Faculty of Economic Sciences, Communication and IT
Department of Computer Science

Andreas Lavén

Multi-Channel Anypath Routing for Multi-Channel Wireless Mesh Networks

Degree Project of 30 ECTS credit points
Master of Science in Information Technology

Date/Term: 10-01-22
Supervisor: Prof. Dr. Andreas J. Kassler
Examiner: Dr. Donald F. Ross
Serial Number: E2010:04

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se

Multi-Channel Anypath Routing for Multi-Channel Wireless Mesh Networks

Andreas Lavén

This thesis is submitted in partial fulfillment of the requirements for the Masters degree in Computer Science. All material in this thesis which is not my own work has been identified and no material is included for which a degree has previously been conferred.

Andreas Lavén

Approved, 2010-01-22

Supervisor: Prof. Dr. Andreas J. Kassler

Examiner: Dr. Donald F. Ross

Abstract

Increasing capacity in wireless mesh networks can be achieved by using multiple channels and radios. By using different channels, two nodes can send packets at the same time without interfering with each other. To utilize diversity of available frequency, typically cards use channel-switching, which implies significant overhead in terms of delay. Assignment of which channels to use needs to be coupled with routing decisions as routing influences topology and traffic demands, which in turn impacts the channel assignment.

Routing algorithms for wireless mesh networks differ from routing algorithms that are used in wired networks. In wired networks, the number of hops is usually the only metric that matters. Wireless networks, on the other hand, must consider the quality of different links, as it is possible for a path with a larger amount of hops to be better than a path with fewer hops.

Typical routing protocols for wireless mesh networks such as Optimized Link State Routing (OLSR) use a single path to send packets from source to destination. This path is precomputed based on link state information received through control packets. The consideration of more information than hop-count in the routing process has shown to be beneficial as for example link quality and physical layer data rate determines the quality of the end-to-end path. In multi-channel mesh networks, also channel switching overhead and channel diversity need to be considered as a routing metric. However, a major drawback of current approaches is that a path is precomputed and used as long as the path is available and shows a good enough metric. As a result, short term variations on link quality or

channel switching are not considered.

In this thesis, a new routing protocol is designed that provides a set of alternative forwarding candidates for each destination. To minimize delay (from both transmission and channel switching), a forwarding mechanism is developed to select one of the available forwarding candidates for each packet. The implementation was tested on an ARM based multi-radio platform, of which the results show that in a simple evaluation scenario the average delay was reduced by 22 % when compared to single path routing.

Contents

List of Figures

List of Tables

Acknowledgement

I would like to thank my supervisor Prof. Dr. Andreas J. Kassler for his guidance and all the knowledge he together with Peter Dely and Marcel Cavalcanti de Castro have given me. Thank you Peter and Marcel for all your support.

My former co-workers, when the ChaCha plug-in was developed, Tomas Hall, Peter Hjärtquist and Andreas Midestad helped me describe the ChaCha plug-in and also to formulate some of my ideas. Peter Hjärtquist was also a co-developer of the Multimetric OLSR, which he also helped describe in this thesis.

Finally, I would like to thank CRL Sweden, and especially Anders Lundström for feedback.

Chapter 1

Introduction

1.1 Introduction

When it comes to connecting nodes in a network, all nodes need to know the best path between them. The question is; how to decide which path is the best? Usually, the path that passes through the fewest number of nodes is considered the best path. This is not always true in wireless networks, where links can vary greatly in matter of quality, which must be considered for a good selection of paths[15][16]. To capture these variations of link qualities, other metrics than hop count can be used.

To increase the capacity of a wireless network, multiple channels are often used[9]. The capacity is increased because with multiple channels it is possible for two nodes within a short distance to simultaneously send information without interfering with each other. Using multiple channels introduces switching delays, which means that nodes might need to wait for a switch of a channel before they are able to send information. This delay should be considered in a routing algorithm.

Anypath routing makes it possible for nodes to forward packets towards the same destination, on different next hops. When multiple next hops exist for a node, for the same destination, the switching delay may be decreased. In this thesis, a novel routing

mechanism will be developed and evaluated that follows the anypath paradigm.

1.2 Purpose of Thesis

The goal of this thesis is to develop and evaluate a routing mechanism for multi-radio multi-channel mesh networks following the anypath paradigm. The routing mechanism aims to decrease the switching delay in the network, and thus the overall end-to-end delay.

To reach the goal of this thesis, a routing algorithm will be developed that is able to find multiple possible next hops for each destination. All these next hops should listen on different channels, because it is then possible to send packets towards a specific destination on several channels compared to one only. With the ability to send on several channels, the probability of being able to send without first requiring a channel switch is increased.

Because of many possible multiple next hop candidates, a node must be able to determine which next hop to forward to. The next hop will then be selected for each packet by a layer 2.5 forwarding algorithm, which takes into account switching cost.

1.3 Subprojects

This project is split into two major subprojects; routing and forwarding.

Routing is a process of finding a path between nodes in the network, and the best path is selected by using a routing algorithm. A difference from most routing algorithms, is that the one in this thesis should find alternative next hops.

Forwarding is the relaying of packets from one node to another. Because several next hops can exist, there must be an algorithm selecting which is the best. The forwarding algorithm should select the best next hop for each packet. The best next hop is the one that will deliver the packet within the shortest amount of time.

1.3.1 Routing

To determine which path is the best, there exist several metrics[19]. The most basic metric is the hop count metric, which counts all links for the path. This metric is not desirable in wireless networks, because of the variation of link quality[15][16]. Instead, there are metrics that are developed specifically for wireless networks. An example of such a metric is the Expected Transmission Count (ETX)[2]. The ETX metric estimates how many times a packet needs to be sent on a link, in average, until the packet is received. This is considered a quality measurement of a link. A retransmission is needed when a packet is lost, for example, due to interference from another transmission.

Another metric developed for wireless networks is Expected Transmission Time (ETT)[10], which is based on the ETX metric. The difference between them is that the ETT metric also considers the capacity of a link. The capacity denotes how many packets that can be received within a time span, which depends on the quality and data rate of the link. By using the ETT metric the delays on the links are captured, but not within the nodes. The switching delay of different nodes to forward a packet may vary, and therefore also a switching cost should be considered in a routing algorithm.

The routing algorithm developed in this thesis will be able to find several next hops for each destination. All these next hops must use different channels, in order to decrease the switching delay. Two next hops using the same channel cannot contribute to a decreased switching delay, because only the one with the lowest path cost will be selected if the specific channel is beneficial. For a next hop selection to be possible, the routing algorithm must also bind a path cost to each next hop.

1.3.2 Forwarding

Usually, there is only one next hop for each destination. It is then sufficient for a node to lookup the next hop in the route table for the entry of the destination, to determine which

node to forward to.

In this thesis, multiple next hops might exist for a destination and therefore the sending node is required to select one of them. To select a next hop, a forwarding algorithm is developed and applied to each forwarded packet. The forwarding algorithm considers both the end-to-end delay of the packet for a given path, and also the delay until the transmission can start. This delay is the time it takes until the required channel is in use.

By considering how long time it takes until the packet can be sent to each next hop, the switching delay can be reduced. A next hop that on average has a greater path cost could be selected if the required channel is already in use. If the difference in path cost, measured in time, is less than the switching delay until the required channel for the average best next hop is in use, then it is beneficial to send the packet on the current channel.

When using anypath forwarding, there is a risk that routing loops will occur. A loop has arisen when a packet is sent back and forth within a group of nodes, without making progress towards the destination. To prevent a packet from circulating in the network, there is a prevention forcing the packet towards the destination presented in this thesis.

1.4 Thesis Structure

The second chapter in this thesis is "Background and Related Work". In that chapter there are explanations of all terms used in this thesis. There is also an overview of the components in this thesis, and related work is presented.

The next chapter is "Design" where the ideas of this thesis are described. It also describes the anypath paradigm in more detail. Also, there are problems explained introduced by the anypath paradigm and solutions for them.

The fourth chapter in this thesis is "Implementation". This chapter explains how the ideas from the "Design" chapter are implemented. Many of the components presented in the overview are modified in this thesis, and these changes are described in this chapter.

In the end of this thesis there is "Evaluation" followed by "Conclusions". The "Evaluation" chapter presents a test performed on the implementation, and there is also topics for further research presented.

1.5 Summary

In this thesis, a design, implementation and evaluation of a new routing and forwarding approach for multi-radio multi-channel wireless mesh networks are performed. Unlike usual single path routing (like OLSR and AODV), this approach creates multiple paths to each destination. The approach also contains a new forwarding algorithm, which selects the best next hop for each packet.

The implementation was tested on an ARM[21] based multi-radio platform, of which the results show that in a simple evaluation scenario the average delay was reduced by 22% when compared to single path routing.

Chapter 2

Background and Related Work

2.1 Introduction

The goal of this thesis is to develop and evaluate a multi-channel anypath routing mechanism for multi-channel wireless mesh networks. In this chapter, there are explanations of what a wireless mesh network is and why the capacity can be increased by using multiple channels. There is also related work presented in this chapter.

2.2 Background

2.2.1 Wireless Mesh Networks

A wireless mesh network (WMN) consists of radio nodes organized in a mesh topology. The nodes within a wireless mesh network are connected by links. There exists a link between two nodes if they have a pair of interfaces that are susceptible to hear one another, which means they need to have the same channel and be within each other's transmission range. A link is also called a hop.

Two nodes in a WMN are able to transmit data to each other if there is a path between

them. A path can contain one or several links. A path between two nodes is found using routing algorithms, for example shortest path first. An advantage of WMNs is that if a node or a link fails, other nodes will automatically find new routes around the no longer working part.

An example of where WMNs are used is within the "One laptop per child" program[23]. The laptops use wireless mesh networking because of the lack of wired connections in the children's environments.

2.2.2 Channels

Information within wireless networks is sent on different frequencies. For the IEEE 802.11b and the IEEE 802.11g protocols, the available frequencies are between 2.400 MHz and 2.485 MHz[18]. The frequency spectrum is divided into 13 slots, which are called channels. The channels are 22 MHz wide and they are overlapping with each other, which means that they may interfere with each other.

Channel Separation

In a network where only two nodes are sending data on the same channel, the capacity is below half compared to what it would be if the two nodes were using different channels. This is because when one of the nodes is sending data, the other node has to wait. To make them both able to send data at the same time, they must use two channels that are at least 5 channels apart.

When there are many nodes in a network all nodes cannot have their own channel because there exist only a limited number of channels, but somehow the nodes must select one. The simplest way of selecting a channel, except randomly pick a channel for each node, is to divide the nodes so there is an equal amount of nodes using each channel. The disadvantage of an algorithm like that is that all inactive nodes might share one channel and all the active nodes share another, yielding low performance for the active nodes while

reserving bandwidth for the inactive nodes.

Orthogonality

Because channels are overlapping, two channels on frequencies near each other do interfere when they are sending at the same time. Channels that are not interfering with each other are called orthogonal channels. Orthogonal channels in the IEEE 802.11b and the IEEE 802.11g protocols are the channels that are at least 5 channels apart from each other. For example, the channels 1, 6 and 11 are orthogonal.

Nodes that are within the interference range are susceptible for interference while transmitting in parallel. This interference can be avoided by using orthogonal channels. The interference range is within the distance the node can send data, usually within two or three hops. The signal may interfere further away than its data is receivable.

2.2.3 Net-X

Net-X is a framework used in wireless networks[8]. It is primarily used for managing multiple channels within the same network, but is also capable of managing multiple interfaces as well as adjusting transmission rates and power levels etc[7].

If data is sent on the same channel from different nodes, the receiving nodes cannot perceive the information. This is called a collision. By making use of more channels, several nodes can send packets at the same time. Because of the possibility of sending packets simultaneously, the capacity of the network increases.

Performance

All nodes in the network have two interfaces, one fixed and one switchable (see figure 2.1 on page 12). The fixed interface is used for receiving packets and the switchable interface is used for sending packets. The switchable interface changes its channel so it matches the channel on the receiving interface of the receiving node.

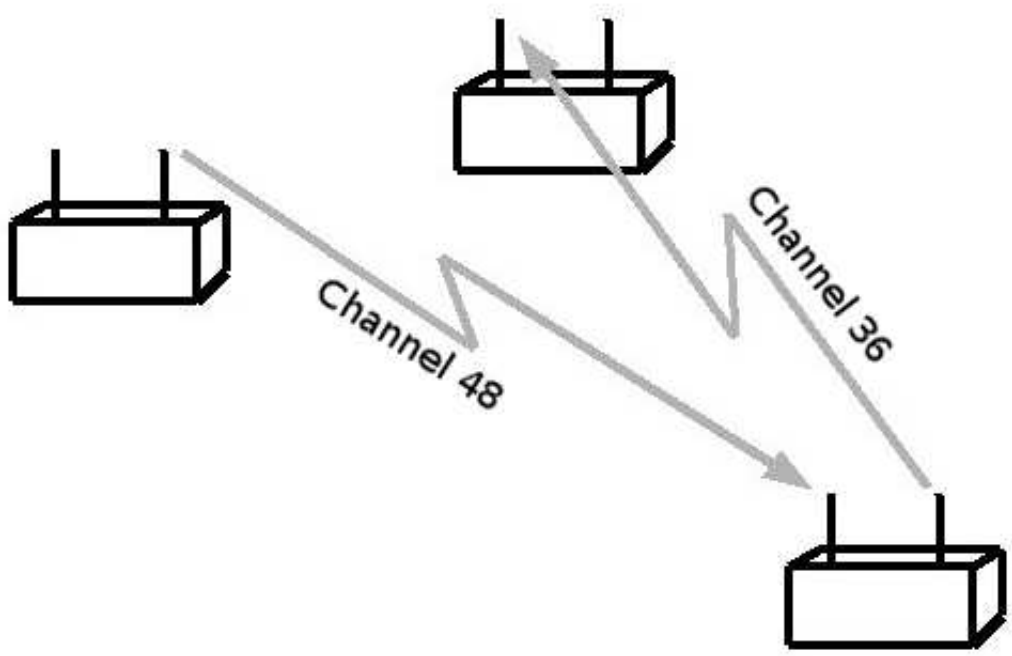


Figure 2.1: Nodes using two interfaces

2.2.4 ChaCha

ChaCha is a plug-in for the Olsrd, which is an implementation of the OLSR protocol (see next section). The purpose of this plug-in is to distribute channel information within a 2-hop environment.

The plug-in may also perform a channel balancing algorithm, which selects the least used channel within the 2-hop environment. When the channel balancing algorithm has selected a channel, it is used for the receiving interface (see Net-X in section 2.2.3). Before a node switches its channel on the receiving interface, it informs its neighbors via broadcast messages.

2.2.5 OLSR

Optimized Link State Routing protocol (OLSR) is a proactive routing protocol, which means the connection setup delay is minimized at the expense of the heavier control traffic load[3]. OLSR uses Hello and Topology Control (TC) messages to discover and disseminate link state information throughout the network. The topology information is used to calculate the next hop destination for nodes in the network.

A key concept of OLSR is the usage of multipoint relays (MPRs). An MPR is a node that is selected to forward broadcast information during the flooding process. The message overhead is substantially reduced due to this technique. In a classical flooding mechanism, every node in the network retransmit broadcast messages.

Another advantage of the OLSR protocol, is that the link state information is only generated by MPRs. This is sufficient because the declaration of MPR nodes link state information is the only requirement for OLSR to calculate the shortest path. When link state information is generated only by MPR nodes, the number of control messages flooded in the network is minimized.

Hello messages

Hello messages are sent to all 1-hop neighbors to sense 1-hop and 2-hop neighbors. Hello messages are also used to perform distributed selection of a set of multipoint distribution relays (MPRs).

TC messages

The TC messages are used to disseminate neighbor information throughout the network. These messages are sent throughout the entire network, but they contain only links that are involving MPR nodes.

2.3 Related Work

Routing is the process of finding a path between the nodes in a network. To determine if a found path is better or worse than other paths, routing algorithms use a metric. There exists many different metrics, whereof three of them is described in this section.

In single path routing, the routing process is satisfied by finding one path between all nodes. There is also routing algorithms finding multiple paths. Routing algorithms that are able to find multiple paths between any two nodes, may have different definitions of what to consider as two separate paths.

Finally, this section is also describing anypath routing.

2.3.1 Expected Transmission Count

Expected Transmission Count (ETX)[11], is a metric that considers the quality of the links. The ETX metric is defined as the number of transmission, on average, that is required until a packet is received on a given link. A retransmission is needed when a packet is lost, for example, due to interference from another transmission.

The goal of using ETX is to find the route with the highest probability of packet delivery.

Equation

The ETX value is calculated using the following equation.

$$ETX = \frac{1}{LQ * NLQ} \tag{2.1}$$

Where LQ is the link quality and NLQ is the neighbor link quality, which is estimated using statistics for received control packets.

2.3.2 Expected Transmission Time

Expected Transmission Time (ETT)[10], is a metric that is based on Expected Transmission Count (ETX). ETT estimates the transmission time for the packet to be successfully delivered to the next hop.

Unlike the ETX metric, ETT considers the capacity of the links. The capacity denotes the quality and the bit rate of the link, that is how many packets that can be received within a time span.

Equation

The ETT value is calculated using the following equation.

$$ETT = ETX * S/B \tag{2.2}$$

Where S is a generic packet size, normally the one used for probing, measured in bytes. B is the link capacity, which is measured using packet-pair technique. ETX is the expected transmission count.

Packet-pair Technique

The packet-pair technique, used to calculate the bit rate of a link, uses two back-to-back probes, one small packet followed by a larger. The receiving node measures the inter-arrival time and reports it back to the sending node. The link capacity is then estimated by the following formula:

$$B = S_L / \min_{1 \leqslant i \leqslant n} d_i \tag{2.3}$$

where B is the link capacity, S_L is the size of the large probe packet, d_i is a delay sample and n is a predefined number of delay samples.

2.3.3 Switching Cost Metric

The switching cost metric (SC)[22] measures the increase in delay a packet experiences on account of channel switching. The delay is zero when a packet could be sent out on the fixed interface, since it is never delayed waiting for the interface to switch channel.

The switching cost is needed, because measuring the time it takes for the packet in the air is not sufficient. Time is also spent when the packet is in a node waiting for the interface to tune in the required channel.

Equation

The switching cost is estimated by calculating the probability of a switch multiplied with the delay of a switch. The probability of switching to a channel k is estimated as follows

$$P_s(k) = \frac{\sum_{\forall i, i \neq k}^{c} Usage_i}{1000000} \tag{2.4}$$

Where i is a channel, k is the channel to switch to, c is the number of available channels and $Usage_i$ is the number of microseconds the channel i has been used the last second.

The delay of a switch is depending on hardware, and is not changing through the execution.

2.3.4 Multiple Paths Routing

Multiple paths routing makes nodes aware of alternative paths between each other. An advantage of awareness of alternative paths is the fast recovery when a node or a link fails[13]. If a node or link fails, the traffic can immediately be sent on an alternative path that does not contain the failed node or link. Another advantage with multiple paths is the possibility of balancing the traffic load.

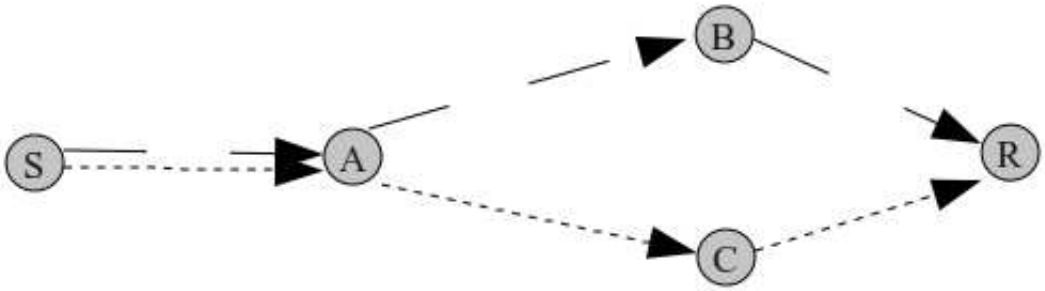
A multiple path routing mechanism can be node-disjoint or link-disjoint. Node/link-disjoint path means that no node/link is a part of two different paths between the source and the destination. It is possible for a node to have more than one alternative path, but then a new path cannot contain a node (or link) that is a part of any of the other paths.

Also, there exists braided multipath where an alternative path is a path not containing at least one node or link from the primary path. See figure 2.2 on page 18 for examples.

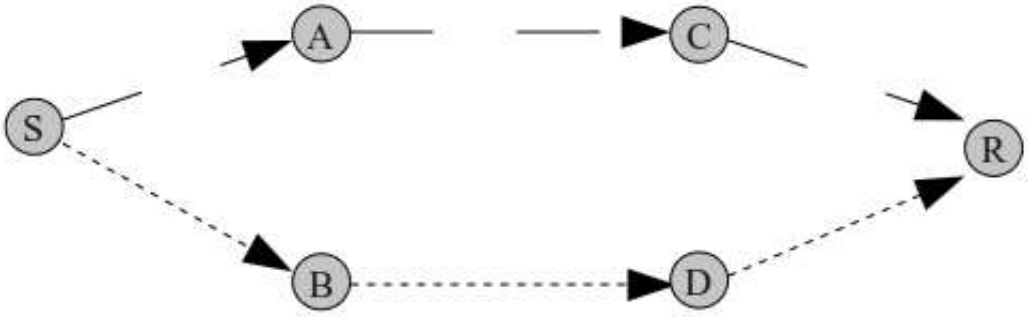
2.3.5 Anypath Routing

Anypath routing means that a packet's path is defined while the packet is traversing the network on its way to its destination. Each node has a set of next hop candidates and the node can forward to any of them. Because of the high loss rate and dynamical quality links in wireless mesh networks[15][16], anypath routing is an advantage because the best possible links for each packet can be selected.

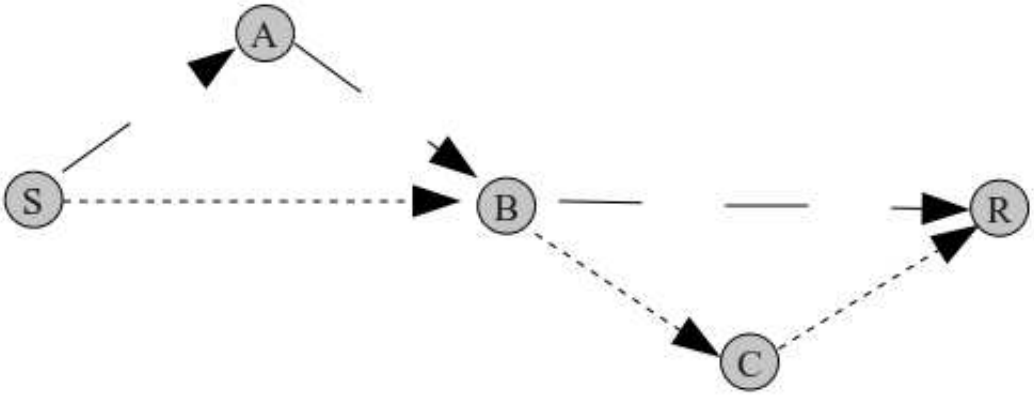
In a paper[14] published in 2008, R. Laufer and L. Kleinrock presents an anypath routing mechanism for single channel wireless mesh networks. The paper presents a routing process finding a set of possible next hops, for each destination. When a packet is to be forwarded towards a given destination, the packet is sent multicasted to all nodes in the next hop set. The first node to receive the packet, forwards it on.



(a) Braided multipath



(b) Node disjoint multipath



(c) Link disjoint multipath

Figure 2.2: Alternative multiple path definitions

The difference to the anypath routing mechanism presented in this thesis, is the usage of multiple channels in this thesis. Because the next hop candidates may use different channels, multicasting packets to the entire set of possible next hops would introduce a large overhead of switching cost. The anypath mechanism presented in this thesis will not make use of multicasting and so the next hop candidate set will contain only one next hop per channel.

2.4 Summary

This chapter covers terms needed to understand the ideas presented in this thesis. In the background section, wireless mesh network and channels are described. The framework Net-X is used to manage multiple network interface cards for each node and the plug-in ChaCha is used to distribute channel information. Also, the OLSR routing protocol is described in this chapter.

There is also related work presented in this chapter. Among the related work, three routing metrics are described, ETX, ETT and Switching Cost metric. There is also a description on multiple paths routing and anypath routing.

Chapter 3

Design

3.1 Introduction

For a routing mechanism to follow the anypath paradigm, a routing algorithm must find multiple next hop candidates. Typically, routing algorithms update their routing table either periodically or on demand when a new route is to be established or when a route breaks. This project aims to decrease the delay by for each forwarded packet select the best next hop out of a candidate set. Decreasing the delay is important for delay sensitive applications, for example VoIP. In this chapter a design will be presented, describing a routing algorithm and a forwarding algorithm enabling the anypath paradigm.

The routing algorithm presented can find multiple next hops for each destination. To determine the cost of a path, a combination of the ETT metric and the Switching Cost metric is used. This combination gives an estimated end-to-end delay for each path.

The selection of a next hop to forward a packet to, is performed by a layer 2.5 forwarding algorithm. This algorithm considers the delay until the channel required to forward the packet to a given next hop is in use, referred to as the switching cost. The selected next hop is the node having the lowest cost, switching cost plus path cost. Therefore, it is important that the routing algorithm metric describes the path cost in terms of time.

3.2 Routing

The goal of a routing algorithm is to find the best path to each destination in a network. To decide which path is the best, there are many metrics that can be used. Two different paths can be considered the best path, dependent of which metric is used. In this thesis, the path that on average takes the smallest amount of time to reach the destination will be considered the best path. Also, to follow the anypath paradigm, the routing algorithm must be able to find multiple next hops for each destination.

The metric used to calculate the path cost in this thesis is a combination of the ETT metric (see 2.3.2 on page 15) and the Switching Cost metric (see 2.3.3 on page 16), which gives the average time it takes for a packet to reach the destination when forwarded on a given path. It is important that the routing algorithm describes the path cost in terms of time, because the next hop selection will add this time to the time it takes until the node is able to start sending to calculate the total cost of a path. A node can start sending packets when the required channel is in use.

The routing algorithm should be able to find multiple next hops for each destination, so that the switching cost can be decreased. By being able to forward packets with the same destination on different channels, the probability for a feasible channel to already be in use without the need to switch increases. Two paths with next hops using the same channel will not increase this probability, therefore only one next hop per channel will be stored in the set of possible next hops for the destination.

To be able to store the alternative next hops, the routing table must be extended (see section 3.2.4 on page 27).

3.2.1 Path Cost Calculation

A routing algorithm searches for the best path from a given node to all other nodes within the network. Usually, Dijkstra's algorithm is used, which walks through all paths in the

network and finds the best path[17]. The algorithm adds all the costs along the path, to get a total cost for the entire path. An example of costs along the path is the number of hops.

In this thesis, the routing algorithm will consider the delay as the path cost. The costs along the path is the time it takes for a packet to be transmitted over the links and the time the nodes must wait because of switching delays. The cost for each link is estimated by the ETT metric and the switching delay for each node is estimated by the Switching Cost metric.

ETT

Expected Transmission Time (ETT) is used to estimate the time it takes for a packet to be transmitted over a link. How the ETT metric value is calculated is described in section 2.3.2 on page 15.

Switching Cost

Switching Cost (SC) is used to estimate the switching delay for all nodes along the path. By using the ETT metric alone, the routing algorithm does not result in a fair estimation for the entire path. Only the links' capacity is considered, not the nodes'.

By adding the switching cost to the ETT metric value, the entire time needed for the packet to be received is considered. For more information on the Switching Cost metric see section 2.3.3 on page 16.

Formula

The following formula is used to calculate the cost for each path:

$$Pathcost = ETT_1 + \sum_{i=2}^{n} ETT_i + SC_i \tag{3.1}$$

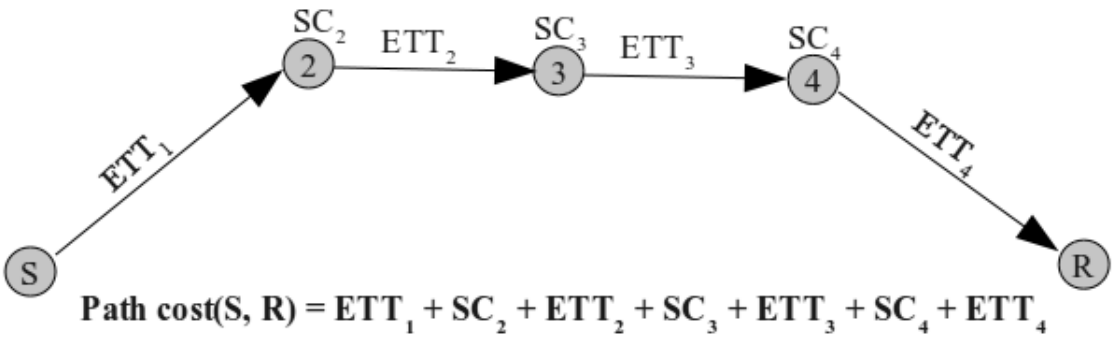


Figure 3.1: Path cost calculation

Where i is a link on the path and n is the number of links. The SC_1 is not part of the path cost, because the exact switching delay will be used in the next hop selection.

See figure 3.1 for an example of a path cost calculation for a route.

3.2.2 Multiple Paths

The decision of whether to store a new found path or not differs between single path routing and multipath routing. When single path routing is used, the routing algorithm must compare a new found path only to a destination with the best path known for the given destination. If the new path has a lower cost than the already known path, the entry in the routing table will be replaced. When the routing algorithm should find multiple paths, where all first hops are using different channels, a new found path must be compared with a possible already known path using the same channel on the first hop.

Figure 3.2 on page 25 is showing three different paths from the source node S to the receiving node R, through the next hops A, B and C. The average best path in the example is through the node C with a cost of 8 time units. This path would in a single path routing be the only selected path. In the routing algorithm presented in this thesis, paths using different channels on the first hop will also be considered. Therefore, also the

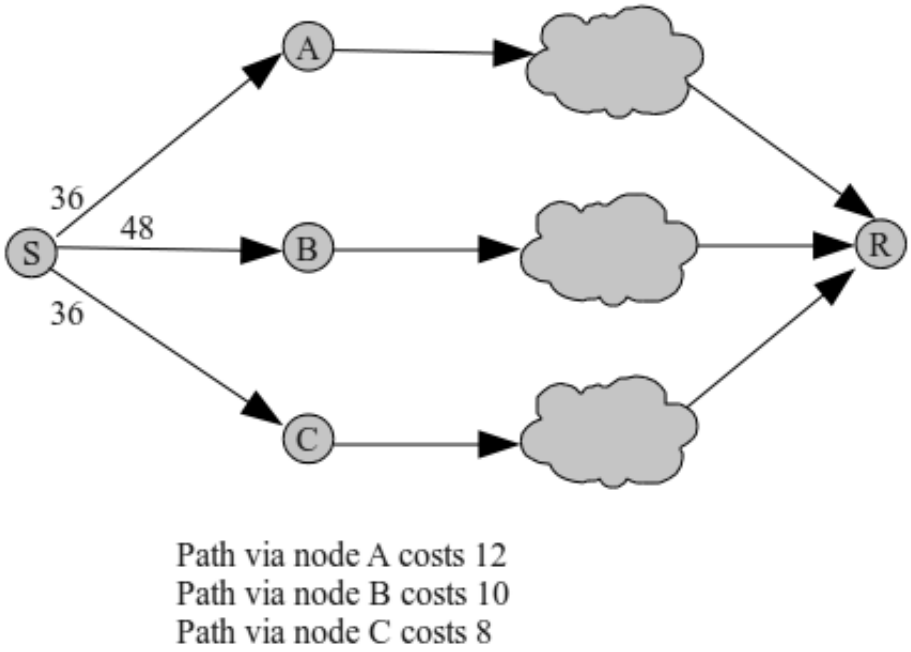


Figure 3.2: Three possible next hops, whereof two are using the same channel

path through the node B will be stored in the routing table. The path through the node A will not be stored in the routing table, because there is another path using the same channel on the first hop with a lower path cost.

3.2.3 Routing Algorithm

The routing algorithm should for each node explore each link to calculate the path cost to the node's neighbors. The path cost is calculated by adding the path cost of the node with the cost of the link to the specific neighbor. See section 3.2.1 on page 22 for more information about cost calculation.

The new path cost is compared with the currently stored path to the neighbor having the same channel on the first hop. If the new path cost is lower than the old path's cost, or if the neighbor's path cost was previously unknown, the new path must be stored. The number of hops must also be calculated, which is done by increasing the previously node's number of hops by one.

A pseudo code example which walks through all links of a known node is presented here:

```
for each node n {
   for each link l {
      if (n.hops = 0) //if first hop, do not include switching cost
         new_cost := l.ett_cost
      else
         new_cost := n.path_cost + l.ett_cost + l.sc_cost

      if (l.node = UNKNOWN OR
            new_cost < get_path_cost(l.node, n.channel) {
```

```
            l.node.path_cost := new_cost
            l.node.next_hop := n.next_hop //same next hop as n
            l.node.channel := n.channel   //same channel as n
            l.node.hops := n.hops + 1
        }
    }
}
```

3.2.4 Routing Table

The routing table needs to be extended for storing the new information from the routing algorithm, multiple paths for each destination and the channel on the first hop. In routing tables of a single path design, there is only one next hop for each destination, which is always used when a packet is to be delivered to a given destination. In the forwarding solution presented in this thesis, a node could have the possibility to forward a packet to a next hop from a set of possible next hops. Therefore, the routing table must be able to store more than one next hop per destination.

When a destination can have several next hops, a forwarding algorithm is needed to select one of them. The forwarding algorithm should select the best next hop for each packet, and therefore it needs information of path cost for the different routes. The channel for the first hop is also needed, so a switch delay can be considered when selecting the next hop. Finally, the number of hops required for each path is needed to prevent loops (see section 3.3.2 on page 31).

An example of the routing table design is shown in table 3.1.

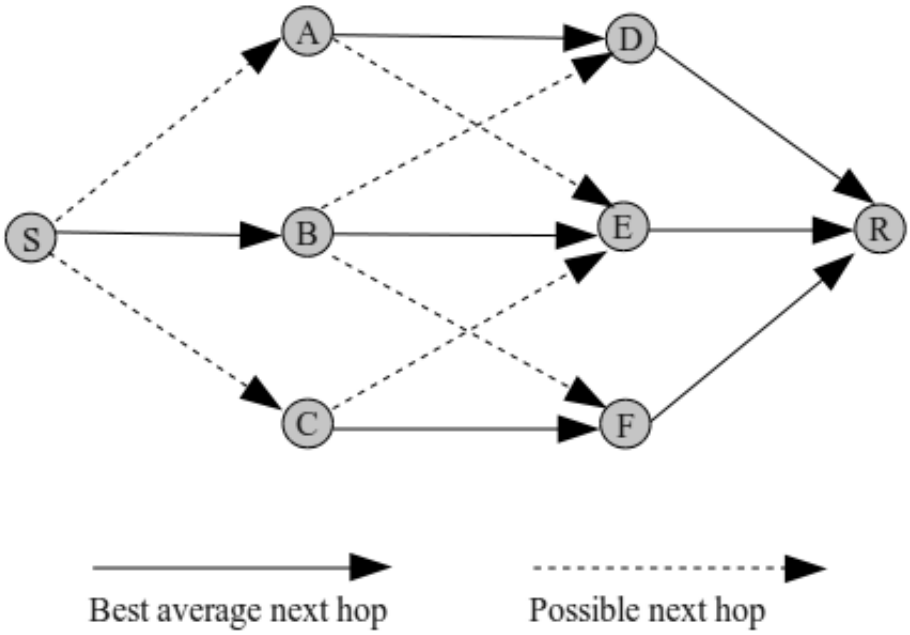


Figure 3.3: Anypath forwarding

Destination	Next Hop	Channel	Cost	#Hops
192.168.1.1	192.168.1.1	36	4	1
192.168.1.1	192.168.1.2	48	6	2
192.168.1.2	192.168.1.2	48	2	1

Table 3.1: Example of the routing table design

3.3 Forwarding

In single path routing mechanisms, nodes always forward packets destined for a specific receiver to the same next hop. This next hop is the one that has the lowest average path cost to the receiving node.

The routing mechanism in this thesis forwards the packet to the next hop that is the best for the moment. Due to the changes made in the routing algorithm, a node can have several possible next hops (see figure 3.3 on page 28 for an example). To decide which next hop that is the best choice for forwarding to for the moment, the total cost for the path must be calculated (see next section). The next hop selection considers both the switching delay and the path cost given from the routing algorithm. The switching delay is the time it takes until the required channel can be in use.

The figure 3.4 on page 30 shows how the channels are used when traffic is to be sent to a destination, and the node has two neighbors that can both be used as a next hop. Since the single path routing can send on one channel only (for example 36), it must switch back to channel 36 before it can continue transmitting (the switch to channel 64 is forced by control messages that needs to be broadcasted). When anypath routing is used, the packets can be sent on any of the two channels and therefore it can resume sending earlier.

Before a next hop can be selected, the forwarding algorithm must also consider the number of hops that is required for next hops to reach the destination. This is needed to prevent loops, described in section 3.3.2 on page 31.

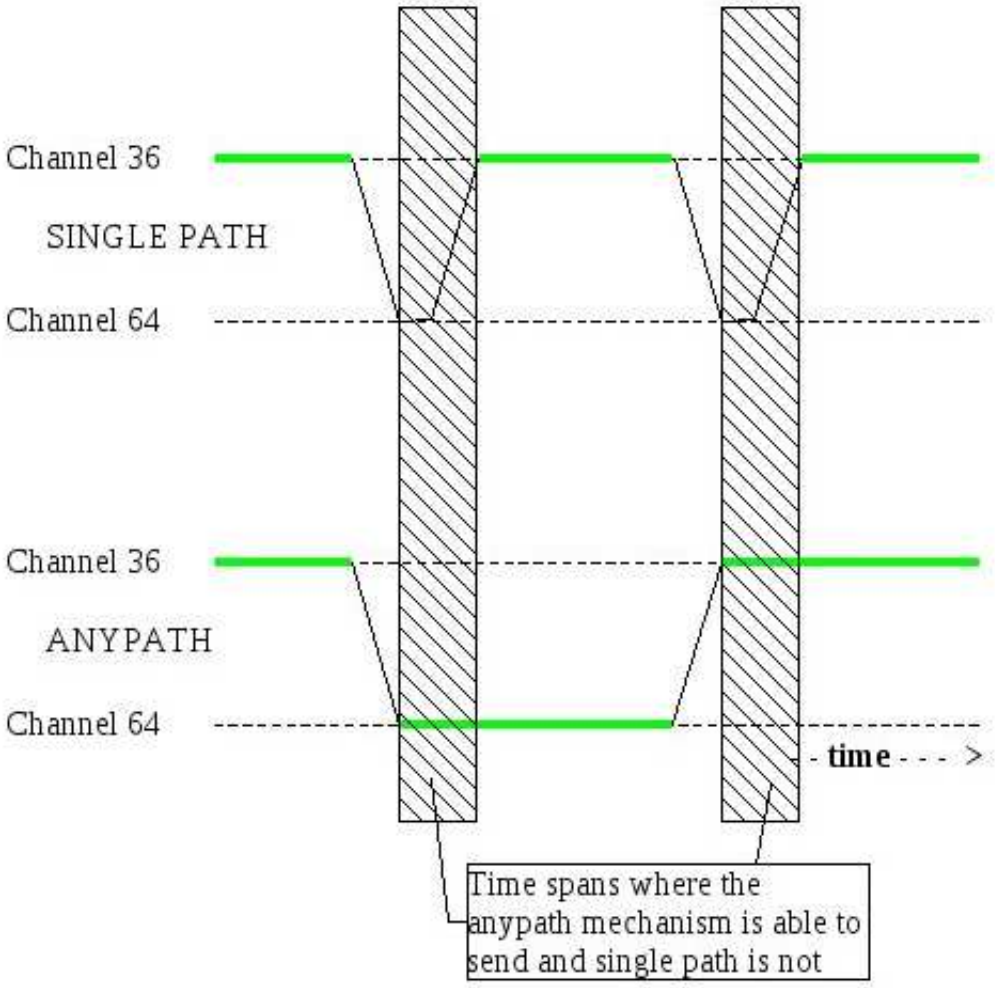


Figure 3.4: Channel usage when both neighbors can be used as next hops

3.3.1 Finding Next Hop with the Lowest Cost

When the routing table contains only one next hop for each destination, it is sufficient to lookup the destination in the routing table and forward to the next hop in that entry. In this thesis, the routing table can have several possible next hops for each destination, and therefore next hop selection is needed.

When a packet is to be forwarded towards a specific destination, the next hop selection calculates the total cost for each of its possible next hops. The next hop with the lowest cost is selected to forward the packet to. To calculate the total cost, the following formula is used:

$$Totalcost = Switchingdelay + Pathcost \tag{3.2}$$

$$Totalcost = SC_1 + ETT_1 + \sum_{i=2}^{n} ETT_i + SC_i \tag{3.3}$$

Where the Switching cost (SC_1) is the time it takes until the required channel to start sending the packet on the specific path can be used.

3.3.2 Loop Issue

A loop is formed when a packet is circulating among a set of nodes, and therefore does not make any progress towards the destination. When each node has only one next hop, the path is a sequence of nodes and therefore always make progress towards the destination, meaning that if the current node is i hops away from the destination, the next hop will be $i - 1$ hops away from the destination.

In the next hop selection proposed, it is no longer certain that the next hop is less hops away from the destination compared to the current node. Therefore, a restriction that a next hop must be closer to the destination, in terms of hops, could remove the possibility

of selecting the best next hop. A node that has two paths towards a destination, where the average best path is 5 hops away and another is 6 hops away, is considered 5 hops away from the destination. If the path that requires 6 hops is selected (by being better for the moment), the packet will still be considered 5 hops away from the destination when the next hop possess the packet.

When two or more nodes within a local environment all have similar cost to the destination, it is possible that they send a packet back and forth between each other due to the reduced switching cost of the currently used channel. This may continue without the packet ever making any progress towards the destination.

Solution

As a solution to the loop issue, the node that creates the packet will also give the packet a maximum number of hops allowed until the destination must be reached. This value will force each packet to make progress towards the destination. A similar solution is presented in [12].

The maximum number of hops that is allowed for the packet is calculated by the number of hops for the best average path multiplied by a constant α. The value of α will define the level of restriction from the node that created the packet. A high value will give the nodes along the path more possibilities to deviate from the best average path. The formula for the maximum allowed number of hops:

$$Hops_{allowed} = Hops_{bestpath} * \alpha \tag{3.4}$$

For each hop the packet makes, the value will be decreased. Before forwarding the packet, each node must check that the next hop node has a path to the destination within the remaining number of allowed hops. This might remove the opportunity for a node to choose the best path, but it forces the packet towards its final destination.

3.3.3 Forwarding Algorithm

The forwarding algorithm loops through all entries in the routing table. When an entry with the required destination is found, the total path cost is compared to the best next hop found so far. The algorithm does also check that the required number of hops for the entry does not exceed the remaining number of hops allowed until the packet must have reached the destination. A pseudo code example of the algorithm is presented here:

```
for each entry e {
   if e.destination = destination {
      if e.hops <= remaining_hops {
         if e.path_cost + switching_delay(e.channel) <
               best_e.path_cost + switching_delay(best_e.channel) {
            best_e := e
         }
      }
   }
}
next_hop := best_e.next_hop
```

The complexity of the algorithm increases by the number of nodes within the network, O(n). There will be one entry per next hop for each destination, but the number of next hops is not affected by the total number of nodes.

3.4 Alternative Anypath Design

An alternative anypath design is to allow a maximum deviation of k hops from the predefined path. Typically $k = 1$ or 2. The predefined path is the one calculated by the node that created the packet. For an example, see figure 3.5.

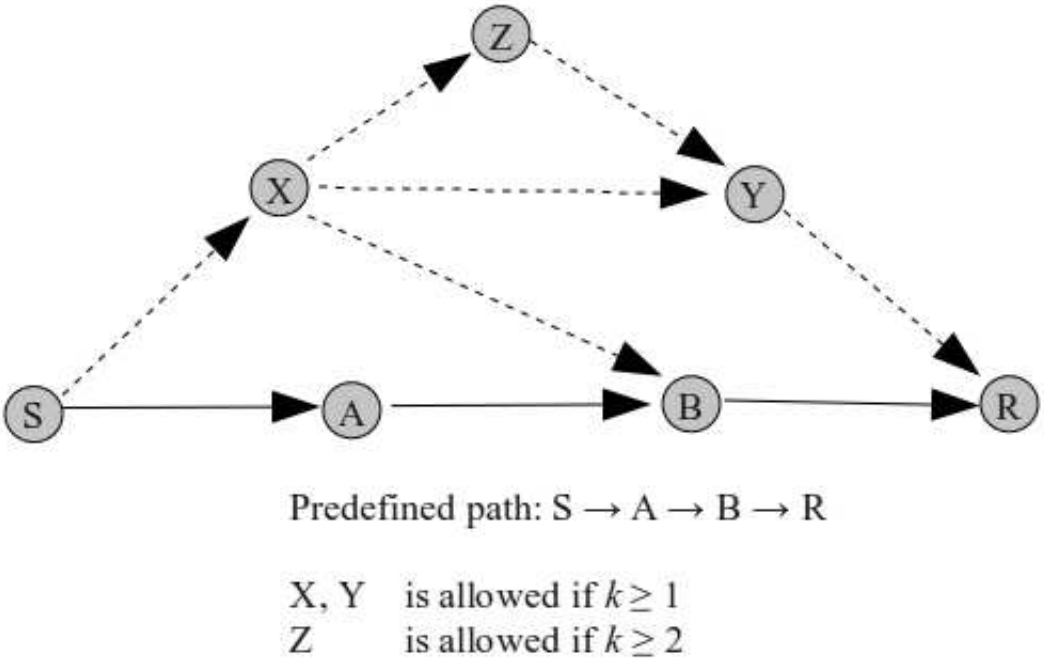


Figure 3.5: Example of alternative forwarding

This design was rejected in this thesis because it does not allow the nodes along the path to independently decide which next hop to choose. Because of the maximum deviation requirement, a node must consider the source node before it can forward the packet. The further away the packet's source is from the current possessor of the packet, the less up-to-date information the source has about the local environment around the current possessor. Therefore, to let the source node have an impact on a next hop selection far away may enforce a bad choice.

Also, for nodes to be able to follow the maximum deviation requirement, they must know which path is the predefined one. The source node can put the path information within all packets, or all nodes along the path calculate the path between the source node and the destination node in order to get the source node's perspective of the network. Either way, this solution will create unnecessary overhead.

3.5 Summary

In this chapter, many ideas around the anypath paradigm is presented. To make the routers in a network support anypath routing, changes are required. The routing algorithm needs to be modified and a next hop selection algorithm needs to be developed.

The routing algorithm needs to be able to find several possible next hops for each destination. The routing algorithm must also be modified to route on time. Also, to make it possible to store several next hops for each destination, the routing table must be extended.

When several next hops exist for a destination, there must be a next hop selection algorithm. This algorithm considers the switching cost to select the best next hop for each packet, which depends on how long time it takes until the required channel is used to be able to transmit to the specific next hop.

Anypath routing also brings a loop issue, which means that packets may be circulating without making any progress towards the destination. A solution of the loop issue is also presented in this chapter.

Chapter 4

Implementation

4.1 Introduction

This chapter will describe the implementation of the routing and forwarding mechanisms presented in this thesis. First, a description of the different components of the routing mechanism and also how they work together is given. After the presentation of the components, the modifications needed to follow the anypath paradigm will be presented.

The routing algorithm, used to find routes to other nodes, must be able to find multiple next hops for each destination. It is also modified to use the Expected Transmission Time metric and the Switching Cost metric to determine the path cost. This functionality is implemented in the AP-OLSR (AnyPath-OLSR), which is developed in this thesis.

The forwarding algorithm, required to select the best next hop within the set of possible next hops in the anypath routing, is implemented in layer 2.5 to be able to select the best next hop for each packet.

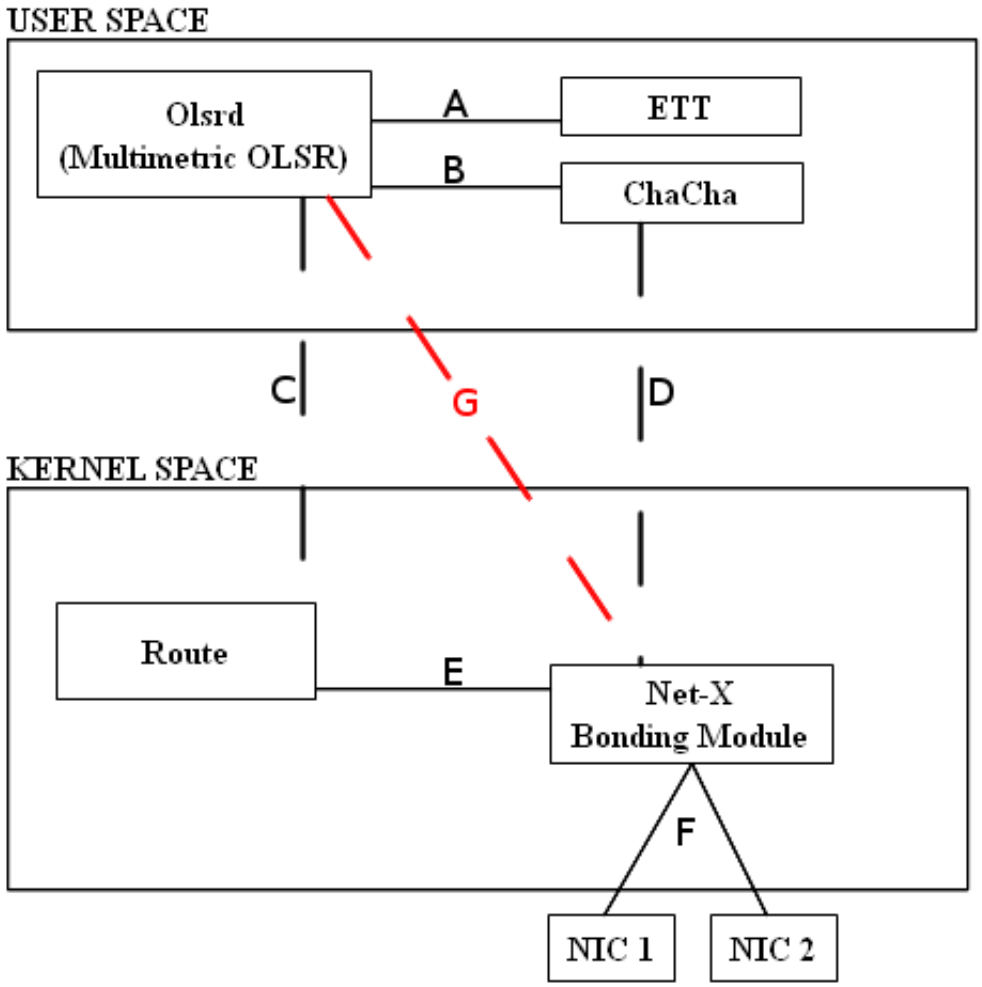


Figure 4.1: An overview of the components

4.2 Components

4.2.1 Olsrd

The Olsrd is an implementation of the routing protocol OLSR, and it was started as a master thesis written by A. Tønnesen at UniK, University Graduate Center.

The major assignment of Olsrd is to find paths to all other nodes within the network. This is done by a routing algorithm which adds up the cost for each link to reach the given

node. The links between the network's nodes and the links' costs are known by sending Hello messages and Topology Control messages.

A difference between the Olsrd and OLSR is that the Expected Transmission Count (ETX) is enabled by default in Olsrd.

4.2.2 Multimetric OLSR

The Multimetric OLSR[6] is an extension of the Olsrd and it is developed at Karlstad University by A. Lavén and P. Hjärtquist. It is based on MF-OLSR[5], which is a metric based framework for OLSR[4]. The main difference between the two is the possibility to use an arbitrary number of metrics in Multimetric OLSR while MF-OLSR supports three of which only one is user configurable. The metrics are distributed by extended Hello and TC messages.

Even though an arbitrary number of metrics can be distributed by Multimetric OLSR, only one metric can be used for the routing table calculation. The metric which is used for routing and which metrics to distribute is set in the configuration file.

If the metric used for the routing table calculation is not distributed from a node in the network, a conservative default value will be used instead. A conservative default value is used in order to maintain the nodes' connectivity, while having a low priority. This is useful when it is the only node providing a unique feature in the network, such as Internet access.

The Multimetric OLSR also contains an implementation of the switching cost metric (see 2.3.3 on page 16).

4.2.3 ETT

The ETT plug-in, see section 2.3.2 on page 15, estimates the expected transmission time for links within a network. The estimation is performed by each node sending probes to its one hop neighbors using a separate socket instead of using built-in Olsrd functionality. The

separate socket is needed to avoid the OLSR protocol packet aggregation, which otherwise is used to save network resources. Packet aggregation is avoided because it hinders the correct operation of the packet-pair technique, which requires inter-arrival time samples[10]. Instead of transmitting the small probe followed by the larger one, OLSR might transmit both in one packet giving no inter-arrival time rendering the sampling useless.

4.2.4 ChaCha

The ChaCha plug-in, see section 2.2.4 on page 13, accepts a couple of parameters which can modify the behavior of the plug-in. The parameters and their values is specified in a configuration file.

Information distributed about a particular node within the local environment is the current channel of the listening interface, the next channel for the listening interface, a countdown to the channel switch, and the IP address of the node. The received information is stored in a list containing instances of the structure *channel_table_entry*, defined as following:

```
struct channel_table_entry {
   olsr_u8_t            channel;      //current channel
   olsr_u8_t            nextChannel;  //next channel
   olsr_32_t            countdown;    //time until switch
   union olsr_ip_addr   ipAddress;    //IP address
};
```

Where *olsr_u8_t* is an unsigned integer with a size of 8 bits and *olsr_ip_addr* is the representation of an IP address within Olsrd. The fields *nextChannel* and *countdown* are set to 0 and -1 respectively if no channel switch is scheduled.

The channel balancing algorithm is executed periodically at an interval which is set in the configuration file. The algorithm loops through the list of *channel_table_entry*

instances calculating the frequency of each channel within the local environment. If the channel with the lowest use frequency differs from the current, a switch must be performed and the countdown field is set to the countdown value defined in the configuration file. The countdown value is continuosly decreased and when it reaches zero the channel of the listening interface is changed to the one in the *nextChannel* field. The *nextChannel* field is reset to zero when the channel is switched.

4.2.5 Route

Route is the part of the Linux kernel where the routing table is defined. The routing table contains an entry for each destination in the network. The routing table is filled by receiving path information from the Olsrd. When a packet is to be sent towards a destination, the routing table is used to lookup the next hop for the destination.

4.2.6 Net-X Bonding Module

The Net-X bonding module[8] is essentially a way to provide a single virtual interface for multiple physical ones. Instead of communicating using a physical interface, a process will use the virtual bonding interface. For this purpose, the bonding module contains two tables, the broadcast table and the unicast table (see figure 4.2 on page 43 for an overview).

The broadcast table tells the bonding module which channels to use to broadcast a message to nearby nodes. For each channel used in the network, the broadcast table should contain an entry consisting of two fields, the channel and the interface to use when broadcasting on this channel. The broadcast table needs to be initialized at the startup, so each channel has an interface to be broadcasted by.

The unicast table tells the bonding module how to send a unicast message to a specific neighbor. The table contains an entry for all known neighbors, each containing the IP address, the fixed channel the neighbor uses, and the interface to send on. The unicast

table requires no initialization, but entries must be added and removed whenever a neighbor is discovered or lost.

If a process wants to send a message via the bonding interface, Net-X will identify whether the message is a broadcast or unicast message. Depending on the type, the appropriate table will be consulted.

In case of a broadcast message, the bonding module will send a copy of the message on each channel in the broadcast table, using the defined physical interface. If more than one channel use the same physical interface, the channel will automatically be switched between broadcasts.

If the message is to be sent unicast, the bonding module will check the unicast table for an entry containing the specific IP address and the packet is put in the correct channel queue. The interface specified in the entry will then be switched to the specified channel if necessary, and the message will be sent. Round Robin scheduling is used to select one of the channels currently having packets queued for transmission.

4.2.7 Component Interaction

Figure 4.1 on page 38 illustrates an overview of the components.

The Olsrd is used to find the paths to all nodes within a network. When a path is found, information about the path is sent to the Route (C). The Route receives this information, and adds it to its routing table.

When a packet is to be sent out on the network, regardless whether it is received from an application within the user space or from another node to be forwarded, the Route will decide the next hop for the packet by a lookup in its routing table. When the packet has a next hop selected, it will be sent to the bonding module (E). The bonding module will determine which network interface card (NIC) the packet will be sent on (F).

ETT and ChaCha are two plug-ins to the Multimetric OLSR. ETT calculates the Expected Transmission Time for the links on the node, and sends the metric values to the

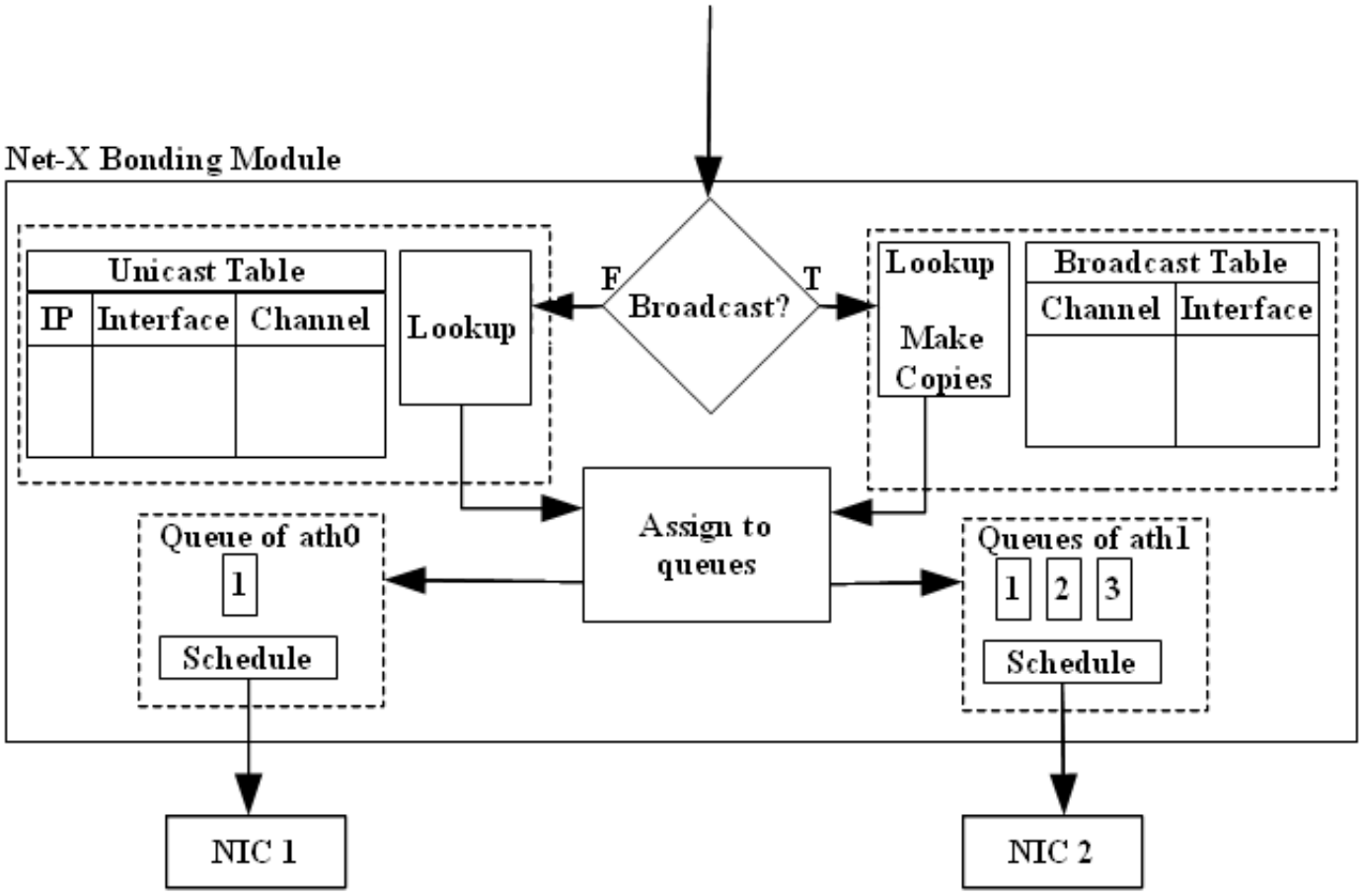


Figure 4.2: Net-X bonding module[8]

Multimetric OLSR which stores the information (A). ChaCha receives channel information from the neighbors, and forwards the information to the Multimetric OLSR (B). When the ChaCha plug-in has performed its channel balancing algorithm, it will, if needed, send a channel switch request to the bonding module to change the channel of the receiving interface (D).

The major difference of component interaction for this thesis is that the routing algorithm will send route information directly to the bonding module, and the bonding module will be responsible to select the next hop for each packet (see section 4.4 on page 48). The direct connection between the routing algorithm and the bonding module is illustrated as the red dashed line (G) in figure 4.1 (page 38).

4.3 AP-OLSR

To make it possible for the nodes to find multiple next hops for each destination, AP-OLSR (AnyPath-OLSR) is developed in this thesis. AP-OLSR is an extension of the Multimetric OLSR, described in section 4.2.2 on page 39. The major differences compared to the single path implementation is changes in the routing algorithm and the routing table, to support multiple paths. AP-OLSR has also direct communication with the bonding module (see section 4.2.6 on page 41).

4.3.1 Route Information

The routing information in AP-OLSR is stored as instances of the structure *tc_entry*. A *tc_entry* is used for each node in the network and it holds information such as next hop and the path cost to reach the node. The structure also contains *edges*, which represents nodes one hop away. In the Olsrd implementation, the *tc_entry* contains only the next hop for the best path, therefore it had to be extended such that in the AP-OLSR it may also contain multiple next hops with different path costs.

For the routing algorithm to be able to compare paths starting on a given channel, the route entry must also contain information on which channel is used on the first hop. Also, the number of required hops for the specific path to reach the destination is needed, because of the loop issue described in section 3.3.2 on page 31.

To make the *tc_entry* structure able to carry this information, it is extended to contain also a new structure called *ap_path_list*. This structure is defined as:

```
struct ap_path_list_entry{
   uint16_t channel;             //Channel used on first hop
   union olsr_ip_addr next_hop;  //IP address of next hop
   olsr_linkcost path_cost;      //Path cost
   uint16_t hops;                //Number of hops required to reach destination

   struct ap_path_list_entry *next;    //Pointer to the next entry
};
```

Where *uint16_t* is an unsigned integer with a size of 16 bits, *olsr_ip_addr* is the representation of an IP address within Olsrd and *olsr_linkcost* is the representation of the cost for a link.

4.3.2 Routing Algorithm

The routing algorithm is an implementation of the Dijkstra's algorithm, shortest path first. The algorithm walks through all nodes by using the *tc_entry* structure declared for each node. This structure contains *edges*, which represent nodes that are connected to this *tc_entry* via one link.

To calculate the path cost, a combination of two metrics is used. The metrics are Expected Transmission Time (ETT) and Switching Cost (SC), described on page 15 and

16 respectively. The metrics are distributed within the topology control (TC) messages and stored for each link for each *tc_entry* (as a list of metrics for each *edge*).

The routing algorithm must also calculate the number of hops required for a packet to reach the destination using the current path. This information is used by the bonding module to prevent loops. The hop count is calculated by increasing a counter for each node along the path.

Alternative Routes

In the original Olsrd, the routing algorithm results only in the best path. To be able to forward packets to the next hop that for the moment is the best choice, the routing algorithm must be able to find more than one next hop per destination.

The routing algorithm should not result in more than one next hop per channel, therefore it has to check the channel used on the first hop. This is done by using functionality implemented in Multimetric OLSR[6]. To get the channel used on the first link, the following line of code is used:

$$channel = mm\_olsr\_get\_metric\_value(new\_tc- > next\_hop- > metrics, LMT\_CHAN); \tag{4.1}$$

Where *new_tc* is the destination, *next_hop* is the next hop to reach the destination and *metrics* is the list of all metrics stored for the link to the specific next hop. The LMT_CHAN tells the function *mm_olsr_get_metric_value* that it is the link metric type *channel* that is requested.

When the channel of the first hop is known, a check is performed to see if the new path has a lower cost than the old path using the same channel. This check uses the following line of code:

$$if(cost < ap\_get\_cost(ap\_path\_list, channel)) \tag{4.2}$$

Where the function *ap_get_cost* returns the cost of the path using the channel *channel* on the first hop, or highest possible value if there is no path using the given channel.

If the found path is the best path that uses the specific channel on the first hop, then the path will be stored. The path will be stored in a list that has been created for each *tc_entry*. This list is called *ap_path_list*, and each entry contains information on which channel is used on the first hops, the IP address of the first hop, the path cost and the number of hops required to reach the destination.

4.3.3 Interaction with AP Bonding

When a new best path is found, on any channel, AP-OLSR must send this information to the bonding module, in order to make the bonding module able to forward packets on this path.

This information is sent directly when a new path is found, by using an ioctl call. The ioctl call is performed within a function called *ap_ioctl_call* which receives the information needed by the bonding module (see section 3.3), and creates an instance of a structure known by both AP-OLSR and the bonding module (AP-Bonding). The structure is presented in section 4.4.1 on page 48.

Channel Switch

Because the forwarding algorithm considers the switching delay until the channel of the first hop is used, it is important to inform the bonding module when a neighbor changes its channel on the listening interface. This information is sent using an ioctl call.

4.4 AP-Bonding

A major difference between the original Net-X bonding module and the AP-Bonding, is that the AP-Bonding is responsible to select the next hop for each packet. In the original implementation, there was only one possible next hop for a destination and it was looked up in the route (see figure 4.1 on page 38).

4.4.1 Routing Table in Bonding Module

To be able to forward packets to alternative next hops depending on which channel is currently tuned in on the sending interface, the bonding module must be aware of all possible next hops. This information is stored in a list containing instances of the structure *ap_entry*, defined as following:

```
struct ap_entry{
   uint32_t dst_addr;  //IP address of destination
   uint32_t nh_addr;   //IP address of next hop
   uint16_t channel;   //Channel used on first hop
   uint16_t cost;      //Path cost
   uint16_t hops;      //Number of hops required to reach destination

   struct ap_entry *next; //Pointer to the next entry
};
```

In single path routing, it is sufficient to find the entry containing the specific destination. When more than one next hop may exist, as in this thesis, a forwarding algorithm is needed to decide which next hop to use. For the forwarding algorithm to be able to select the best path, path cost and required channel is needed.

The structure *ap_entry* must also contain the number of hops required for each path. This value is used to prevent loops (see section 3.3.2 on page 31).

4.4.2 Loop Issue

Because of the loop issue, described in section 3.3.2 on page 31, the bonding module must be aware of the allowed number of hops for a packet until its destination must be reached. This value will force the packet towards its destination.

The value of allowed number of hops is set by the source of the packet. It is calculated by multiplying the number of hops required by the best path, the path with the lowest cost, with a constant α.

The value of allowed number of hops must be decreased by 1 for each node the packet pass by. All nodes must also check that the next hop can reach the destination within the allowed number of hops before the packet is forwarded. A node currently possessing a packet does always have at least one next hop that can reach the destination within the allowed number of hops, otherwise the previous node would not have been able to forward the packet to it.

Packet Processing at Source Node

Before a node sends a packet through the network, it has to set a number of allowed hops until the destination must be reached. The number of hops required by the path with the average lowest path cost is the minimum allowed number of hops. This value is multiplied with the constant α to get the maximum number of allowed hops for the packet.

A function named *ap_add_opt* is implemented to set the maximum allowed number of hops as a part of the IP packets, within the IP options. The function receives a socket buffer (a structure representing an IP packet within the bonding module) and the number of allowed hops. The function moves the IP header and stores the value in the gap where the IP header previously was placed. Finally, it updates the IP header by recalculating the checksum.

Packet Processing at Forwarding Nodes

When a packet is to be forwarded in the network, the nodes must prevent the packet from circulating without making any progress towards the destination. This is prevented by checking the required number of hops for a next hop, compared to the remaining number of hops until the destination must be reached. The value is reached via a function named *ap_get_remaining_hops*, which returns the data from the socket buffer.

When a node receives a packet, it decreases the number of allowed hops by one. This is done because the packet has passed a link to reach the current node. To decrease the number of remaining number of allowed hops, a function named *ap_update_incoming_opt* is used. Because the function modifies the IP packet, it must also recalculate the checksum.

4.4.3 Forwarding Algorithm

The original usage of the Net-X bonding module was to lookup the channel and network interface card (NIC) that a packet is to be sent on. The next hop is already selected by the Linux route. The channel and NIC lookup is performed by looping through a table of entries, searching for the specific next hop. This table contains an entry for each neighbor of the node.

The bonding module in this anypath routing implementation is also responsible for selecting the next hop to forward the packet to, because of the possibility of having several next hops. The selection of next hop is performed for each packet, and before the original bonding module performs its task.

The forwarding algorithm is implemented in a function named *get_next_hop*, and is located in the file *ap_bonding.c*. The forwarding algorithm is implemented as:

```
for(struct ap_entry tmp = ap_entries; tmp != NULL; tmp = tmp->next) {
   if (tmp->dst_entry == destination &&
       tmp->hops <= ap_get_remaining_hops(skb)) {
```

```
      cur_cost = tmp->cost + get_switch_time(tmp->channel);
      if(cur_cost < min_cost) {
         min_cost = cur_cost;
         nh = tmp->nh_addr;
      }
   }
}
return nh;
```

Where *get_switch_time* is a function returning the delay until the required channel is used, *ap_get_remaining_hops* is a function returning the remaining number of allowed hops for the packet and *skb* is the socket buffer (a structure representing the IP packet).

4.4.4 Interaction with AP-OLSR

The interaction between the bonding module and AP-OLSR is performed by ioctl calls. AP-OLSR sends information for all possible next hops for each node in the network. This information is then stored in a table developed in this thesis, see section 4.4.1 on page 48.

When the bonding module receives an ioctl message containing route information from the AP-OLSR, it must update the table. The information that is sent is the addresses of the destination and the next hop, the channel used for the first hop, the cost for the path and also the required number of hops.

When path information is received from the AP-OLSR, the new entry is added to the table. If the path's first hop uses a channel already in the table for the specific destination, the old entry must be deleted. Otherwise, the old path will never be used. If two next hops to the same destination use the same channel the one with the lowest path cost will always be used, because the delay for a channel switch will be equal for both of them.

The AP-OLSR must also send information about removal of an entry. This happens when a link becomes unusable, and also when a neighbor change its channel on the listening interface.

4.5 Summary

This chapter presents the implementation of the ideas presented in this thesis. The routing protocol implementation is based on Olsrd[2], and the result is named AP-OLSR. Changes have also been made to the Net-X bonding module, so it is able to select the best next hop from a set of possible candidates. The name for the resulting bonding module is AP-Bonding.

The AP-OLSR has got its routing table extended, so it matches the requirements of anypath routing (see section 3.2.4 on page 27). The routing algorithm has been modified as well, so it is able to find multiple next hops for each destination. Finally, there has been a new ioctl connection that goes to bonding module. This ioctl message is used to send new information needed for anypath forwarding.

The bonding module is modified to catch ioctl messages from the AP-OLSR, containing route information to a destination. The information received is added to a routing table, created for this purpose. A forwarding algorithm is also implemented to select a next hop for each packet, because the possible existence of multiple next hops for given destination.

Chapter 5

Evaluation

5.1 Introduction

This chapter contains an evaluation of the anypath routing mechanism implemented in this thesis. To evaluate the implementation, a small test is performed. This test shows how the components work and how the traffic flow will be divided when multiple next hops exist. The test is also performed for a single path routing mechanism, and a comparison of delay in the test environment is made.

This chapter also presents tests that still remains to be performed, and topics for future work within the anypath paradigm.

5.2 Test

5.2.1 Test Case

To test the implemented anypath routing mechanism, a traffic flow between two nodes within a small network will be measured. The network contains of four multi-channel nodes placed in a diamond formation, with MAC filters simulating an obstruction blocking

the direct link between the source node and the receiving node (see figure 5.1 on page 55). Because of the obstruction, the sending node S must send the packets via the next hops A or B to reach the destination node D. The two next hops A and B are listening on different channels. The same test is also performed using the original single path routing mechanism.

There is also other traffic within the network which is the control messages that are sent by AP-OLSR as well as the plug-ins ChaCha and ETT. Because of this traffic, the node S must switch its channel on the sending interface occasionally to be able to send to both next hops. When single path routing is used, the traffic flow from the node S and the node R can only be sent through one of the two possible next hops and therefore the source node S experiences a switching delay when the sending interface is tuned to the channel of the other one.

When anypath routing is used, the traffic flow can be sent via both possible next hops. The selected next hop will be determined by the current channel on the sending interface, because both paths have an equal cost.

The bonding module is configured to use a channel for a minimum of 25 ms and a maximum of 60 ms, if there is packets in another channel queue and thus a switch is required. The packet stream is generated by mgen[20], with a packet size of 1000 bytes and a rate of 100 packets per second. All links have a bandwidth set to 6 Mbps.

5.2.2 Performance

This section will describe the performance of the AP-OLSR and the AP-Bonding of the source node S.

AP-OLSR

AP-OLSR found its neighbors, A and B, via control messages, which also contained the cost to reach them. Thereafter, it found two routes to the node R, one via the node A and

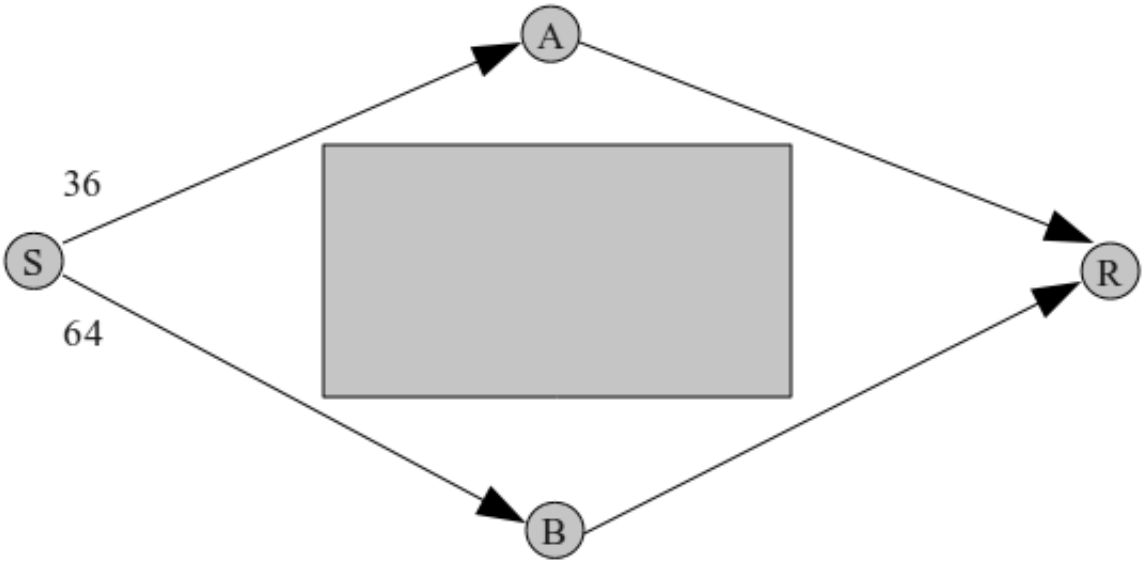


Figure 5.1: Test environment

Destination	Next Hop	Channel	Cost	#Hops
IP address of A	IP address of A	36	x	1
IP address of B	IP address of B	64	x	1
IP address of R	IP address of A	36	2x	2
IP address of R	IP address of B	64	2x	2

Table 5.1: Routing table of paths sent from AP-OLSR

the other via the node B.

The AP-OLSR passed all paths to the bonding module, in total 4 paths (see table 5.1)

AP-Bonding

The bonding module received the paths that were passed from the AP-OLSR and added them to its routing table. When a packet was to be sent to the node R, the forwarding algorithm decided which next hop to forward to.

The path costs were equal for both possible next hops and the selected path was there-

fore determined solely by the switching time. When the channel 36 was in use, the packets were sent to the next hop A, and when the channel 64 was in use the packets were forwarded to the next hop B.

5.2.3 Result

Single Path Routing

When the single path routing was used, traffic was only sent via one of the two possible next hops. The next hop used were the node A. Since the two possible next hops listened to different channels, the source node S had to wait for a channel switch when control messages had been sent to the node B. The next hop node A also used two different channels, because the node R listened to a third channel.

Because all nodes sent on two channels, a packet could either reach the destination after two, one or no channel switches. The graph 5.2 on page 57 shows the probability for a packet to be received within a given delay.

The average time until a packet was received using single path routing was 8.2 ms.

Anypath Routing

When the anypath routing was used, traffic was sent via both the nodes A and B. Therefore, the node S was always able to forward the packets, and the switching delay at this node was zero. Hence, no packets experienced two switching delays.

The traffic from the source node S to the receiving node R was equally distributed over both next hops. Although the flow was divided into two different paths, all packets arrived in the same order as they were sent in.

The graph 5.2 on page 57 shows the probability for a packet to be received within a given delay. The average time until a packet was received using anypath routing was 6.4 ms.

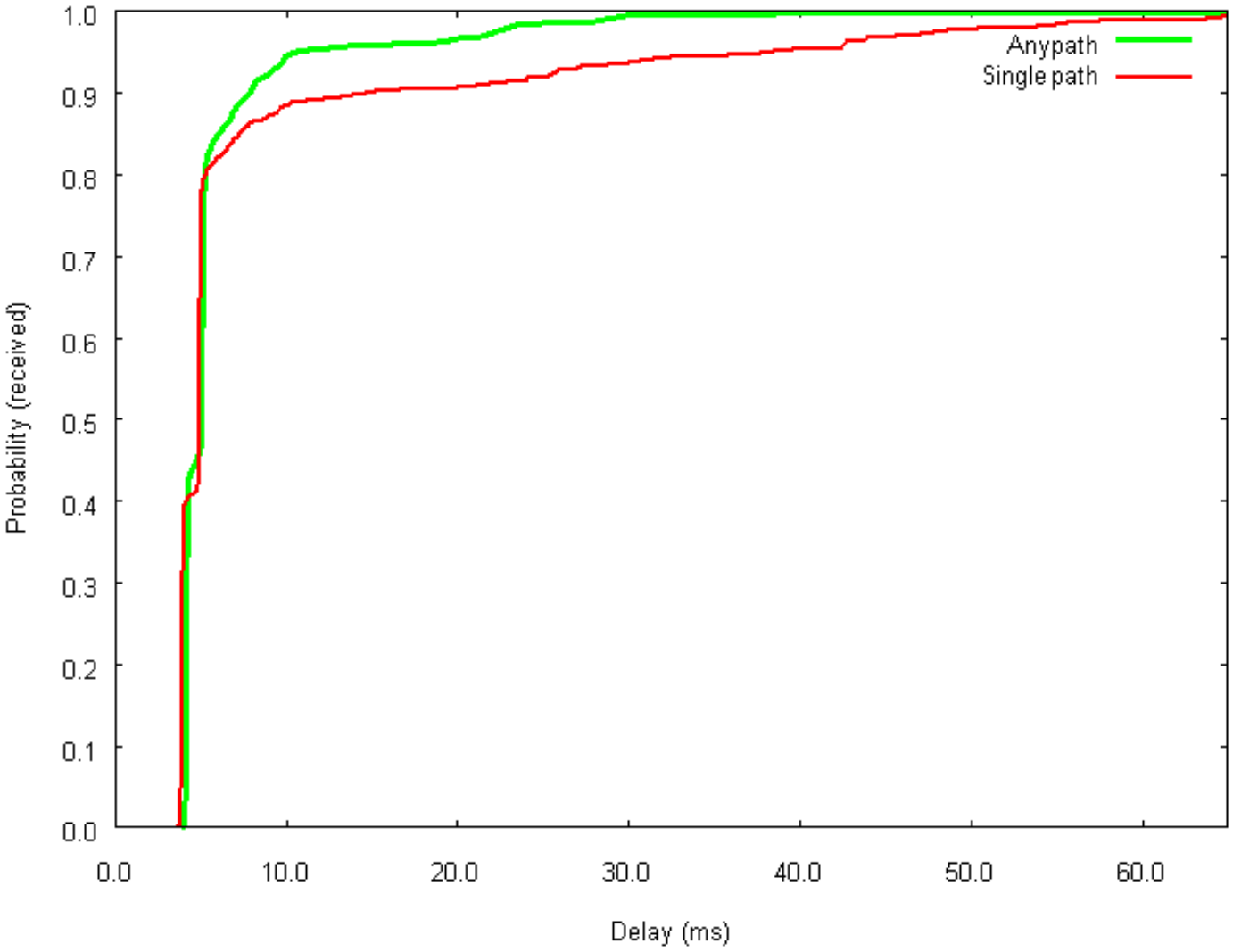


Figure 5.2: Graph showing the probability that a packet is received

5.3 Summary

The implementation of the anypath routing mechanism has been tested in a small multi-channel wireless mesh network. The test showed the performance of both the AP-OLSR and the AP-Bonding. Both components worked as expected, the AP-OLSR found both possible next hops and both next hops were used by the bonding module when packets were forwarded. The same test was also performed using single path routing, which resulted in a 28 % longer average delay.

Chapter 6

Conclusions

6.1 Discussion

The goal of this thesis was to develop and evaluate a novel routing mechanism for multi-radio multi-channel mesh networks following the anypath paradigm. In this thesis a design for an anypath routing mechanism has been developed, and an implementation of the design has been tested.

The test shows that routing mechanism following the anypath paradigm can decrease the end-to-end delay in multi-channel wireless mesh networks. For example, in the simple scenario tested in this thesis, the end-to-end delay was decreased by 22 %. The end-to-end delay can be decreased compared to single path routing due to the possibility for the nodes to have several next hops for each destination. Having more next hop candidates for a destination increases the probability of being able to send without a required channel switch on the sending interface.

6.2 Future Work

This section presents topics for future work for the anypath routing mechanism from this thesis. Tests in larger networks, encompassing additional nodes, need to be performed to identify a proper value for the constant α (used in the loop prevention mechanism). Tests in larger networks will also detect possible re-ordering problems.

Other than more tests, routing algorithms have also much room for improvements. If the end-to-end delay between nodes is decreased due to the anypath routing, the routing algorithm should consider this when calculating the path costs. Different paths are affected differently by the anypath routing mechanism.

6.2.1 Alpha Value

The α value is used to calculate the number of allowed hops for a packet before its destination must be reached (see section 3.3.2 on page 31). Because of the network topology and size in the test performed in this thesis, no loops could occur and therefore the loop prevention was not tested. A test in a larger network, containing more nodes, is needed to find a proper value for α.

Since the number of allowed hops is calculated as the required hops of the average best path multiplied with α, the α value sets the restriction for each packet to choose its path. Different values of α need to be tested in order to see the effects on the packets' arrival time. The lowest value for α is 1, which means that no node along the path is allowed to forward a packet to a next hop resulting in a detour. A high value gives the nodes many opportunities to gain time by selecting next hops that are better for the moment, but if the nodes start circulating the packets the arrival time will increase.

6.2.2 Re-ordering

A problem that could occur when using anypath routing is re-ordering, which means that the packets arrive in a different order than they were sent in. Re-ordering could happen when packets are sent in parallel paths.

In anypath routing, packets within the same stream might be forwarded to different next hops because of channel switches. When the stream is split, the packets may arrive re-ordered because the actual path delay may differ from the estimated average delay.

Within the test performed in this thesis, there was no re-ordering among the packets. The test were performed in a small network, which decreases the probability of re-ordering to occur. The probability for re-ordering increases with the number of nodes along the path that have several next hops.

6.2.3 Routing Algorithms

Routing algorithms of today have much room for improvements to better suit the anypath paradigm regarding path cost calculation. Those routing algorithms consider only the average best next hop, instead of the entire set of possible next hops. If the anypath routing mechanism is able to select the best next hop for each packet, the end-to-end delay between two given nodes should be lower than the average best single path.

See figure 6.1 on page 62 for an example topology, where the environment from the test (see section 5.2 on page 53) is a part of the network. When the routing algorithm calculates path costs for single paths routing mechanisms, the average path cost between node B and node R is around 8 ms, but when the anypath routing is used the actual path cost between the two nodes is lower (see results from test on page 56).

A routing algorithm should consider the set of next hops to better suit the anypath paradigm. A regular routing algorithm adds the links one by one, and therefore it cannot consider alternative next hops to reach a node further away. When alternative next hops

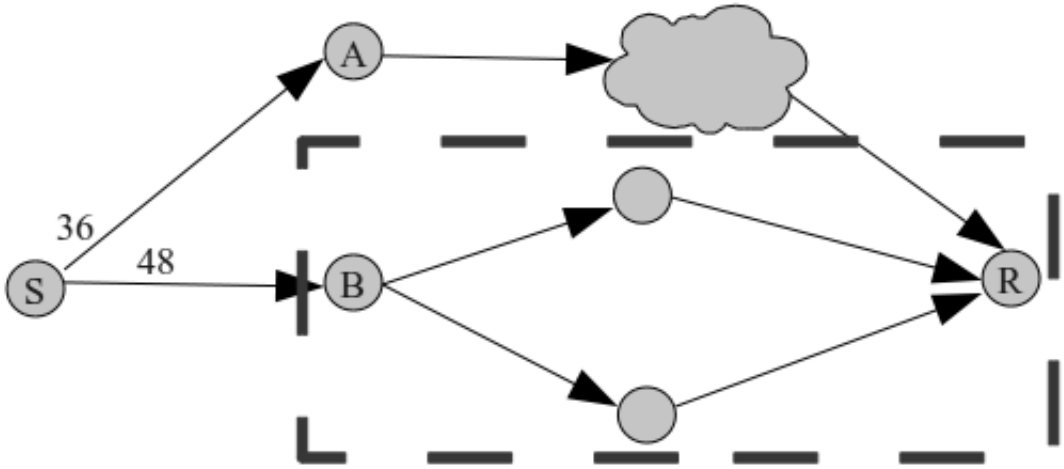


Figure 6.1: A network which the test environment is a part of

are considered, the routing algorithm is capable of estimating the switching cost for any of the feasible channels and an estimated average cost for the different paths.

References

[1] Thomas Clausen and Philippe Jacquet, Optimized Link State Routing Protocol (OLSR), IETF RFC 3626, Oct. 2003, Available: http://www.ietf.org/rfc/rfc3626.txt [Accessed Sept. 17, 2009].

[2] UniK OLSR daemon software, Available: http://www.olsr.org/ [Accessed Sept. 16, 2009].

[3] Implementing a fully distributed certificate authority in an OLSR MANET Dhillon, D. (RSA Security Inc., Vancouver, BC, Canada); Randhawa, T.S. Wang, M. Lamont, L. Source: 2004 IEEE Wireless Communications and Networking Conference (IEEE Cat. No.04TH8733), p 682-8 Vol.2, 2004

[4] Thomas Aure, Metric-based routing framework and radio-aware shortest path selection for OLSR, May 2008, Available: http://www.velocitynetworks.org/olsr/master-thomasau.pdf [Accessed Sept. 16, 2009].

[5] MF-OLSR implementation, Available: http://www.velocitynetworks.org/olsr/eolsrd-0.5.zip [Accessed Sept. 16, 2009].

[6] Multimetric OLSR and ETT, A. Lavén, P. Hjärtquist, 5th OLSR Interop & Workshop, October 2-4, 2009 in Vienna, Austria

[7] Net-X webpage, http://www.crhc.illinois.edu/wireless/netx.html [Accessed Sept. 16, 2009].

[8] Pradeep Kyasanur, Chandrakanth Chereddi and Nitin H. Vaidya, Net-X: System eXtensions for Supporting Multiple Channels, Multiple Interfaces and Other Interface Capabilities, August 2006, Available: http://www.crhc.illinois.edu/wireless/papers/kyasanur2006Tech.pdf [Accessed Sept. 16, 2009].

[9] P. Kyasanur and N. Vaidya, "Routing and Link-layer Protocols for Multi-Channel Multi-Interface Ad Hoc Wireless Networks" Source: ACM SIGMOBILE Mobile Computing and Communications Review , Issue 1 (January 2006), p 31 - 43 Vol.10, 2006

[10] Implementing the expected transmission time metric for OLSR wireless mesh networks Esposito, P.M. (Grupo de Teleinformatica e Automacao - PEE/COPPE - DEL/POLI, Univ. Fed. do Rio de Janeiro, Rio de Janeiro, Brazil); Campista, M. Moraes, I.M. Costa, L. Duarte, O. Rubinstein, M.G. Source: 2008 1st IFIP Wireless Days, p 5 pp., 2008

[11] A high-throughput path metric for multi-hop wireless routing De Couto, D.S.J. (Comput. Sci. & Artificial Intelligence Lab., MIT, Cambridge, MA, USA); Aguayo, D. Bicket, J. Morris, R. Source: Wireless Networks, v 11, n 4, p 419-34, 2005

[12] A channel and rate assignment algorithm and a layer-2.5 forwarding paradigm for multi-radio wireless mesh networks Avallone, Stefano (Department of Computer Engineering, University of Naples, 80125 Naples, Italy); Akyildiz, Ian F. Ventre, Giorgio Source: IEEE/ACM Transactions on Networking, v 17, n 1, p 267-280, 2009

[13] Highly-resilient, energy-efficient multipath routing in wireless sensor networks Ganesan, D. (California Univ., Los Angeles, CA, USA); Govindan, R. Shenker, S. Estrin, D. Source: MOBIHOC 2001. Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking and Computing, p 251-4, 2001

[14] Multirate anypath routing in wireless mesh networks Laufer, R. (Comput. Sci. Dept., Univ. of California, Los Angeles, CA, USA); Dubois-Ferriere, H. Kleinrock, L. Source: 2009 Proceedings IEEE INFOCOM. 28th International Conference on Computer Communications, p 37-45, 2009

[15] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, Link-level Measurements from an 802.11b Mesh Network, in Proceedings of the ACM SIGCOMM04 Conference, Portland, OR, USA, Aug. 2004, pp. 121-131.

[16] M. Campista, P. Esposito, I. Moraes, L. H. Costa, O. C. Duarte, D. Passos, C. V. de Albuquerque, D. C. Saade, and M. Rubinstein, Routing Metrics and Protocols for Wireless Mesh Networks, IEEE Network, vol. 22, no. 1, pp. 612, Jan.-Feb. 2008.

[17] Data Structures and Algorithm Analysis in C++
Mark Allan Weiss
Addison Wesley

[18] Computer Networking, a Top-Down Approach Featuring the Internet
James F. Kurose, Keith W. Ross
Addison Wesley

[19] Mobile Communications
Jochen Schiller
Addison Wesley

[20] Multi-Generator, Naval Research Laboratory (NRL)
http://cs.itd.nrl.navy.mil/work/mgen/

[21] ARM Architecture
http://www.arm.com/products/CPUs/architecture.html

[22] P. Kyasanur, "Multichannel wireless networks: capacity and protocols", Ph.D. dissertation, University of Illinois at Urbana-Champaign, IL, 2006

[23] One laptop per child
http://laptop.org