 [6].

DLEP is a pure radio-router protocol. There is no extra overhead transported over the air since it only acts locally between the radio and

the corresponding router. The figure below shows a network with three nodes in which all routers and radio devices are running DLEP [6].

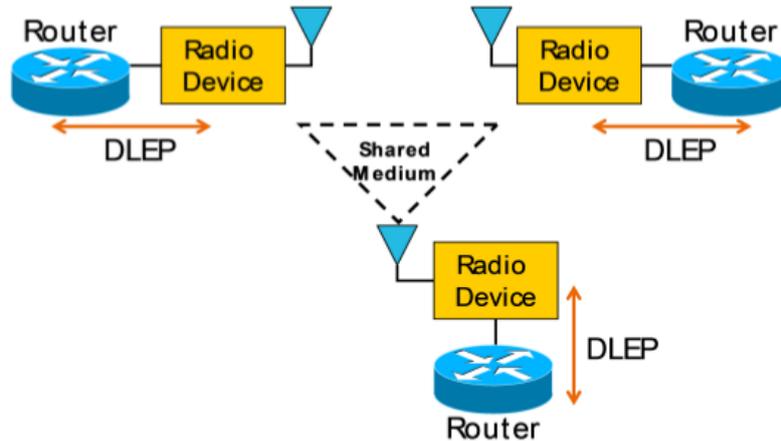


Figure 4: Setup using DLEP [3].

In DLEP there is a session for each connected radio and router pair. There are different ways how a session can be established. In DLEP there is an optional feature which allows DLEP radio devices to send a UDP packet to a predetermined link-local multicast address in order to request a new DLEP session with any local routers. These routers will respond to that request with a unicast IP address and a port number which will be used by the radio device in order to create a new session over TCP. The other way of setting up DLEP sessions is to pre-configure the radios with an IP address and a port number to one or several routers [6].

Once a session has been established it is kept alive by regularly sending heartbeat messages. When the radio notices that a new connection to another radio is available or when an already existing connection has been lost, it informs the router by sending either a *destination up* or a *destination down* packet, respectively. The radio also sends data to the router to inform it of any changes to the radio's current links using *destination update* packets. There is no specific time interval specified for these updates, rather the DLEP specification states that they are to be sent as needed.

Similarly to PPPoE, there is no decision made by the DLEP specification on how the data link information will be utilized by the router. DLEP merely provides the means for the router to get the information [6].

Each data packet sent using DLEP is formatted as a type-length-value (TLV) packet, specifying what type of message it is, the length of the payload and finally the payload itself [6].

The DLEP specification draft (version 5) specifies an optional credit-based system for implementing a data flow mechanism. This mechanism is similar to the corresponding flow control mechanism in PPPoE. There are two credit-windows: *modem receive window* controls how much data the router may send to the radio and *router receive window*, which works in the same way but in the other direction. Both the radio and the router are responsible for granting credits to their receive window, respectively.

2.2 Utilizing the Link Feedback

When it comes to how the routers utilize the link feedback, one can separate the different techniques into two categories. Spatial control, i.e. routing, allows the router to select a specific route for each packet. There is also temporal control, which allows the router to regulate the rate of traffic. Both of these techniques can be dependent on source and destination addresses as well as the type of traffic. This makes it possible to do very specific optimizations based on the topology of the network and on the type of application traffic that is going to be routed.

2.2.1 Routing

By using the link-metric provided to the router it is possible to dynamically alter the parameters used for path selection, e.g. by calculating link costs. By doing so, data traffic might be routed away from low quality links. It is also important to be able to respond to messages from the radio-router protocol indicating that either a new link is available or that a previous connection has been lost. Responding accordingly to such information should make it possible to reduce the network convergence time by not depending on the routing protocol, which is usually based on timeouts. As an example, OSPF waits for a pre configured amount of time for a *hello* message to arrive from its peer. If no message is received during this time the link is declared as down [10]. A radio-router protocol could reduce this time and make the network respond to topology changes faster.

The effects of using dynamic link cost as well as appropriately responding with OSPF *hello* messages to *destination up* events from the radio-router protocol has been previously tested in a paper written by Charland et al [2]. They set up a network of 10 nodes in an emulated test environment and used experimental implementations of R2CP, DLEP and PPPoE in an open source routing software suite called Quagga [2], [11]. The authors believe that, due to the experimental implementations of the radio-router protocols used in their tests, their result should not be seen as a performance evaluation. Instead, it should be regarded as a proof of concept indicating that it is possible to increase end-to-end availability using a radio-router protocol.

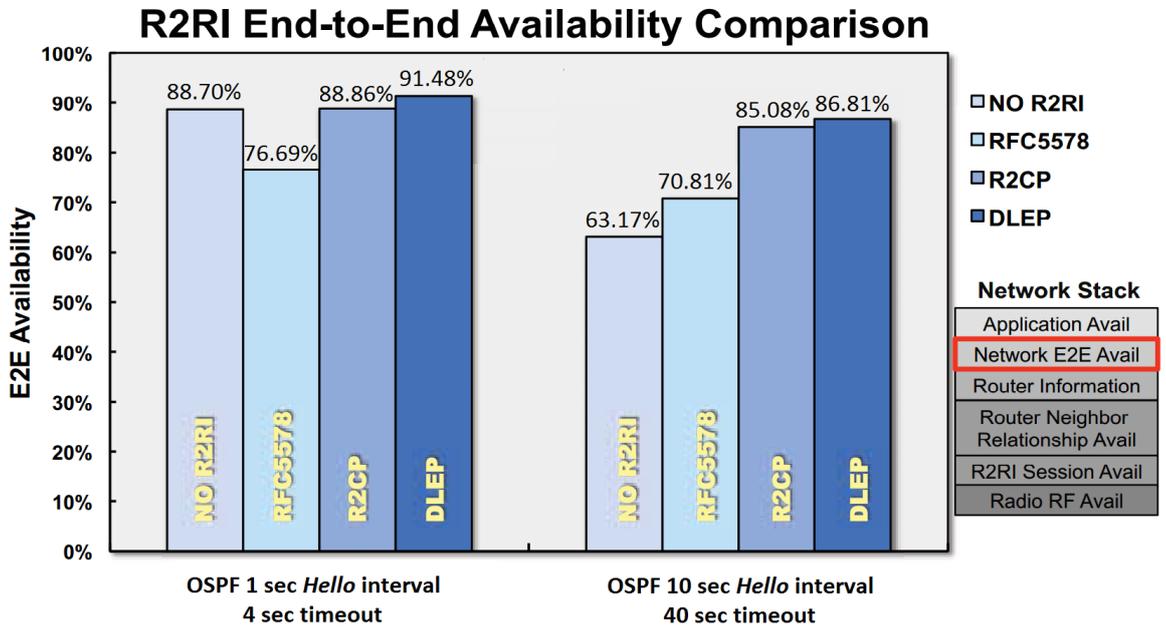


Figure 5: End-to-end availability measured by using ICMP pings [2].

Figure 5 shows the end-to-end availability measured by sending all-to-all Internet Control Message Protocol (ICMP) ping messages with a one second interval. ICMP ping messages are a simple tool that can be used to test the reachability to a remote host as well as to record the round-trip time. In their experiment they try three different radio-router protocols with two different settings for the OSPF routing protocol.

It is clear by looking at Figure 5 that there are performance gains to be made by using a radio-router protocol, especially in the case where OSPF *hello* and *dead* timers were set to 10 and 40 seconds respectively, which are very relaxed timers. It is often desired to keep over the air overhead down to a minimum when there are low bandwidth radio-links in the network. The high interval timers reduce the amount of overhead generated by the protocol. A raise of 24 percentage points in end-to-end availability can be seen when using OSPF (10s/40s, *hello* and *dead* timers respectively) and DLEP compared to not using any radio-router protocol.

When the OSPF timers are lowered to 1s/4s, the gain in end-to-end availability is drastically decreased. This can be explained by the OSPF protocol reacting more quickly to the network and thus the use of the radio-router protocol diminishes.

2.2.2 Quality of Service

In addition to performing spatial routing, it is useful to have link quality information available at the router in order to perform temporal control. Radio links are often intermittent and their bandwidths tend to vary. By providing information such as the current data rate to the router, it is possible to shape the data output stream in order to prevent overrunning the radio with data. It is also possible to have different kinds of traffic

sent at different rates. These rates can be set dynamically according to the current link state in order to avoid starvation [2].

Some traffic, voice communication for example, requires a minimum amount of bandwidth in order to provide a high enough quality of service. By knowing the current link capacity it is possible for the router to deduce how much bandwidth any given traffic flow will have access to, according to some prioritization scheme. The router can then decide to completely block the traffic if the currently available bandwidth is not sufficient enough to uphold the quality needed. This will free up resources to other application traffic being routed through the router.

2.3 Discussion

It is clear that there is no well established standard for radio to router communication. Nevertheless, in recent years there has been an increased interest in arriving at one. This study has identified the three most prominent protocols: PPPoE, R2CP and DLEP.

Due to the lack of stable specifications of both R2CP and DLEP there are not many open source implementations available. Many radio manufacturers do not have support for any of the three mentioned protocols. Rather than relying upon pending standard specifications, radio manufacturers implement their own, proprietary protocols. In the case of supporting DLEP, there has been a lot of changes to the specification since its initial release making it difficult to stay compliant to the specification. One possible solution to deal with all the proprietary interfaces is to use a proxy server in front of the radio device which can communicate with the radio using the radio's non-standard interface and then transform and provide link state information according to some open radio-router protocol. This has been successfully done in several experiments and might speed up the process towards a working standard since it makes it possible to use legacy devices [2], [12].

It is clear that by using a radio-router protocol the convergence time of a network running OSPF can be reduced [2], [3], [12]. The link-layer feedback allows routers to more quickly respond to the changes in a dynamic network. Out of the three protocols covered in this literature review, DLEP seems to be the most promising. PPPoE has become quite dated and has some design traits making it inconvenient to use when the underlying network has broadcast support.

Although R2CP is a great improvement over PPPoE, we find no reason as to why the protocol so strictly specifies how to utilize the link information that the radio provides. This creates an unnecessary coupling between R2CP and OSPF. We would rather have R2CP functioning just as DLEP, as a pure information interface between the radio and router. The development of R2CP seems to have been stalled as the latest published RFC by The Internet Engineering Task Force (IETF) has reached an informational status and is now expired since September, 2011.

DLEP also has its disadvantages. It is still not a well defined standard and its specification is changing quite rapidly, but it seems that there have been several lessons learned over the last few years which has turned DLEP into a better protocol than its predecessors.

3 Capabilities of Dynamic Link Exchange Protocol

Version 5 of the DLEP specification draft [6] mentions 16 protocol messages. These messages are sent between the radio and the router and may contain none or several out of the 22 different types of data items, depending on the type of message. Some messages have a mandatory set of data items that must be supplied with each message while some data items are optional and might not be supported by all DLEP implementations.

More than half of the DLEP messages (number 1-8 and 14 in Table 1) are used for initializing and maintaining the DLEP session between the client and server. The *peer discovery* and *peer offer* messages allow for a discovery mechanism to be used which reduces the amount of a-priori configuration necessary in order to setup a session. The *peer initialization* packet is used to initialize a session and to declare which data items the radio supports as this can differ between different devices. For example, not all devices are able to report the current data rate. Table 1 and Table 2 present a complete listing of the messages and data items that DLEP consists of.

Table 1: DLEP Messages

Number	Message name	Description
1	Peer Discovery	Sent by radios to a broadcast address in order to find any local routers to connect to.
2	Peer Offer	Sent by routers in response to a <i>peer discovery</i> message.
3	Peer Initialization	Message used to initialize a session between a client and a server.
4	Peer Initialization ACK	
5	Peer Update	This message can be used to indicate bulk changes that applies to all remote destinations.
6	Peer Update ACK	
7	Peer Termination	This message is used to terminate a session.
8	Peer Termination ACK	
9	Destination Up	This message is used to indicate that a new remote destination is available.
10	Destination Up ACK	
11	Destination Down	This message is used to indicate that a destination is no longer available.
12	Destination Down ACK	
13	Destination Update	This message is sent to inform the receiver of changes in the information base of some remote destination.
14	Heartbeat	Heartbeat message used to keep the session alive.
15	Link Characteristics Request	Used to request a change of a link to some remote destination, e.g. request a new waveform.
16	Link Characteristics ACK	

Table 2: DLEP Data Items

Number	Data item	Description
1	DLEP Version	Specifies DLEP version number.
2	DLEP Port	Mandatory data item in <i>peer offer</i> message. Used to specify which port the DLEP server is listening on.
3	Peer Type	An optional data item which can provide a string of additional information about a DLEP peer, e.g. 'Satellite Radio 1'
4	MAC Address	MAC address to remote destination.
5	IPv4 Address	IPv4 address to remote destination.
6	IPv6 Address	IPv6 address to remote destination.
7	Maximum Data Rate (Receive)	
8	Maximum Data Rate (Transmit)	
9	Current Data Rate (Receive)	
10	Current Data Rate (Transmit)	
11	Expected Forwarding Time	Optional data item used to indicate the typical latency between the arrival of a packet at the transmitting device and the reception of the packet at the other end of the link.
12	Latency	Used to indicate the amount of latency on a link.
13	Resources (Receive)	Optional data item used to indicate how much (in percentage) of a resource (e.g. battery power) that is devoted to receive data.
14	Resources (Transmit)	Optional data item used to indicate how much (in percentage) of a resource (e.g. battery power) that is devoted to transmit data.
15	Relative Link Quality (Receive)	A value in the range 0-100 used to indicate the relative link quality for inbound traffic.
16	Relative Link Quality (Transmit)	A value in the range 0-100 used to indicate the relative link quality for outbound traffic.
17	Status	The status data item is sent as part of an acknowledgment message, indicating the status of the request.
18	Heartbeat Interval	This data item is used to specify at what interval the heartbeat messages will be sent when a session has been initialized.
19	Link Characteristics ACK Timer	Optional data item sent within a <i>peer initialization</i> message. It is used to indicate how long to wait for a link characteristics request.
20	Credit Window Status	Optional data item used by any peer to indicate the latest credit window status.
21	Credit Grant Request	Optional data item used to indicate the amount incremented to the current credit window by the sender.
22	Credit Request	Optional data item used to request an increment of a credit window.

In the following sections we will present a more detailed explanation of the messages that we have utilized to implement the feedback mechanism used in our experiments. This is followed by detailed explanations of the feedback mechanisms themselves.

3.1 Destination Update Message

The *destination update* message is sent when a radio detects a change in one of its radio links. For example the latency of a link or its capacity might have changed. The specification does not specify how often these messages are to be sent or whether a heuristic for dampening the frequency of the update messages should be used. This is left up to the implementations to decide. In our DLEP implementation a *destination update* message is generated as soon as the radio link emulator changes any property of an emulated link. No heuristic is needed since we are in full control of the link as it is emulated. All the data items that are listed in Table 3 can be added to the *destination update* message.

Table 3: Destination Update Data Items

Data Item	Mandatory
MAC Address	Yes
IPv4 Address	No
IPv6 Address	No
Maximum Data Rate (Receive)	No
Maximum Data Rate (Transmit)	No
Current Data Rate (Receive)	No
Current Data Rate (Transmit)	No
Latency	No
Resources (Receive)	No
Resources (Transmit)	No
Relative Link Quality (Receive)	No
Relative Link Quality (Transmit)	No
Credit Window Status	No
Credit Grant Request	No
Credit Request	No

It is important to note that the *destination update* message has the ability to carry IP addresses. This makes it possible for radio devices to have several connections with different link properties which can be differentiated by the IP address to the remote destination. This is a common use case since radio devices typically are able to reach more than one remote radio device, and the latencies and bandwidth are not likely to be the same for each radio link.

DLEP has support for a credit mechanism which allows each participant to limit the incoming rate of traffic. This is implemented by the following three data items that can be added to a *destination update* message: *credit window status*, *credit grant request* and *credit request*. One of the strengths of having each server and client pair form a session is that it allows the DLEP participants to have different credit windows for each session. This makes it possible to evenly distribute the radio's resources to the connected routers. It would also be possible to perform a per

client quality of service by using these credit windows. This can be done by assigning a priority to each credit window and making the radio always dequeue the highest prioritized, non-empty queue first. Having each radio and router pair form a session also induces some drawbacks. Sessions increase the number of messages that the protocol must support and increases the amount of traffic between each pair forming a session. Furthermore, handling session initialization, timeouts and storing the state of each session are error prone tasks to implement correctly.

3.2 Destination Up and Destination Down Messages

A *destination up* message is sent by any DLEP participant to indicate that a new destination is reachable from the sender. *Destination up* messages can be sent both by radio devices and routers. When the message is sent by a radio, it is typically to inform the router that a new radio link has been established to a new remote radio device, whereas messages originating from a router are used to announce that some new logical destination is reachable from the router, e.g. a multicast group. The *destination down* message is used to indicate that a destination is no longer reachable.

Table 4 shows a list of all the data items that can be sent with a *destination up* message. It is mandatory to add all data items that the new destination has support for to the initial *destination up* message. This is to inform the receiver what information is to be expected in future destination updates and to set default values.

Table 4: Destination Up Data Items

Data Item	Mandatory
MAC Address	Yes
IPv4 Address	No
IPv6 Address	No
Maximum Data Rate (Receive)	No
Maximum Data Rate (Transmit)	No
Current Data Rate (Receive)	No
Current Data Rate (Transmit)	No
Latency	No
Resources (Receive)	No
Resources (Transmit)	No
Relative Link Quality (Receive)	No
Relative Link Quality (Transmit)	No
Credit Window Status	No

3.3 Feedback Mechanisms

There are several different feedback mechanisms that can be implemented when using the information provided by DLEP. In the following subsections we describe the mechanisms we have implemented and used in our experiments. We also describe some other mechanisms in Chapter 7.

It is worth mentioning that our DLEP server is running in a router, but this does not have to be the case. It would be possible to utilize DLEP information in client computers as well, either by forwarding the information from the router to a client computer, or by giving a client direct access to a radio device. Utilizing the DLEP information in client computers would enable more fine-grained, per application optimizations. For example, it would be possible for each application to have the radio feedback affect its output rate, choice of codecs and so forth. One drawback is that each application would require it's own system to handle the feedback. By having the feedback mechanisms in the router you get less control, but it will be possible to affect the network even when using applications that does not have any support for DLEP. The router will consume the radio feedback and transparently, from the applications point of view, perform some optimizations according to the configuration of the router. In this thesis we only study the feedback mechanisms which are of interest when the DLEP server is in a router connected to a radio.

3.3.1 Dynamic Traffic Shaping

One key feature of any packet switched network is the use of buffers. Any packet that is to be transmitted over a busy link can be queued in the outgoing packet buffer instead of being dropped. This drastically reduces the packet loss and throughput of a network, but the use of buffers can also reduce the network performance, in terms of latency. The more packets that are queued in a buffer, the higher the latency will be and eventually interactivity will be lost. It is important to understand that different scenarios might have different needs. For example, applications with low latency requirements, such as voice communication, might suffer from a high latency due to a large buffer even though the overall throughput might be increased [13].

The ideal use of a buffer is to have it not full nor empty. This means that as soon as the link associated with a buffer is free there should be packets waiting to be transmitted. When the link is busy, we can queue up the packets in a buffer instead of dropping them. In this way, we utilize the link as much as possible, not having to wait for new packets to be sent when the line becomes free, but still provide a low latency and packet loss since we are able to buffer up more packets. In order to have such scenario, we must adjust the rate of outgoing packets to conform with the rate of handling incoming traffic; otherwise we will eventually overflow the buffer and packets will be dropped. Typically, this is done either in the application layer or the transport layer. For example, TCP has a well proven mechanism for regulating the data rate. The way these algorithms usually work is that as soon as they start to notice that an increased number of packets are being dropped, they decrease the outgoing rate and then slowly increase it again in order to find an equilibrium [14]. Under dynamic conditions this becomes more difficult since the convergence process has to restart every time the rate changes.

By using the feedback from the radio, we can shape the outgoing traffic from the router so that it matches the rate reported by the radio. This will prevent the router from overrunning the radio with more traffic than the radio can handle; consequently, the latency will be kept to a minimum and packets will not be dropped by the radio. By knowing the maximum link capacity we are also able to distribute the capacity evenly between different types of traffic. This is possible to do without

knowing what the current maximum data rate is, but only by having different buffers for each type of traffic and then by using, for example, a round-robin algorithm to decide from which buffer to dequeue a packet. However, given the maximum current link capacity we are able to deduce how much bandwidth will be allocated to each type of traffic. Certain types of traffic, audio and video streams for example, require a specific amount of bandwidth in order to provide a required quality of service. This means the router can make well-informed decisions and completely cut off certain traffic routed towards the radio, if the capacity is not high enough for the type of traffic in question. This frees up the radio's resources to other applications.

Our DLEP server implementation has support for the described traffic prioritization, with the capability to specify a cutoff rate. Figure 6 shows part of a configuration file used by the dynamic traffic shaper.

```
# Format:
# <port>      <capacity in %> <cutoff rate (bit/s)> <priority>
501           30      96000                3
502           30      96000                2
503           20      32000                1
```

Figure 6: Excerpt of a dynamic shaper configuration file.

With this configuration file we define three different types of traffic that are running on ports 501, 502 and 503. Each type of traffic has a reserved percentage of the current available bandwidth and has a cutoff limit. As can be seen from Figure 6, traffic using port 501 has been reserved 30 percent of the bandwidth. This means that as soon as 30 percent of the total available bandwidth is less than 96000bit/s, which is the cutoff rate, the flow with the lowest priority will be cut off and the bandwidth it was using will be evenly distributed amongst the rest of the applications. This enables us to utilize a dynamic link with as much useful bandwidth as possible, rather than just maximize the transmitted bits per second. We have implemented this system by using the traffic control (tc) subsystem, which is part of the Linux kernel and iptables. The graph in Figure 7 shows how the different components of tc are interconnected. At the top there is a *hierarchy token bucket* (htb), a queueing discipline which allows to split up the outgoing traffic into different classes. Each class has the ability to control the rate of outgoing data. Filters are used to assign each packet to a class, based on the port number.

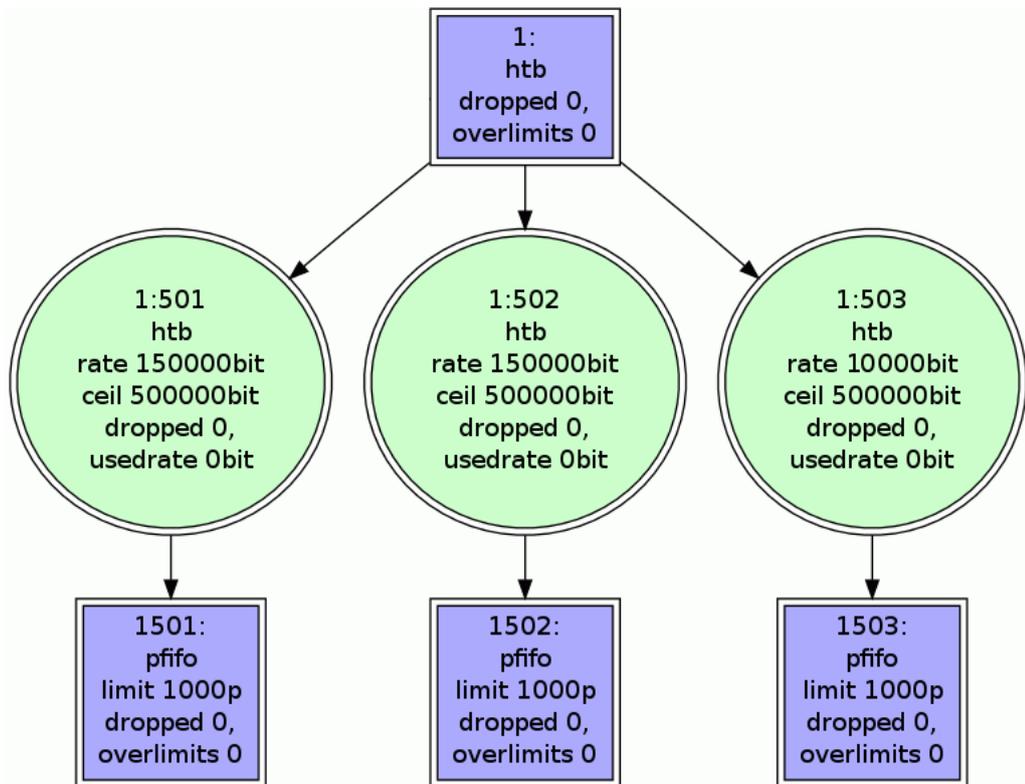


Figure 7: An example of a *traffic control* setup.

In Figure 7 there are three different classes, presented as circles. Inside each circle, we can see the class handle followed by the *rate*, which is the reserved data rate as well as the class' *ceil* rate. The *ceil* rate is the maximum outgoing data rate. Having both a *rate* and a *ceil* assigned to each class allows different classes to borrow bandwidth from each other when there is a superfluous of bandwidth. There is also some debugging statistics inside of each circle, such as the amount of dropped packets due to the queue being full.

If a service needs to be completely blocked, due to the lack of available bandwidth, it is done by using *iptables*. This is more efficient than blocking the service using *tc* as it will cause less processing in the Linux kernel. Following each class there is another queueing discipline, a *pfifo*. *pfifo* is a simple first-in first-out queue. This is the last stage of the traffic control system before each packet is given to the network interface card.

Please see Section 4.1.7 and Section 4.1.5 for more information about *tc* and *iptables* respectively.

3.3.2 Reactive Routing

Any network comprised of several smaller network needs a routing protocol. The routing protocol makes it possible to route traffic between any pair of nodes in the internetwork. A routing protocol typically consists of three major components:

- A system to detect other nodes in the network.

- A way of distributing up-to-date topology information.
- A path finding algorithm.

In order for a router to function, it must first find its neighbouring routers. In most routing protocols this is done by sending out either a broadcast or a multicast packet to some predetermined address. By convention, these are typically called *hello* messages. All routers listening for *hello* messages and are in range, will be able to respond with a unicast message back to where the original message came from or reply with a second broadcast; this differs amongst different routing protocols. When two routers have successfully identified each other, they can begin to communicate, e.g. share topology information and so forth. In the same way as most routing protocol detects new neighbours, they also detect lost neighbours. That is, when a router has not received a *hello* message from a neighbour in a certain amount of time, the connection to that particular neighbour is considered to be lost. This technique of relying on a timer and constantly sending out *hello* messages has its drawbacks. It is difficult to know how long to wait before declaring a link as lost and how often to send out *hello* messages. The more often we send out *hello* messages, the faster the network will converge, but the routing protocol will have a higher overhead.

Traditionally, computer networks were a lot more static as compared to today's networks. New connections between networks were not installed very often and using a low frequency of *hello* messages was more than enough to automatically configure the network, only sacrificing a relatively small amount of network overhead. The military network scenario is different. Network nodes are typically mobile, and connections between different nodes can vary as soldiers move around. In order to have a network with a high end-to-end availability in the dynamic settings of a military network, one is forced to have aggressive timeout timers and a high frequency of *hello* messages. This is problematic since military networks also typically have a lower average bandwidth and are less reliable than static networks, due to using more wireless links.

One way to improve the situation would be to let the routing protocol utilize feedback from the link layer. If a DLEP server is running on the router, it can give indications to the routing protocol whether a remote radio connection has been established or lost. The routing protocol can then take further actions to initialize communication with a new router by sending out *hello* messages immediately, without waiting for the timer to expire. It would also be possible for the router to immediately remove dead links from its routing table. This would allow more relaxed *hello* timers, yet keeping a low convergence time.

We have implemented a plugin to the *olsrd* [15] implementation of the OLSR specification. This plugin is capable of receiving two commands: *destination up* and *destination down*. When a *destination up* command is received it will begin to send out three *hello* messages within a 1.5 second interval in order to speed up the process of creating a neighbourhood with the newly detected remote router. When a *destination down* command is received, OLSR's local link database is updated and the lost link is removed.

4 Testing Environment

This chapter provides an overview of the different software tools and hardware that were used in the experiments performed in this thesis.

4.1 Software

4.1.1 VirtualBox

VirtualBox [16] is a general-purpose, full virtualizer for x86 hardware. The source code to VirtualBox is freely available under GPL license. VirtualBox has been used extensively during this project to simulate the different network nodes in the test environment. Both routers, radio link emulators and network clients have all been emulated using VirtualBox.

It is possible to create private virtual networks between a set of emulated machines using VirtualBox. This is done through VirtualBox's interface by supplying the network interface with a specific network handle. Each network interface with the same network handle will then be treated as being on the same virtual local network. This feature makes VirtualBox very convenient for setting up testing environments as it greatly reduces the number of computers needed, and it is much easier to manage the entire setup from one computer.

It is worth mentioning that emulating machines, for example by using VirtualBox, rather than using discrete computers, reduces performance. However, for this thesis this is not an issue since the computations we are doing are well within the limits of what our virtual machines are capable of.

4.1.2 Radio Link Emulator

Saab has developed a radio link emulator (RLE). With the RLE it is possible to emulate real world network conditions such as packet loss and latency in a controlled and repeatable manner. The RLE is written in Python and can run in several Linux environments, as well as in an emulated machine using VirtualBox. The RLE works by using two sets of Linux tools, *ebtables* [17] and *tc* [18]. *ebtables* is a network filtering tool which makes it possible to analyze and mark certain packets that we want to associate for a specific emulated network link. In addition to marking packets, we can also route them by using *ebtables*. This allows the RLE to couple two network interfaces and let traffic flow through them.

By using *tc*, which is a part of the Linux kernel, we can then emulate packet loss and add latency to outgoing packets by looking at their corresponding mark which we created using *ebtables*. Please refer to Section 4.1.7 for more information on *tc*.

4.1.3 Optimized Link State Routing

Optimized Link State Routing (*OLSR*) is a routing protocol optimized for large and dense networks [19]. It is a proactive routing protocol, which means that each node maintains a local copy of a topology database. When a node wants to send data, it can use a path finding algorithm and the information in the topology database in order to find the shortest route. The way that OLSR distributes routing information differs from most other link-state routing protocols. Typically, this is done by flooding

the information, broadcasting it out on all links and making it propagate through the network. This technique generates a lot of overhead in dense networks since nodes get flooded with the same data several times. To alleviate this, OLSR has a concept called *multipoint relays* (MPR). Nodes that are elected as MPRs are the only nodes who get to retransmit flooded topology information. Such protocol operation will greatly reduce the number of retransmissions that would occur in dense networks.

OLSR is using *hello* messages in order for routers to find any one-hop neighbours. By the exchange of *hello* messages two routers can form a neighbourhood relationship. By using the flooding mechanism of OLSR, each router can spread information about its links throughout the network using *topology control* (TC) messages.

We used an implementation called *olsrd* [15], and its source code is freely available under BSD-license. In all our test we used the version 0.6.6.2. *olsrd* has a framework for creating plugins to the routing engine. This was used extensively when developing our reactive routing mechanism, please see Section 3.3.2 for more details.

4.1.4 iperf

There is a simple utility program called *iperf* which can be used to generate a stream of data traffic. It can act both as a sender and a receiver, and record information about sent and received data, such as: data rate, packet loss and jitter. We used iperf version 2.0.5 extensively in our experiments, both to send and receive data.

4.1.5 iptables

The Linux kernel comes with a built in firewall. *iptables* is a user-space program which allows configuration of this built in firewall. We used *iptables* version 1.4.12 in order to effectively implement the dynamic traffic shaper described in Section 3.3.1. *iptables* allows us to drop packets at an early stage as they enter the Linux kernel, based on properties of the IP header, such as port number.

4.1.6 tcpdump

By using a utility program called *tcpdump* we were able to inspect network traffic on one or several network interfaces. *tcpdump* has support for advanced packet filtering on traffic direction, i.e. inbound or outbound traffic, protocol type, fields in the IP header and so forth. *tcpdump* has the ability to store the traffic logs in a binary format called *pcap*. *pcap* files can then be analyzed by several other tools in order to get statistics such as average delay, number of dropped packets and so on. We have used *tcpdump* version 4.5.1 extensively during our experiment execution in order to record network activity. It has also been used as a tool for inspecting network traffic whilst developing our DLEP implementation.

4.1.7 Linux Traffic Control

Modern versions of Linux comes with a system for managing the rate of outgoing, and to some extent incoming network traffic. This system is called *traffic control* and it is part of the Linux kernel, operating in between the IP-layer of the network stack and the network interface device

driver. One of the tasks of the traffic control subsystem is to provide the network interface driver with IP packets to transmit on the interface [4]. By applying different filters, which are part of the traffic control subsystem, we can place packets in different output queues. These filters are configurable and can differentiate packets by values in the IP header and to some extent fields in the transport protocol header, e.g. TCP and UDP port numbers. By utilizing different scheduling schemes we are then able to prioritize certain types of traffic as well as to shape the rate of outgoing data. There is a user-space command called *tc* which can be used to interact with the traffic control subsystem and control how these different queues work. Figure 8 shows how the traffic control subsystem integrates with other parts of the kernel and how we can use the *tc* user space program to interact with it.

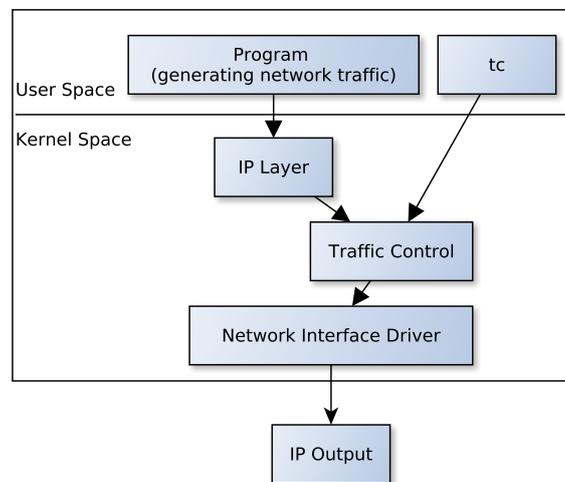


Figure 8: Interaction of the traffic control subsystem with the IP stack [4].

The Linux kernel has support for a wide variety of different scheduling algorithms. The default scheduling algorithm that is used for any network device is a three-banded first-in-first-out (*FIFO*) queue. This queueing discipline internally consists of three different queues with different priority, which means that a low priority band will not be dequeued until all bands with a higher priority become empty. The type-of-service field in the IP header is inspected on incoming packets and this value decides on which of the three internal queues the packets should be placed in.

4.1.8 Dynamic Link Exchange Protocol

The implementation of the first draft of DLEP is available under MIT license. The source code is available at [20] and it is written in C. It comes coupled with a set of simulation tools for router and radio devices. The draft which the implementation is based upon dates back to November of 2010 and the protocol has gone through some major changes. As of writing this, the most recent draft is now version 5. The code has not been updated since 2012 and the project no longer seems to be actively

maintained. Proprietary implementations are also available. For example, Cisco IOS [21] has support for DLEP.

In order to be able to run tests in a repeatable manner, part of this project has been to design, implement and integrate the latest draft of DLEP (draft 5) in the toolset used at Saab for emulating radio links and network environments. Since the specification is a draft and not yet a well defined standard, there are some ambiguities in the specification which forced us to make certain decisions, which might lead to our implementation not being compatible with other DLEP implementations. This is unfortunate, but necessary given the state of the specification.

DLEP uses a client-server model. A client is the program that initiates a DLEP session between two peers, typically a radio and a router. The client resides in the radio and the server in the router. The specification states that all DLEP messages should be sent using TCP as the underlying transport protocol as TCP provides reliability and the link connecting the router and radio is most likely of a high quality where the overhead caused by TCP is of little concern.

There are 16 different DLEP messages. They are formatted in a nested type-length-value style. The outer part encodes what sort of message it is, and the message's value is composed of a set of either optional or mandatory data items, depending on which message is being sent.

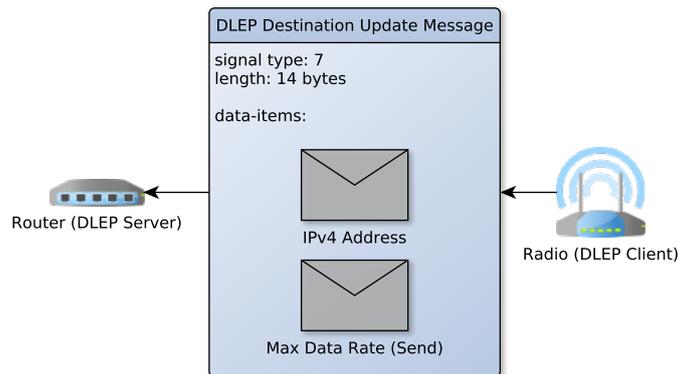


Figure 9: A DLEP message with nested data items.

Figure 9 shows a *destination update* message being sent from the radio to the router. The message has two nested data items: an IPv4 address and a maximum data rate value, which are both in turn type-length-values.

4.1.8.1 DLEP Server

The DLEP server is running as a daemon, typically in routers, that are connected to a radio device. The task of the DLEP server is to receive and utilize the provided link state information.

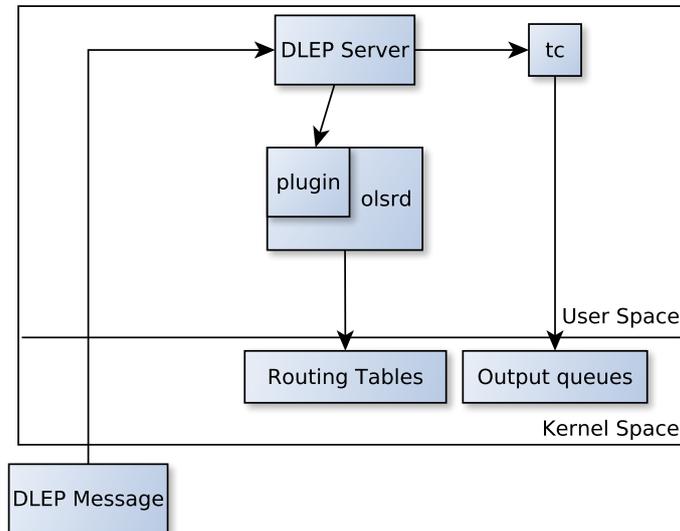


Figure 10: An overview of the DLEP server interaction with different subsystems.

Since DLEP does not specify how the link state information should be utilized and which actions it should make in order to increase network performance, it is desirable to have a DLEP server that can be easily modified to employ the information provided by DLEP in any appropriate manner. In Figure 10 we can see different modules that our DLEP server can affect directly or indirectly.

First, it can have an indirect effect on the Linux kernel routing table. It does so by sending commands to a plugin inside *olsrd*. This plugin knows of two commands, *destination up* and *destination down*. The plugin will interact with the rest of the *olsrd* system and make *olsrd* initialize a session with newly available remote destinations and remove lost links from its internal database. This mechanism of altering the routing table as a response to certain DLEP events is what enables us to perform dynamic spatial routing.

Secondly, it might also be the case that the DLEP server performs traffic shaping by issuing a command to *tc*. The DLEP server is designed in such a way that it is easy to extend it with new functionality. There is no hard coupling between the DLEP server and *tc*, nor with *olsrd*. For example, it would be an easy task to add support for OSPF or some other routing protocol. This makes it a good starting point for future work and further experiments.

4.1.8.2 DLEP Client

Since Saab has its own radio link emulator (RLE), the DLEP client had to be able to run in the same environment as the RLE. Some modification has been done to the RLE core in order to be able to provide the link status information needed by DLEP.

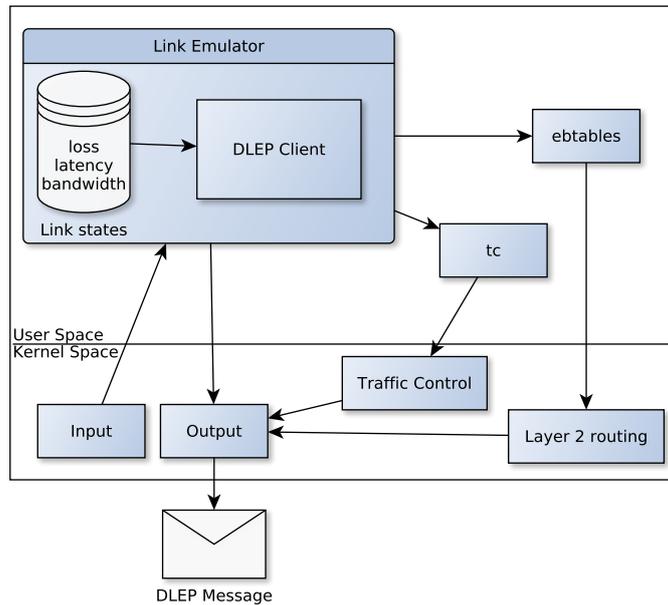


Figure 11: An overview of the radio link emulator and how DLEP is integrated.

The radio link emulator can mimic the characteristics that are typical for a radio link, such as high packet loss, latency and low bandwidth. These properties are configurable on a per bi-link basis (i.e. one configuration for each link direction) and are loaded at system startup. They can also be changed while the RLE is running in order to make experiments more realistic as radio links often fluctuate and have a very dynamic behavior as opposed to wired links. These configuration values are stored in an in-memory database, which contains a list of all currently active links, as can be seen in Figure 11. This database holds information such as current latency, packet loss and transmission rates. The DLEP client is able to access this information in order to construct DLEP messages which will be sent to the DLEP server.

4.2 Hardware

All the experiments has been executed using VirtualBox instances running on a discrete host PC. The host PC has 8GB or RAM, an Intel Core i7-3770 CPU running at 3.4GHz with four cores.

5 Experiments

In this section we present the experiments performed in this master thesis. First a general description of each scenario is presented, followed by a more detailed description of how the scenario was executed. For details on what software was used in the experiments, please see Chapter 4.

5.1 Scenario 1 - Quality of Service

5.1.1 Description

This scenario depicts a group of vehicles out on a mission. One of the vehicles (*Vehicle A* in Figure 12) has a radio connection back to headquarters. The radio link has varying bandwidth due to change of the link's transmission waveform and the effects of the surrounding environment. In addition to the connection back to headquarters, *Vehicle A* is connected to several other vehicles, forming a local network. This network configuration makes it possible for all vehicles to utilize the radio link to headquarters. Each vehicle is carrying a number of soldiers. Soldiers are carrying computers that are connected to a router in their corresponding vehicle.

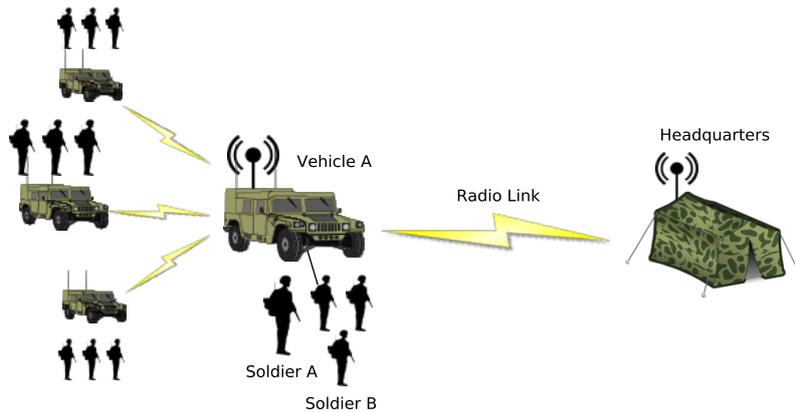


Figure 12: The topology of this scenario.

There are three types of applications utilizing the network. Some soldiers are running video, and some audio recording applications which are streaming real-time data back to headquarters. In the headquarters, this information can be utilized in making tactical and strategic decisions. Each vehicle has an on-board computer that is running a battle management system (BMS). The BMS is used for issuing commands and tracking the position of the soldiers and vehicles.

Given the sparse and varying bandwidth of the radio link back to headquarters, the router in *Vehicle A* has a pre-configured percentage of bandwidth allocated to each type of application. This allows the router to distribute the radio links bandwidth in order to provide better quality of service. Due to the radio link's dynamic bandwidth, it can be the case that an application's reserved bandwidth is insufficient to sustain

a desired level of quality. Therefore, each application has a cutoff rate. When the reserved bandwidth to an application is lower than the cutoff rate, the application's traffic gets blocked and stops being sent to the radio from *Vehicle A*. In addition to distributing the bandwidth of the radio link amongst the applications, there is a priority assigned to each application's traffic. An example of this would be if *Soldier A* was of higher rank than *Soldier B*. *Soldier A*'s traffic would then be assigned a higher priority, making *Soldier A*'s traffic less likely to be cutoff when bandwidth is sparse.

5.1.2 Execution

In our simulation we reduced the number of soldiers to two and the number of vehicles to one. This makes the experiment less time consuming to set up, yet provides interesting scenarios where applications are competing for the available bandwidth. Figure 13 presents an overview of our network topology. *Soldier A* and *Soldier B* are represented by *Client A* and *Client B* respectively. The computer and router inside *Vehicle A* are represented by *Client C* and *Router A* respectively. The headquarters is represented by *Server A*.

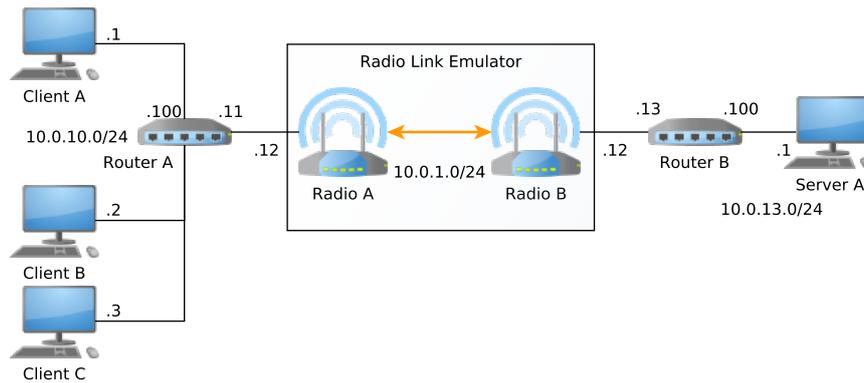


Figure 13: Network topology

To simulate the dynamic condition of the radio link we used an RLE. The emulated radio link has zero percent packet loss, a 14 ms round-trip time, and a bandwidth that is varying between 150 Kbps and 500 Kbps. The experiment runs for 60 s, changing bandwidth every 10 s. Under normal conditions the radio link is expected to have a bandwidth of 600 Kbps. Each application's traffic is simulated by generating a fixed rate of UDP packets using *iperf*. The data rates used for the generated traffic are approximations of how much bandwidth each type of application requires in order to provide a high quality of service. Each generated packet is 1470 bytes which makes it fit inside an IP packet. This choice of packet length makes counting dropped datagrams easier, since it prevents datagrams from being fragmented over several IP packets. The static data rate of the generated traffic might not reflect the characteristics of a typical audio or video stream under certain situations since these are known to have a varying rate. Each application's traffic also has a cutoff rate assigned to it.

The cutoff rates are approximated to represent the minimum acceptable quality of service. Figure 14 shows how the radio link’s bandwidth varies over time as well as the cutoff rate for each application. Table 5 presents details of the traffic being generated.

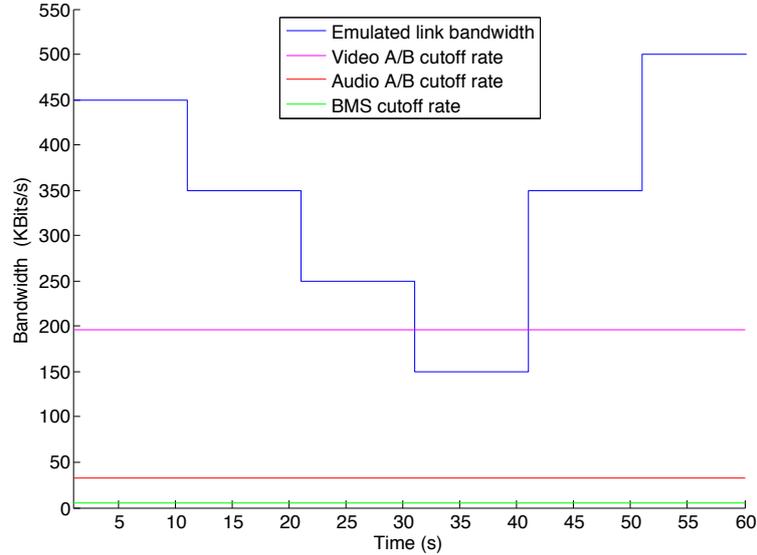


Figure 14: Emulated link bandwidth and stream cutoff rates.

Transmitter	Receiver	Port	Type of traffic	Data rate (Kbits/s)
Client A	Server A	501	Video Stream A	350
Client B	Server A	502	Video Stream B	350
Client A	Server A	503	Audio Stream A	96
Client B	Server A	504	Audio Stream B	96
Client C	Server A	505	BMS	10

Table 5: Traffic generated by the four applications.

Router A is running a DLEP server, which has support for the dynamic traffic shaper (see Section 3.3.1 for details), and is connected to the DLEP client in the RLE, which emulates the radio link. Figure 15 shows the configuration file used for the dynamic shaper that specifies cutoff rates and priority for each traffic, specified by its UDP port number.

```
# Format:
# <port>      <capacity in %> <cutoff rate (bit/s)> <priority>
501           40      196000                2
502           40      196000                1
503           20      32000                 4
504           20      32000                 3
505           100     5000                  5
```

Figure 15: The configuration file used for the dynamic traffic shaper.

The BMS traffic, running on port 505, is deemed mission critical, therefore it has the highest priority, followed by the audio streams and finally

the video streams. The BMS is allowed to use all available bandwidth in order to reach its cutoff limit if necessary. The video and audio streams are not as important and are limited to a bandwidth allocation of 40 and 20 percent respectively. Such allocation will suffice to provide a high quality of service given a bandwidth of 500 Kbps, which is the maximum capacity of the radio link in this simulation.

5.1.3 Results and Discussion

The scenario was executed twice. First we executed the scenario without DLEP and the traffic shaping system described in Section 3.3.1. The first execution will be referred to as the control execution. Then we activated the shaping mechanism and executed the same scenario a second time. This allows us to compare the results of the control execution with the results from the scenario execution using the traffic mechanism.

During the experiments we measured the rate of data received in Server A, for each type of traffic described in Table 5. Both traffic generation and reception was done using *iperf*. We also measured average packet jitter, which is the smoothed mean of differences between consecutive transit times, and the round-trip time between *Client C* and *Server A*. Due to the packet loss, several consecutive ping messages might be lost which makes it difficult to measure round-trip time since we will get gaps in our recorded data. In order to provide a more accurate round-trip time, we send 10 ping packets per second and calculate the average per second.

5.1.3.1 Without shaping mechanism

The following results were obtained in the control experiment execution when no dynamic shaping was activated.

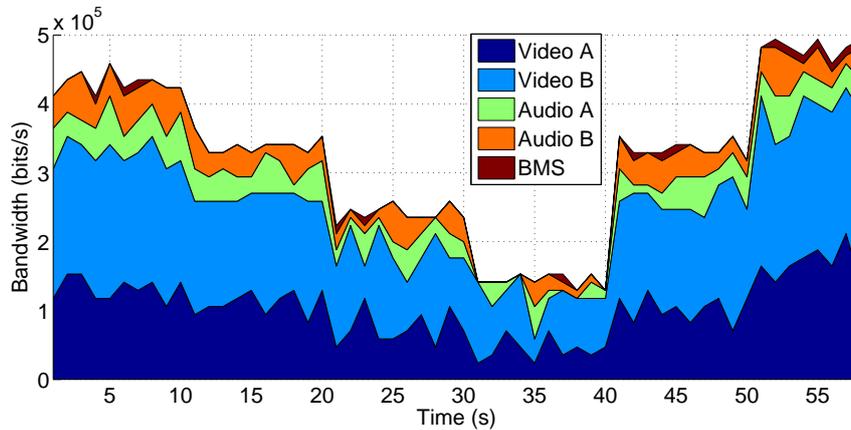


Figure 16: Radio link utilization. Without DLEP and traffic shaping.

In Figure 16 we can see the received data rate in *Server A*. Each application's traffic is indicated by a unique color. Since no traffic shaping nor prioritization is in place, the router forwards data at the speed it is generated, which is above the bandwidth of the radio link. As the input buffer in the radio gets full, packets begin to get dropped. This causes all applications' traffic to compete over the radio links bandwidth. It is

difficult to predict which application's packet is going to be queued next and which ones are going to be dropped. One interesting thing to note here is that there are several times when the BMS has a 100 percent packet loss. The BMS is mission critical and only uses a small share of the radio links bandwidth. Both audio streams are most of the time below their cutoff rate, taking up bandwidth but producing output that is discarded because of its quality being too low, essentially wasting bandwidth.

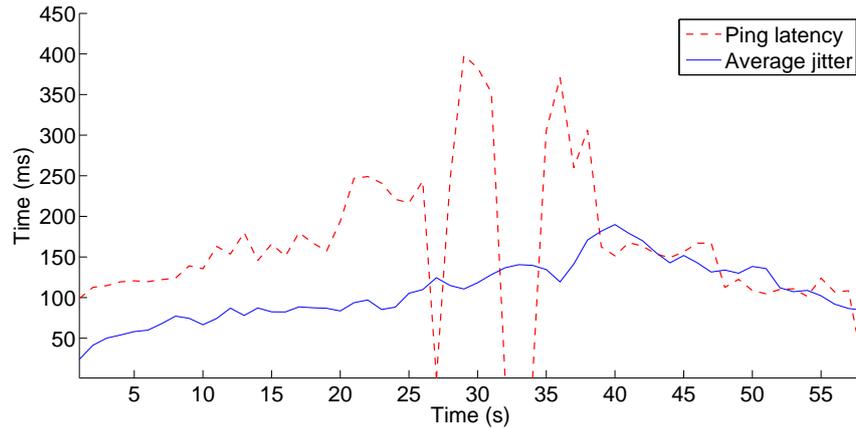


Figure 17: Latency between *Client A* and *Server A* and incoming packet jitter.

As the radio link's bandwidth is decreased, the end-to-end latency between *Client C* and *Server A* increases. This can be explained by the increased time that packets have to wait in output queue of the RLE. The RLE can queue up to five packets. Given that the size of each packet is 1470 bytes, that gives a total of 7350 bytes in the buffer that has to be dequeued before the ping packet can be transmitted. When the bandwidth of the radio link is 450 Kbps this will cause a 131 ms delay in the transmit buffer.

When the bandwidth is at its lowest, 150 Kbps, there are two periods when no ping messages get through, resulting in zero latency being reported. This can be explained by the packet loss, observable in Figure 18 below.

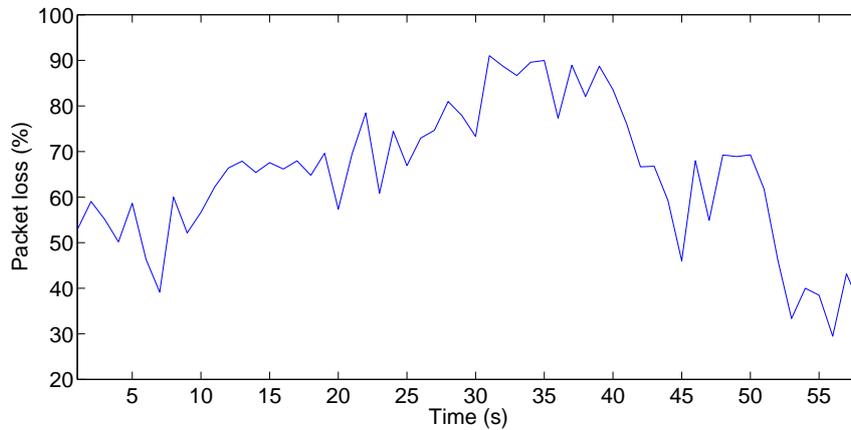


Figure 18: Amount of dropped packets versus time.

In Figure 18 we see the percentage of dropped UDP packets that were sent to *Server A*. In the interval from 30 to 40 s, when the bandwidth is 150 Kbps, the packet loss goes up to 91 percent. This explains the lack of reported round-trip time due to the loss of several consecutive ping packets.

5.1.3.2 With shaping mechanism

The following results were obtained using DLEP and the dynamic shaper. Refer to Figure 15 for the configuration file used for the dynamic shaper.

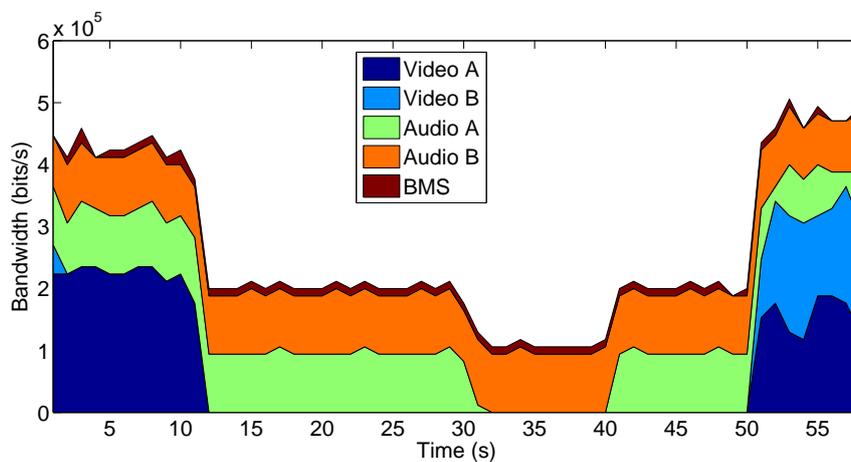


Figure 19: Radio link utilization. Configuration with DLEP and the dynamic traffic shaper.

In Figure 19 we can see how each type of traffic follows the rate specified in the dynamic shaper configuration. There is a small delay of approximately one second, at the beginning of the test, before the shaping mechanism is in action. A result of this is that Video B can send some bytes before it is blocked. Overall, it is clear that the throughput of

the radio link that will be discarded due to being of too low quality has been drastically reduced compared to the control results. Whenever a new data rate is reported, each application's traffic rate almost instantaneously adapts itself to the new conditions.

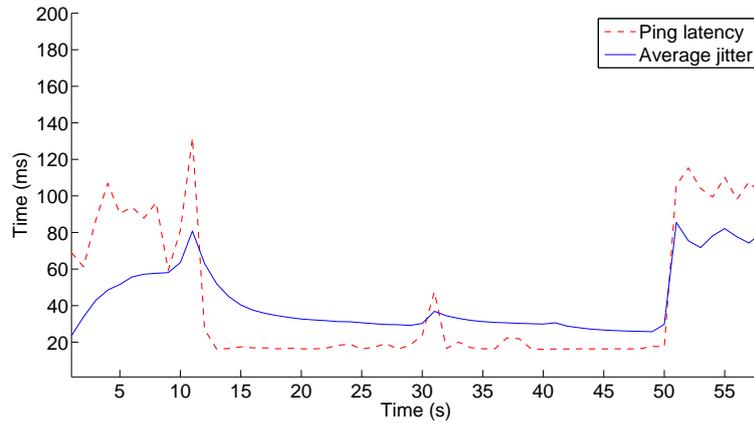


Figure 20: Latency between *Client A* and *Server A* and incoming packet jitter.

As can be seen in Figure 20 the round-trip time is much lower overall compared to the control data. The round-trip time of the first and last 10 second periods indicate that the buffer has been full during these periods. This is likely to be the case as 100 percent of the reported link bandwidth is distributed to the different types of traffic. A solution to this would be to under-subscribe the link with a few percentage rather than keeping the link at full load, which was found to be a poor design choice in our implementation. This would decrease latency at the cost of bandwidth.

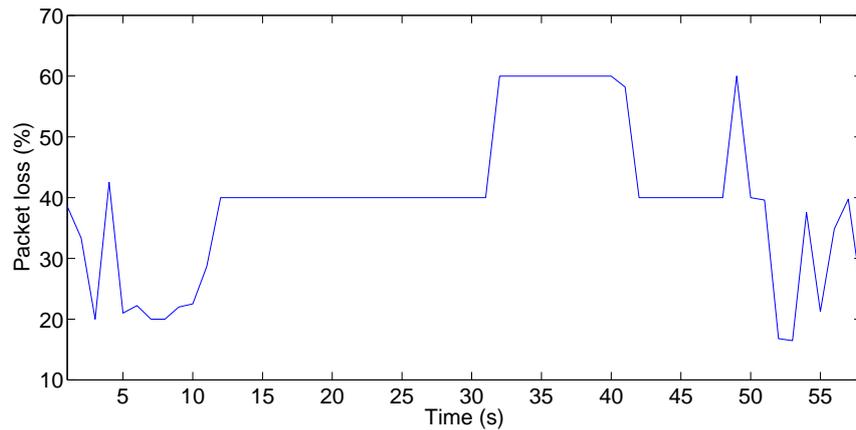


Figure 21: Amount of dropped packets.

In Figure 21 we can see how many packets that are lost during the experiment execution. The number of dropped packets now forms a more regular pattern than in the control data set. This behavior can be ex-

plained by packets being dropped by *iptables* rather than being dropped by a full buffer in the radio.

5.2 Scenario 2 - Dynamic Routing

5.2.1 Description

The second scenario depicts an armada of warships sailing open waters. On board each ship there is a local network connected to a router. Each router is connected to a radio device making it possible for all ships that are in range of each other to communicate. The router in each ship acts as a node in a mesh network, making it possible to route traffic through intermediate ships, if necessary, for a packet to reach its destination.

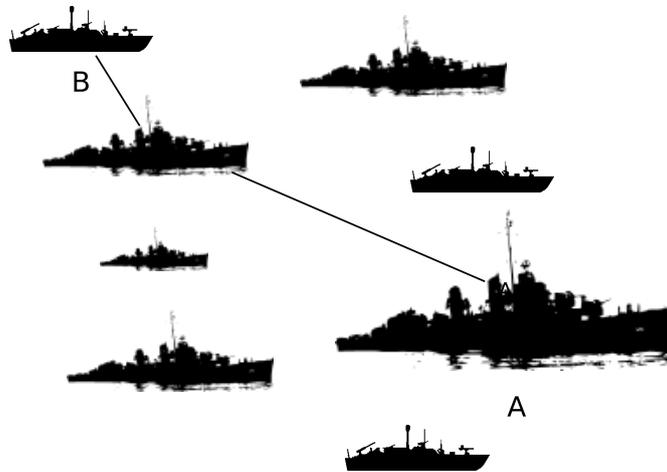


Figure 22: Showing one possible router from A to B

As the distance between different ships increases, certain connections will fail and others will be formed. The ships' radio devices are running DLEP clients capable of detecting link changes and will notify a DLEP server that is running in the router on each ship allowing the routers to quickly react to network changes.

5.2.2 Execution

We emulate this scenario by having five routers, *A* through *E*, running OLSR. Each router represents an individual ship and has an ethernet connection to the radio on its corresponding ship. All radio links are emulated by using an RLE. This provides us with the means to enable and disable links as ships go out of range of each other. Each router is running a DLEP client which has a reactive routing system in place (see Section 3.3.2 for details), which will notify OLSR that a layer two link, i.e. a radio connection, has been lost or established. In order to utilize this information, we have created a plugin to OLSR. This plugin will automatically generate three *hello* messages during a 1.5 s interval when notified by the DLEP server that a new radio link has been established. It

will also update OSLR’s internal link database and automatically remove links that are reported as lost by the radio.

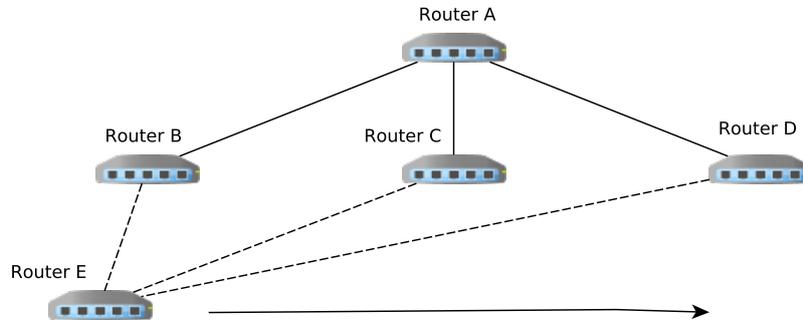


Figure 23: Network topology

The experiment runs for three minutes and Figure 23 shows the network topology. We denote each link with the routers it connects, that is, the link between router *A* and *B* is called link *AB*. In Figure 23 there are three links which are dashed. These are the links which will fluctuate during the experiment, in accordance to Table 6.

Table 6: Link status

Time (s)	Link name	Link transition
60	BE	Down
	CE	Up
120	CE	Down
	DE	Up

At the beginning of the experiment the network is converged and there is a connection established between *Router B* and *Router E*, that is, the link *BE* is up and a OLSR neighbourhood has been established. The reader can imagine *Router E* moving towards east, eventually losing contact with *Router B* and at the same time forming a new connection with *Router C*. Note that the old link will always be taken down before a new link is established. This is done in order to reduce complexity since we are focusing on the convergence time of the network.

During the experiment execution we measure two things: overhead generated by OLSR and end-to-end availability. All traffic generated by OLSR in each router is logged by *tcpdump*. End-to-end availability and latency is measured using ICMP ping packets. These ICMP packets is sent from *Router E* towards *Router A* during the entire experiment and the responses are recorded by *tcpdump* in *Router E*.

5.2.3 Results and Discussion

We run our experiment six times using three different timing configurations for the OLSR daemon. Each configuration is executed once without DLEP—this will be referred to as the control data—and once with DLEP and the reactive routing mechanism activated. The three configurations

can be seen in Table 7 below. For all other parameters than the ones which are listed in Table 7 the default values were used.

Table 7: OLSR configuration settings.

	Execution 1	Execution 2	Execution 3
HelloInterval	2.5 s	5 s	10 s
HelloValidity	5.0 s	10 s	20 s
TcInterval	2.5 s	5 s	10 s
TcValidity	5.0 s	10 s	20 s

The first experiment run had the most aggressive timings. As can be seen in Figure 24 the amount of OLSR data generated is close to 100 KB during the three minutes test execution.

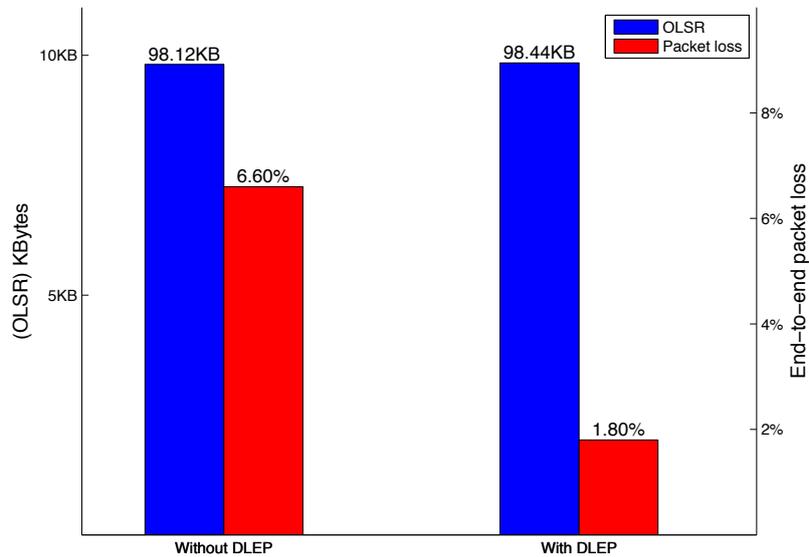


Figure 24: Overhead and packet loss using 2.5 and 5.0 seconds timings.

It is clear that using DLEP and the reactive routing system has had an effect on the end-to-end availability between *Router A* and *Router E*. The packet loss has been reduced by 4.8 percentage points at a cost of 320 bytes. The increase of OLSR overhead comes from the increased number of *hello* messages which are generated by each *destination up* message that the DLEP client sends to the respective router. Figure 24 indicates that these additional *hello* messages helped to improve the network convergence time, as packet loss is reduced.

The second experiment execution had less aggressive timings. As can be seen in Figure 25, the amount of OLSR traffic generated is halved. A lower OLSR overhead is to be expected due to the lower broadcasting frequencies compared to the first scenario execution. The low frequency also leads to a higher packet loss. Again, DLEP is increasing the end-to-

end availability between *Router A* and *Router E*. This time the relative increase of end-to-end availability is not as high as with the previous OLSR settings.

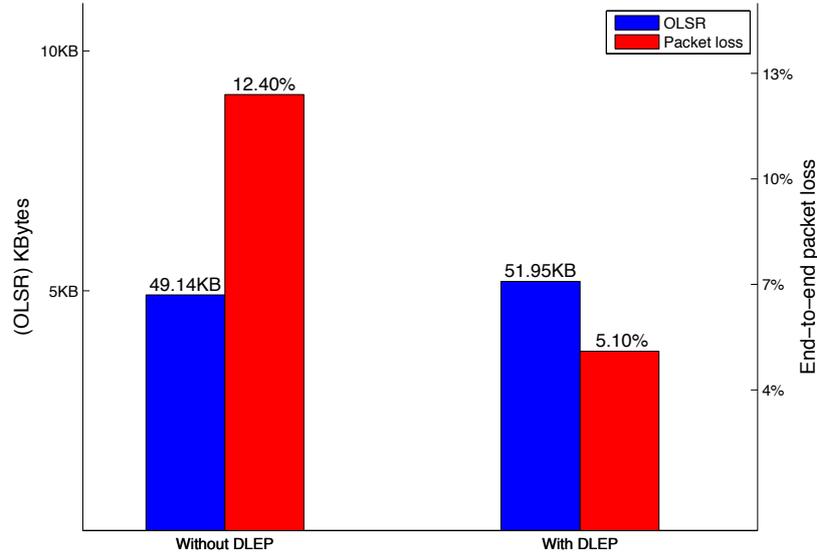


Figure 25: Overhead and packet loss using 5 and 10 seconds timings.

The last experiment execution had the least aggressive timings. Again, the OLSR generated overhead was halved and the packet loss increased, as can be seen in Figure 26. One might expect to always get twice as high packet loss when the timers are halved, which is not the case in this test execution. This is likely to have been caused by the difference in the amount of time passed after the experiment started until the first OLSR message is generated between the three experiment executions. This is difficult to control as OLSR has a built-in jitter mechanism in order to prevent multiple routers to flood the network at the exact same time.

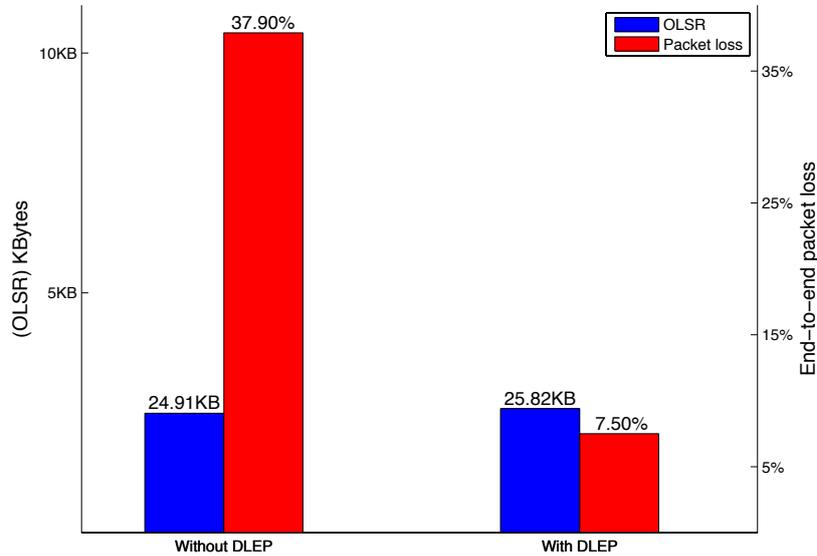


Figure 26: Overhead and packet loss using 10.0 and 20.0 seconds timings.

The last experiment execution shows the largest gain in end-to-end reachability, both in percentage points and relative to the control data. It is interesting to note that, even when using DLEP, the packet loss gradually seem to be increasing as the OLSR timers are increased. This indicates that the mechanism used in the OLSR plugin, which is triggered once a new link has been discovered, is insufficient to fully set up an OLSR neighbourhood and update the routing table. If the OLSR plugin alone would be able to fully establish an OLSR neighbourhood relation and make the new destination routable, the packet loss would be independent on the timer used for OLSR *hello* message interval, since any *hello* messages other than the ones caused by the OLSR plugin would be superfluous. However, this is not the case as the packet loss tends to increase, even when DLEP and the reactive routing mechanism is activated.

In our experiment we only used a small set of routers. It would be interesting to expand the network and increase the number of nodes, in order to see how the reactive networking plugin for OLSR scale with an increased network size. The changes in topology, caused when new links are added and old links are lost, will have to be spread throughout the network. It is likely that the OLSR overhead will grow with the number of nodes in the network. Initially, we planned on having a larger experiment setup, using 20 virtual machines. Unfortunately, we got very mixed results, most likely caused by hitting the limit for the number of virtual machines our hardware can manage. This led us to downscale the experiment. It would be possible to split the experiment setup over several machines to overcome this issue.

6 Discussion

In this thesis we have investigated how the use of link-state information can improve the quality of service in tactical networks as well as reducing the network convergence time. We have focused on DLEP and implemented and tested two ways of improving network performance. By conducting our first experiment, we have proven that it is possible to utilize current data rate and maximum data rate metric provided by a radio device, using DLEP, in order to implement a dynamic traffic shaper. We have also shown how the concept of assigning each type of traffic a cutoff rate can increase the quality of service during dynamic conditions.

Our second test focused on spatial routing. We have shown that by actively reacting on events in the network, e.g. new links reported by the radio, makes it possible to increase the end-to-end availability of the network. We observed that even when the underlying routing protocol used aggressive timers the reactive routing system in place increased the end-to-end availability with a low cost in terms of increased OLSR traffic.

It is worth mentioning that neither of the mechanisms we have implemented and tested do rely on any particular feature of DLEP. They could have been implemented using some of the other protocols mentioned in Section 2. DLEP is a rather complex protocol, with many messages, session handling and so forth, which are superfluous for the type of mechanisms presented in this thesis. It should be possible to achieve the same results with a more simple protocol.

It will be interesting to see what the development of radio-router protocols will lead to and if they will be extensively used in networks on the tactical edge. It is clear that link-status information can improve quality of service and decrease the converge time of a network. However, it might be the case that the protocols that exist today are not mature enough. In order to get more devices with support for open source radio-router protocols we need to see a continuation of the development of the protocol specifications. Whether DLEP will be the de-facto standard for radio-router protocols in a few years is not certain. It is up to the radio manufacturers to decide whether they want to use open standards or proprietary solutions.

7 Future work

In this thesis we have experimented with two ways of utilizing link-layer information in routers. There are many other ways to utilize link-layer information, not only in routers but in clients as well. Amongst these are the following:

- **Dynamically calculate link costs:** In dense networks it is likely that there are more than one path between two nodes in the network. By assigning dynamic costs based on the metrics reported by some radio-router protocol one might be able to more efficiently utilize the links and avoid congestion. It would most likely be necessary to develop some heuristic for dampening fluctuations in link costs, in order to prevent packets in the same flow to take different paths. Spreading network traffic over two or more different paths could have a negative impact on the network performance, especially when using TCP as transport protocol [22].
- **Protocol specific feedback mechanism:** It would be interesting to see the effects of utilizing the link-layer information in, for example, a video or audio streaming system. It should be possible to adjust to new link conditions by dynamically changing bit rate and other parameters of the data stream. This work would not be as generic and widely applicable as the work described in this thesis, although there might be bigger gains in quality of service.
- **More sophisticated routing protocol integration:** The OLSR plugin used in our experiments is rather naive. It would be interesting to implement a more sophisticated plugin which could extend OLSR with some other approach of initializing a router neighbourhood. For example, one could use the technique described in [23], which presents an explicit unicast handshake mechanism for setting up the relationships between routers rather than just relying on *hello* messages. Doing so will likely further reduce the network convergence time.

References

- [1] B.-N. Cheng, J. Wheeler, and L. Veytser, "Radio-to-router interface technology and its applicability on the tactical edge," in *IEEE Communications Magazine*, vol. 50, (USA), pp. 70 – 77, 7 2012.
- [2] R. Charland, L. Veytser, and B.-N. Cheng, "Integrating multiple radio-to-router interfaces to open source dynamic routers," in *Proc. 2012 IEEE Military Communications Conference (MILCOM) 2012*, (Piscataway, NJ, USA), pp. 1 – 6, 2012.
- [3] C. Barz and H. Rogge, "Improved community network node design using a DLEP based radio-to-router interface," in *Proc. IEEE 8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2012)*, (Piscataway, NJ, USA), pp. 636 – 642, 2012.
- [4] J. Vehent, "Journey to the center of the linux kernel: Traffic control, shaping and QoS," *Gnu/Linux Magazine France*, vol. 127, May 2010.
- [5] Cisco, "Internet protocol and radio frequency networks: Creating robust military networks." <http://www.wintertime.com/Prof/Other/RAR.pdf> - Fetched at 15th of August, 2014.
- [6] S. Ratliff, B. Berry, G. Harrson, S. Jury, and D. Satterwhite, "Dynamic link exchange protocol (draft 05)," Internet-draft, Mobile Ad hoc Networks Working, 2014.
- [7] D. Dubois, A. Kovummal, B. Petry, and B. Berry, "Radio-router control protocol (R2CP)," Internet-draft, Networking Group, September 2011.
- [8] L. Mamakos, K. Lidl, J. Evars, D. Carrel, D. Simone, and W. R, "A method for transmitting PPP over Ethernet (PPPoE)," RFC 2516, Network Working Group, 2 1999.
- [9] H. H. B. Berry, "PPP over ethernet (PPPoE) extensions for credit flow and link metrics," RFC 5578, Network Working Group, 2 2010.
- [10] J. Moy, "OSPF version 2," RFC 2328, IETF, 4 1998.
- [11] "Quagga routing suite." <http://www.nongnu.org/quagga/> - Fetched at 5th of May, 2014.
- [12] B.-N. Cheng, R. Charland, P. Christensen, A. Coyle, I. Pedan, L. Veytser, and J. Wheeler, "Comparing radio-to-router interface implementations on experimental CoTs and open source routers," in *Proc. IEEE Military Communications Conference (MILCOM) 2012*, (Piscataway, NJ, USA), pp. 1 – 6, 2012.
- [13] J. Kurose, K. Ross, *Computer Networking - A Top-Down Approach*. Addison Wesley, fifth ed., 2010.
- [14] W. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," tech. rep., Network Working Group, 2001. <http://tools.ietf.org/html/rfc2001> - Fetched at 24 of July, 2014.
- [15] "Olsrd - an adhoc wireless mesh routing daemon," tech. rep. <http://www.olsr.org> - Fetched at 12th of August, 2014.
- [16] "Virtual Box." Internet page (manpage). <https://www.virtualbox.org/> - Fetched at 16th of June, 2014.

- [17] “ebtables.” Internet page. <http://ebtables.sourceforge.net/> - Fetched at 16th of June, 2014.
- [18] “Traffic control (tc).” Internet page (manpage). <http://lartc.org/manpages/tc.txt> - Fetched at 16th of June, 2014.
- [19] P. Jacquet, T. Clausen, “Optimized link state routing protocol (OLSR),” tech. rep., Network Working Group, October 2003.
- [20] “DLEP Tools.” Internet page. <http://dleptools.sourceforge.net/> - Fetched at 12th of May, 2014.
- [21] Cisco, “Cisco IOS and NX-OS software.” Internet page. <http://www.cisco.com/c/en/us/products/ios-nx-os-software/index.html> - Fetched at 12th of March, 2014.
- [22] U. Ranadive and D. Medhi, “Some observations on the effect of route fluctuation and network link failure on TCP,” in *Proc. Tenth International Conference on Computer Communications and Networks, 2001.*, pp. 460–467, 2001.
- [23] Y. Huang, S. Bhatti, and S.-A. Sorensen, “Reducing neighbour detection latency in OLSR,” in *Proc. (PIMRC) 2007. IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–5, Sept 2007.

XR-EE-LCN 2014:005