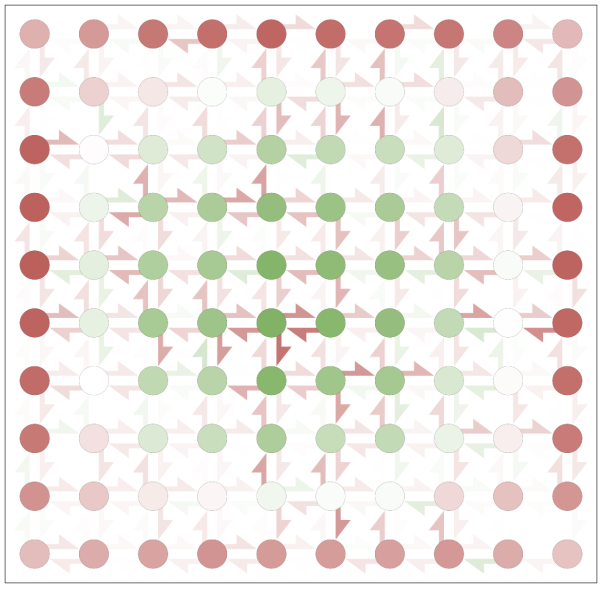
Protocol independent link state estimation and a novel throughput aware routing strategy for B.A.T.M.A.N. V

Communication Technology

Master thesis
Group 1052
June 3, 2021



Aalborg University
Department of Electronic Systems
Communication Technology
10th semester

Department of Electronic Systems
Communication Technology
Fredrik Bajers Vej 7B
9220 Aalborg
http://es.aau.dk/

Title:
Protocol independent link state estimation and a novel throughput aware routing strategy for B.A.T.M.A.N. V

Theme:
Master thesis

Project Period:
Spring Semester 2021

Project Group:
1052

Participant(s):
Daniel Greth Sørensen
Jeppe Nellemann Andersen
Rolf Krøyer

Supervisor(s):
Tatiana Kozlova Madsen
Sebastian Bro Damsgaard

Copies:
Digital version

Page Numbers:
106

Date of Completion:
June 3, 2021

Abstract:

This project investigates how to adapt the ad hoc network protocol B.A.T.M.A.N. V for use in a specific context of military communication. Making routing independent of lower protocol layers, along with improving the routing to avoid links with high packet loss, in an attempt to lower overall packet loss. Implementation and evaluation of this is done through simulation.
Part one of this project investigates the influences of interference and how it is modelled and simulated.
Part two of this project proposes a solution for estimation of link throughput independently of lower layers. This method relies on packet trains and information sharing between neighbours. Results show that this method lowers overall packet loss compared to B.A.T.M.A.N. V when assuming no information is available from lower layers.
Part three investigates how to make a representative estimation of multi-hop routes end-to-end throughput. It is found that B.A.T.M.A.N. V's hop penalties are not representative of this, and therefore a new method is proposed that is meant to be more representative and better adapted to heterogeneous links. Results show that this method generally lowers overall packet loss, and that routing avoids areas with high packet loss by the use of longer routes.

Department of Electronic Systems
Communication Technology
Fredrik Bajers Vej 7B
9220 Aalborg
http://es.aau.dk/

Titel:
Protokol uafhængig link tilstands estimering og ny thorughput opmærksom routing strategi til B.A.T.M.A.N. V

Tema:
Master thesis

Projektperiode:
Forårssemestret 2021

Projektgruppe:
1052

Deltagere:
Daniel Greth Sørensen
Jeppe Nellemann Andersen
Rolf Krøyer

Vejledere:
Tatiana Kozlova Madsen
Sebastian Bro Damsgaard

Oplagstal:
Digital version

Sidetal:
106

Afleveringsdato:
3. juni 2021

Abstract:

Dette projekt undersøger hvordan ad hoc protokollen B.A.T.M.A.N. V kan tilpasses til brug i konteksten af militær kommunikation. Ved at gøre routing uafhængig af lavere protokol lag, samtidig med at forbedre routing til at undgå links med høj pakketab. Implementering og evaluering er gjort igennem simulering.
Del et af projektet undersøger indflydelsen af interferens and hvordan det modelleres og simuleres.
Del to af projektet foreslår en løsning til estimering af link throughput uafhængigt af lavere lag. Denne metode gør brug af pakketog and informationsdeling mellem naboer. Resultater viser denne metode generelt sænker pakketab sammenlignet med B.A.T.M.A.N. V, når det antages at ingen information er tilgængelig fra lavere lag.
Del tre af projektet undersøger hvordan en repræsentativ estimering af flere hops ruters throughput. Der findes at B.A.T.M.A.N. Vs hop tab ikke er repræsentativ af dette, derfor foreslås en ny metode der er tiltænkt at være mere repræsentativ og bedre tilpasset heterogene links. Resultater viser at denne metode generelt sænker pakketab og at routing undgår områder med højt pakke tab ved brug af længere ruter.

Preface

This project is written by group 1052 from Aalborg University during the 10^{th} semester of Communication Technology.

Acronyms used in the report are shown under the section 'Acronyms' on the following page. The Vancouver system is used for citation, where a number is inserted within the text to refer to a specific reference, listed in the section 'References' on page 87. This page contains all details about the used references.

The code used for the simulations can be found in the following GitLab repository [1]. Here the folder called `inet` needs to be imported in OMNeT++ for it to work.

Without the patient guidance of our supervisors, this project could not be what it is today.

Acronyms

B.A.T.M.A.N. Better Approach To Mobile Adhoc Networking

BUS Bitrate, Usage and Success rate

COD Cross Over Distance

CSMA/CA Carrier-sense multiple access with collision avoidance

ELP Echo Location Protocol

ELSV Expanded Link State View

FSPL Free-space path loss

IFS Inter frame spacing

MANET Mobile ad hoc network

OGM Originator Message

OGMv2 Originator Message version 2

OLSR Optimized Link State Routing Protocol

PLR Packet loss ratio

PSR Packet success ratio

RTS/CTS Request-To-Send / Clear-To-Send

SPAN Smartphone ad hoc network

SWAP Sliding Window Adjusted Penalty

TAPT Triple Asymmetric Packet Train

TRGR Two-ray ground-reflection

TTL Time-to-live

TVLV Type Version Length Value

VANET Vehicular ad hoc network

Contents

1 Introduction

Networking and wireless communication is an important element of consumers and industries daily routines, often involving the internet. Today the majority of these needs are covered by WLANs or cellular networks such as 4G or 5G. These technologies rely on a wired backbone infrastructure to provide end-to-end communication over the internet. [2, 3]

However, this project focuses on another type of networks, namely ad hoc networks, these are often wireless and do not require any large or complex infrastructure to operate.

Over the years ad hoc networks have received a lot of attention due to their flexibility, their ability to self-form and heal. However, they have never received much adoption or usage compared to, for example, Wi-Fi. But especially with the recent increase in the number of IoT devices or other machine-to-machine communication scenarios, the adoption and usage has increased in the later years. Therefore, ad hoc networks are nothing new, but the usage and scenarios where they make sense to use are increasing. For example, there is an increase in the number of IoT devices or other machine-to-machine communication scenarios such as self-driving cars or satellites, where ad hoc networks are being applied. [3, 4, 2, 5]

These are scenarios where the ad hoc networks exist as intermediary or part of the previously mentioned infrastructure of the Internet, where they facilitate a different or often specialised kind of communication between devices in a more local area. Connecting either the devices with each other or with gateways that can bridge the ad hoc network's nodes with the larger Internet's infrastructure. [6]

Scenarios where ad hoc networks still function when isolated from other infrastructures also exists. Perhaps an easily imaginable use case is the military, which would like to have communication with no dependence on existing infrastructure. This can either be for resilience against said infrastructure going down or for being deployed in an area where using the existing infrastructure is not an option. [7, 8]

Ad hoc networks dynamic approach to organising networks allows for interesting properties and is one of the initial interests and motivations for this project's work. Furthermore, using ad hoc networks in a context such as large scale military communications is also of special interest, as ad hoc networks can provide the needed flexibility, robustness of communication and self-reliance. [7, 8]

This is where this project takes its initial inspiration from and guides how the project works with ad hoc networks. Furthermore, this project as a starting point uses a specific use case presented by Systematic to this project's authors, based in a meeting held in the beginning of the project. This meeting provided a better understanding of the communication scenario and the associated limitations and challenges. Along with insight into Systematic's wishes for usage of certain protocols. [9]

Additionally, this project follows a project done by the same authors investigating full-stack simulations of the ad hoc protocol Better Approach To Mobile Adhoc Networking

(B.A.T.M.A.N.) and will work from previous knowledge gained in that project. [10]

Both Systematic's use case and the previous work acts as the initial direction of the project, the preliminary analysis will investigate Systematic's use case, as well as the previous work and answer the following question:

What types of ad hoc networks exist, and what specific ad hoc protocol can be used in Systematic's use case, along with how can this be prototyped and evaluated?

2 Preliminary analysis

This section presents the initial analysis and knowledge about ad hoc networking and a more detailed description of Systematic's use case. Then followed by a short introduction to the ad hoc protocol B.A.T.M.A.N., ending with the choice of a test environment.

In ad hoc networks, nodes actively participate in the network and help with the communication between distant nodes by sending and forwarding control packets used for routing and likewise forwards data packets closer towards their final destination. Generally, there are three types of different ways of performing routing in an ad hoc network.

- **Proactive** routing
- **Reactive** routing
- **Hybrid** routing

In a proactive routing scheme, routes are maintained by periodically sharing information, thereby proactively keeping track of the current possible routing options, such that when needed a route is already known. Compared to a reactive routing scheme where routes are found only on demand. The difference here is that the overhead created by reactive routing can often be less because control packets are only sent when a route is needed. However, the trade-off is then in the delay it takes to send some data. Because in a reactive protocol the routes need to be established before information can be sent, compared to a proactive protocol that maintains all of these beforehand. However, these are generalisations, as the usage of the network or how routes are maintained while in use, heavily affects the performance trade-offs for specific protocols no matter the type. Lastly, there also exist hybrid routing schemes that attempt to do a mixture between the two. [11, 12, 13]

Ad hoc networks can be used in many different unique contexts, to which they can be adapted, such that the protocol can perform a specialised purpose in the best way possible. Some of the more well-known applications of ad hoc networks are:

- **Mobile ad hoc networks (MANETs)**
- **Vehicular ad hoc networks (VANETs)**
- **Smartphone ad hoc networks (SPANs)**
- **Wireless mesh**

The routing is one of the main parts of an ad hoc network or protocol, as it is this that defines performance trade-offs and functionality. One of the main trade-offs is the overhead versus the delay, as mentioned with reactive and proactive protocols. Another main element of ad hoc protocols is the scalability of the networks, this can either be

in size measured in the number of nodes, the amount of data traffic the network can support or the density of nodes, where all of these elements most often are tied together in trade-offs, for example, it is easier to have many nodes if the data traffic is low. [11, 12, 13]

Some of the concepts that persist across most ad hoc protocol:

- **Overhead** is an important metric in ad hoc networks, as it is often large and is determined by the control packets that in turn define functionality. The overhead metric itself is the indirect amount of airtime required for delivering the payload itself. This airtime can both be active usage of a medium or the idle periods required by the protocol.
- **Delay** is another important metric in ad hoc networks, as it tells how fast data can propagate through a network or how long discovery times are within the network, as multi-hop configurations are a fundamental part of this type of network. The delay often comes as a trade-off with the overhead.
- **Flooding** is an important mechanism in ad hoc networks as it is often used to share some information or find a route. Flooding works by forwarding an incoming packet onto outgoing links at every node. This is done so that the message is spread out across the entire network in a flood of packets, and often generates a large overhead as a result.
- **Loop avoidance** is essential in ad hoc networks, as routing or flooding loops can generate a large overhead. Loops occur when it is possible for a packet to reach a node more than once and the node does not realise this and repeats the message, sending the same messages around in a loop of nodes. Often it is handled with sequence numbers, time-to-live and specific routing rules.

2.1 Military communication context

This project works from some specific interest that is presented by the company Systematic. Early on in this project, a conversation was had with Systematic since they have an interesting specific use of ad hoc networks in one of their products. They have a product suite called SitaWare they have been working on. This product suite facilitates communication and organisation between parties in the context of defence, military or other strategic forces. This includes troops, vehicles and many other units, and the system should work between different forces that are cooperating, either nationally or internationally. [9]

The market for these types of solutions is large but also difficult, as the solution needs to integrate into existing equipment. Equipment that will differ from device to device and the underlying radio equipment can be anything from some of the most modern technologies to technologies as old as 30-40 years. Systematic, therefore, have a desire to be able to deploy this ad hoc network solution in many differing existing setups. To make this easier, the solution should be as independent of the underlying hardware as

possible. Therefore, there is a goal of making the ad hoc network protocol independent of the lower layers that provide the radio functionality.

Furthermore, Systematic has worked on this for quite a while and were willing to share some specifics regarding what traffic patterns could be expected in their use case and the network sizes and structure of their current solution, this is very useful for making more representative simulation scenarios.

They provided information regarding how such a network could look in size and hierarchy. They see the network structure in parallel with the hierarchical structure of the military, with a headquarter at the top branching out into more and more layers. The smallest unit of the network is what they call a community consisting of up to 500 nodes each, equating to something like a battalion or a company. All the communities can route between themselves and are connected together in an overlay network that can cover up to 70000 nodes in total. The focus of this project is only on the communities, as it is these that will be using an ad hoc protocol to facilitate communication between nodes.

The choice of ad hoc protocol is not necessarily obvious as there over the years have existed many different. But through conversation with Systematic, they explained that they consider B.A.T.M.A.N. as one of the more modern and promising ad hoc protocols under current development, and they even consider it the de facto standard in this area. This is a sentiment shared by the authors of this project as they previously also have worked with B.A.T.M.A.N. for similar reasons. Therefore, the choice of ad hoc protocol for the purpose of Systematic's use case and this project falls on B.A.T.M.A.N..

However, Systematic also knows that B.A.T.M.A.N. needs to be adapted to their use case, specifically, one of the main elements they focused on is that B.A.T.M.A.N.'s routing functionality needs to be made independent of lower protocol layers, as this is not the case currently. This is therefore the main problem this project will focus on. But Systematic also mentioned many other problems and modification that they imagine needs to be considered in B.A.T.M.A.N., of these, two additional problems are deemed interesting for this project and are presented below:

1. Being aware of throughput variation, as Systematic care about not just the shortest or fastest route.

2. Avoid links with high packet loss, this also ties into caring about not just the shortest or fastest route.

2.2 B.A.T.M.A.N. protocol

This project works with the B.A.T.M.A.N. protocol specifically B.A.T.M.A.N. V, but for simplicity as this is the only protocol version addressed in this project any reference of B.A.T.M.A.N. refers to this specific version. B.A.T.M.A.N. is made by the German community Freifunk who created it as a replacement for Optimized Link State Routing Protocol (OLSR) which is another older ad hoc routing protocol. Freifunk uses B.A.T.M.A.N.

for large noncommercial computer networks, that is meant to provide free and decentralised Wi-Fi access. Shown below in figure 2.1 is a picture of the B.A.T.M.A.N. communities that exists within Germany.

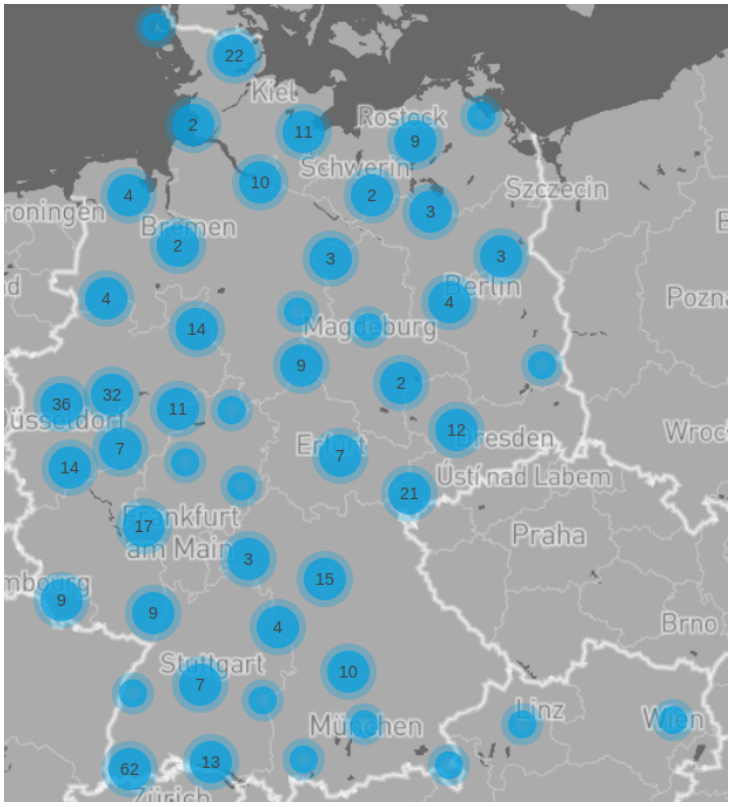


Figure 2.1: Map of mainly Germany, showing locations and a count of existing networks within the Freifunk community that provide free Wi-Fi access to these regions using B.A.T.M.A.N.. [14]

B.A.T.M.A.N. is a proactive routing protocol that locally at each node keeps a list of all nodes in the network and what the best hop "direction" is to reach any other node. Each node will forward packets to a given destination through a specific neighbour. The used protocol version of B.A.T.M.A.N. relies on two different control packets for this routing functionality, Originator Message (OGM) and Echo Location Protocol (ELP) packets. [15, 16, 17]

By only saving the direction of other nodes, B.A.T.M.A.N. decentralises the knowledge about a route across the network. Making it so that no individual node has all the global information of the network or routes between nodes.

A routing direction is found by every node periodically sending an OGM packet, this packet is then flooded through the network both showing the presence of the device within the network, but also the direction the OGM packet arrived from, is the direction back to that node. Furthermore, as B.A.T.M.A.N. only keeps one next best hop for a node, a routing metric is used, this helps a node determine what direction is the best towards a node in case it sees an OGM packet from multiple directions. The routing

metric B.A.T.M.A.N. uses is a bitrate transfer speed, this metric is firstly measured between every node and is done by the Wi-Fi module in a node. Then when an OGM packet is sent or forwarded this metric is sent along as well. Then depending on the link the OGM packet is received on, this metric in the OGM packet is either applied a penalty or set to that links estimated bitrate transfer speed, depending on what is the lowest, as this would be the limiting factor. [15, 16, 17]

The ELP packets are generally used for identifying direct neighbours and estimating the link capacity between them.

2.3 Choice of test environment

For testing different solutions and evaluating them, a test environment is needed. Ideally, this is done on a real-world testbed, as that is the most representative. But there are some difficulties with doing this, data needs to be collected across many devices and furthermore be synced up, as well as setting up large scale tests with many devices is both expensive and time-consuming.

Therefore, simulations are considered instead, as simulations offer a faster and cheaper setup of tests with a large number of nodes. This in turn also allows for faster prototyping and iterating through potential solutions faster. Equally, evaluation can also be done faster as many different network configurations can be generated fast and tested concurrently. However, simulations are limited by how accurately they model reality. However, good simulations often get close enough for most metrics. Therefore, simulation is chosen as the test environment for prototyping solutions and evaluating them.

For the choice of the simulation tool, OMNeT++ is chosen, it is a discrete-event simulator which means that time jumps from the current event directly to the next event in time. This often has the advantage of being fairly efficient since time is not simulated when no system state change is happening, which is the case for simulating in discrete fixed-size time steps. OMNeT++ allows for simulating a full protocol stack, which is essential for the work with B.A.T.M.A.N. in the context of other lower layer protocols. Furthermore, OMNeT++ can easily be integrated with many accurate frameworks and models from INET.

The choice of OMNeT++ is partly because of existing experience by the authors of this project [10]. From this work, a simulation model with B.A.T.M.A.N. is already made and will be reused as the base for this project's simulations. This has the benefit of allowing for reaching the prototyping phase faster.

2.4 Previous work

This project uses the results and simulation setup from an earlier project by the same authors, titled "Full stack simulation of B.A.T.M.A.N. V", as a foundation [10]. This project examined the behaviour of a simulation model of the ad hoc routing protocol B.A.T.M.A.N. and compared the observed behaviour to the expected. Knowledge about

B.A.T.M.A.N.'s expected behaviour was acquired by examining and cross-referencing the B.A.T.M.A.N. source code, the documentation and a network trace file. Since no real-world B.A.T.M.A.N. network was available, the evaluation of the behaviour of B.A.T.M.A.N. was based on three different simulation scenarios. In these scenarios, the simulated nodes were either placed on a line, in a grid or moved randomly around. Besides examining the behaviour of B.A.T.M.A.N., the performance was evaluated through the use of common network metrics, such as end-to-end delay, Packet loss ratio (PLR) and protocol overhead.

The project generally concluded that the simulation model of B.A.T.M.A.N. behaved according to the available protocol information. This was apparent in the simple line scenario where the network metrics followed the expected trends. While both the grid and mobility scenario were more complex, the results somewhat followed the predicted tendencies for networks with a low number of nodes. However, for medium- and large-sized networks, the PLR was drastically higher than expected.

This unexplained discrepancy between the expected and observed packet loss also serves as one approach for this project, as it ties in with Systematic's intention of using B.A.T.M.A.N. in large scale networks. While also using the same simulation environment as a basis for the work and simulations in this project.

3 Problem Statement

In the previous section, ad hoc protocols are introduced as the main interest of this project, specifically the interest in the use case presented by Systematic, of a real-world application of ad hoc networking. This application desires a protocol with functionality that is as self-contained within the protocol itself as possible, allowing for ease of deployment on different radio setups. With further specific interests into bandwidth variation awareness for routing not just along the shortest routes and focus on stable communication by avoiding links with packet loss. Furthermore, B.A.T.M.A.N. is considered as a potential ad hoc protocol for this purpose, which aligns well with the authors existing experience with B.A.T.M.A.N. and simulating B.A.T.M.A.N. in a full protocol stack [10]. This full-stack simulation is how this project will prototype and evaluate solutions to the specific adaptions of the B.A.T.M.A.N. implementation. Solutions that could suit the specific needs and interests of the use case, and with the hopes that general improvement suggestions for the B.A.T.M.A.N. protocol are also possible.

Below is the main problem that this project will involve itself with, along with how this is broken down into more detailed subproblems:

What can be done to adapt B.A.T.M.A.N. to the use case given by Systematic, such that B.A.T.M.A.N. becomes more independent of the lower layers and chosen hardware?

This problem is considered in three smaller and more independent parts that each investigates different elements for collectively answering the main question.

Part one considers the simulation environment used for solutions to the main question and evaluation thereof. With a focused investigation into interference and packet loss that stems from experience gained in previous work [10]. The goal is to have a more representative simulation environment for more accurate results, and a better idea of what can be done to reduce the overall system-wide PLR. This is summarised as the following subproblem:

1. *In previous work [10] a larger than expected amount of packet loss was observed in simulation, what is the reason for this, and what can be done to alleviate this?*

The second part addresses the problem of making B.A.T.M.A.N. independent of the lower layers. As B.A.T.M.A.N. uses information from the network hardware to get a bitrate, that is then used as a routing metric. This dependency is what this project aims to remove, by replacing it with a way of estimating the bitrate within the protocol layer itself. The initial ideas for this are based on measuring the time between packets in a way that is usable for estimating the underlying bitrate the radio operates at.

Besides finding a different way of estimating the same underlying bitrate it would also be nice if further optimisations can be made to the routing so that it factors in packet loss, allowing for routing around problematic areas. As it is expected that the first part will show that nodes lose packets unevenly and that the system-wide PLR can be alleviated

by avoiding links with high loss. The initial idea for incorporating this into the link metric is to measure usage for considering congestion, and by measuring sent packets versus received packets for packet loss. Resulting in using a link and routing metric that is more representative of the possible throughput of a link. All while still keeping the link metric estimation independent of the lower protocol layers. This is summarised into a subproblem that considers the independence of the routing metric and link conditions affecting the routing metric:

2. *How can the routing metric be made independent of the lower layers, while considering differing link conditions in the routing metric so that it is more representative of the usable transfer rate of individual links?*

The final part addresses how B.A.T.M.A.N. makes routing decision for multi-hop routes, as there is a suspicion that B.A.T.M.A.N. is a bit too punishing of long routes and B.A.T.M.A.N. generally goes for the shortest route, partly due to a rudimentary hop penalty. But an element of the main problem is that links with high packet loss should be avoided and longer routes might be needed for this, therefore, a more representative hop penalty should be applied so that the multi-hop metric at each node is closer to a route's actual possible throughput, especially if the link metric is modified. This is summarised into the following specific subproblem:

3. *How does B.A.T.M.A.N. handle multi-hop routing decisions based on individual link metrics and can a more representative end-to-end route metric help?*
    - *If a different routing metric is suggested, that consider link differences, how does multi-hop routing handle these heterogeneous links?*

4 B.A.T.M.A.N.

B.A.T.M.A.N. is a multi-hop ad hoc protocol developed by a community called Freifunk stationed in Germany. B.A.T.M.A.N. was originally released in 2008 and the protocol has since been through several iterations, the algorithm is currently at its fifth iteration called B.A.T.M.A.N. V, which is also the algorithm focused on in this project. The protocol was originally developed as a large-scale decentralised network to interconnect wireless LANs, in urban areas. Currently, there are several of these networks running in Germany, for example, Hamburg has a very extensive network as shown in figure 4.1. The Freifunk community is also maintaining an implementation of B.A.T.M.A.N. called B.A.T.M.A.N.-Advanced. This implementation functions as a Linux kernel module and is placed between a normal network interface, for example, Wi-Fi or Ethernet, and the IP-layer.

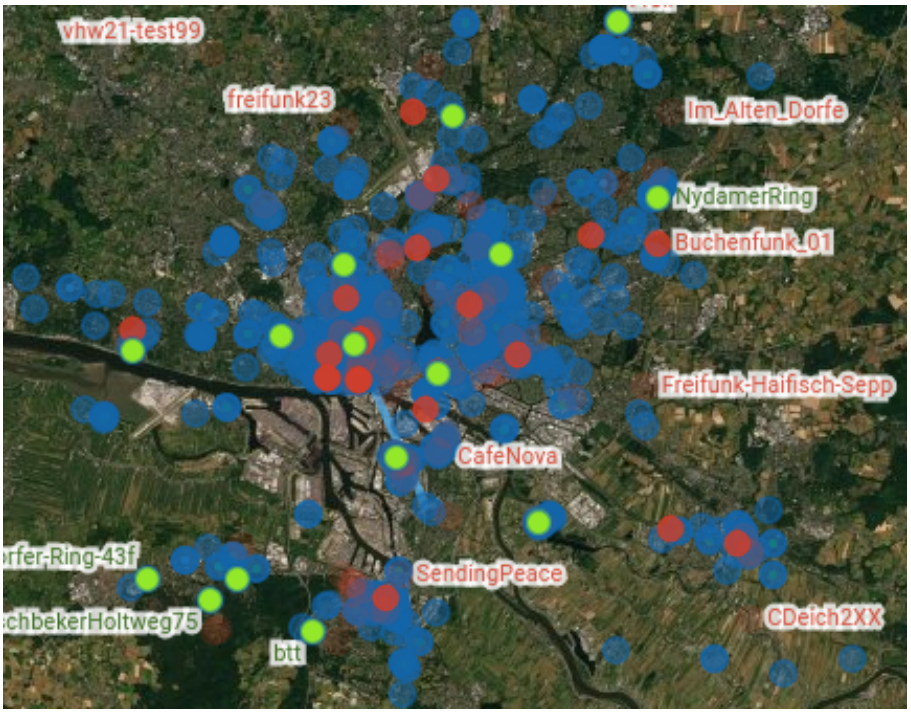


Figure 4.1: B.A.T.M.A.N. nodes in Hamburg. [18]

4.1 Definitions

The following is a list of definitions of some of the essential terms used in B.A.T.M.A.N. and are taken and quoted directly from the documentation [15].

- Node: A mesh router which utilizes the B.A.T.M.A.N. protocol as specified in this document on at least one network interface.
- Originator: A node broadcasting its own OGMs that is therefore addressable within the mesh network routing layer. It is uniquely identifiable by its originator address.

B.A.T.M.A.N.-Advanced uses the MAC address of its primary hard interface.

- Neighbor: An originator within one hop distance.
- Router: A neighbor which is a potential, loop-free next hop for forwarding data packets towards a specific originator.
- Link throughput: link throughput from one interface to a neighbor's interface.
- Best link throughput: The best of all link throughput values towards a neighbor.
- Path throughput: The incoming OGM's throughput combined with the best link throughput of the according neighbor.

4.2 Control packets and routing tables

The rules in this section are quoted from the B.A.T.M.A.N. documentation[15].

B.A.T.M.A.N. is a proactive routing protocol, which means that every node in the network maintains a list of other nodes in the network and how to route towards any given node. Which is in contrast to a reactive routing protocol that first establishes a routing table to a given node at the time it is needed. In the case of B.A.T.M.A.N. V, it maintains two lists, a neighbour list and an originator list.

The neighbour list contains neighbour nodes and parameters attributed to the neighbour nodes, for example, the estimated link throughput from the neighbour and a last seen timestamp. This neighbour list is created with the help of ELP packets. Each node broadcasts an ELP packet with a given interval, as standard this is every $500ms$, to let neighbour nodes known of its existence. The ELP packet frame is shown in appendix G.

The ELP packet can also be sent as a unicast packet called an ELP-probe packet. Every node sends two ELP-probes to all its neighbours at the same interval as the broadcast ELP. The ELP-probe uses the same frame as the ELP but is appended with $184B$ of data.

The originator list functions as a routing table and contains a list of all reachable participating mesh nodes and parameters attributed to these nodes, for example:

- Next best hop, which is what neighbour to send to
- Estimated path throughput to the given node
- Last seen timestamp

This list is created and maintained through the Originator Message version 2 (OGMv2) packets, which every node broadcasts with a given interval, as standard once every second. The OGMv2 packet contains the address of the originator, the estimated throughput from the originator to the node receiving the OGMv2, a sequence number and a Time-to-live (TTL), the OGMv2 packet frame can be found in appendix G. These OGMv2 packets are

flooded through the network and when a node receives such a packet, some preliminary checks are made where the most relevant check is:

- Own Message Check: If the originator address of the OGM is our own the message must be silently dropped as this OGM originated from this node.

If the packet passes the preliminary checks, the packet is checked if the neighbour is a potential router. If the node has received an OGMv2 packet from the originator before, the sequence number of the packet is checked, if it is newer than the last received packet, it is allowed to pass.

Then the internal node statistics are updated, according to the rules from the B.A.T.M.A.N. documentation[15]:

- The last seen timestamps of the router and the originator are updated
- The last sequence number and ttl values are adopted
- If the link throughput to the neighbour this OGMv2 was forwarded by is lower than the path throughput of the OGMv2, then this lower link throughput is adopted.
- Forward penalties are applied:
  - If the considered interface is the default interface, no penalty is applied
  - If the incoming and considered outgoing interface is the same half duplex interface and the reported throughput is larger than 1 MBit/s, the throughput is reduced by 50%
  - Otherwise, a hop penalty is applied and the throughput is reduced by the according value (default 5.8% or 15/255). This is especially useful for "perfect" networks to create a decreasing metric over multiple hops.
- The throughput value with the penalties applied is stored for the router

Lastly, a check is done if the router shall be updated and the OGMv2 shall be rebroadcast

- If the OGMv2 has been received from the best router, no change is necessary
- If no router has been selected yet, the received router becomes the selected router immediately
- If the throughput from the received router is higher than the throughput via the selected router, the received router becomes the selected router
- Also, if the sequence number is by at least OGM_MAX_ORIG_DIFF higher than the last received sequence number from the selected router, the received router becomes the selected router.

Then the OGMv2 packet is prepared for rebroadcast unless another packet with the same sequence number has already been forwarded. Before the packet is sent the TTL is decremented, if it reaches zero the packet is dropped and not forwarded.

The OGMv2 packet also carries what B.A.T.M.A.N. calls Type Version Length Value (TVLV) frames. These TVLV frames can carry data such as placement of gateways or information of non-mesh nodes. These TVLV frames are not relevant to the project and will not be discussed further.

4.3 Link state evaluation of B.A.T.M.A.N. V

Which route a packet is going to take in a B.A.T.M.A.N. V network is evaluated through a throughput metric. This throughput metric utilises a function found in most Wi-Fi interfaces, that supplies a throughput between two Wi-Fi interfaces at the MAC layer when there is traffic on the link. Because there needs to be traffic on a link for the Wi-Fi interfaces to negotiate the throughput between the nodes, B.A.T.M.A.N. V utilises the ELP-probes so that every $0.5s$ a node will send ELP-probes to its neighbours, in the case where there has been no other traffic on the link in the last $0.1s$. If the interface used by a node is not one of the Wi-Fi interfaces capable of supplying the throughput, a standard throughput value of $10Mbit/s$ is set for the link. This way of setting the throughput metric is referred to as *throughput override* in the documentation for B.A.T.M.A.N. and going forward any mention of *throughput override* will similarly refer to this method.

To evaluate routes a throughput value is set in the OGMv2 packet, this value is then used to determine what route a packet should take. The originator of the OGMv2 starts by setting this value to the max throughput value, which has a standard value of 0xFFFFFFFF, and then broadcast the OGMv2. The recipients then look up what the throughput of the link between the originator and the recipient is, and compares this to what is in the OGMv2 packet, and chooses the lesser of the two values. In figure 4.2 this means that if **A** broadcast an OGMv2 and **B** receives it, **B** will set its throughput toward **A** to $20Mbps$.

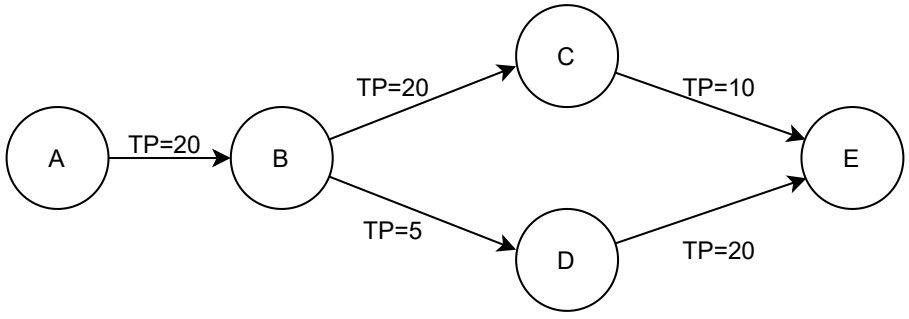


Figure 4.2: Network example with throughput values in Mbps.

When a node forwards an OGMv2 it needs to know if the interface is full or half-duplex, if the interface is half-duplex and the throughput is above $1Mbps$ the throughput is halved. If the interface is full-duplex or the throughput is below $1Mbps$, a hop penalty is applied,

this hop penalty is set to reduce the throughput by 5.88% at each hop. Again in figure 4.2 **B** forwards the OGMv2 from **A**, and sets the throughput to $10Mbps$. This OGMv2 is then received by **C** and **D**, here **C** has estimated the throughput of the link from **B** to be $20Mbps$ and will thus keep the $10Mbps$ as the throughput toward **A**. **D** on the other hand has estimated the throughput to be $5Mbps$ and will set this to be the throughput toward **A**. Both **C** and **D** will again forward the OGMv2, with the throughput halved, and **E** will receive it. Here **E** will choose **C** as the next hop toward **A** since it carries the highest throughput.

5 Part 1 - Simulation environment and interference modelling

This part describes the simulation environment used throughout this project. Then followed by an investigation into the influence of interference in varying sized networks, and the path loss model used for simulations.

5.1 Simulation environment

Throughout this project, the method of testing and evaluating different possible solutions is via simulation. The simulation tool chosen is an open-source discrete event simulator called OMNeT++[19], specifically version 5.6.2, which is written in C++. Furthermore, the INET Framework[20], which is a model library containing detailed models of various common network protocols from all layers of the network stack, is used in conjunction with OMNeT++. As an example, INET provides models for protocols such as IEEE 802.11 (Wi-Fi), IPv4, Ethernet and TCP/UDP and in addition, it also provides some simple models for simulating mobility. The version of INET used for this project is 3.6.7.

However, INET does not provide any model for the B.A.T.M.A.N. protocol. So for simulating B.A.T.M.A.N., a model, which originally was written for a paper by Benjamin Sliwa et. al.[21], is used. The source code for this model is based on version v2018.0 of the batman-adv Linux implementation and is an almost one-to-one copy of this, providing confidence in the model.

Furthermore, the project group has previously worked with the mentioned B.A.T.M.A.N. model in conjunction with OMNeT++ and INET, where some changes were made to the model, such that it more correctly plays together with the rest of the network stack. Among these changes were:

- Changed the encapsulation for the packets entering and leaving the B.A.T.M.A.N. layer so that they are encapsulated with the correct headers, compared to a real network trace file
- Added functionality for easier generation of random pairs of transmitter and receiver nodes used for UDP traffic
- Added a new startup jitter, offsetting nodes more at time zero, allowing for faster settling time
- Added more logging of metrics and networks statistics, mostly specific to the B.A.T.M.A.N. layer

Only one unique protocol stack is simulated throughout this project. This protocol stack is shown below according to the TCP/IP model with Wi-Fi chosen as the lowest layers. This protocol stack is equivalent to how B.A.T.M.A.N.-Advanced would be used by Freifunk in a wireless setting.

- UDP header
- IP header
- Ethernet frame
- B.A.T.M.A.N.
- Wi-Fi (IEEE 802.11 g(erp))

A visual representation of how a data packet is encapsulated using this protocol stack is shown in appendix G. All layers in this protocol stack except B.A.T.M.A.N. are simulated using the aforementioned INET modules.

5.2 Interference's impact on packet loss

As mentioned in section 2.4 there is observed a higher PLR at a lower amount of nodes than expected. The cause of this loss is important to understand to see what can be done in regards to lowering it.

5.2.1 Previous results

A subset of the previous results are shown in figure 5.1 and come directly from the previous project [10]. This subset of results is from the scenario were the highest PLR is observed.

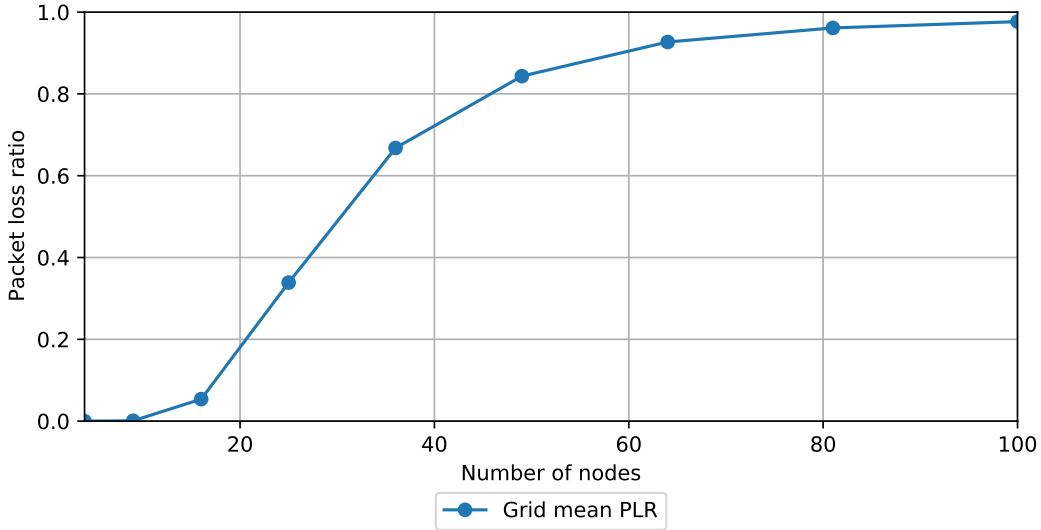


Figure 5.1: Accumulated PLR for multiple UDP data streams in a grid scenario.

This figure shows the results of the previous project, with the nodes' radio using a $54Mbit/s$ bitrate, placed in a grid pattern and every node being in communication range with its immediate neighbours, vertically and horizontal, not including their diagonal neighbours. The traffic generated is $1200B$ packets every $120ms$ from every node to another random node. The results presented in figure 5.1 shows a large increase in the PLR at already 24 nodes being above 30% and increases to above 90% at 64 nodes and slowly converging towards 100% after this.

The numerical values of this set of measurements are not addressed in [10], because the tendencies of the three measured metrics are the interest of that project and these generally behaves as expected. One of the concerns that these numerical values might show, is that the scalability of B.A.T.M.A.N. in the grid scenario is generally not very promising with the used configuration parameters.

The use case presented by Systematic describe a network with communities where each community is expected to be an independent B.A.T.M.A.N. or similar type of network, with each community containing up to 500 nodes. Therefore, the packet loss observed in [10] needs to be investigated and addressed, for Systematic's use case specific application of B.A.T.M.A.N., to be as scalable as desired.

During the previous project[10], this warranted some considerations, the main speculation, during [10], is that the majority of the packet loss occurs due to interfering radio signals, causing congestion or loss. However, this is not fully confirmed or documented, since it is not the main focus of that project.

5.2.2 Free-space path loss model

The interference in the configuration shown in figure 5.1 is modelled by the INET v3.6.7 radio propagation model `FreeSpacePathLoss`. The Free-space path loss (FSPL) equation is shown in equation 1. For further details regarding this equation see appendix A.1.

$$FSPL = \frac{P_t}{P_r} = \left(\frac{4\pi d}{\lambda}\right)^2 \tag{1}$$

Changing the radio propagation is the starting point for this investigation, where the main idea is to test the effects of the interference by removing it and seeing the difference in performance. The main performance metric used is the same PLR as in the previous project [10], which is the system-wide PLR. This will be simulated in the same grid scenario as the previous project as this scenario is still a good synthetic benchmark setup with random communication pairs and 2-4 connections per node.

This is tested by adjusting the range of the interference, to being equal to the communication range, and one where it is so low that no nodes experience interference at all. Then this is simulated with a varying sized grid network, for these two interference ranges and the default value for reference. See appendix A.2 for more information regarding the specific simulation parameters and a figure showing the size difference of the interference ranges. The result of this simulation is shown in figure 5.2.

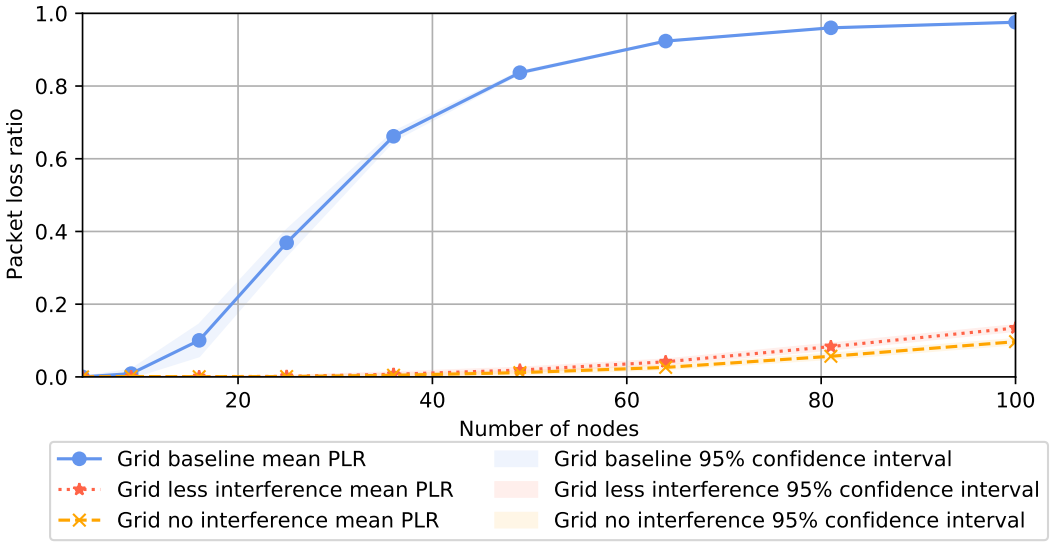


Figure 5.2: The mean PLR measured for the grid scenario, with interference, with interference equal to communication range and no interference.

The reduced interference shows a large difference, reducing the PLR to around 10% for no interference and 15% for interference reduced to communication range. This seems to confirm that the majority of packet loss is indeed caused by interference since it is the only element that is different between these simulation scenarios. This shows the impact of interference for the PLR.

The first attempt to address this is by considering how representative the FSPL model is compared to real life and whether there are better alternatives. As the FSPL model in some cases can be considered the worst case for how far the radio signals travel. The goal is to find a model that is more realistic and comparable to a physical test environment, while also hoping that this will lower the general amount of interference and thereby lower the packet loss.

5.2.3 Two-ray ground-reflection model

For finding an alternative path loss model that is more representative, Associate Professor Troels Bundgaard Sørensen from Aalborg University's Department of Electronic Systems is consulted since he has extensive knowledge regarding radios. From this, it is gathered that a more fitting model for modelling realistic outdoor radio propagation is the Two-ray ground-reflection (TRGR) model. This model considers two paths between the transmitting and receiving antenna; the direct line of sight path and a path where the signal is reflected once of the ground. The two paths will either constructively or destructively interfere with each other due to the differing travel distance to the receiving antenna. However, at longer distances the difference between the two paths becomes less and almost parallel, this generally causes a faster attenuation compared to the FSPL model because a reflection causes the radio waves to change phase by 180°. This means

the two almost parallel paths will destructively interfere with each other, making the signal attenuate faster, which leads to less interference or noise at longer distances compared to the FSPL model.

The constructive and destructive interference create a much more complex slope, compared to the FSPL model which is simple and smooth. An example, of how this complex interaction can look is shown in figure 5.3.

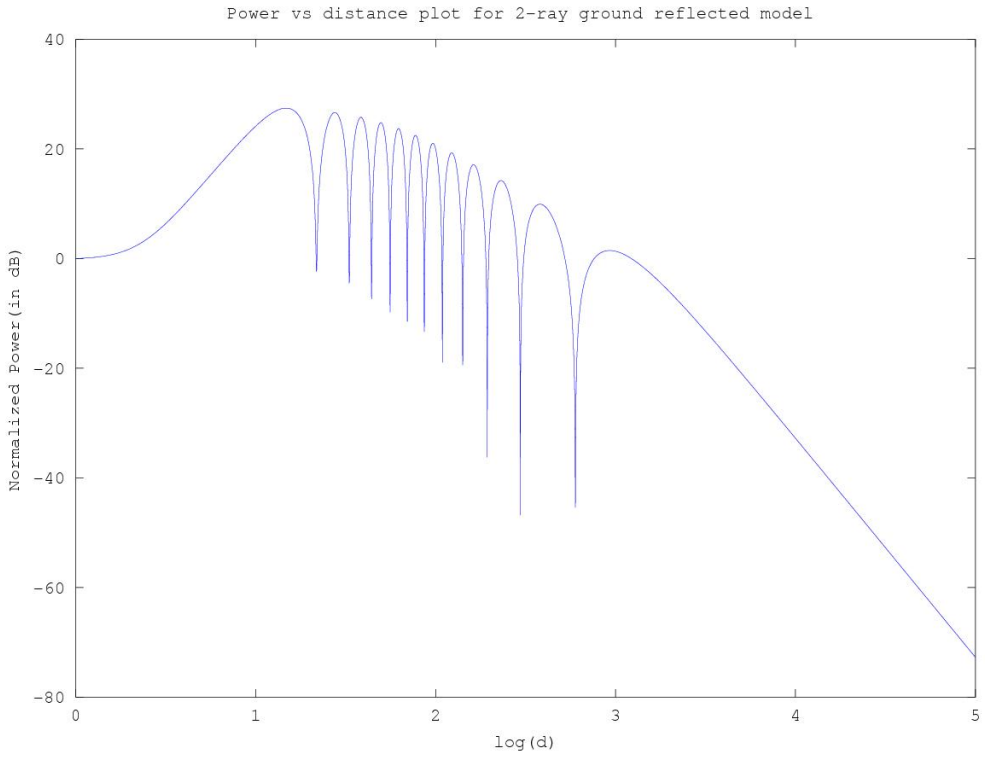


Figure 5.3: This graph shows how the signal power decrease with the distance between antennas, from [22].

Furthermore, Associate Professor Troels Bundgaard Sørensen also showed how the radio propagation can be approximated by two simple slopes, and that this from his experience and experiments approximately matches a real physical environment. The way this model works is that at close ranges the signal is modelled after the FSPL model and after a Cross Over Distance (COD) the model will change and follow a more attenuated relationship between distance and radio propagation. Due to the simplicity of this, the multi-slope approximation will be used.

Equation 2 shows the far field approximation for the TRGR model:

$$P_r \approx P_t \frac{G h_t^2 h_r^2}{d^4} \tag{2}$$

Where:

P_r is the power received

P_t is the power transmitted

h_t is transmitter antennas height

h_r is receiver antennas height

G is the combined antenna gain of the transmitter and receiver antenna $G = G_t G_r$

d is the distance between antennas

The COD at which the two models changes is calculated as shown in equation 3:

$$COD = \frac{4\pi h_t h_r}{\lambda} \tag{3}$$

Where:

COD is the Cross Over Distance

h_t is transmitter antennas height

h_r is receiver antennas height

λ is the wavelength of the signal

An example, of this multi-slope model, is shown in figure 5.4. Here the COD is plotted as the vertical line and shows where the slope changes. Both the TRGR and the FSPL model is plotted to show how they compare. Mainly the comparison shows that they are identical at shorter ranges and at the COD the attenuation will increase for the TRGR model.

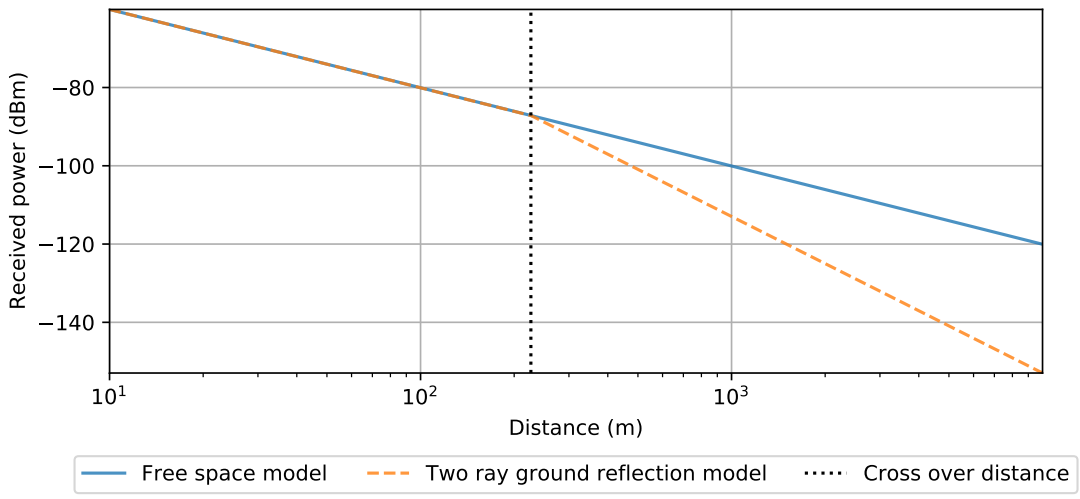


Figure 5.4: An example, of the multi-slope approximation of the TRGR model plotted as the orange line. The vertical line marks the COD, while the blue is the previously used FSPL model as a comparison.

To see how the interference ranges of the FSPL model and TRGR model compare see figure A.1 in appendix A.3.

To use the TRGR model the INET `TwoRayGroundReflection` model replaces the currently used `FreeSpacePathLoss` model in the simulation setup. How this can be done is shown in code snippet 1. The two main things that need to be changed is the path loss type and then defining a ground surface which the TRGR model interacts with and the height of the nodes above the ground. For all simulations in this project using the TRGR model all nodes are placed $1.5m$ above ground.

Code snippet 1: Configuration for using the TRGR model from INET.

```
1 Batmanadv.physicalEnvironment.groundType = "FlatGround"
2 Batmanadv.physicalEnvironment.ground.elevation = 0m
3 Batmanadv.radioMedium.pathLossType = "TwoRayGroundReflection"
4 *.batman5Router[*].mobility.initialZ = 1.5m
```

After implementing the TRGR model in the simulation setup, a simulation is run to see how it performs and compares to the previous results. This entails comparing against the FSPL model and seeing how much of a difference the new model makes, this simulation is run with the same parameters as the results of the FSPL model in figure 5.1, namely the same default parameters and 150 meters of separation between nodes. The results are shown in figure 5.5.

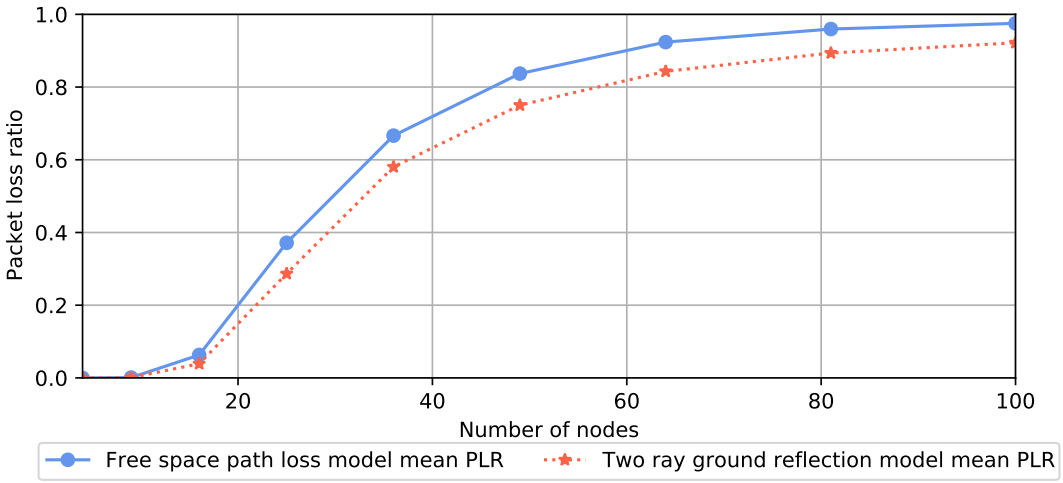


Figure 5.5: Comparison of the two radio propagation models' PLR under the same conditions.

This test actually does not show as much of a difference as expected, the hope is that the interference will be limited and can approach performance closer to the results with no or less interference. However, the TRGR model is more realistic and representative, while at the same time giving slightly better performance. Therefore, it seems suitable for being used as the propagation model going forward.

5.2.4 Local packet loss

Another consideration is to look at the latest simulation with the TRGR model, but instead of measuring the system-wide PLR, each node's individual links' PLR are measured. There is a general expectation that the nodes in the middle both have more radio noise reaching them and have more traffic routed through them as well and would therefore be more prone to experiencing packet loss.

To measure this, more logging information is gathered during simulation, so that every node's amount of sent and received packets are logged, and where they are sent and received from. This information is then combined with the network topology graph for showing the results in figure 5.6, with the colour of the arrows representing the PLR of the links.

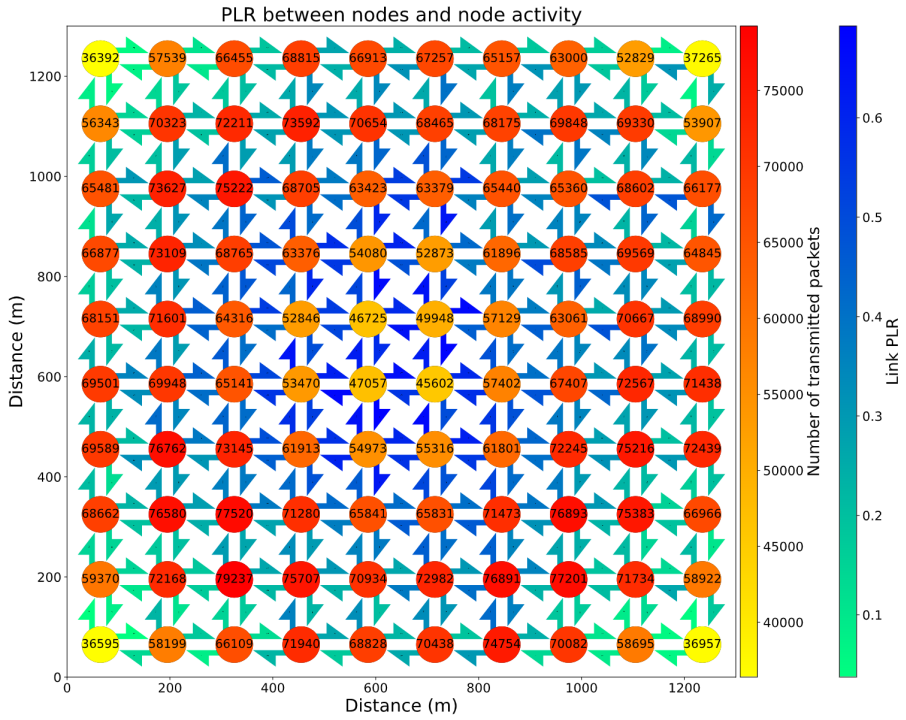


Figure 5.6: Grid scenario topology, with each node coloured according to the amount of traffic sent by it, and with the PLR for each unidirectional link shown as well.

The colouration of the arrows shows that there is a difference between each node's PLR. The nodes in the middle experience more packet loss than the nodes closer to the edge. For example, a node in the middle sends a total of 49948 packets and the PLR of its outgoing links is $\approx 67\%$, whereas for an edge node it is 70438 total packets sent with a PLR of $\approx 19\%$ to $\approx 33\%$.

The uneven spread of packet loss is important to be aware of, as packet loss could be a local issue for each node, these issues cannot be seen by only looking at system-wide packet loss. For example, some nodes at the edge of a network might experience little to no loss, while others in the center might. Therefore, addressing system-wide loss should

be focused on handling the nodes who experience the loss and not the network as a whole. These local differences, therefore, needs to be measured by the nodes so that this difference can be considered. Also, it is worth noting that the packet loss is not symmetrical between node pairs.

For example, maybe packet loss can be reduced by routing around hot spots of interference in an attempt to spread the load across the network. This hopefully could thereby reduce the overall system-wide PLR. For example, routing from one diagonal corner to the opposite corner should be prioritised to be done around the edge of the network as the hop count or Manhattan distance is the same as a route through the middle of the network.

5.2.5 Interference range and network utilisation

To see how the interference range is tied to the packet loss a new set of simulation scenarios are devised, to see what effects differing amounts of overlapping interference radii has on the nodes ability to utilise the network.

This experiment uses the new TRGR model but with the interference range set such that it in steps includes one node less in the horizontal and vertical directions in the grid topology until no nodes are within range, starting with a radius of 5 nodes within range.

Furthermore, this simulation is done for four different scenarios:

A 100 nodes 80000$bits/s$ mean utilisation

B 100 nodes 8000$bits/s$ mean utilisation no RTS/CTS

C 25 nodes 80000$bits/s$ mean utilisation

D 25 nodes 8000$bits/s$ mean utilisation no RTS/CTS

During this setup, the total amount of bits sent on the radio are logged to measure the utilisation of the radio since this is expected to also be closely tied to the interference, namely how much the radio medium is utilised.

For this simulation setup, there is also experimentation with disabling Request-To-Send / Clear-To-Send (RTS/CTS) for networks with lower loads, however, this is quickly abandoned to keep things more simple and only stick to one scheme, where it is enabled for all packets.

These are the interference range steps used: $75m$, $151m$, $301m$, $451m$, $601m$, $751m$, $843.51m$.

The nodes are separated by 150 meters, and the 75 meters equate to zero nodes within range and $751m$ and $843.51m$ both equating to 5 nodes being within range, with the latter being the previously used default value.

The results of this simulation are shown in figure 5.7 and figure 5.8 showing respectively the mean radio utilisation and the PLR.

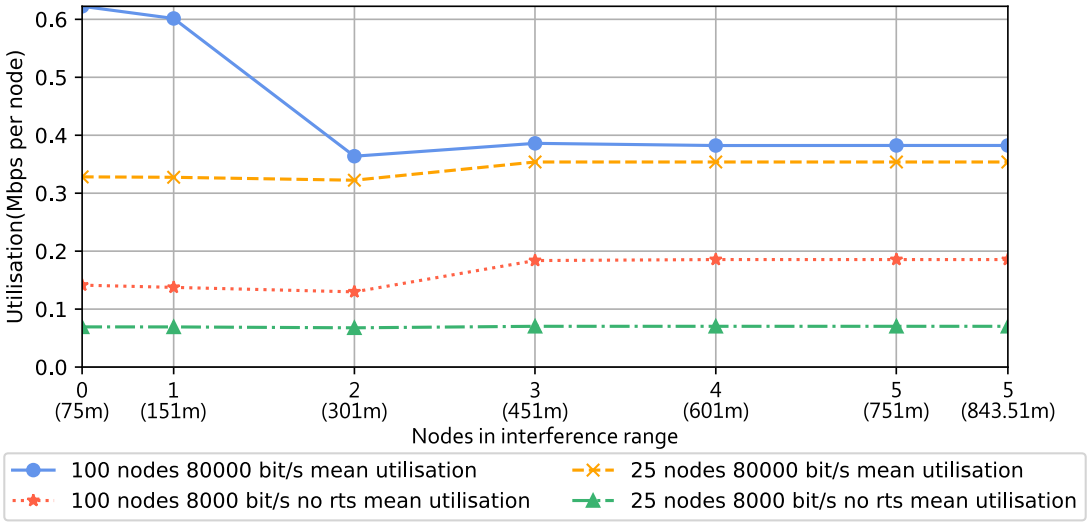


Figure 5.7: Network utilisation for varying ranges of interference and for four different network configurations. The used ranges radii equates to different number of nodes in any cardinal direction, and are shown right above the distances on the X-axis.

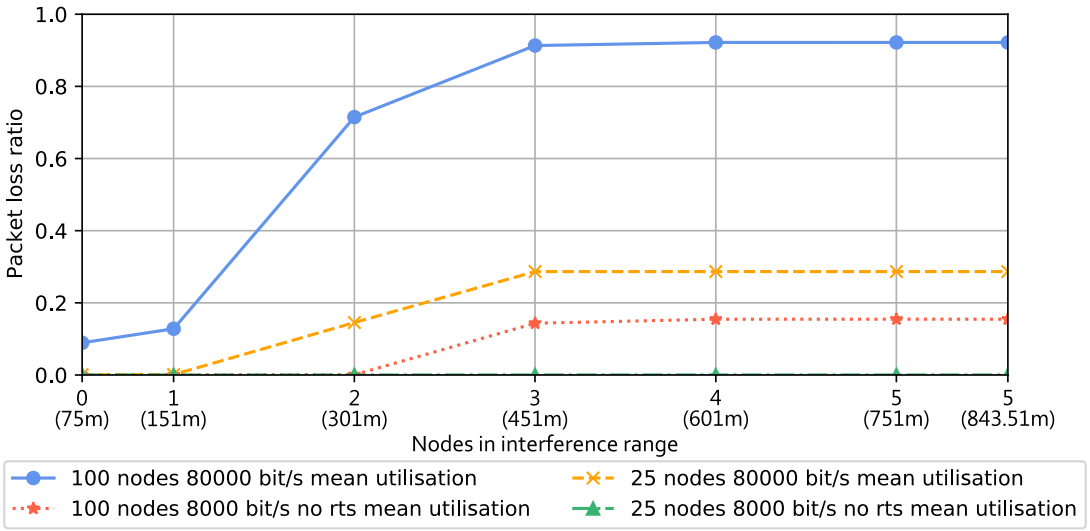


Figure 5.8: Mean PLR for varying interference ranges and for four different network configurations. The used ranges radii equates to different number of nodes in any cardinal direction, and are shown right above the distances on the X-axis.

These two graphs show that the packet loss already starts increasing with an interference radius of two or three nodes except for scenario **D**. However, it is only scenario **A** that increases above 30% going all the way up to the previously observed 90%. This shows that both the network size and the amount of data generated and sent by each node heavily affect the packet loss, along with the interference itself. Interestingly, the utilisation graph shows different effects for the different simulation setups. Scenario **A**'s utilisation

decreases with a large amount, the utilisation of scenario **B** and scenario **C** increasing slightly and scenario **D** staying constant.

Scenario **A** and scenario **C** for the default interference range also seem to utilise the radio almost the same amount but with around 90% and 30% PLR respectively. With the traffic being the same, the network size determines how much data is being sent on the network, since larger networks mean longer routes and more forwarding for the used traffic with random pairs, and more generated traffic in general. Being aware that the network has an upper capacity limit is important and that this depends on the size and routes packets are sent. Furthermore, this could highlight that overloading the network can have negative effects and diminishing returns for everyone. This could also indicate that the previously used traffic of $80000 bits/s$ is too much for the large networks that are relevant for this use case. Therefore, fully understanding what traffic the use case lends itself to, is crucial for the scalability and performance of the solution.

The changes also seem to happen as the packet loss increases. One argument for this making sense would be that packet loss causes retransmissions and therefore higher utilisation. However, that clearly is not the case for the largest network.

An explanation for utilisation decreasing could be related to the packet loss being measured as unsuccessful end-to-end deliveries and that if the packet loss becomes large the packets will with high likelihood be lost early on. This then causes a whole chain of packets not to be transmitted or retransmitted resulting in less utilisation. To investigate this a new graph is made with additionally logged data for each hop.

The graph in figure 5.9 shows the mean hop count for different network sizes. The theoretical hop count is based on the mean hop count equation $E[HopCount] = \frac{2}{3}n$ from [10] where n is the width or height of the network in nodes. The actual hop count is measured in the scenario **A** with a default interference range of $843.1m$.

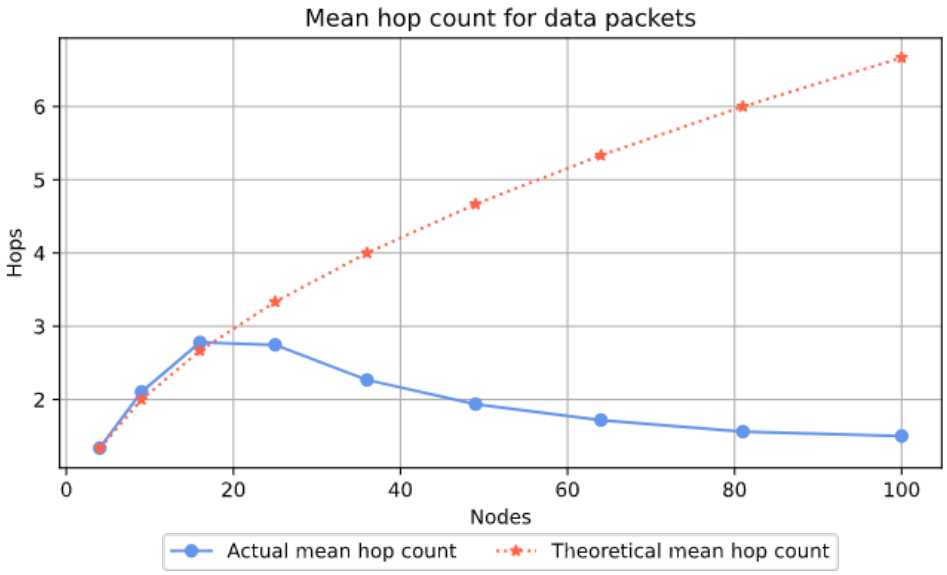


Figure 5.9: Comparison of the mean theoretical and mean observed hop count for different network sizes.

Figure 5.9 does seem to support the idea showing that the mean hop count for the 100 node network becomes low. This means that the packets will be lost early on and not be sent the 6 hop average which is so low that even with retransmissions on the first hop this still has a lower utilisation than the whole end-to-end multi-hop route.

5.2.6 Summarised results

Many factors on the physical radio level have an impact on the layers above themselves in the protocol stack, this makes sense as the physical radio is what provides the basis for the other protocols to exists on top of. This section is about experimenting with the physical layer in the pursuit of better performance with respect to packet loss. This lead to finding a better and more realistic model, while also realising the parameters used have a large impact on the achievable performance and how scalable the network is. The physical radio is, however, not the main focus of this project and therefore, will as such not be investigated further.

What is taken away from this investigation is a new and better model of how the radio signals propagate and a realisation that depending on the setup and configuration, the results can show widely differing performance and scalability measures.

From the network utilisation simulations, it is also apparent that the effects of loss or congestion can be counterintuitive. As packet loss can both increase or decrease pressure on other data streams by lowering or increasing general network utilisation. Similarly, high network utilisation increases the risk of packet loss. Additionally, the interference, loss and load are also observed to vary locally at each node depending on their location within the network.

Going forward the TRGR model will be used on the basis that it is a more realistic model and that the work is done to understand and use it. Along with a better understanding of how important topology, traffic and radio parameters are, showing the importance of fully understanding and specifying the use case, to figure out what is possible in terms of scalability and performance.

While also realising that the packet loss is much more present in the middle of the grid scenario. These are local differences and should be considered in the end-to-end routing or the routing metric, so that traffic loads get routed around problematic areas, unlike what is currently done in the B.A.T.M.A.N. implementation used in the simulation setup. These thoughts also support Systematic's own ideas and thoughts about avoiding links with a high PLR.

6 Part 2 - Protocol independent link state estimation

This part focuses on the link metric used to estimate the amount of traffic the links between the nodes can carry, and how this metric can be estimated from the B.A.T.M.A.N. protocol layer. First this is done by estimating the bitrate of a link, and later it is extended with estimations of usage and Packet success ratio (PSR).

6.1 Bitrate metric estimation

From the Systematic use case, there is a need for being able to deploy the network solution on top of almost any radio hardware. However, the solution that B.A.T.M.A.N. utilises is based on getting a throughput or bitrate estimation from the Wi-Fi driver or equivalent hardware. Therefore, if this could be replaced by a solution that can exist within the same protocol layer as B.A.T.M.A.N., it would fulfil the desired flexibility of the network solution making it deployable on most hardware.

The implementation of B.A.T.M.A.N. used in the simulation environment has disabled getting information from the Wi-Fi interface, therefore it defaults to a value of $10Mbit/s$. This is not representative of B.A.T.M.A.N.'s functionality in real life, as most Wi-Fi interfaces probably can provide this information. However, this should work on other hardware that is unlikely to be Wi-Fi. So this information being unavailable is actually more representative of the intended usage by Systematic and shows how B.A.T.M.A.N. would behave.

B.A.T.M.A.N. generally refer to the information it uses for routing and which it obtains from the Wi-Fi interface, as throughput. The exact information the Wi-Fi interface provides seems to be dependent on the specific Wi-Fi interface, but most often this information is the rate at which the radio transmits data. However, it is impossible to say if this is always the case, but for simplicity, it is assumed that when this information is available, it is most likely to be the transmission rate, which for this project will be referenced to as the bitrate. This is important to note as this project will later define throughput as a different metric that uses the bitrate along with other metrics.

Going forward all simulations will assume no information is available from the lower layers, as both the new solution can not rely on this and the simulated implementation of B.A.T.M.A.N. is incapable of using this. Any performance evaluation will therefore only be representative for the deployment conditions but not of B.A.T.M.A.N.'s performance when used with Wi-Fi for its original intended purpose.

6.1.1 Asymmetric links

Before addressing how the bitrate will be estimated as the quality of a link, some assumptions of the link in question will be made in order to have better grounds for a representative estimation.

Figure 6.1 shows a simple example with two nodes and one link.

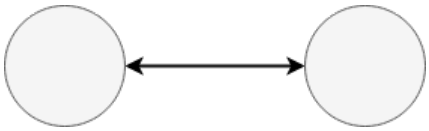


Figure 6.1: Simple setup of two nodes, with a bidirectional symmetric link.

For the general use case, the communication link between the two nodes will be done over a wireless medium, so in reality, there can only occur communication in one direction at a time and the conditions for reception might not be identical in both direction. Therefore, the quality of links is calculated as two unidirectional links. This means that the link between two nodes are two asymmetric unidirectional links that each needs to be estimated individually.

6.1.2 Bitrate definition

The bitrate is important as it tells how much data can be sent per time unit. Knowing the bitrate is needed for cases where the bitrate differs from link to link and for being able to prioritise routes with higher capacity, exactly as B.A.T.M.A.N. would do under normal conditions with Wi-Fi.

Since the assumption is that no information is available from the lower layers, the bitrate must be estimated within the B.A.T.M.A.N. layer itself. From the B.A.T.M.A.N. layer the only things that can be controlled and fully trusted or relied on are which packets are sent, and measuring the reception events when they reach the B.A.T.M.A.N. layer.

It is important to consider what is being estimated as bitrate can refer to multiple things. This project considers the physical bitrate and the effective bitrate, and it is important to make a distinction between the two.

- **The physical bitrate** is the rate at which the radio can send individual bits.
- **The effective bitrate** is for this project defined as how many B.A.T.M.A.N. payload bits can be transferred on a single successful transmission.

The difference is that the effective bitrate accounts for the protocol overhead generated from the lower layers, which differ depending on the lower layers and are therefore unknown within the B.A.T.M.A.N. layer. Generally, both are usable, it is just a matter of how the channel usage is accounted for, as the effective bitrate accounts for the overhead and the physical bitrate does not.

It is also assumed that there is no time synchronisation between nodes. This is important as time differences calculated using timestamps between nodes cannot reliably be used, only timestamps on individual nodes can be used.

6.1.3 Simple bitrate estimation

The essence of how to estimate both types of bitrate, when assuming no knowledge about the lower layers, is to look at some time frame and some data amount in order to figure out at which rate($\frac{bits}{second}$), bits can be transferred. One simple way to do this is to look at the packets successfully received over some time frame. Figure 6.2 below shows a sketch of two different scenarios with the same time frame, but with different saturation of the communication channel.

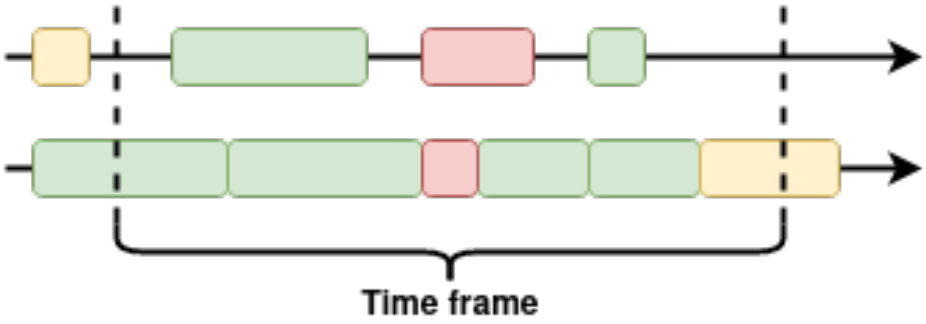


Figure 6.2: Sketch of a time window, showing the difference in link saturation, the successful packets transfers(green), the failed packets transfers(red) and packets transfers outside the time frame .

The figure shows the number of packets or data within a time frame can vary. Therefore, the estimated bitrate would depend on whether or not the link is utilised as much as possible, similar to what is sketched on the lower timeline. For a good estimation of a links full capacity, the transmitting nodes would artificially need to saturate the link as much as possible, and the receiving node then sees how much it receives over a time frame. This measurement also indirectly includes packet loss, as shown in figure 6.2, that while only successfully packets are received, failed packets also populate the timeline, and as such these take up time while providing 0 bits of data, affecting the overall time frame. Therefore, this estimation is more than just the effective bitrate when measured in the B.A.T.M.A.N. layer, as the packet loss will also indirectly be accounted for. Not having to find the packet loss separately does make things more simple. However, the artificial link saturation is not desired, because sending a lot of probes will generate a much larger overhead and will interrupt other transmissions, therefore other solutions are investigated.

6.1.4 Double Packet Train

One solution that is tested is the packet train concept, inspired by a paper by Lars Møller Mikkelsen et al. from 2015 [23]. This paper for example estimates the throughput from the TCP/IP layer. This concept measures time per data, rather than data per time. By sending packets with a certain known interval, or right back to back, and by knowing the size of the packet it is possible to measure how long it took to transfer the last packet's amount of data. Figure 6.3 shows a sketch of a timeline where two probe packets are

sent in a row, whereby looking at the reception time of both packets and then knowing the size of the packet or payload it is also possible to calculate the effective bitrate. The packet reception measurements are illustrated in figure 6.3, highlighting that the time difference between them measures the full duration of a packet transmission.

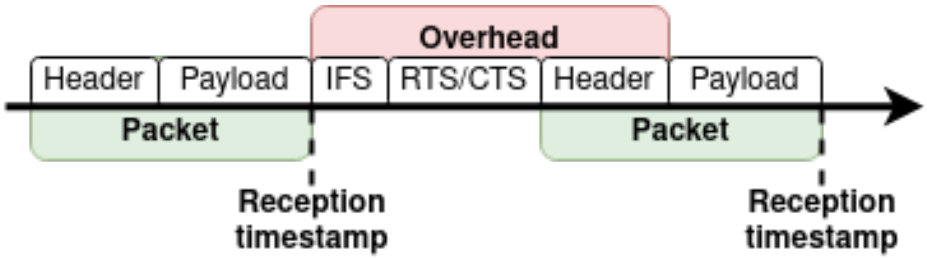


Figure 6.3: Sketch of a packet train of two packets in a row, with the overhead highlighted as both the B.A.T.M.A.N. packet header but also, for example, Inter frame spacing (IFS) or RTS/CTS.

This estimation estimates the previously defined effective bitrate, as figure 6.3 illustrates the overhead is included in the time measurement, while only the size of B.A.T.M.A.N. packets' headers and payloads are known as seen from the B.A.T.M.A.N. layer.

The effective bitrate is calculated by knowing the size of the second probe's payload and the duration it takes to transfer this packet.

The calculation is shown in equation 4:

$$EffectiveBitRate\left(\frac{bits}{second}\right) = \frac{PayloadSize(B)}{PacketDuration(s)} \tag{4}$$

Since the $PacketDuration(s)$ is both the overhead and payload duration, it becomes clear that, as the payload size can differ from $0B$ to $1464B$, so can the effective bitrate, as a larger payload lowers the ratio between payload and overhead and more efficiently uses the physical bitrate.

Additionally, the estimation so far has assumed two transmission successfully transmitted in a row as fast as possible, but in reality, the overhead will vary due to other traffic and other temporal randomness, for example affecting the contention period in Wi-Fi. However, all packets are equally likely to be affected and by relying on the law of large numbers, if enough packet trains are received the mean time difference between them should approach the true mean. This applies to all packet train methods, as there is no way to guarantee that a single packet train takes the true mean duration. Therefore, any implementation using the packet train methods will be using some sort of window of packets to compensate.

The double packet train in itself is similar to what B.A.T.M.A.N. already does, namely by sending two probe packets, but in functionality, it is very different. Original B.A.T.M.A.N. send probes for a different reason, as they force the Wi-Fi protocol to negotiate a transmission rate with the recipient, that B.A.T.M.A.N. in turn then rely on. Therefore, this

is done for Wi-Fi by just sending padded probe packets, and leaving the network interface to do the rest. This is also why these probes are only sent when no other traffic is present.

However, B.A.T.M.A.N.'s implementation of these padded probes can be reused for the double packet train estimation. Furthermore, as the same packets are reused, the double packet train would not generate much further overhead. The only increase in overhead is caused because these probes previously are only sent if no other communication occurs. But since the bitrate is needed for routing functionality, these probes are changed to also be sent periodically, and will therefore create more overhead as compared to the conditional probes. To save some overhead, the payload of the first probe is set to be empty.

This concept is tested in simulation with a simple setup of two nodes, where the underlying physical bitrate is varied. The goal is to estimate the effective bitrate. These simulations also vary the second probe's size, as these affect the effective bitrate.

The estimated effective bitrate is calculated as the mean of all successful probe pairs received and as a mean across the two nodes since they have identical communication conditions in both directions.

The results from this are shown in figure 6.4.

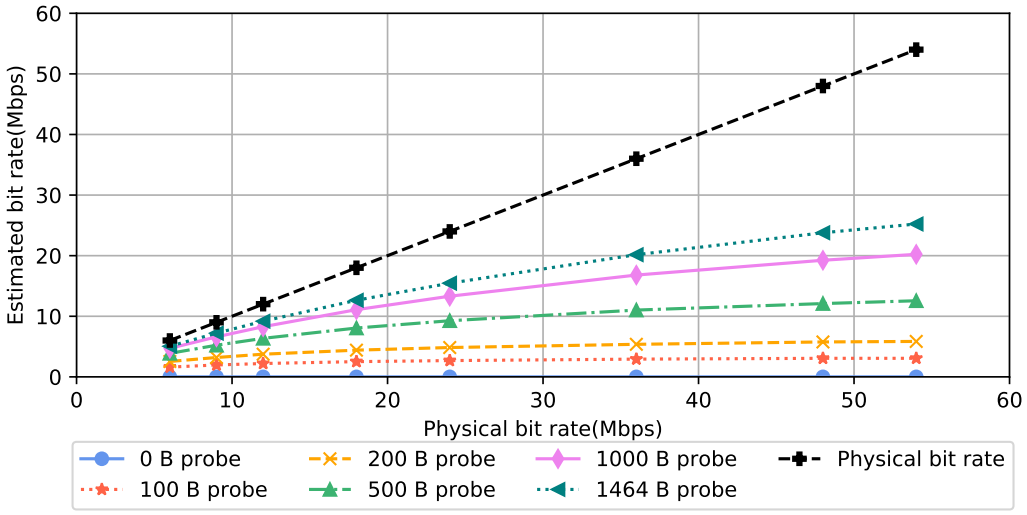


Figure 6.4: Packet train estimation in a 2 node simulation setup with varying physical bitrate, and different probe sizes.

These results show that the effective bitrate is lower than the physical bitrate as expected, the bitrate also does not increase linearly with the physical bitrate which is initially expected. However, further investigation revealed that for the simulation run, not every part of the overhead is scaled equal, explaining why a less than linear effective bitrate is achieved.

The main takeaway from these simulations is that the probe size matters a lot for the

estimated effective bitrate. Furthermore, the goal is to estimate the effective bitrate, however, the ground truth is never defined so the estimations do not have anything to compare them against. Therefore, theoretical values are calculated with varying physical bitrates and varying probe sizes for comparison. These values matched the results from the simulation in size and tendencies and can be found in appendix C.1 figure C.1. But these calculations rely on knowing the protocol stack. From within the B.A.T.M.A.N. layer, a different approach could be to have a specific probe size defined as the reference one, in order to enable comparing bitrates, as all nodes bitrates would be estimated using the same probe size.

However, thinking forward here the later parts of this project considers usage as an important metric, and therefore the double packet train might not be ideal, as determining the bitrate usage now becomes more difficult. Because packets will be sent with differing payloads sizes. For example, how much of the effective bitrate does a packet with size $1400B$ use when compared to the reference bitrate estimated using $200B$ probes. This is difficult to do without knowing the overhead or ratio between payload and overhead.

This problem is not fully investigated, but a preliminary idea for doing this is by sending different sized double packet trains to determine the relationship between effective bitrate and probe sizes. This could then be used to estimate the overhead or ratio between overhead and payload. Which then could enable the conversion between packets at different sizes and their equivalent usage at a specific probe size's effective bitrate, or alternatively a conversion from effective bitrate to physical bitrate.

6.1.5 Triple Asymmetric Packet Train

Another concept that is tested is an expanded version of sending the double probe packets, and that is by sending a total of three packets in a row. As with the double packet train, the first probe can be empty to save on the overhead, then the following two need to have different B.A.T.M.A.N. payload sizes. For simplicity a payload of $0B$ and $184B$ is chosen, as $0B$ again saves on the overhead and the $184B$ matches the default probe size. This concept going forward is referenced to as Triple Asymmetric Packet Train (TAPT). The goal of TAPT is to estimate the physical bitrate by isolating the duration of the overhead and the duration of the payload which the double packet train did not do.

For the specific Wi-Fi protocol stack used, the overhead created by the double packet train versus TAPT is investigated at $54Mbit/s$ and shows that the TAPT is $794.5\mu s$ and the double packet trains is $559\mu s$, a more detailed description of how this is reached can be found in appendix B.

By measuring the reception time of the packets and looking at the time difference between these packets it is possible to isolate the duration it takes to send a payload of $184B$, and thereby estimating the physical bitrate. To illustrate this figure 6.5 shows that the time difference between timestamps ΔT measures the duration of overhead and the duration of overhead plus payload between packet pair 1→2 and 2→3 respectively.

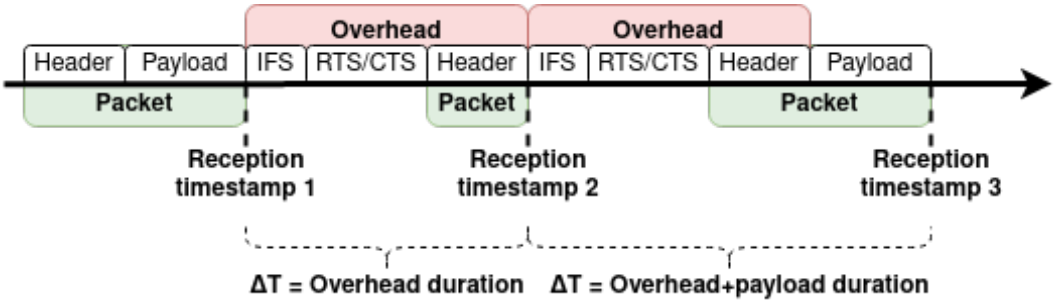


Figure 6.5: Sketch of a packet train of three packets in a row, showing two "empty" probes follows by a third. Highlighting that the ΔT time difference measures the duration of overhead, and overhead and payload respectively.

From this, it is possible to calculate the physical bitrate by having a known amount of data per time. This is shown in equation 5.

$$PhysicalBitRate\left(\frac{bits}{second}\right) = \frac{PayloadSize(B)}{\Delta TPayload(s)} \tag{5}$$

As with the double packet train, there is made no assumption that a single set of successful packet train is representative of the true mean transmission duration. Especially with TAPT where a single packet train might be unevenly affected over the three packets, therefore it is important to use an average duration between probe 1→2, and 2→3.

A simple simulation is set up between two nodes, where TAPT is implemented and the physical bitrate is varied, along with different sizes of the third probe packet. The time measurements from these tests are then averaged for the whole simulation and across two nodes since they have identical communication conditions in both directions. The results from this are shown in figure 6.6.

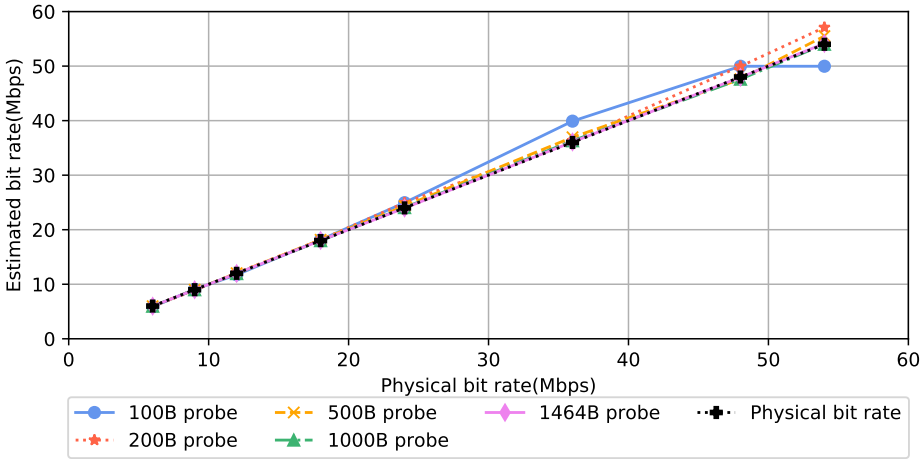


Figure 6.6: TAPT estimation in a 2 node simulation setup with varying physical bitrate, and different probe sizes.

The results from this quick test simulation show promising results, here unlike the double packet train, it is much easier to determine if the estimation is close to the ground truth, as this is the known physical bitrate. For all probe sizes the estimate is very close to the actual physical bitrate, with perhaps a slight tendency that the larger probe sizes are slightly closer to the physical bitrate. This does make sense since the difference in time measurements become larger, and therefore could be less influenced by small random variations.

Furthermore, now that the physical bitrate and the overhead duration is estimated, it makes it much easier to estimate any given B.A.T.M.A.N. packet's total use of the capacity determined by the physical bitrate. Therefore, the TAPT method is then one used going forward as it provides a good estimate and makes it straightforward to calculate the used capacity.

Through experimentation, it is learned that the using TAPT it is important to consider the size of the ELP-probes, as this has a small effect on the accuracy of the TAPT methods estimation. It is found that to get the best estimation the two first ELP-probes can be kept at $0B$ while the third needs to be $205B$ (equivalent to $189B$ payload). For more detail why that is see appendix C.2. Therefore, going forward these values for the ELP-probes using TAPT are used. This modification could be considered protocol specific, as the 205 is chosen by knowing the protocol stack and modulation. But this could also be achieved within the B.A.T.M.A.N. layer by varying the probe size slightly until minimum, maximum or similar is found and thereby allowing estimation of an accurate bitrate without knowing what the modulation is and therefore not considered specific to the protocol stack.

6.1.6 Implementation and adjustments

The results from the TAPT estimation showed a good estimation in a simple setup, therefore a more complete implementation is simulated in two larger networks. Both scenarios have 100 nodes placed in a grid, the first one have a random traffic load (scenario 1), and the other has the same random load with an added edge-to-edge traffic on top (scenario 3). These scenarios are meant to push the network to the limit, with 100 nodes and more traffic than the network can carry, see appendix C.5 for a detailed description of these two scenarios. The system-wide PLR for scenario 1 and scenario 3 with the original B.A.T.M.A.N. implementation is:

Scenario 1: 31.39% system-wide loss with 17.27% lost due to no routes available

Scenario 3: 39.01% system-wide loss with 24.29% lost due to no routes available

6.1.6.1 OGM self interference

From running the initial simulation of the TAPT implementation it is noticed that a large amount of OGMv2 packets are lost causing problems with routing not acting as

expected. This turned out to be because of the way OGMv2 packets flood the network and interfere with themselves. For a more detailed description of this see appendix C.4. In this appendix the solution for this is also presented, which is to add a random jitter right before an OGMv2 packet is forwarded. Furthermore, as this is a general problem with flooding a network with broadcast packets, this is not necessarily B.A.T.M.A.N.'s fault, and therefore this is also added to the original B.A.T.M.A.N. implementation, as well as all the new version going forward in this project.

Rerunning the simulation of scenario 1 and 3 with the original B.A.T.M.A.N. implementation gives:

Scenario 1: 25.12% system-wide loss with 0% lost due to no routes available

Scenario 3: 42.11% system-wide loss with 0% lost due to no routes available

6.1.6.2 Expanded Link State View

Packet trains rely on measuring the reception time on the receiving node, but it is the transmitting node that needs to know the bitrate metric for when it is routing. For sharing this information back to the transmitting node the concept of Expanded Link State View (ELSV) is introduced to enable nodes to know more information from their neighbours and surroundings than just their own local information. See appendix C.3 for a detailed description of how this information is shared back. The basic concept is to use the padding in the periodic ELP-probes to share this information without generating further overhead. Beware, that this appendix also addresses information sharing beyond what is needed for the bitrate estimation, but also addresses a new throughput metric that is defined later in this report.

6.1.6.3 Window estimation

Going back to the simulations, the goal of them is to show that the estimations across nodes and time are consistent and still gives good estimations even in a chaotic environment. From running the TAPT implementation with windows on scenarios 1 and 3 a few times, a couple of things are learned regarding the window size and estimating within a window.

Using the mean time difference initially gave estimations that are way off. The suspected reason is that these mean values gave poor results due to either unfortunate queuing or the exponential back-off mechanisms, as one bad contention period could skew the mean by quite a large amount, and therefore in turn requires a large window size. Therefore, another method is tried, namely using the smallest observed time difference between probe one and two, and probe two and three. The initial logic behind doing this is that instead of being affected by randomness, find two good measurements that are sent as fast as they can and use only these two instead. This seemed to work really well, therefore going forward the estimation over the window of measurements, is done by finding the minimum time differences stored at each node.

Then these scenarios are run again with TAPT, where the window size is varied to see what size is suitable in order to ensure a good estimate. The window sizes $100s$, $125s$, $150s$, $175s$ and $200s$ are all tested, with $200s$ being the only one that gives a perfect estimation over the entire simulation for all nodes.

Using a window size of $200s$ guaranteed at least two good measurements, combined with using the minimum measurements, this gives good estimations across all 100 nodes during the entirety of the simulation. Actually, once every node had two good measurements they would get a physical bitrate estimation of either $53999kbits/s$ or $54000kbits/s$ compared to the underlying $54Mbits/s$ ground truth bitrate. Furthermore, these estimations are always reached within the time it took to fill the window of $200s$, and once the correct estimation is reached they never changed it to a different one.

6.1.6.4 Addressing overhead

Then looking at the performance, initially, the TAPT solution gives a higher system-wide PLR of 52.93% in scenario 1, this is not really ideal as the solution should be a direct replacement. But the reason for this is speculated to be most likely caused by the additional overhead from guaranteed periodic ELP-probes, and three probes instead of two whenever they are sent.

Because even as the three probes combined are smaller in padding than the two large probes previously sent, this does still equate to a larger overhead similar to what is calculated in section 6.1.5.

Therefore, to get the same performance the interval between probing is increased to get the same overhead. TAPT is tested with the following intervals $0.8s$, $0.75s$, $0.725s$ and $0.7s$, with the last one approximately matching the system-wide PLR of the original B.A.T.M.A.N. implementation with a PLR of 24.22%.

Following this, the amount of ELP-probes per second is decreased, and the window size is doubled to $400s$ to ensure that it still works, less could probably work, but the window size is not optimised, as reactivity is not evaluated here. But the window size should probably not be larger than 400 times the ELP interval or even better be dependent on the variance or packet loss of the link for optimisation.

6.1.7 Comparison

So far this section has only been about how to estimate the bitrate, which is also what B.A.T.M.A.N. does. However, the presented concept and implementation of TAPT differs in that the TAPT estimation is designed to work with no assumptions on lower protocol layers and works without relying on the Wi-Fi interface. Therefore, to show the advantage of TAPT, a scenario is constructed, where B.A.T.M.A.N. is forced to use the *throughput override* metric value of $10Mbit/s$, instead of the value provided by the Wi-Fi interface.

Both the TAPT implementation and original B.A.T.M.A.N. is tested in the same scenario. The scenario consists of 7 nodes evenly spaced along a circle, with a data stream going

from one side to the opposite side. The data stream is 280 packets per second each with a payload size of $1408B$. The data stream then has two possible routes to the destination, one with two hops and one with three hops, going different ways around the circle each. Then along the short route, the two intermediary nodes are limited to only transmit with $6Mbit/s$ and all other nodes are allowed to transmit with $54Mbits/s$. This setup is illustrated in figure 6.7.

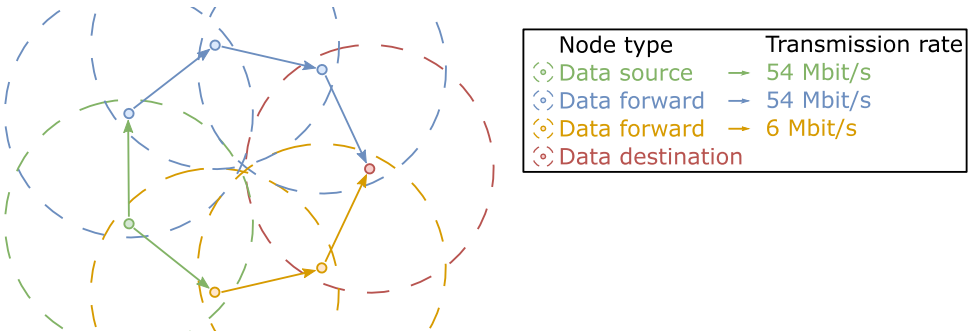


Figure 6.7: An illustration of the placement of nodes, along with their transmission speeds for the two routes towards the destination node.

The simulation time used for this scenario is one hour, with the last 50 minutes being logged and the first 10 minutes is used for letting the network settle.

Given that B.A.T.M.A.N. is used in a case where the Wi-Fi interface cannot provide a bitrate estimation, B.A.T.M.A.N. defaults to one specific value for all links, in this case, the default bitrate is $10Mbit/s$. The expectation is that this should then cause the original B.A.T.M.A.N. implementation to choose the shortest path not realising the bottleneck, compared to the TAPT implementation that should choose the slightly longer route but with no bottleneck.

The results show that the original B.A.T.M.A.N. implementation sends all 840000 logged packets towards the short route, with 414534 reaching the destination node for a packet loss of 50.7%. While the TAPT implementation sends all 840000 logged packets along the long route with 804683 reaching the destination for a packet loss of 4.2%. This confirms that the TAPT method works to better identify bitrate bottlenecks, resulting in correctly identifying the longer route as the one with higher capacity, when no information is available from the lower protocol layers.

6.1.8 Summary

For replacing the throughput metric B.A.T.M.A.N. relies on getting from the Wi-Fi interface, a bitrate metric is estimated from the B.A.T.M.A.N. layer. This bitrate metric is estimated by using packet trains, and both the double packet train and TAPT is usable for estimating the effective and physical bitrate respectively.

The TAPT is determined to be the best of the two options as it for the later work in this project seemed to make the most sense, specifically enabling easier usage estimations.

The implementation of the TAPT shows it is possible to get good estimations, for example estimating either of $53999kbits/s$ or $54000kbits/s$ when compared to an underlying $54Mbits/s$ bitrate. However, the TAPT does generate a larger overhead compared to the original B.A.T.M.A.N. implementation and therefore the interval between ELP-probes is adjusted to $0.7s$ compared to $0.5s$ to get equal performance of the system-wide PLR.

While these simulations show that the implemented version of TAPT gives good estimations, these simulated scenarios do not consider scheduling or processing time and what effect this could have on the estimations. This would need to be considered or tested on a real testbed to fully verify the method. But if these do not impact the estimation, or only affect the estimation slightly the method seems to work well.

Furthermore, a scenario is also constructed to show the advantage of being able to estimate the bitrate when no knowledge of the lower layers is assumed. This scenario shows that a longer route can be identified as the better option and lower the PLR from 50.7% to 4.2%.

6.2 Throughput metric

The TAPT concept enables B.A.T.M.A.N. to get a bitrate independently of the lower protocol layers. But this metric is not representative of how much data can be transferred between two nodes on a single link.

Ideally, the link metric is representative of the actual throughput on a link such that the routing can be done based on this, as this is also what Systematic wants. To reiterate, Systematic wants to avoid links with a high PLR and be aware of throughput variation of links. So that the network prioritises transferring as much data as possible.

The goal would then be to improve the routing metric so that this is considered. The initial idea is to consider what can affect a link's ability to transmit and receive and combine this with the bitrate estimation using the TAPT concept. This new metric should also work independently of the lower protocol layers as well.

This section investigates how to estimate the throughput of a communication link within the B.A.T.M.A.N. protocol itself, for use as the main routing metric in B.A.T.M.A.N., therefore, this section also starts out with a description of what determines a link's quality in order to specify what needs to be estimated within the B.A.T.M.A.N. layer.

6.2.1 Defining throughput

Similar to goodput, this project is interested in what rate it is possible to transfer data from B.A.T.M.A.N. to the layers above. As this is what they will be able to utilise and the goal is then to make the link metrics representative of this.

The idea is to define the throughput of B.A.T.M.A.N. to be based on the physical bitrate, but consider overhead generated from B.A.T.M.A.N. and the layers below, and the loss experienced for packets passing the B.A.T.M.A.N. layer. So that it is possible to

figure out the possible rate at which data can effectively be transferred as seen from the B.A.T.M.A.N. layer.

While they might be similar they are not identical, as calculating the goodput is dependent on the packet size, the throughput is not, because it is calculated as the capacity minus the combined consumption of overhead and payload. However, this also means that one bit of payload in the B.A.T.M.A.N. layer does not equate to one bit consumed of the throughput's capacity. Therefore, the throughput can be considered as the remaining unused capacity of the physical layers bitrate.

6.2.1.1 Types of overhead

The basis for the possible throughput is the bitrate that the physical layer allows for. However, several factors make the actually achievable throughput of data lower, there is the overhead of protocol headers and spacing between packets. The headers are usually a fixed size of bits wrapped around every payload according to the protocols used, the spacing between packets is determined by how the channel is accessed, in this case, Wi-Fi is used, which uses both Carrier-sense multiple access with collision avoidance (CSMA/CA) and RTS/CTS control packets.

The CSMA/CA used in this project uses IFS and an exponential back-off mechanism, this creates moments where the medium is not used and results in overhead. Additionally, RTS/CTS has in this project been enabled for all packets, in a non-ad hoc Wi-Fi network it is normally either not used or only used for larger packets, however, for simplicity it is enabled for all. RTS/CTS is used to avoid collisions caused by the hidden node problem. The overhead caused by time spent waiting and negotiating a transmission and the overhead from packet headers are shown in figure 6.8.

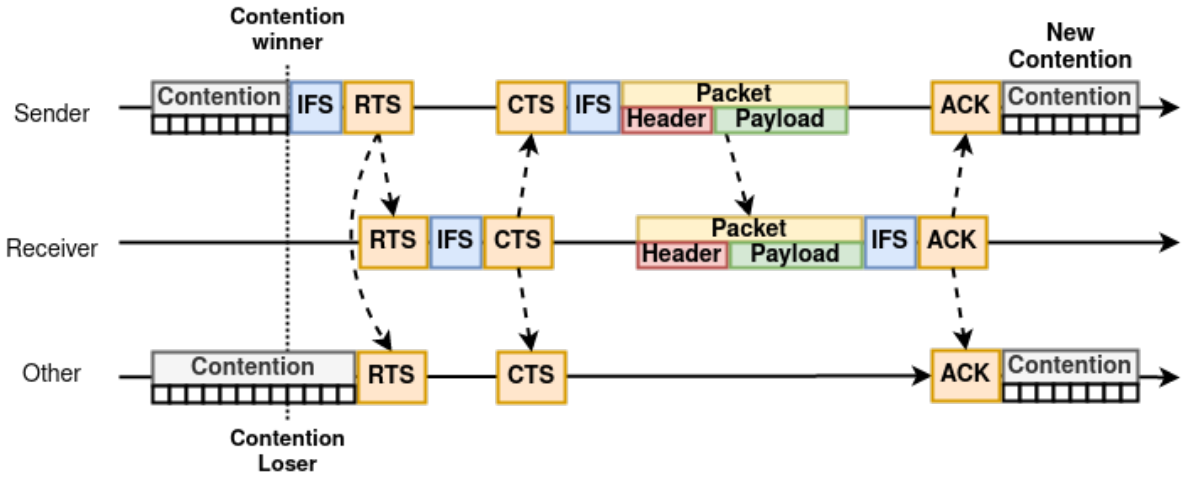


Figure 6.8: Sketch of a simplified timeline of a packet being sent, showing the contention period and IFS from CSMA/CA and the negotiation process from RTS/CTS creating overhead, besides that of the packet header.

Figure 6.8 is meant to show that besides the packet and payload itself there are other

mechanisms that affect the usage of a local channel. Worth noting here is that this sketch of the overhead is specific to the used underlying Wi-Fi radio and MAC layers. However, other radio setups could apply something functionally equivalent to RTS/CTS in order to address the hidden node problem, which is especially prevalent in an ad hoc network. This is important to note since the solutions need to work with as many radio configurations as possible, without relying on that specific configuration. Therefore, the specific overhead could be different but the cause and type of the overhead would remain similar.

6.2.1.2 Shared channel

Figure 6.8 also shows that there are other neighbours than just the ones participating in the exchange itself. This is meant to show that the channel used between nodes is most likely going to be shared with other neighbouring nodes that are also communicating with their surroundings. These are also important to consider since they also use the same underlying channel and therefore also uses some of the capacity of the local channel for their communication.

This is illustrated in figure 6.9, here, the link that is being estimated is shown with a solid line, however, the transmitting node's channel is also used by the receiving node and other neighbours within the communication range. These neighbouring nodes are on the figure shown in grey and the green background illustrates that the neighbouring nodes are within communication range and the dotted lines show that when these neighbouring nodes transmit, they use the transmitting nodes local channel.

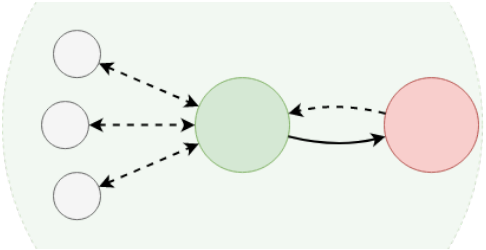


Figure 6.9: The green node transmitting to the receiving red node, with the green node's communication range shown as a faded background, showing other nodes within range as well sharing the local wireless channel of the green node.

The receiving node's channel is likewise affected by the transmitting node and its other neighbours. The nodes affecting the receiving node are in a similar manner as figure 6.9 illustrated in figure 6.10.

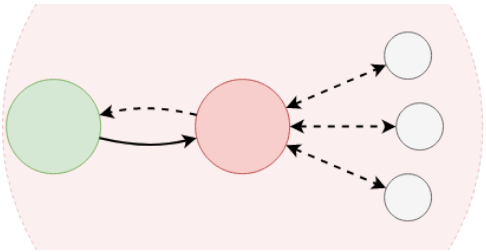


Figure 6.10: The green node transmitting to the receiving red node, with the red node's communication range shown as a faded background, showing other nodes within range as well sharing the local wireless channel of the red node.

Therefore, the available bitrate depends on the existing utilisation of both of these local channels and how the usage is contended between nodes. The immediate neighbours will due to CSMA/CA use up the capacity of a node's local channel. Furthermore, the use of RTS/CTS can make a node two hops away prompt a node one hop away to send a CTS control packet that will block a node's local channel also using up the capacity.

However, this is just how much it is possible to send from one specific node, but it should also be considered that the receiving node might experience radio interference that leads to a lost packet. Therefore, the number of packets lost will also affect the throughput. This interference can stem from general noise or nodes outside communication range transmitting and is generally hard to predict and it therefore also appear random in nature.

6.2.1.3 Usage

The above explanation of the wireless unidirectional links leads to a starting point of a simple relationship between a specific sending and receiving node, where the throughput initially can be described by the physical bitrate minus the usage within a node's local channel.

The available bitrate is the physical bitrate of the channel minus how much of it is used by other neighbouring nodes, or alternatively how large a ratio of the bitrate is not utilised yet as shown in equation 6.

$$AvailableBitrate = PhysicalBitrate \cdot (1 - Usage) \tag{6}$$

Where:

$AvailableBitrate$ - is the unused capacity(in $\frac{bits}{second}$) of the underlying channels radio medium

$PhysicalBitrate$ - is the capacity(in $\frac{bits}{second}$) of the underlying channels radio medium

$Usage$ - is the ratio of used capacity versus the capacity determined by the physical bitrate.

6.2.1.4 Success rate

Besides usage affecting throughput, packet loss can randomly occur from radio interference and noise, also affecting throughput. This means that only a certain ratio of data is received compared to the sent data. This ratio is the success rate or probability of data being received. The following equation 7 shows one way of expressing this, as the ratio of packets received. However, packet size matters as smaller packets take less time to transmit and are therefore less likely to experience noise. Therefore, alternatively, it can also be expressed as the ratio of bits received. Both express the probability of successfully receiving a payload of data just weighted differently, but for this project, only the ratio of packets is considered.

$$PacketSuccessRate = \frac{NumberofReceivedPackets}{NumberofSentPackets} \tag{7}$$

Where:

$PacketSuccessrate$ - is the success rate or probability of a packet being received

$NumberofReceivedPackets$ - is the total number of packets received by the receiving node

$NumberofSentPackets$ - is the total number of packets sent by the transmitting node

6.2.1.5 Throughput metric

Having defined the available bitrate and PSR leads to the following equation 8 of throughput described from the product of available bitrate and the probability of packets not being lost. Where the available bitrate is written out into its subcomponents as these are all the unknown variables that are expected to be estimated individually.

$$Throughput = PhysicalBitrate \cdot (1 - Usage) \cdot PacketSuccessRate \tag{8}$$

Where:

$Throughput$ - is the rate(in $\frac{bits}{second}$) at which data can be successfully transferred between a specific pair of transmitter node and receiver node.

$PacketSuccessRate$ - is the success rate or probability of a packet being received

$PhysicalBitrate$ - is the capacity(in $\frac{bits}{second}$) of the physical channel

$Usage$ - is the ratio of used capacity versus the capacity determined by the physical bitrate.

This throughput metric will, going forward, be refereed to as the Bitrate, Usage and Success rate (BUS) metric.

6.2.2 Usage estimation

This section focuses on the estimation of the $Usage$ term that is part of the throughput calculation from equation 8 at the end of section 6.2.1.5.

As described in section 4.3 B.A.T.M.A.N. sets its throughput routing metric using *throughput override*, in the case where a node's physical layer cannot provide an estimate. This value can be replaced by the estimated bitrate from section 6.1. However, if all the nodes in an ad hoc network transmit using similar hardware, there should not be any difference in the routing decisions when using either B.A.T.M.A.N.'s *throughput override* metric value or the estimated value from section 6.1. This means that making routing decisions solely on either of the mentioned metrics would result in nodes all trying to route through the shortest paths if the network's nodes are reasonably homogeneous. Which in some cases would be problematic since multiple nodes would end up sending their traffic through the same route, making it more congested and potentially sending more data than the link can carry, resulting in packet loss. This is where the usage term from equation 8 comes into play, since each node tries to estimate how much of the communication channel it and its neighbours uses. And by multiplying the estimated bitrate with 1 minus the estimated usage ratio, nodes should be better equipped to route around congested areas when possible.

To estimate the usage, two approaches are proposed where the first approach serves as a base for the second.

Approach 1: In this approach, a node keeps track of how much of the communication channel itself uses, by accumulating the number of bits and packets it has transmitted during the last x amount of time. Furthermore, each node already has an estimate of their bitrate and an estimate for the average overhead time for an unicast packet. This allows each node to calculate its usage ratio during the last x seconds using equation 9:

$$OwnUsageRatio = \frac{Overhead(s) \cdot AmountOfPacketsSent + (\frac{TotalBits(b)}{EstimatedBitrate(bps)})}{x(s)} \quad (9)$$

Each node then shares its estimated usage ratio with its neighbours via the last of the three ELP-probes, transmitted during probing. This is achieved by setting a field in the last ELP-probe to the estimated usage ratio as described in appendix C.3.

Upon receiving the third ELP-probe from a neighbour, a node then records this received usage ratio in its neighbour table for the corresponding neighbour. When the throughput metric then is required, every node knows their own usage ratio and the usage ratio of each of their neighbours and can therefore calculate the total usage by accumulating them all, as expressed in equation 10.

$$TotalUsageRatio = OwnUsageRatio + NeighboursUsageRatio \quad (10)$$

Figure 6.11 illustrates this first approach, where the green node knows about the traffic itself has generated (green arrows), while its red neighbours have to share information

about their usage (red arrows) before the green node can make a decent estimate of how much of the local channel is used.

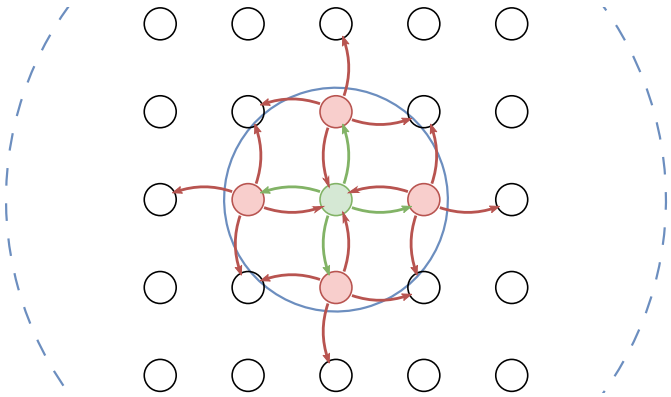


Figure 6.11: When using the first approach every node keep track of how much of their own local communication channel they use, and share this information with ELP-probes. From the perspective of the green node, its neighbours, the red nodes inside the blue communication range, shares their usage ratio, such that the green node can make a qualified estimate of the usage. The white nodes, which are in interference range (dashed blue circle), are not part of the usage estimate.

Approach 2: The problem with the first approach is that the estimated usage ratio does not consider nodes that are two hops away. This is problematic in the case where something like RTS/CTS is used since nodes that are two hops away can directly affect a node's local channel by communicating with their neighbours. This is illustrated in figure 6.12 where the yellow nodes, which are outside the green node's communication range, can directly affect the green node's local channel by communicating with any of the red nodes, or white nodes.

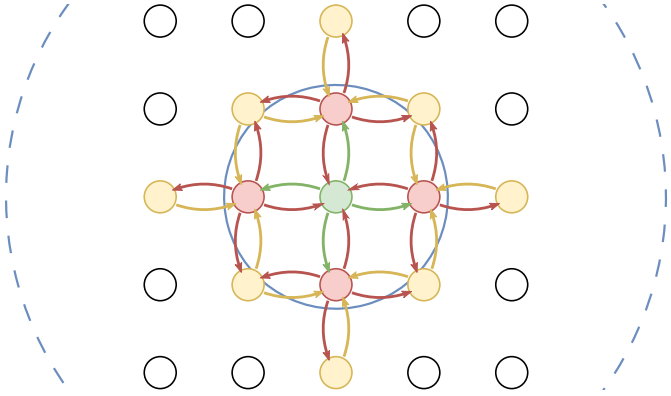


Figure 6.12: When using the second approach, the communication from the yellow nodes are also part of the green node's usage estimate, due to RTS/CTS, which inhibits the green node's ability to communicate with the red nodes.

To account for this, the nodes in the second approach additionally records the estimated usage ratio toward each of their neighbours. All individual neighbour usage ratios, together with the corresponding node id (e.g. MAC address), is then packaged in pairs in

the last ELP-probe and shared like the rest of the information. Upon reception of an ELP-probe, a node can now examine whether or not it knows about the nodes listed in the probe. If an id corresponds to a node that is already a neighbour, its usage ratio should be disregarded to ensure that it is not counted twice. While the usage ratio for unknown ids should be added to the total usage ratio for the local channel, which is the value used for calculating the usage metric.

$$TotalUsageRatio = OwnUsageRatio + NeighboursUsageRatio + NeighboursNeighboursUsageRatio \quad (11)$$

In equation 11 $NeighboursNeighboursUsageRatio$ does not include usage ratios of links already accounted for in $NeighboursUsageRatio$.

As an example of how the nodes individual estimation of the usage ratio is illustrated in appendix D in figure D.1, where each node is coloured according to its usage ratio estimation.

6.2.2.1 Showing the effect of approach 2

To show the effect of the added usage term, described in **approach 2**, a simple simulation scenario is constructed. Figure 6.13 shows a visual representation of the placement of the nodes.

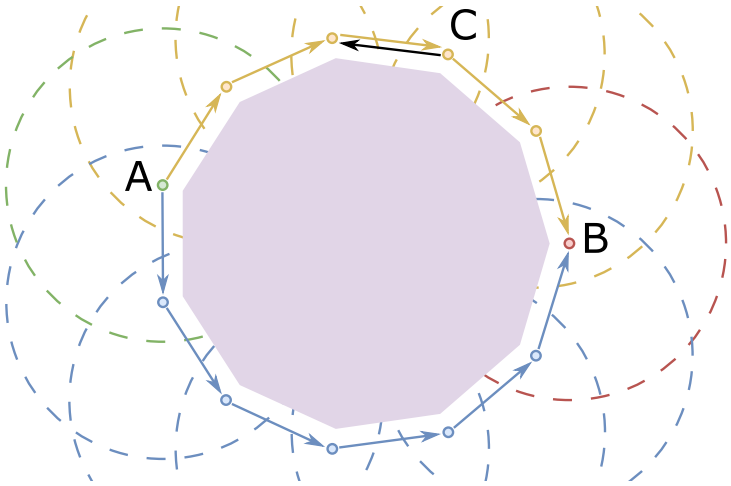


Figure 6.13: Simulation scenario for testing effect of usage ratio term. The green node (**A**) communicates with the red node (**B**), through either the blue or yellow route. The blue route is longer while the yellow route already has a high amount of traffic. The object in the middle absorbs any incoming radio signals to limit effect of interference.

In the scenario depicted in figure 6.13 **A** is configured to transmit 280 packets per second of $1408B$ each to **B**. It can either do this by transmitting it to the blue or the yellow neighbour who will then forward it towards its destination. Furthermore, the third yellow node (**C**) is configured to transmit packets every $0.9ms$ with a size of $1408B$ to its

left neighbour, presented by a black arrow. Since all links in this scenario are set to transmit at $54Mbit/s$, both B.A.T.M.A.N.'s *throughput override* throughput metric and the proposed replacement, which only uses the estimated bitrate, would choose the yellow route since the default calculation only considers the length of the route. However, choosing the yellow route in this scenario would result in a high packet loss since the existing traffic between **C** and its neighbour already takes up a significant portion of the available bandwidth. This means that in this scenario the optimal route should be the blue route despite it being one hop longer. To isolate the nodes from interference in this made-up scenario, a polygonal object that absorbs any incoming radio signals is placed in the middle between all the nodes.

Figure 6.14 shows how B.A.T.M.A.N. handles the simulation scenario using only the bitrate metric found using TAPT. The number written in each node corresponds to the number of transmitted data packets, while the number in the arrow shows the specific links PLR. The arrows are colour coded such that a higher PLR results in a darker blue colour.

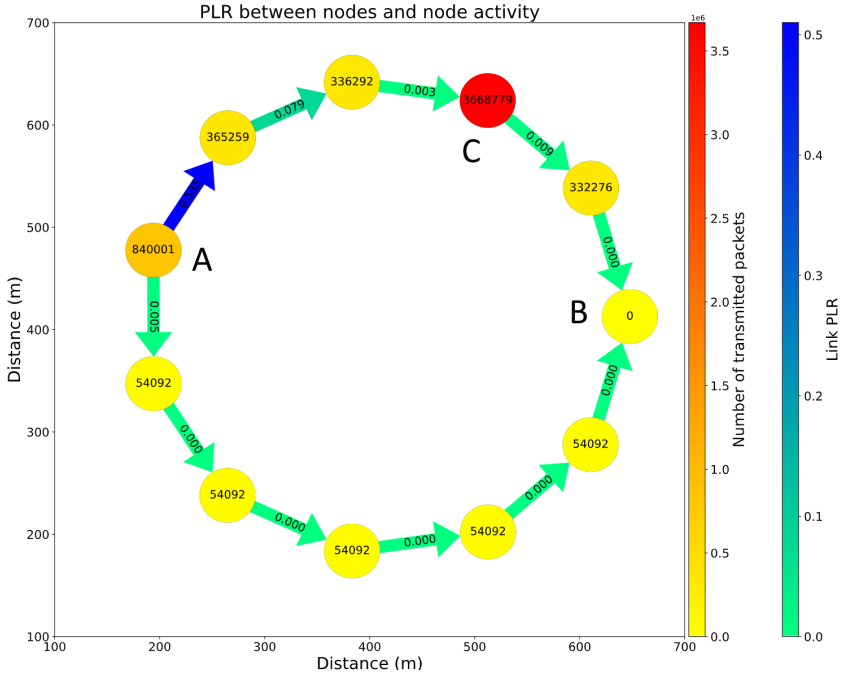


Figure 6.14: Simulation results using only the bitrate metric found using TAPT. This simulation is simulated for an hour, while the logging started after 600 seconds.

Here it is shown that during the simulation **A** sends 840001 data packets, of which the majority is routed through the short route, where there is a packet loss of 53.5% on the very first hop and a smaller packet loss on every subsequent hop. Worth noting is that a small amount of the packets are routed through the long route, where only 0.5% are lost. In total, only 46.1% of the data send from **A** managed to reach **B**, using the estimated bitrate for routing. Figure 6.15 shows which route each of the data packets used during the simulation.

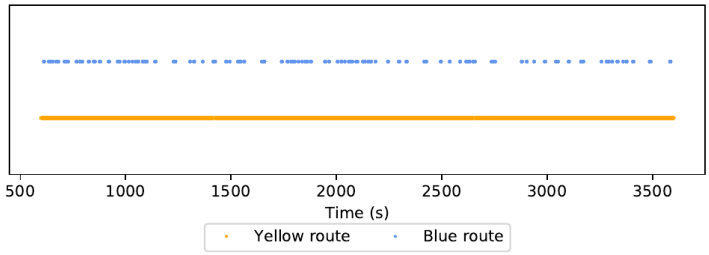


Figure 6.15: Shows data packets over time as either a blue or yellow dot to represent which route it travelled in reference to figure 6.13, using B.A.T.M.A.N.'s *throughput override* metric.

Similarly, figure 6.16 shows how the proposed usage and bitrate estimation performs in the same simulation.

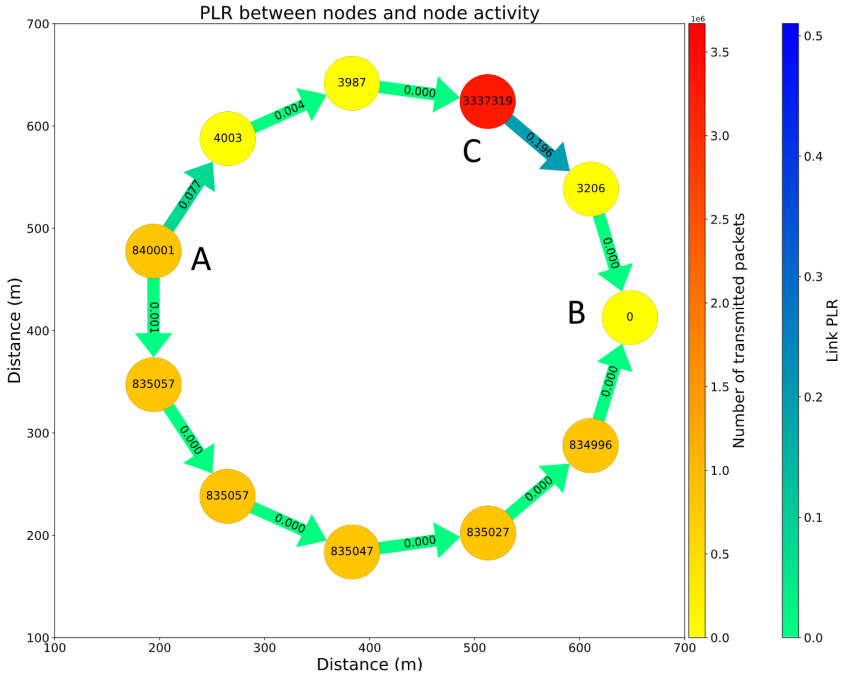


Figure 6.16: Simulation results using the proposed bitrate and usage estimation for the throughput metric. This scenario is simulated for an hour, while the logging started after 600 seconds.

This time **A** decides to route 99.48% of the packets through the long route, where only 0.1% of the packets were lost in the first hop, with a minuscule packet loss on the subsequent hops. However, a small number of packets were still routed through the short route where approximately 27.1% of these packets were lost, before reaching **B**. Finally, using the second proposed way of estimating the usage ratio resulted in 99.78% of the data packets reaching **B**, compared to the 46.6% from standard B.A.T.M.A.N..

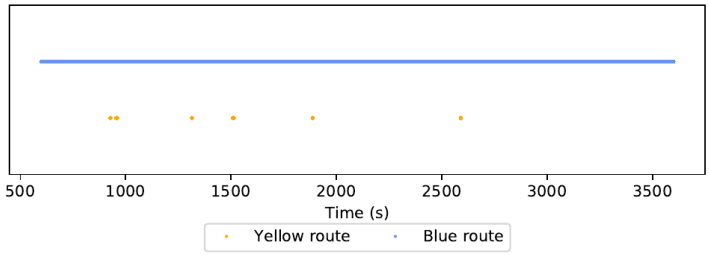


Figure 6.17: Shows data packets over time as either a blue or yellow dot to represent which route it travelled in reference to figure 6.13, using the proposed bitrate estimation together with **approach 2** for usage estimation.

Like figure 6.15, figure 6.17 visualises which of the two routes are chosen throughout the simulation using **approach 2** for the usage estimation. Going forward the usage estimation is estimated and calculated as described in **approach 2**.

6.2.3 Packet success rate estimation

Another important metric in the throughput estimation is how reliable a link carries the packets. Here this is considered as a rate at which packets successfully arrives at the recipient. A packet is considered lost if it is successfully sent from the B.A.T.M.A.N. layer but does not arrive at the recipient's B.A.T.M.A.N. layer. In this project, three different packet loss levels are considered:

- The system-wide PLR, which is the collective loss of the full system
- The end-to-end packet loss, where a packet is lost somewhere in a route between two nodes in the network
- The link packet loss, where a packet is lost on a specific link between two neighbouring nodes

These levels are related to each other, if there is a high packet loss in a lot of routes it results in a high system-wide PLR and if there is a high loss in a link it results in high loss for routes using that given link. Therefore, to reduce the system-wide PLR it is of interest to route around links with a high packet loss.

The system-wide PLR is in this project measured as the sum of all end-to-end transmissions versus the end-to-end receptions. This metric is used for general performance evaluation when comparing solutions. The link packet loss is in this project all the sent unicast packets from the B.A.T.M.A.N. layer, versus the received at the destination node, and is used for both evaluation and is also used for incorporating the local links' state into the throughput metric.

The PSR, as mentioned in section 6.1, is a measure of the probability of a packet successfully reaching its destination. Where the packet loss is the probability for a packet to be lost, so $PacketLoss = 1 - PacketSuccessRate$. One method of finding the PSR,

especially if B.A.T.M.A.N. is to function on any physical interface, is to track how many packets are sent and received at the B.A.T.M.A.N. layer between node pairs. Since the success rate on a link may change over time, it may be preferable to track how many packets are lost in a time frame and not indefinitely.

For tracking the loss in a link, two methods can be used, the nodes can either track how many packets are lost, or they can track how many bits are lost. This can be done by knowing the sizes of the packets sent and packets received. These two methods will give two slightly different results because they will be dependent upon the sizes of the packets. In the method where it is counted how many bits are lost, there is a bias toward it being worse to lose larger packets than smaller packets. In a scenario where there is a proportional large overhead compared to the time spent sending a packet, as with for example RTS/CTS enabled, it might not be preferable to make a loss of a large packet count more than the loss of a small packet. On the other hand, in a scenario where the majority of the time spent sending a packet is used sending the actual payload, it might be preferable to track the bits lost. Since this project assumes some kind of flow control, it is chosen to count how many packets are sent and not how many bytes are sent.

For estimating the PSR on the nodes forming a link, the nodes need to share how many packets they have received from the other node via the link, as mentioned in appendix C.3 the ELP-probe packet is used to carry data to the neighbours. One way of tracking the amount of sent and received packets is to save a timestamp every time a packet is sent or received and then delete timestamps older than the chosen time window, when the number of packets is needed the timestamps are counted. This method might require a lot of memory at the nodes depending on how much traffic is sent and go through that node.

Another method is to define a bin size and then accumulate the number of packets sent or received into the bins. Then a window size can be controlled by having a maximum number of bins, and when the list of bins grows to the maximum number of bins, then a bin is deleted at the front of the list. This method requires less memory since it only depends on the number of neighbours. However, due to nodes being unsynchronised a discrepancy arise that in the worst case is equal to the bin size used. Because the bins used for counting packets between nodes can end up being offset between nodes and thereby give measures for different time periods. Therefore, if the traffic varies a lot, then the estimated packet loss might be slightly wrong, as one node might count a burst of packets while the other might not. Figure 6.18 is a sketch of how the bins on two nodes can cover different slightly offset time periods. This discrepancy can be combated to some extent by having a large time window or smaller bins so that the error becomes relatively small. But having a large window also makes the system slower to react to changes in the network, so if it is a static scenario it may be preferable to have a large window, but in a dynamic scenario, it may be better to have a smaller window. This method is chosen going forward.

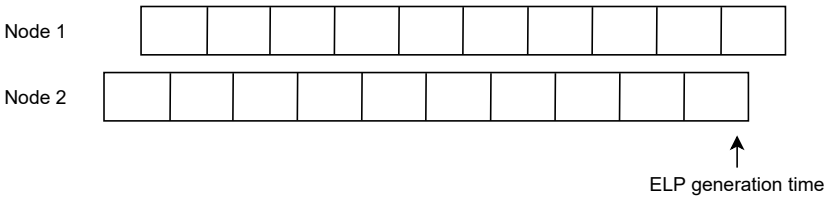


Figure 6.18: An example showing how the lists of bins can be offset due to nodes not being synchronised.

6.2.3.1 Evaluation of packet success rate metric estimation

To see if the estimation of PSR works a scenario is made consisting of four nodes as shown in figure 6.19 where the green nodes and the blue nodes are pairwise within communication range, but the green nodes are just outside the communication range of the blue nodes. Both the green nodes and the blue nodes are sending UDP-packets with a payload of $1408B$. The green nodes are sending with an interval of $3.57ms$ and the blue nodes are sending with an interval of $0.9ms$ which is chosen to produce a lot of interference for the green nodes resulting in packet loss for the green nodes, but still allow some packets to arrive so the OGMv2 and ELP packets can still arrive and maintain the network.

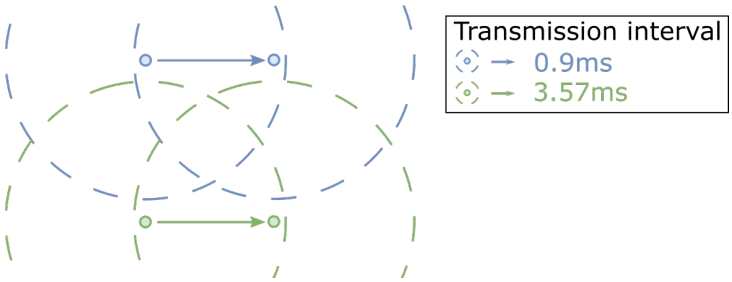


Figure 6.19: Illustration of the scenario for simulating the link PSR estimation.

Figure 6.20 shows how the implemented PSR estimation holds up against how many packets have actually been sent from the one green node to the other, when using a window of 60 bins and a bin size of $0.5s$.

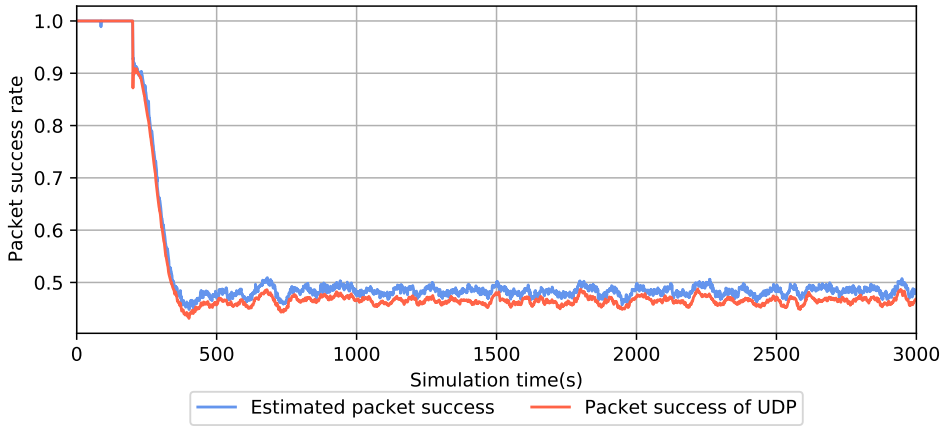


Figure 6.20: Estimation of the PSR over time.

In the figure it is shown that the tendency of the estimation is very similar to what is expected. The estimated PSR overestimates with about 2%, with the mean PSR having a mean from 500s to 3000s of 0.49 and the mean for the calculated success rate being 0.47.

6.2.3.2 Packet success rate in routes

For each link the PSR should be applied as equation 8 states. But if 10% packets are lost on the first link and again 10% packets are lost on the second link these losses accumulate. Therefore, the success rate of routes needs to be applied for every link, in practise this means that after the node has chosen the minimum throughput value this is where the PSR is applied.

To evaluate how the success rate metric performs, a setup as shown in figure 6.21 is made. A UDP stream at an interval of 3.57ms, and a payload size of 1408B, is set to be sent from the green node (**A**) to the red node (**B**) and thus has to choose between the longer blue route of 6 hops or the shorter yellow route of 5 hops. In the middle of the circle, a polygon is placed set to have the material of concrete without reflection. At the same time, the nearest purple node is set to send UDP packets, with an interval of 2ms to the other purple node, and thereby causing interference in the yellow route. All nodes are set to have a transmission rate of 54$Mbit/s$. The simulation is run for 3600s with logging and the UDP traffic from **A** to **B** starting after 600s.

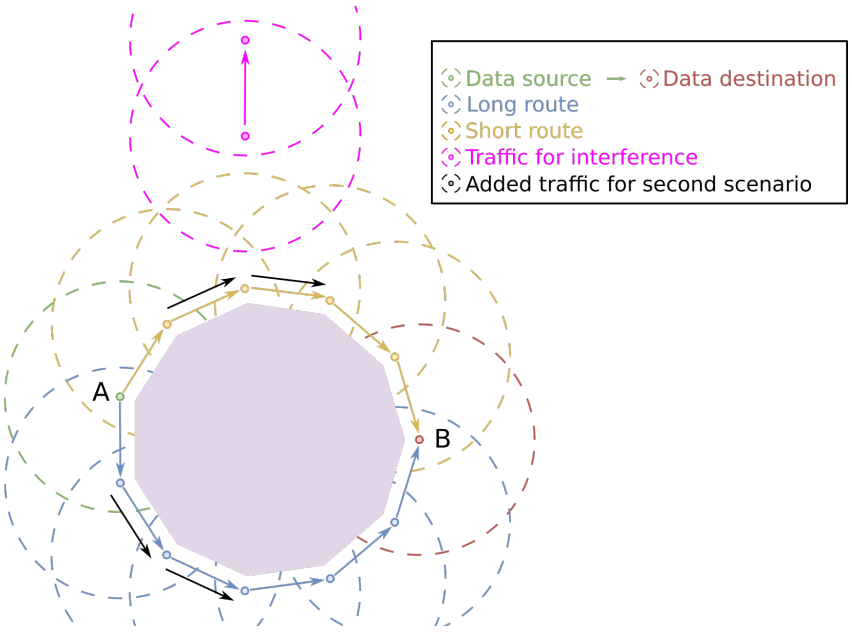


Figure 6.21: Simulation scenario for evaluating the PSR metric in routing, with traffic flowing from **A** to **B**, while the purple nodes generates interference for the shorter route.

This simulation is first run with the routing metric only consisting of the bitrate estimation found in section 6.1, resulting in a packet loss of 39.4%, with 13.5% of the packets being routed through the long route. Then the simulation is run with the routing metric consisting of the bitrate and the PSR. This simulation results in a packet loss of 39.2%, with 14.1% being routed through the long route. This result is somewhat expected since there needs to be traffic on a route to register the loss of packets, but since the ELP-probes are also counted toward the PSR the original assumption was that these were enough to trigger the loss. However, in a scenario like this ELP-probes creates such a small amount of traffic that most of them arrive at the neighbours.

Because of this, a new scenario is set up, as shown in figure 6.21, but now with some traffic along the black lines, with an interval of $50ms$. Because of the added traffic to the network, the interval **A** is sending with is adjusted to $10ms$. Again with only the bitrate as the metric the packet loss from **A** to **B** is 82.1%, with 8.7% taking the long route, but with the success rate metric, the packet loss is 23.2%, with 72.0% taking the long route. Looking at the data it is apparent that most of the time the short route is chosen with the success rate metric, this happens in the first $200s$ and then less and less until nearly never choosing the short route after $1200s$, this signifying that the network needs time to stabilise, as shown in figure 6.22.

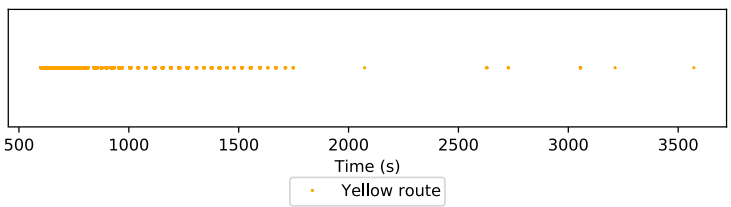


Figure 6.22: A timeline showing when the short route is chosen, signified by the dots.

Figure 6.23 shows how the packets are lost throughout the network. The main data stream is sent from node **A** to **B**. Here it is shown that a lot of packets are lost in the first two links on the short route, and the majority of packets have taken the long route.

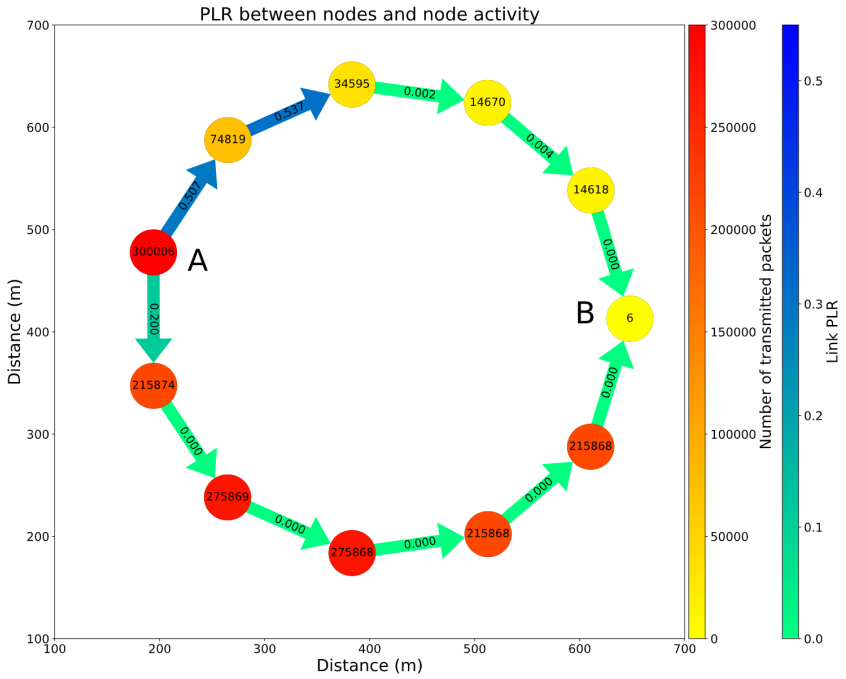


Figure 6.23: Simulation results using the proposed bitrate and PSR estimation for the throughput metric. The nodes describe how many B.A.T.M.A.N. unicast packets is sent, and the link describes the PLR in that link. Simulation is run for an hour with logging after 600*s*.

6.2.4 Throughput estimation

The previous sections have addressed bitrate, channel usage, and PSR and to combine these into the throughput of a link equation 12 is deduced.

$$Throughput = Bitrate \cdot (1 - Usage) \cdot Successrate \tag{12}$$

If the constellation from section 4.3 again is considered, as shown in figure 6.24, one problem here stemming from original B.A.T.M.A.N. is that the throughput of the route is evaluated in the opposite direction of what is actually the direction the data is sent. This means that if **E** wants to send a packet to **A** the throughput is actually evaluated from **A**→**E**. This works fine when all links are symmetric, but if asymmetric links are introduced this can be a problem.

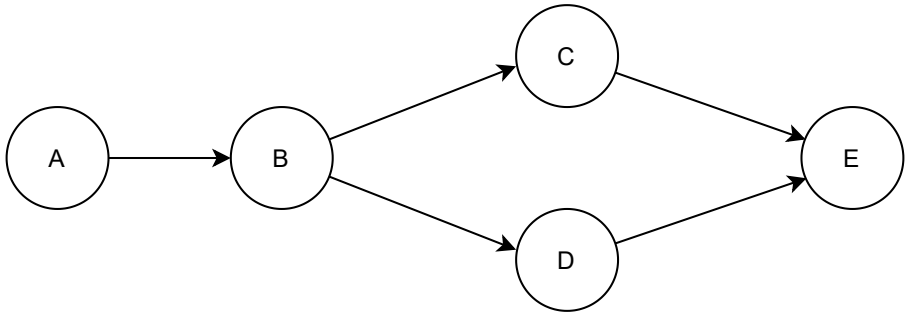


Figure 6.24: Example network, for illustrating throughput metric.

This problem is eliminated with ELSV since now **B** knows what the bitrate is from **B**→**A**. So when **B** receives an OGMv2 from **A**, **B** uses the estimated bitrate from **B**→**A**, and the same is again the case in the context of channel usage and PSR.

To apply the BUS metric to the routes in a network, using the standard B.A.T.M.A.N. hop penalty discussed in section 4.3. When an OGMv2 arrives at a node, the node calculates the $bitrate \cdot usage$ of the link the OGMv2 arrives through. Then the node chooses the lowest throughput of what is in the OGMv2 packet and what the node has estimated the link to be. Then the PSR is applied, and the throughput toward the originator from this node is estimated. Then, when the OGMv2 packet is forwarded the throughput is handled according to the B.A.T.M.A.N. hop penalty, described in section 4.3.

To evaluate how effective the BUS metric is versus original B.A.T.M.A.N. with *throughput override*, two network scenarios are used. Both the networks used here are grids with 100 nodes and the scenarios used are scenario 1 and scenario 2, as mentioned in appendix C.5. So the following simulations are run:

F Original B.A.T.M.A.N. with random traffic.

G B.A.T.M.A.N. with the BUS metric, with random traffic.

H Original B.A.T.M.A.N. with cross traffic.

I B.A.T.M.A.N. with the BUS metric, with cross traffic.

For **H** and **I** respectively, the system-wide PLR is 48.73% and 41.05%. In figure 6.25 it is shown how many B.A.T.M.A.N.-unicast packets carrying UDP are transmitted from each node, signified by the colour of the node, while the colour of the arrows signifies the PLR of the links, for simulation **H**.

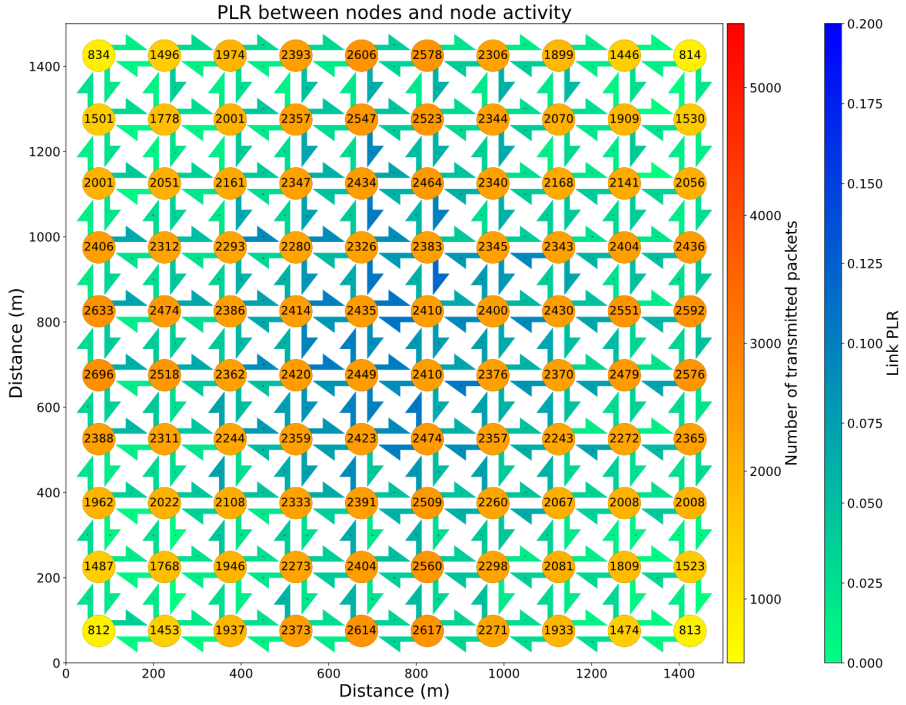


Figure 6.25: Each node showing how many B.A.T.M.A.N.-unicast packets carrying UDP is transmitted and the arrows showing the PLR for simulation **H**.

In figure 6.25 it is shown that a lot of the traffic is routed through the middle of the grid where also a high PLR is present.

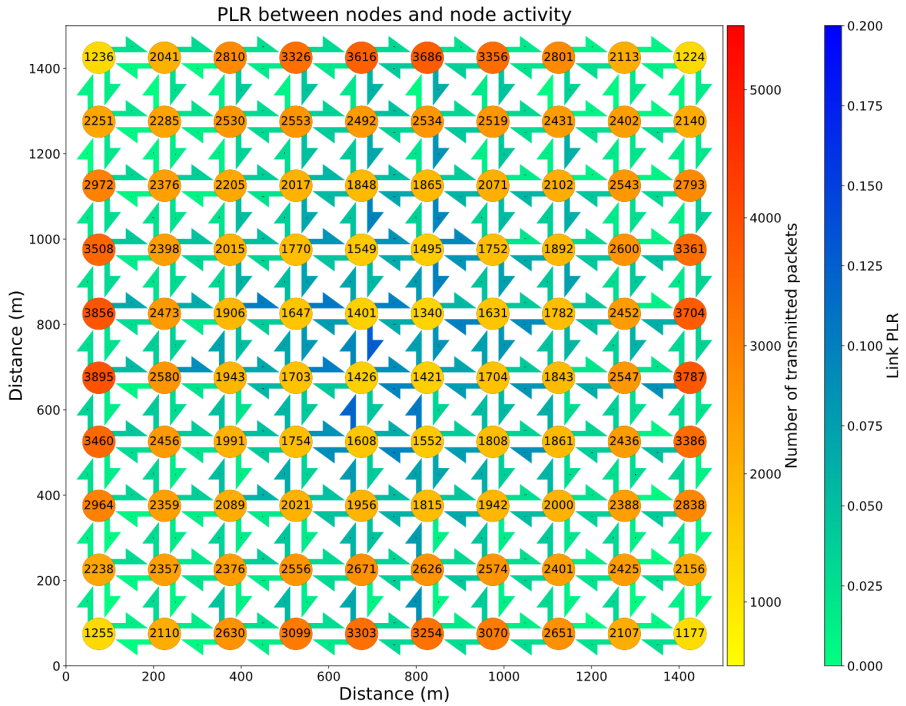


Figure 6.26: Each node showing how many B.A.T.M.A.N.-unicast packets carrying UDP is transmitted and the arrows showing the PLR for simulation **I**.

Figure 6.26 shows PLR and packets transmitted for simulation **I**. Here the traffic is moved more toward the edge of the grid where the links do not suffer from such a high PLR as the middle of the grid does. This is indicated by the edge nodes being slightly more orange, compared to the nodes in the middle.

For **F** the simulation shows a system-wide PLR of 37.41% while **G** has a loss of 36.14%. Illustration of the network traffic for these simulations are shown for **F** in figure E.1 and for **G** in figure E.2 in appendix E.

A larger improvement between **H** and **I** is observed compared to **F** and **G**. This does make sense when considering that for **F** and **G** a lot of the traffic is generated in the middle of the grid and this traffic can not be routed away from the middle, where a high amount of interference is present. Whereas for **H** and **I** all the traffic originates from the edge of the grid and the destinations are at the opposite edge. This gives the possibility for many packets to route around the middle of the grid.

To illustrate that the link metric is actually acting as intended, figure 6.27 shows what the throughput metric is for each link at the end of the simulation time in simulation **I**.

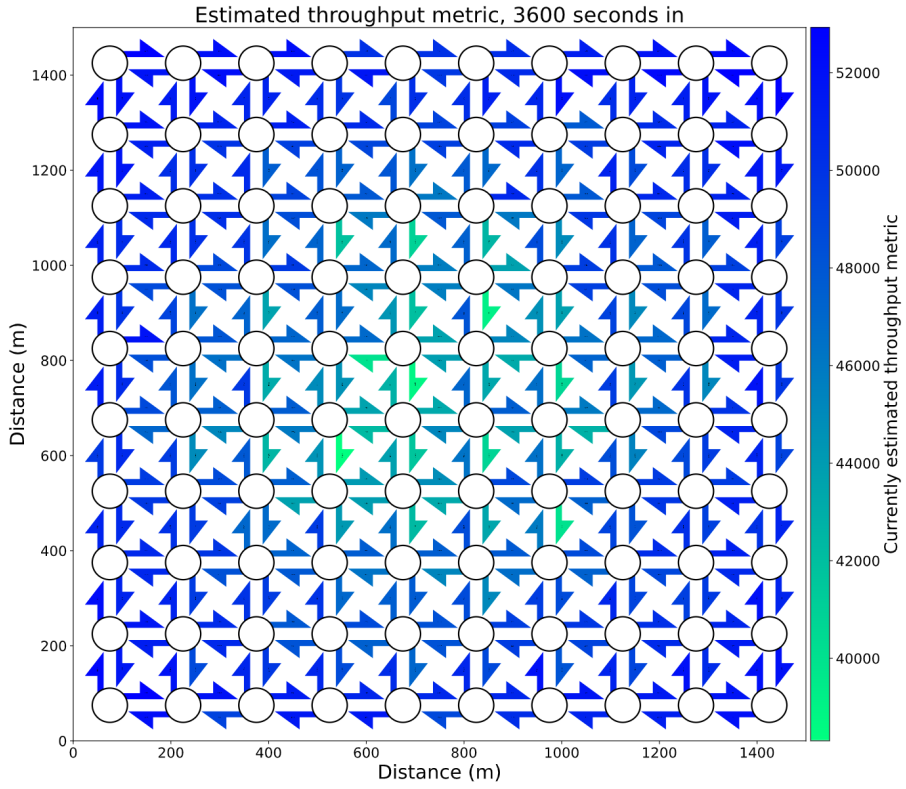


Figure 6.27: Throughput metric estimation of each link for simulation **I**

Figure 6.27 shows that at the edges the throughput metric is nearly estimated to be the full $54 Mbit/s$, but closer to the middle of the grid the link throughput metric drops below $40 Mbit/s$. This also confirms why the traffic is moved outward in the grid.

7 Part 3 - Hop penalties for multi-hop routing

This part focuses on the estimation of end-to-end capacities of routes that are used by B.A.T.M.A.N. to route packets through the network. First B.A.T.M.A.N.'s forward penalties are discussed and how this method deviates from literature and experimentations. Later a new solution is proposed and evaluated.

7.1 Forward penalties

One of the things that seem weird about original B.A.T.M.A.N. is the penalty that it applies to the throughput metric when it receives an OGMv2 packet. As described in section 4.3, B.A.T.M.A.N. halves the throughput metric, in the received OGMv2 packet, if it is above $1Mbit/s$, and reduces it by a factor of $\frac{15}{255}$ or 5.88% if it is below. The B.A.T.M.A.N. documentation claims that this halving is due to the store and forward property of wireless communication, and the 5.88% reduction is chosen with no specific reasoning, quoting the documentation itself:

"The default hop penalty of '15' is a reasonable value for most setups and probably does not need to be changed." — [24]

The halving does make sense since a node on a single radio channel is unable to send and receive at the same time, which means that any node forwarding traffic would be subject to this. However, applying the forward penalty at every hop seems unreasonable because a route consisting of 50 hops would heavily reduce the metric to a point that does not necessarily represent what is possible to transfer. This simple nature of the hop penalties therefore on longer routes favours the shortest routes as they would be the ones reduced the least.

The paper by Jinyang Li et al. from 2001, gave inspiration to using a different forwarding penalty [25]. The paper itself is generally about some specifics of 802.11 MAC and investigate the capacity of different networks and the individual nodes the networks are comprised of. One of the scenarios they address is the capacity of a chain of nodes. In this scenario, they address how the amount of nodes in the chain that can transmit simultaneously is one of the limiting factors for the capacity of the whole chain. For example, in a chain of nodes, with identical links, a message is sent towards the last node, how many hops does it need to take before a new message can be sent, without them colliding or contending.

Jinyang Li et al. states that theoretically, it should be possible for $\frac{1}{3}$ of the nodes to transmit at the same time, with each link utilising $\frac{1}{3}$ of the capacity, assuming all link capacities are equal. The paper assumes RTS/CTS is enabled for all packets and therefore after two hops the transmission should no longer affect the next transmissions directly in the form of direct contention or negotiation. This is shown in figure 7.1 where both node 1 and 4 can transmit at the same time without having collisions at the receiving nodes.

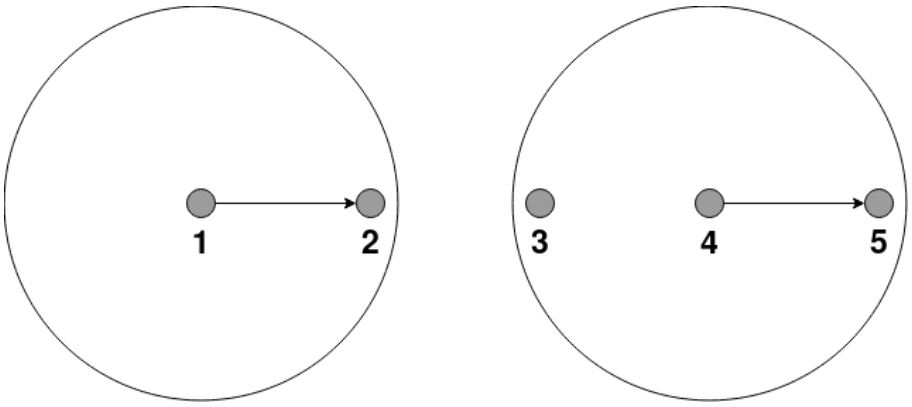


Figure 7.1: Sketch of two transmissions separated by two hops.

However, as Jinyang Li et al. also addresses, this does not consider interference range. The paper then states that if the interference range is twice the transmission range, then transmissions from node 4, as shown in figure 7.2, will interfere with node 1's transmission, reducing the theoretical utilisation of the capacity of the links to $\frac{1}{4}$ as only every fourth node can transmit at the same time.

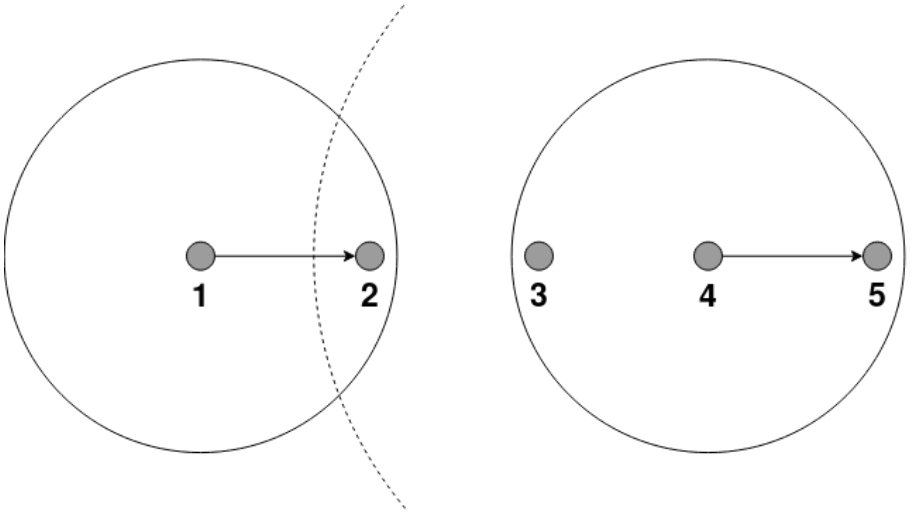


Figure 7.2: Sketch of two transmissions separated by two hops, with the interference range from node 4's transmission illustrated with the dotted circle[14].

Furthermore, Jinyang Li et al. simulates chains of different lengths, to test the end-to-end capacity of the chains. They observe that in reality, the capacity decreases to approximately $\frac{1}{7}$ of the maximum capacity, and that the capacity at four hops almost has reached this value and that the capacity only decreases slightly after this before settling in at $\frac{1}{7}$.

Generally, this project uses a different interference model with a larger range where the severity of the interference is dependent on the distance to the transmission. But for both single-hop and two-hop routes, the hop penalty of the routes is expected to follow B.A.T.M.A.N.'s assumptions. However, for routes longer than two hops, the hop penalty

should settle towards a specific fraction of the capacity being able to be utilised.

Single-hop
For a single-hop, there is only one transmitter and one receiver and if neither of the participants in this route switches roles, it is possible to send with the link's full capacity.

Tw hops
For a two-hop route, it is only possible to send with half the capacity of the links due to the store and forward property, assuming identical link capacities.

Three or more hops
For a route of three or more hops, the capacity will be lower than $\frac{1}{3}$ dues to RTS/CTS but what it converges to is determined by the interference and randomness of the back-off mechanisms. But as Jinyang Li et al. mentioned with interference, it is most likely $\frac{1}{4}$ or lower.

7.2 An alternative to B.A.T.M.A.N.'s hop penalty

To figure out what the hop penalties should be, to more accurately portray how much data can be sent through a route, two scenarios are constructed. One where the nodes are equally spaced on a line, similar to the chain in Jinyang Li et al., but still using the more realistic interference TRGR radio model. In the second scenario, the nodes are placed in a zigzag pattern, with walls placed so that each node only is in line of sight with its neighbours, effectively removing interference between nodes.

These two scenarios are illustrated in figure 7.3.

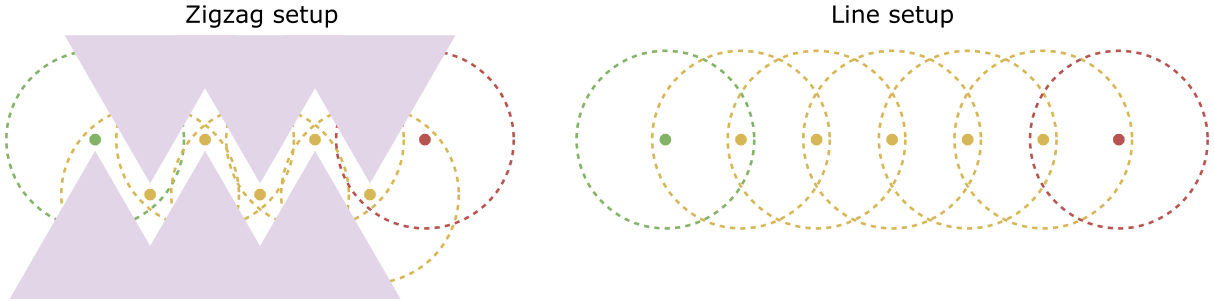


Figure 7.3: To the left is a sketch of the setup in a zigzag pattern, and to the right the nodes are placed on a line.

Then for both of these scenarios, routes of different lengths are simulated, and the amount of traffic is adjusted to the maximum that is possible to be transmitted from one end to the other, without observing significant packet loss. The number of packets sent per second for the different route lengths and the corresponding PLR is shown in table 1. Similar to the Jinyang Li et al. all links are set to the same underlying physical bitrate of $54Mbit/s$.

Table 1: Packets per second send from end-to-end in routes with varying amount of hops, in both the line and zigzag scenario. Each packet with a size of $1408B$, with all links transmitting with $54Mbit/s$.

Hops	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**25**
Zigzag pps	2130	1144.16	655.74	567.86	527.98	505.82	484.96	471.48	471.48	448.43
Zigzag PLR	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Line pps	2130	1144.16	373.29	266.67	219.78	194.17	154.69	121.19	114.29	85.65
Line PLR	0.0	0.0	0.0015	0.0026	0.0029	0.0020	0.0006	0.0004	0.0017	0.0035

Assuming that all links are equal then the values for one-hop, for both zigzag and line in table 1, are the maximum amount of traffic that can be sent over any link. An estimation of what the hop penalty should be, can be calculated using the one-hop data as a reference. Figure 7.4 shows how the hop penalty should decrease the routing metric for routes of varying length. B.A.T.M.A.N.'s hop penalty is also shown for comparison.

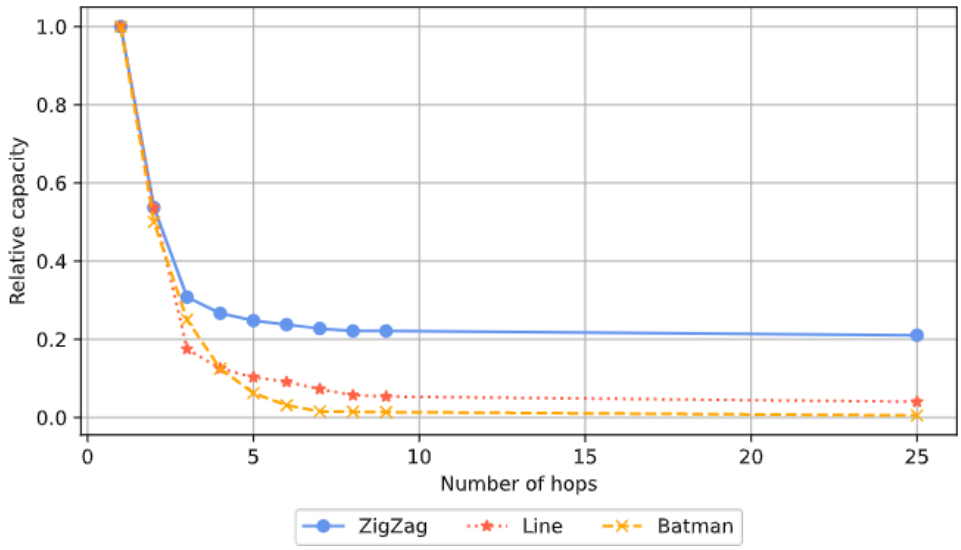


Figure 7.4: Shows what the hop penalty should decrease the routing metric with at a specific hop. Both for the ideal zigzag scenario with essentially no interference, but also for the line scenario where interference plays a roll. B.A.T.M.A.N.'s hop penalty is also shown in comparison.

The measured decrease in possible throughput is generally higher than what B.A.T.M.A.N. estimates, with the only exception being at three hops in the line being measured lower. Confirming that B.A.T.M.A.N. does seem to be too punishing regarding long routes. But even as it might seem like B.A.T.M.A.N. is close to the scenario with noise, B.A.T.M.A.N.'s estimation is often much smaller relatively. For example, at 25 hops for the zigzag scenario, the relative capacity is 0.21, the line scenario 0.04 and for B.A.T.M.A.N. it is 0.005, where B.A.T.M.A.N.'s estimated capacity is 8 times smaller than the line scenario and 42 times smaller than the zigzag scenario. This relative difference is however dependent on what the starting capacity is, because for B.A.T.M.A.N. if it starts higher it will decrease faster and if it is lower it will decrease slower.

The measured relative capacities are what the penalties should match, with the results measured in the zigzag being the highest possible throughput as interference has been removed, while the line scenario is closer to what would be experienced with interference. In reality, the most representative hop penalties therefore probably depend on the interference in the specific scenario, making it difficult to say what the exact penalties should be. But as the BUS metric on its own considers interference and congestion, a combination between the BUS metric and the zigzag penalties are considered as they would end up with the multi-hop routing metric somewhere between the zigzag scenario and the line scenario. Which even if it is not the most representative, it will be closer to the actual capacity of a route.

One of the big advantages of having a more representative hop penalty is that the estimated route capacities are much closer to their actual capacities. Furthermore, this hop penalty differs from B.A.T.M.A.N.'s in such a way that longer routes are more likely to be more equal further helping the BUS metric to make a difference in routing, with longer routes being more likely to be chosen.

7.3 Forward penalty with heterogeneous links

The forward penalty considered in B.A.T.M.A.N. is as shown inaccurate both in the values it generates for the metric, but also in the relative metric from hop-to-hop. This is even assuming that B.A.T.M.A.N. has access to an accurate throughput metric. The paper by Jinyang Li et al. generally argues that the forward penalty should be determined by the number of nodes that concurrently can transmit. The values they measure in simulation seems to indicate that the capacity of a chain of nodes converges to $\frac{1}{7}$ the capacity of a single link's capacity, compared to the $\frac{1}{4}$ they predicted. [25]

However, Jinyang Li et al. assumes that the nodes and links are identical and the previous measurements, in table 1, are also done on links with the same underlying bitrate. As previously mentioned, this project assumes links are generally different, and the throughput metric is used to estimate a link capacity representative of the difference. The paper does not address what happens when the links differ and the network becomes heterogeneous.

The same logic can perhaps be adapted to such scenarios with the goal to devise a method for calculating any routes' capacity with arbitrary individual link capacities. Because it still applies that due to RTS/CTS only $\frac{1}{3}$ of nodes can transmit at the same time, and interference and nonoptimal scheduling further lower this ratio. This method should also match the results gathered so far for hop penalties where all hops have an underlying bitrate of $54Mbit/s$.

For simplicity interference and scheduling is not considered, so theoretically it is again only possible for $\frac{1}{3}$ of the nodes to transmit concurrently.

Single-hop
For routes that are only one hop, it is possible to transmit data at the throughput capacity of that one link.

Two hops

For routes with two hops the links' throughput capacity can differ, for example, one hop could have a capacity of $54Mbit/s$ and the other could be $6Mbit/s$. Additionally, they could be combined two different ways, in the order of $54Mbit/s$ then $6Mbit/s$ or $6Mbit/s$ and then $54Mbit/s$. These two scenarios are illustrated in figure 7.5.

Figure 7.5: Sketch of the two combinations of $54Mbit/s$ and $6Mbit/s$ routes, in both cases traffic is going from 1 towards 3.

Previously the store and forward property determined that the capacity should be halved because a node cannot send and receive at the same time, however, the halving is done on the assumption that the capacity is the same on both links. If one considers that only one of these hops can actively transmit at a time and that the $6Mbit/s$ capacity is 9 times slower than $54Mbit/s$ capacity. The highest possible route capacity can only be achieved by using the links unevenly but balanced so that the lowest link is active as much as possible, as it is the bottleneck in this scenario. The optimal usage ratio between the two links is $\frac{1}{10}$ and $\frac{9}{10}$. As the node in the slow link takes 9 times as long to use the slow link as the fast one. The route throughput would then be $\frac{1}{10}$ of $54Mbit/s$ or $\frac{9}{10}$ of $6Mbit/s$ which is:

$$\frac{1}{10} \cdot 54Mbit/s = \frac{9}{10} \cdot 6Mbit/s = 5.4Mbit/s$$

This value is also the same no matter the order in which the link capacities are placed.

Three hops

For routes consisting of three hops, the same applies as in the case of the two-hop routes. Only one link can be active at a time, with the slowest link being as active as much as possible. For example, in a scenario with the following links $54Mbit/s$, $18Mbit/s$ and $6Mbit/s$ in any order. Now the medium is shared between all three links, the relative capacity between links is that $18Mbit/s$ is three times slower than $54Mbit/s$ and $6Mbit/s$ is 9 times slower. Here the ideal ratio of activity is $\frac{1}{13}$ activity for the $54Mbit/s$ link, $\frac{3}{13}$ activity for the $18Mbit/s$ link and $\frac{9}{13}$ activity for the $6Mbit/s$ link. The total route throughput would then be equal to any of the links multiplied with their activity ratio:

$$\frac{1}{13} \cdot 54Mbit/s = \frac{3}{13} \cdot 18Mbit/s = \frac{9}{13} \cdot 6Mbit/s = 4.15Mbit/s$$

Longer routes

This concept continues for longer routes, but as it then becomes possible to have more than one transmission on the route concurrently, the routes throughout is equal the three consecutive links whose nonconcurrent transmissions equals the lowest throughput of the entire route. This is because it would never be possible to transmit faster end-to-end on the whole route than the slowest bottleneck of three consecutive links.

7.3.1 Implementing in B.A.T.M.A.N.

The way this is handled in practice on arbitrary length routes, is by having a sliding window in each OGM packet that contains the last two links' throughput capacity metric, along with the local information of the opposite direction link the OGM packet just arrived from. Using these three links to then calculate the route capacity of those links, is done at every hop the OGM packet takes. Within the OGM packet is also stored the lowest observed route throughput capacity so far, as it would never be possible to have a higher capacity towards a destination than the lowest bottleneck. This method of using a sliding window is termed Sliding Window Adjusted Penalty (SWAP), and will be referenced as such going forward.

As a side note, for this to work with the BUS metric the accumulated PSR is stored in the OGM packet, so that it can be applied more correctly.

In figure 7.6 it is shown how the OGM packet travels away from its originator node to any node **N**, with the first three consecutive links towards that originator sketched.

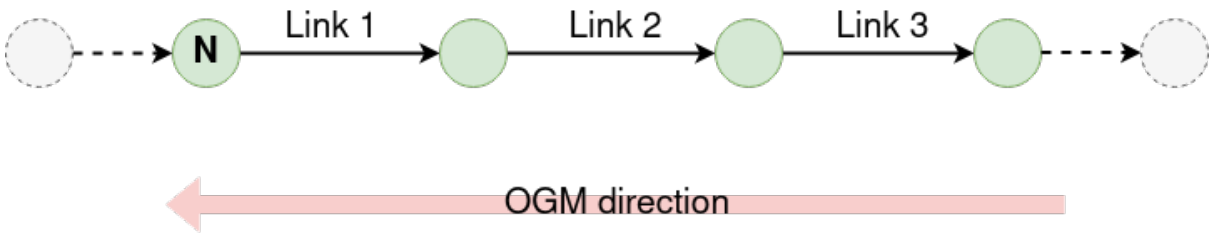


Figure 7.6: Sketch of three consecutive nodes, that sends data non concurrently from node **N** along the links towards a destination on the right, with the OGM packet travelling away from the destination.

A route throughput calculation, done at node **N** for the next three consecutive links towards the destination, is shown in equation 13. Here the information of $Link_1$ is locally available at that node, and the capacity knowledge of $Link_2$ and $Link_3$ is known from the OGM packet that has just reached node **N**.

$$\frac{Link_1}{1 + \frac{Link_1}{Link_2} + \frac{Link_1}{Link_3}} = Route \tag{13}$$

Where:

$Route$ - is the throughput capacity of three consecutive links at any node **N** towards the destination.

$Link_1$ - is the capacity of the first link from node **N** towards the route destination.

$Link_2$ - is the capacity of the second link from node **N** towards the route destination.

$Link_3$ - is the capacity of the third link from node **N** towards the route destination.

This expression can be adjusted to two hops by removing the fraction $\frac{Link_1}{Link_3}$ from the equation, or likewise be increased to consider more consecutive and non-concurrent links by adding a fraction more $\frac{Link_1}{Link_n}$ for every link added.

7.3.2 Evaluating SWAP

To compare SWAP, which can handle heterogeneous links, to B.A.T.M.A.N.'s hop penalties described in section 4.3, this method's hop penalty is shown together with the penalties from figure 7.4 in figure 7.7. Since the figure 7.4 only considered homogeneous links, so does figure 7.7.

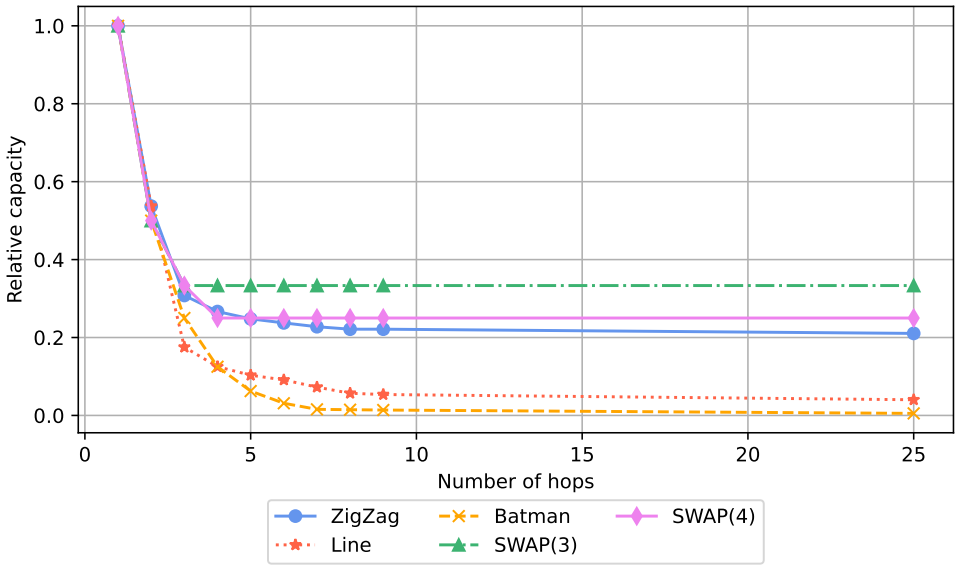


Figure 7.7: Shows the hop penalties effect on the throughput metric in the case of homogeneous links. The three hop penalties from figure 7.4 is shown compared to SWAP's way of calculating the hop penalty discussed in this section, while considering the last 3 and 4 hops indicated by SWAP(3) and SWAP(4).

Figure 7.7 shows that using SWAP for calculating the hop penalty, considering only the last three hops, would end up penalising the routing metric considerably less than the hop penalty for the ideal zigzag scenario. However, considering the last four hops in SWAP, yields a result more similar to the zigzag scenario this is used instead and is referenced as SWAP(4). This also fits into the findings of Jinyang Li et al., where theory and experimentation say that due to interference and nonoptimal scheduling, the route capacity is $\frac{1}{4}$ of a single-hop's capacity, or lower, on longer routes.

Worth noting here is that SWAP(4) does not reduce the routing metric after four hops. But it is speculated that if SWAP is combined with the BUS metric, a similar small decrement of the routing metric on increasingly long routes, as the zigzag and line scenarios experience, would be observed.

Similarly, a simulation is set up, where the nodes again are placed in the same way as for

the zigzag pattern, shown to the left in figure 7.3. However, the difference here is that the focus is on heterogeneous links, so the simulation is run twice; once where the first link in the chain acts as a bottleneck and once where it is the last link. The simulations are run for chains of nine hops, where all the links have an underlying physical bitrate of $54Mbit/s$, except for the bottleneck of $6Mbit/s$. The results for the simulations is shown in table 2.

Table 2: The relative capacity after 9 hops, compared to the maximum of a single-hop on a $54Mbit/s$ link, calculated using each method. Along with the measured relative capacity of 9 hops in the two heterogeneous scenarios.

	B.A.T.M.A.N.	**SWAP(3)**	**SWAP(4)**	**Measured**
Bottleneck first hop	0.0099	0.0909	0.0833	0.1139
Bottleneck last hop	0.0138	0.0909	0.0833	0.1145

The table shows that the relative capacity of a 9 hop route is measured to be 0.1139 and 0.1145 of the capacity of what is possible to be sent over a single hop. Furthermore, the table also shows what the different hop penalty methods estimates is possible to send over a route consisting of 9 hops.

Generally, the measured results are much closer to SWAP(3) and SWAP(4) than B.A.T.M.A.N.. Compared to the measured results, SWAP(4) underestimates by 26.86% and B.A.T.M.A.N. underestimates by 91.31% in the case with the bottleneck as the first hop, and this difference is slightly smaller in the case where the bottleneck is on the last hop. Generally, SWAP is more punishing than expected but still much more representative than B.A.T.M.A.N. and is, therefore, a better option. While SWAP(3) is closer than SWAP(4) it is not considered the better option of the two, as it would then overestimate in the cases shown on the graph 7.7. Overestimating is considered worse than underestimating because longer routes generate more usage and interference, therefore SWAP(4) is still considered the best option and is used going forward. Additionally, for SWAP the placement of the bottleneck does not have an impact on the last nodes metric, while for B.A.T.M.A.N. it gives two different results depending on the location of the bottleneck. This property is more representative of the measured results, as these results also show that the capacity is approximately the same no matter the placement of the bottleneck.

8 Evaluation

As a final evaluation, all the suggested improvements are combined and evaluated together for comparison against the original B.A.T.M.A.N. solution, to see the difference SWAP makes when combined with the BUS metric. As the metric on its own is evaluated in section 6.2.4.

The first overall improvement from this project is **better simulations**, compared to the previous work in [10]. The improvement comes from using a more representative radio propagation model and this model is used to improve the results for all simulations, in this section and others. So that both the performance of the original B.A.T.M.A.N. implementation and the new implementations are more accurate and representative of real-world performance.

The second suggested improvement, the **BUS metric**, is designed to estimate the outgoing links' capacity, independent of the lower protocol layers, and considering packet loss and usage at each link individually. This also makes all the links in the network less homogeneous and more heterogeneous. The effects of using this BUS metric for routing is shown in section 6.2.4, where it is shown that both the BUS metric varies across the network and that this helps in routing around areas with high interference and loss.

The last suggested improvement, **SWAP**, is designed to be more representative of the possible throughput for multi-hop routes, compared to original B.A.T.M.A.N. that has a larger penalty that favours shorter routes. Furthermore, the method for calculating the hop penalty is designed to also provide more accurate throughput metrics for multi-hop routes when the links are heterogeneous. Using SWAP should allow the BUS metric to also have a larger impact as longer routes should be less penalised.

To verify the combined impact of all the suggested improvements, a set of simulation scenarios, with both the BUS metric and SWAP implemented, are run to show the effects of potentially longer routes. But also to verify that in networks with even more heterogeneous links that they are handled better.

Therefore, the exact same scenarios from section 6.2.4 are run again to see the effects of SWAP. But also a similar set is run, where the nodes have different underlying physical bitrates, to even further make the links vary across the network.

8.1 Node configuration and traffic patterns

Homogeneous network: This setup is the same as for the throughput simulations run in section 6.2.4 and described in appendix C.5, with the nodes placed in a grid of 10 by 10 nodes. Where all nodes have a physical bitrate of $54Mbit/s$.

Heterogeneous network: This setup is again the same as for the throughput simulations run in section 6.2.4 and described in appendix C.5, with the nodes placed in a grid of 10 by 10 nodes. But for the heterogeneous network, the nodes physical bitrate is randomised, between $6Mbit/s$ and $54Mbit/s$, assigned in an approximately uniform

distribution as shown in table 3.

Table 3: The distribution of bitrates in the heterogeneous node network setup.

Bitrate	6 Mbit/s	9 Mbit/s	12 Mbit/s	18 Mbit/s	24 Mbit/s	36 Mbit/s	48 Mbit/s	54 Mbit/s
Amount of nodes	12	12	12	13	13	12	12	12

Random and cross traffic: As with the scenarios in section 6.2.4 the two types of traffic patterns simulated are *"Scenario 1 Random Traffic"* and *"Scenario 2 Cross Traffic"* as described in appendix C.5.

8.2 System-wide evaluation

For the four combinations of node configurations and traffic patterns, these scenarios are each simulated for one hour, with the last 50 minutes being logged. The homogeneous networks are only simulated once, whereas the heterogeneous are simulated with 7 realisations, where the assignment of physical bitrates are randomised.

For the traffic patterns, the expectation is routing matters more in the cross traffic scenario due to the longer routes than the random traffic scenario. As more varied link metrics and longer routes should allow the throughput routing metric and the hop penalties to make more of a difference.

In table 4 the system-wide PLR for these simulations are shown, with the heterogeneous results being the average of all the realisations. The table also contains the homogeneous results from section 6.2.4 where only the BUS metric is implemented, as this section used the same simulation scenarios.

Table 4: Table of the system-wide PLR, for two different traffic patterns and homogeneous and heterogeneous nodes.

	Homogeneous Random traffic	Heterogeneous Random traffic	Homogeneous Cross traffic	Heterogeneous Cross traffic
Original B.A.T.M.A.N.	37.41%	30.08%	48.73%	48.52%
BUS metric	36.14%	Not evaluated	41.05%	Not evaluated
BUS metric and SWAP	35.66%	31.63%	39.15%	42.69%

When using both BUS and SWAP, the system-wide PLR for the homogeneous nodes is decreased by 1.75% for the random traffic and by 9.59% for the cross traffic when compared to the original B.A.T.M.A.N. solution. However, the improvement between the BUS routing metric and adding SWAP only see a decrease of 0.48% for the random traffic and 1.91% for the cross traffic. This seems to indicate that the potential for longer routes does not make that much of a difference as the routing metric does on its own, but still gives a slight improvement for both types of traffic.

That SWAP helps less is not unexpected, since there are no expectations as to how large of an impact the two solutions would have compared to each other, except that both should

improve the system-wide PLR. But one speculation on this is that the grid scenario has many alternative routes with equal length, and therefore it is also the metric that has the largest impact, in choosing routes, as compared to taking longer routes. Specifically, the corner nodes can avoid the middle without extra hops, while only some edge nodes cannot avoid the middle without extra hops due to how the nodes are connected. This also leads to the belief that the impact of these solutions depends very much on the scenario and traffic patterns.

For the heterogeneous scenario, the comparison is only between original B.A.T.M.A.N. and the combination of BUS and SWAP. For the cross traffic the system-wide PLR is reduced by 5.83%, but for the random traffic, the system-wide PLR actually increase by 1.55%%. This is not ideal, as the hope is to get equal or better performance in all cases, and this shows that that is not the case. But the improvement gained in the cross traffic is larger than the setback of random traffic.

The observed increase in the system-wide PLR did cause some speculation as to why this happens, one idea is that since the routes in random traffic on average are much shorter, it could be the case that the relatively high increase in route length is not worth the trade-off of adding more interference for all other routes. Hinting at that there is a length of route at which this trade-off becomes worth it and increases performance, as it is observed with the longer routed cross traffic, where the relative increase in route length is much smaller.

For further understanding the impact of the BUS and SWAP and performance evaluating them, the average end-to-end delay of successful UDP transmission is also measured and these are shown in table 5.

Table 5: Table of the average end-to-end delay of successful UDP transmission measured in milliseconds, for two different traffic patterns and homogeneous and heterogeneous nodes.

	Homogeneous Random traffic	Heterogeneous Random traffic	Homogeneous Cross traffic	Heterogeneous Cross traffic
Original B.A.T.M.A.N.	5.32 ms	6.96 ms	14.18 ms	25:00 ms
BUS metric	5.58 ms	Not evaluated	14.21 ms	Not evaluated
BUS metric and SWAP	6.08 ms	7.51 ms	15.38 ms	24.33 ms

For the homogeneous scenarios and the heterogeneous random traffic scenario, the delay increases when using BUS and slightly more when using SWAP, with an overall increase versus the original B.A.T.M.A.N. implementation of 14.31%, 8.03% and 8.44%. But for the heterogeneous with cross traffic scenario the delay decreases by 2.68%. the latter is unexpected as the trade-off of using longer routes is an increase in end-to-end delay. This could be caused by the shortest routes becoming more successful.

For the homogeneous scenarios it is observed that when using both BUS and SWAP the delay increase is larger when compared to only using BUS. This does make sense as BUS focuses on using alternative routes no matter the length, whereas SWAP focuses on punishing longer routes less than original B.A.T.M.A.N.. This also shows that SWAP maybe has a larger effect than originally speculated based on the measured system-wide

PLR, where it is speculated that longer routes are not needed to avoid the middle.

Overall the system-wide PLR is reduced when using the BUS metric and is reduced slightly more when combined with SWAP for the simulated scenarios. But this is achieved by using different routes that are either equally long or longer and in turn, this increases end-to-end delay.

8.3 Individual node evaluation

Another evaluation of all the suggested changes is done by examining the difference it makes on a node to node basis. This is done by looking at each links' PLR and how many unicast packets containing UDP messages are sent by each node. These results are shown on top of the network topology as figures.

8.3.1 Homogeneous scenarios

For the combined BUS and SWAP implementation in the cross traffic scenario in a homogeneous network, the traffic load is shown as both numbers and with the nodes colourised accordingly. Furthermore, the PLR of the individual links are shown as colourised arrows between the nodes, being more blue in areas with high loss, shown in figure 8.1.

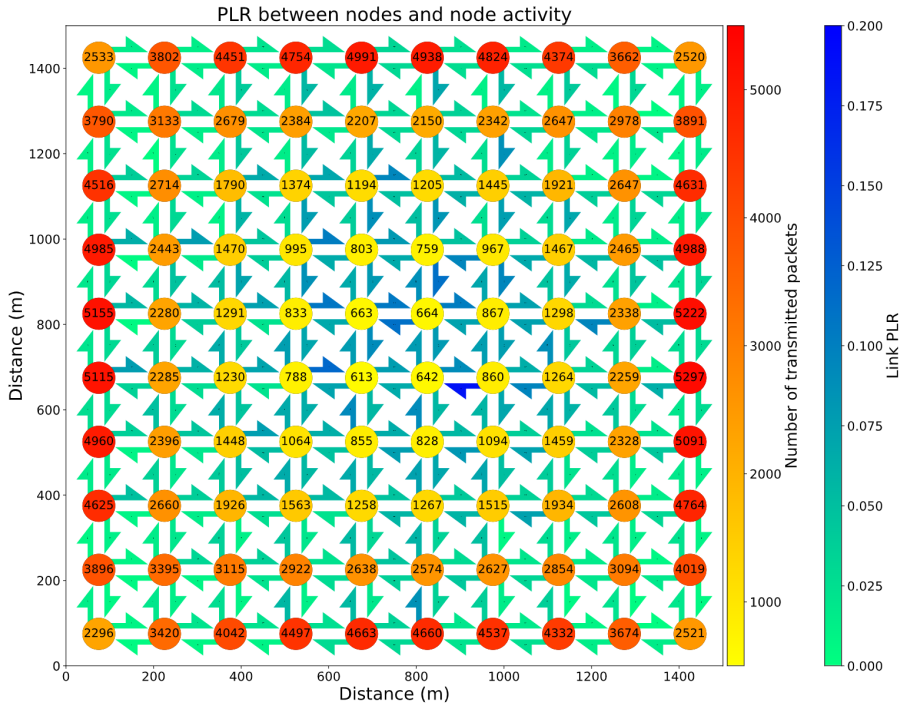


Figure 8.1: The UDP traffic forwarded by each node shown in the nodes, and the link arrows showing the PLR of the specific links. For the cross traffic scenario in the homogeneous network using both the BUS metric and SWAP.

The difference between figure 8.1 and the results from section 6.2.4 in figure 6.26 with

the BUS metric and figure 6.25 with *throughput override* is interesting as even more traffic is routed around the middle as opposed to through it. This is illustrated by the higher intensity of red nodes along the edge, signifying a higher traffic load, as well as a more yellow middle, indicating less load. As opposed to the previously mentioned figures, which were more balanced where most of the nodes are more orange in colour. This fits well with the goal of the SWAP concept as this seems to allow for longer routes resulting in less traffic through the middle. This, furthermore, confirms the speculation from the system-wide evaluation, that the addition of SWAP has the impact of avoiding the middle more, presumably through longer routes.

For the BUS metric and SWAP implementation in the random traffic and homogeneous network scenario the individual node results are shown in figure E.3 in appendix E and should be compared against the figure E.2 and figure E.1 both in appendix E as well. The comparison shows that like with the cross traffic, here when SWAP is used, traffic is to a larger degree routed around the middle.

8.3.2 Heterogeneous scenarios

For the heterogeneous scenarios, a single example is chosen to show the difference the new implementations make versus the original B.A.T.M.A.N. implementation, in a scenario with the cross traffic. This realisation is specifically chosen, as it is close to the average system-wide PLR of all realisations. The configuration of nodes in this specific realisation is shown in figure 8.2. Here the colour of the nodes indicates the physical bitrate for the specific node, red nodes are the slowest while green nodes are the fastest.

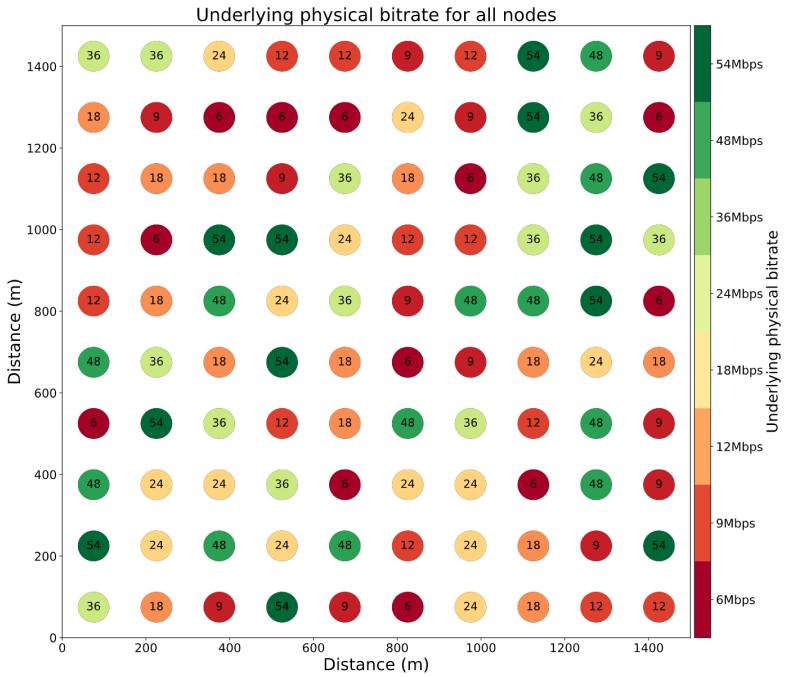


Figure 8.2: The specific distribution of physical bitrates in the heterogeneous network realisation referenced in figure 8.3 and figure 8.4.

The results of simulating this heterogeneous realisation is shown in figure 8.3 and figure 8.4.

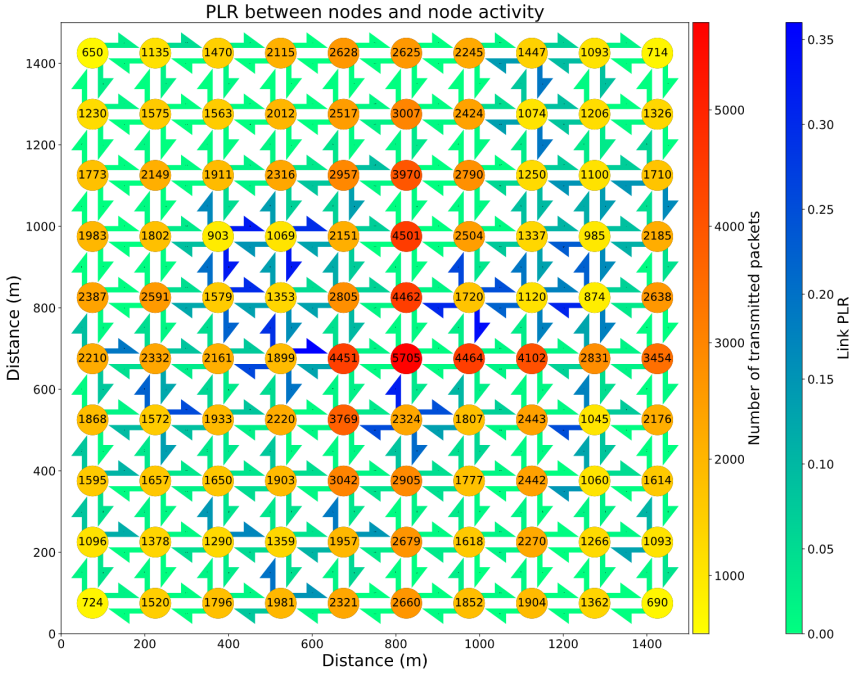


Figure 8.3: The UDP traffic forwarded from each node shown on the nodes, and the arrows showing the PLR of the specific links. For the cross traffic scenario in the heterogeneous network using original B.A.T.M.A.N..

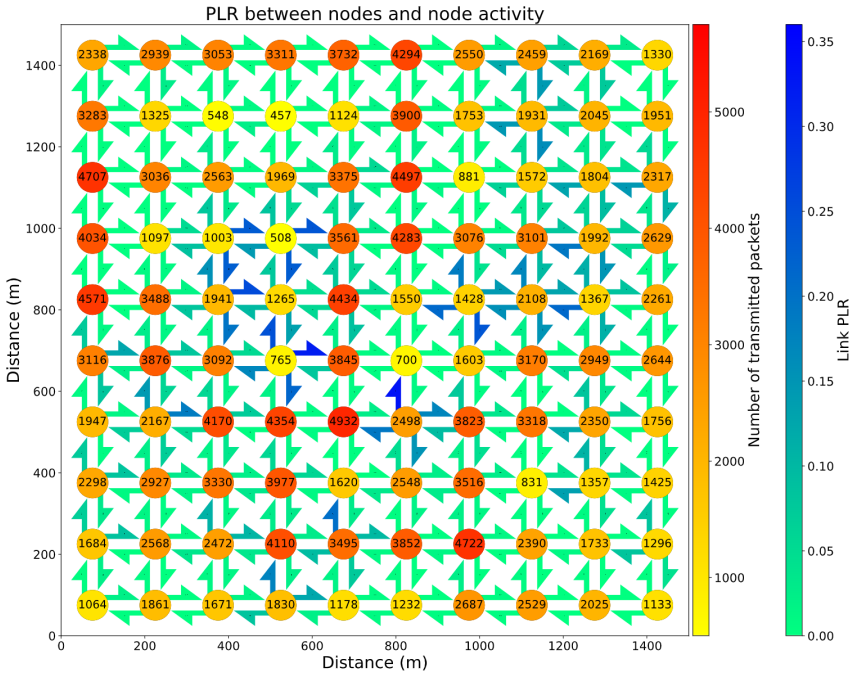


Figure 8.4: The UDP traffic forwarded from each node shown on the nodes, and the arrows showing the PLR of the specific links. For the cross traffic scenario in the heterogeneous network using both the BUS metric and SWAP.

The results from this specific heterogeneous realisation using the random traffic pattern are shown in figure F.1 and F.2 in appendix F.

The figures for heterogeneous nodes are much more chaotic compared to the homogeneous nodes. The shown heterogeneous scenario with cross traffic, shows that some nodes are avoided when using BUS and SWAP, for example, there is a node in the middle that have the most traffic through it, while also being one of the biggest bottlenecks, this node then becomes one of the nodes with the least traffic despite it being in the middle of the network. This observed by the node going from bright red to yellow. Another tendency observed is that some nodes act almost like a "highway" with a lot of traffic going through them compared to their immediate surroundings. These "highways" are seen as lines of bright red nodes. This is generally the effects expected from using the BUS and SWAP implementations, and it results in a lower system-wide PLR for this scenario.

For the random traffic, much of the same is observed, but here it ends up resulting in a higher system-wide PLR. It is hard to decipher from these data why this is the case, but it is speculated that previous short routes are forced onto the "highways" of the network, instead of being stubborn and going through the bottleneck causing congestion and more interference for other routes due to longer routes being used.

9 Discussion

This section discusses the tools, approaches and assumptions and what effect they could have on the results reached, along with some limitations of this project. This is done both generally but also for each specific part.

For this project OMNeT++, together with the INET Framework, is used to run the simulations. INET provides all the modules that mimics different parts of network protocols or wireless networks, while OMNeT++ uses these modules for the actual simulation. However, since this project is mostly based on simulations, it is assumed that the modules function like their real-world counterparts.

Generally, this project focuses on the system-wide PLR for evaluating performance and determine if improvements are made. However, the system-wide PLR does not describe how the packet loss is distributed across a network. This might make it difficult to evaluate performance of two widely different networks or routing schemes since they might experience the same system-wide PLR, but distributed across the networks in different ways. But to counteract this and get a more detailed understanding of how the network experience packet loss, the individual links' PLR are visualised along with the forwarded unicast packets. This helps understand where packets are lost and to see how the traffic load is changed between different simulations, and fills in the details the system-wide PLR does not describe.

For all results, the first period of time is never used for evaluation. This is done to allow the network to settle, since a scenario where all nodes start at the same time is not representative of likely real-world scenarios. But there are some interesting metrics in these periods, such as discovery time and the time it takes for a node to have accurate link metrics estimated. However, the mechanisms for discovery have not been modified besides the added OGMv2 jitter which on average adds one millisecond of delay per hop. For the time it takes to estimate the link metric, the window sizes are chosen to generally provide a good metric within that time frame.

Another important point, is that all the grid scenarios are unrealistic, in the sense that the nodes are placed evenly spaced out on a flat plain, where there are no other obstacles. This is far from the real-world, where there often are lots of different obstacles, which can affect the radio propagation negatively but also reduce the interference in a positive way.

Furthermore, the grid network as a topology is not always representative of what would be observed during real-world usage. As the size and connectivity will often vary making the network topology much different. But all nodes are also assumed to be static which is also unlike the use case of Systematic where nodes are expected to move around, and thereby changing topology over time. However, this is purposely not considered, as static networks are more simple to work with, and is therefore just a stepping stone for later considering movement.

Generally two types of traffic patterns are considered for the traffic load in the simulations. They might not be representative of all possible scenarios Systematic's use case would

experience. But they purposely differ both in the average length of routes, the distribution of load and the size of the load, in an attempt to still have representative results.

All the results and conclusions are only based on one protocol stack, namely Wi-Fi and TCP/IP. As one of the main parts of this project is to be independent of other layers, it is not ideal to make conclusions regarding this, without testing the proposed improvements on other protocol stacks. This is done due to time scope, however, at every step of developing the proposed solutions it is considered in a way to avoid making it Wi-Fi dependant.

RTS/CTS is assumed enabled, and this cannot be guaranteed to always be the case, and knowing if it is the case or not, is important for, for example, calculating the usage. For usage this will make it inaccurate by overestimating usage, however, the additionally considered nodes in this case still affect a single node's ability to transfer successfully, and is therefore perhaps not that much of a discrepancy. Furthermore, for ad hoc networks having RTS/CTS or similar mechanics makes a lot of sense as the hidden node problem is very present here, so assuming it is enabled is probably more likely to be correct than not.

All the results would benefit from more realisations and longer simulation time for better results and more confidence in them.

One element this project has ignored is the processing trade-off the proposed improvements have, since nodes' capabilities could be limited. As the implementation of the presented improvements uses more processing power and memory than the original B.A.T.M.A.N.-advanced. For the extra processing power required it is difficult to estimate how much this is, since it depends a lot on the implementation and the complexity of the underlying operations. But for the memory it is estimated to use around $20kB$ of extra memory per neighbour. This is due to various queues keeping track of data for the last x seconds, used to estimate the bitrate, usage and PSR towards each neighbour. It is important to note that this specific implementation is not focused around optimising neither memory consumption nor the processing power, and could probably be lowered.

Part 1 - Simulation environment and interference modelling
From section 5.2.2 it is shown that the interference considerably affects the system-wide PLR, and that being able to reduce the interference would similarly also reduce the system-wide PLR. By switching the path loss model, which was used in previous work [10], from the FSPL model to the TRGR model, the interference range is lowered significantly. However, because of the placement of the nodes in the simulated scenarios, a large part of the nodes are still inside the interference range of each other, which is why the switch between the models did not improve the system-wide PLR by much more than $\approx 6\%$. So while the switch between the two models, did not have a major impact on the PLR, it did result in using a path loss model which is more realistic.

Part 2 - Protocol independent link state estimation
The TAPT concept introduced in this part relies on a small measure of time, and as such also requires a high precision time resolution, and this should be considered if it is a problem for any piece of hardware. Additionally, the results gained from using TAPT in the simulator does not consider processing time and scheduling which would

be present on real hardware and would create additional jitter between packets reaching the B.A.T.M.A.N. layer, the effects of this are not considered, but are expected to have a negligible impact on the estimations.

The non-grid scenarios used to evaluate the bitrate, usage and success rate estimations, are created to highlight the advantage of considering each metric. The observed performance gained from these scenarios might therefore be exaggerated compared to how they would appear within a larger scenario or network.

When TAPT is used, it is done with probe sizes tailored to Wi-Fi for accurate estimations of the physical bitrate. As other probe sizes resulted in overestimations due to the specific modulation of Wi-Fi. This does not make the solution dependent on Wi-Fi, but only the accuracy. Furthermore, this accuracy is not an issue if all nodes are using the same protocol stack and hardware. Alternatively, high accuracy can be archived independently of any modulation by varying the probe size over time, to find the ideal size, with the only potential drawback being a higher settling time.

Part 3 - Hop penalties for multi-hop routing
The scenarios used to measure the actual capacities of multi-hop routes and prove that B.A.T.M.A.N. heavily underestimates, are not necessarily representative of all cases, as a grid case would probably fall in between B.A.T.M.A.N. and the line scenario. However, it is still expected to be higher than B.A.T.M.A.N., and it is also generally impossible to say which scenario is most representative, as the likelihood of each topology's appearances are unknown.

The SWAP method is only evaluated on its own in two small heterogeneous scenarios, where it is shown that SWAP is much more representative than B.A.T.M.A.N.. However, the average difference SWAP makes compared to B.A.T.M.A.N. is uncertain as there exists many combinations of heterogeneous nodes and only a very small set is used, but it is generally expected to underestimate less and therefore be more representative.

Furthermore, the SWAP method is designed to match the measured capacities of a zigzag scenario of nodes in a line topology with minimum interference. However, this could be considered inaccurate as interference is generally always going to be present, however, the idea it to combine it with the BUS metric, which on it own consider interference and would when combined be more representative of scenarios with interference.

10 Conclusion

In this section, a conclusion will be formed around the problem statement in section 3 and the questions presented there.

1. *In previous work [10] a larger than expected amount of packet loss was observed in simulation, what is the reason for this, and what can be done to alleviate this?*

Section 5.2 concludes that the effect of interference on the system-wide PLR is large, as it is observed, that with and without interference, the loss goes from 97.5% to $\approx 10\%$ in the simulated scenarios. This lead to finding a new radio propagation model that is more representative for the outdoor use case, namely the TRGR. This is done in the hope of lowering the range of the interference, however, this model only improves the loss from 97.5% to 92%.

This then lead to other considerations such as considering the loss on an individual node basis instead, this showed an uneven distribution of loss across the nodes, with the most central of nodes being the most heavily affected.

With the new radio propagation model, a set of simulations are run, where the size and traffic load parameters are varied. These scenarios all show differing amounts of loss depending on these parameters. Furthermore, these scenarios' network utilisation all react differently to varying interference, even if they all follow the same packet loss tendencies, this is speculated to be caused by the differing impact of early loss and late loss on an end-to-end route.

With this knowledge, the focus is instead turned towards how loss can be handled within the B.A.T.M.A.N. protocol itself.

2. *How can the routing metric be made independent of the lower layers, while considering differing link conditions in the routing metric so that it is more representative of the usable transfer rate of individual links?*

In section 6.1 three methods for estimating the bitrate from the B.A.T.M.A.N. protocol layer is presented, the first two estimate an effective bitrate that is dependent on the overhead and payload size. While the third method TAPT is chosen as the best method because of its ability to estimate the overhead and physical bitrate. This method is observed in simulation to be capable of matching B.A.T.M.A.N.'s *throughput override* method in routing and the system-wide PLR for a network of homogeneous nodes. But also outperforming B.A.T.M.A.N.'s *throughput override* method in scenarios with bitrate bottlenecks.

It is also concluded that B.A.T.M.A.N. effectively uses the link metric in the opposite direction of how it is measured. To make up for this, the empty space in the ELP-probes are used to share link data between neighbours, in a concept aptly named ELSV.

To consider the uneven spread of loss across the network, the link metric is changed to a more representative metric of the possible link throughput instead. This is achieved by incorporating estimations of individual links success rate and the usage of each node's local communication channel, documented respectively in section 6.2.3 and 6.2.2. For both the usage and success rate, it is also shown that they individually change the routing for the better by routing around areas with high usage and a high packet loss.

The combined throughput metric, of bitrate, usage and success rate, is evaluated in section 6.2.4, here it is shown that the throughput metric changes routing decisions away from the middle with high loss. Consequently, lowering the system-wide PLR in all simulated scenarios, with the scenario using the longer routed cross traffic pattern experiencing the largest improvement going from 48.73% loss to 41.05% loss.

3. *How does B.A.T.M.A.N. handle multi-hop routing decisions based on individual link metrics, and can a more representative end-to-end route metric help?*

    - *If a different routing metric is suggested, that consider link differences, how does multi-hop routing handle these heterogeneous links?*

Section 7.1 compares the hop penalties B.A.T.M.A.N. uses versus the actual possible routable capacity, here it is found that B.A.T.M.A.N.'s hop penalties are too punishing and not representative at all, resulting in often choosing the shortest route, even if it is not the route with the highest capacity. Through observed capacities of varying route lengths, two sets of hop penalties are measured and are considered as the most representative for multi-hop capacities, with and without interference. These are what the multi-hop routing decisions ideally should be based on, or alternatively closer to, as opposed to B.A.T.M.A.N.'s penalties for choosing routes more representative of the available capacity.

Furthermore, the ideal hop penalties are measured in a network of homogeneous nodes, but the BUS routing metric estimations differ depending on the local link conditions, resulting in a network of heterogeneous link metrics. For multi-hop routing decisions, it is unclear how to apply the previously found hop penalties. Therefore, a new method based on the same principles is designed, called SWAP, for handling this in a way that is still representative of the actual capacity.

Then for two heterogeneous scenarios the actual capacity is measured and compared against the estimates of SWAP and B.A.T.M.A.N.'s hop penalties. Here it observed that SWAP underestimates the capacity by 26.86% and B.A.T.M.A.N.'s hop penalties underestimates it by 91.31%. Even if the SWAP method is not fully representative of the capacity, it is much closer than B.A.T.M.A.N.'s hop penalties.

Evaluating the combined performance of BUS and SWAP, the system-wide PLR is further decreased in all simulated homogeneous scenarios compared to only using BUS. But the system-wide PLR of the heterogeneous scenarios compared to original B.A.T.M.A.N., is only decreased in the scenario with the longer routed cross traffic and slight increased in the shorter routed scenario with random traffic. See table 6 for the system-wide PLR in the different scenarios using different methods.

Table 6: Table of the system-wide PLR, for two different traffic patterns and homogeneous and heterogeneous nodes.

	Homogeneous Random traffic	Heterogeneous Random traffic	Homogeneous Cross traffic	Heterogeneous Cross traffic
Original B.A.T.M.A.N.	37.41%	30.08%	48.73%	48.52%
BUS metric	36.14%	Not evaluated	41.05%	Not evaluated
BUS and SWAP	35.66%	31.63%	39.15%	42.69%

What can be done to adapt B.A.T.M.A.N. to the use case given by Systematic, such that B.A.T.M.A.N. becomes more independent of the lower layers and chosen hardware?

It is shown that the TAPT concept is capable of estimating metrics independently of the lower protocol layers, and when used in the BUS metric it is possible to avoid the links with high packet loss that Systematic wants to avoid, further helped by using SWAP for allowing routing along longer routes. All of this is simulated using a representative path loss model of Systematic's outdoor use case.

11 Further work

While this project propose changes that might improve the system-wide PLR, it is observed that the overhead generated by control packets at 100 nodes does not show promising scalability properties for Systematic's use case. Therefore, to further reduce the system-wide PLR and make larger feasible, it might be a good idea to consider reducing this overhead somehow. One way of doing this could be the interval of how often the OGMv2 packets are sent depending on how many changes each node observes of the network, as a static and stable network does not need constant maintenance. Other parameters can also be dynamically changed, for example, if the variance of the link metric is low it might be a possibility to lower the interval the ELP-probes are sent with. Changing some parameters dynamically may help with the scalability of the network.

In the use case presented by Systematic, troop movement is also part of the scenarios. Therefore, it would also be relevant to look at how the networks will perform in more realistic scenarios, that would mimic the movement of troops in the terrain. Here, it could also be relevant to look at different terrain types with different obstacles influencing the performance.

During this project some unexpected observations are made that maybe can be attributed to the impact of packet loss and where in a route it occurs. This phenomenon would be interesting to investigate further and perhaps model. This could help determine the impact of longer routes on other nodes, as a higher number of hops create more overall interference and that comes at the cost of other transmission. One idea is to change the weighting of individual metrics depending on how far it has travelled. Changing the weight of metrics such as usage and packet loss, so that a routing considers what is best for the whole network and not just itself. For example, it is much better to be risky with the route choice early and conservative late, as failing fast is less costly in terms of wasted transmissions.

Many of the improvements made in this project, even though this project is focused on the Systematic use case, is also relevant for the original implementation of B.A.T.M.A.N.. Therefore, further work could be to implement some of the features in the Linux kernel implementation of batman-adv.

References

[1] D. G. Sørensen, J. N. Andersen, and R. Krøyer. *Omnetpp-batman*. Last visited: jun 2, 2021. URL: https://gitlab.com/ct101/batman.

[2] Ericsson. *Cellular networks for Massive IoT*. Last visited: may 31, 2021. 2016. URL: https://www.ericsson.com/en/reports-and-papers/white-papers/cellular-networks-for-massive-iot--enabling-low-power-wide-area-applications.

[3] Wi-Fi Alliance. *Value of Wi-Fi*. Last visited: may 31, 2021. URL: https://www.wi-fi.org/discover-wi-fi/value-of-wi-fi.

[4] Ntefeng Ruth Moloisane, Reza Malekian, and Dijana Capeska Bogatinoska. "Wireless machine-to-machine communication for intelligent transportation systems: Internet of vehicles and vehicle to grid". In: *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 2017, pp. 411–415. DOI: 10.23919/MIPRO.2017.7973459.

[5] *Starlinks website*. Last visited: may 31, 2021. URL: https://www.starlink.com/.

[6] Wikipedia. *Thread (network protocol)*. Last visited: may 31, 2021. URL: https://en.wikipedia.org/wiki/Thread_(network_protocol).

[7] C. E. Fossa and T. G. Macdonald. "Internetworking tactical MANETs". In: *2010 - MILCOM 2010 MILITARY COMMUNICATIONS CONFERENCE*. 2010, pp. 611–616. DOI: 10.1109/MILCOM.2010.5680456.

[8] C Toh, Elizabeth Lee, and Naomi Ramos. "Next-Generation Tactical Ad Hoc Mobile Wireless Networks". In: (May 2021).

[9] Systematic. *SitaWare Suite*. Last visited: may 31, 2021. URL: https://systematic.com/en-gb/industries/defence/domains/land/sitaware-suite/.

[10] Daniel Greth Sørensen, Rolf Krøyer, and Jeppe Nellemann Andersen. "Full stack simulation of B.A.T.M.A.N. V". In: (2020).

[11] M. Ahmad et al. "State-of-the-Art Clustering Schemes in Mobile Ad Hoc Networks: Objectives, Challenges, and Future Directions". In: *IEEE Access* 7 (2019), pp. 17067–17081. DOI: 10.1109/ACCESS.2018.2885120.

[12] A. Ali and U. V. Kulkarni. "A State-of-The-Art of Routing Protocols for Mobile Ad-Hoc Networks (MANET)". In: *International Journal of Computer Applications* 127 (2015), pp. 40–45.

[13] Wikipedia. *Wireless ad hoc network*. Last visited: may 31, 2021. URL: https://en.wikipedia.org/wiki/Wireless_ad_hoc_network#Routing.

[14] freifunk.net. *Community finden*. Last visited: may 11, 2021. URL: https://freifunk.net/wie-mache-ich-mit/community-finden/.

[15] B.A.T.M.A.N. developers. *Welcome to open-mesh.org's documentation!* Last visited: jun 03, 2021. URL: https://www.open-mesh.org/doc/index.html.

[16] B.A.T.M.A.N. developers. *Originator Message version 2 (OGMv2)*. Last visited: jun 03, 2021. URL: https://www.open-mesh.org/doc/batman-adv/OGMv2.html.

[17] B.A.T.M.A.N. developers. *Echo Location Protocol (ELP)*. Last visited: jun 03, 2021. URL: https://www.open-mesh.org/doc/batman-adv/ELP.html.

[18] Freifunk Hamburg. *Zugangspunkte*. Last visited: feb 19, 2021. URL: https://hamburg.freifunk.net/wo-wird-gefunkt.

[19] OpenSim Ltd. *OMNeT++ Homepage*. Last visited: may 31, 2021. URL: https://omnetpp.org/intro/.

[20] INET developers. *INET Framework Homepage*. Last visited: may 31, 2021. URL: https://inet.omnetpp.org/.

[21] B. Sliwa, S. Falten, and C. Wietfeld. "Performance Evaluation and Optimization of B.A.T.M.A.N. V Routing for Aerial and Ground-Based Mobile Ad-Hoc Networks". In: *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*. 2019, pp. 1–7. DOI: 10.1109/VTCSpring.2019.8746361.

[22] Wikipedia user Catverine. *Power vs distance plot*. Last visited: mar 1, 2021. 2021. URL: https://en.wikipedia.org/wiki/Two-ray_ground-reflection_model#/media/File:2-ray_ground_reflected_Power_vs_distance.jpg.

[23] Lars M Mikkelsen, Nikolaj B Højholt, and Tatiana K Madsen. "Performance evaluation of methods for estimating achievable throughput on cellular connections". In: *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*. Springer, 2015, pp. 422–435.

[24] B.A.T.M.A.N. developers. *Tweaking B.A.T.M.A.N. Advanced - hop penalty*. Last visited: may 10, 2021. URL: https://www.open-mesh.org/doc/batman-adv/Tweaking.html?highlight=hop%20penalty.

[25] Jinyang Li et al. "Capacity of ad hoc wireless networks". In: *Proceedings of the 7th annual international conference on Mobile computing and networking*. 2001, pp. 61–69.

[26] "IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications". In: *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)* (2012), pp. 1–2793. DOI: 10.1109/IEEESTD.2012.6178212.

Appendices

A Appendix: Interference

A.1 Free-space path loss

The INET v3.6.7 radio propagation model `FreeSpacePathLoss` is an implementation of the FSPL. This model is derived from Friis transmission formula, which describes the ratio of power between an antenna receiving radio waves to a transmitting antenna. Friis transmission formula is shown in equation 14.

$$\frac{P_r}{P_t} = D_t D_r \left(\frac{\lambda}{4\pi d}\right)^2 \tag{14}$$

Where:

P_r is the power received

P_t is the power transmitted

D_t is transmitter antennas directivity

D_r is receiver antennas directivity

λ is the wavelength of the signal

d is the distance between antennas

From this the FSPL model is derived, here the directivity is assumed to be 1 for both antennas, this corresponds to isometric antennas, which is what is used throughout this project. Secondly, it flips Friis transmission formula on its head so that it now describes the ratio of power at the transmitting antenna compared to the receiving antenna. The FSPL equation is the one shown in equation 1 in section 5.2.2.

A.2 Adjusting interference range

The simulation run for the results in figure 5.2 from section 5.2.2 is created to see the effects of removing interference. This is achieved by adjusting the radio module's interference range, to be the same as the communication range and a second setup where it is very small, and no other nodes are within interference range. This is done by changing the `minInterferencePower` from the default value of $-110dBm$, to $-85dBm$ and $-10dBm$ respectively. These ranges are illustrated to the left in the figure A.1 in appendix A.3, showing the default interference range as a red circle, and the communication range is a green circle.

The grid scenario with interference and the ones with less and no interference are all run for one hour of simulation time with the last 50 minutes of the simulation being logged. In each scenario the grid size n is varied, with n being the width or the height of the grid, with $n = 2..10$ resulting in 9 different grid sizes of n^2 nodes being tested. Additionally, for each scenario there are 10 realisations with different seeds to get a higher confidence in the results. The results of this simulation is shown in figure 5.2 in section 5.2.2.

A.3 Vsiual compariosn of FSPL vs TRGR

A visual comparison of the interference and communication range between the FSPL model and the TRGR model.

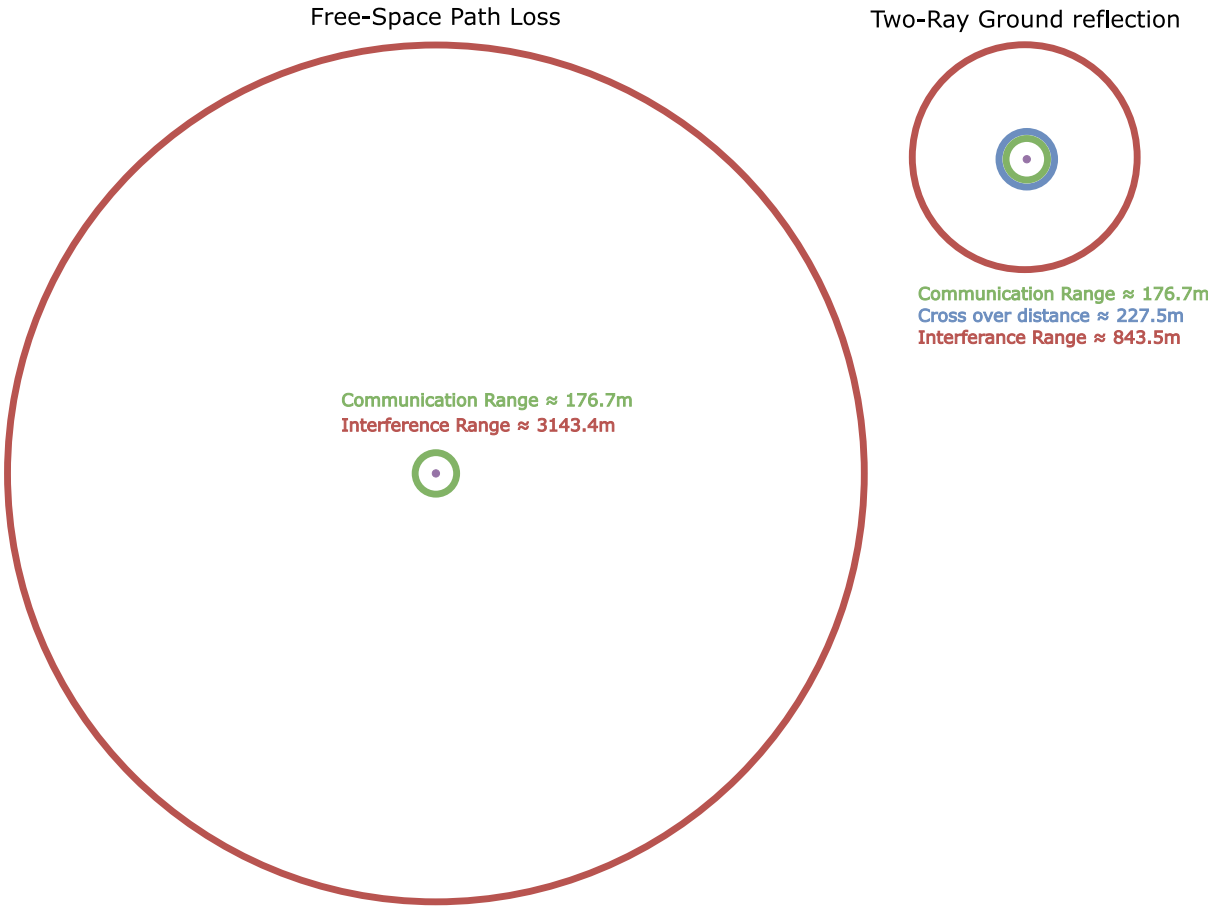


Figure A.1: Showing the size difference of the interference range when using the FSPL model versus the TRGR model.

B Appendix: Theoretical throughput estimation

For this project OMNeT++ is configured to simulate 802.11g (Wi-Fi), or more specifically ERP-OFDM only mode (Extended Rate PHY - Orthogonal Frequency Division Multiplexing). Table 7 specifies some of the important parameters used in this mode together with a short description. The information and specific values used in this section is gathered from [26].

Table 7: 802.11g ERP-OFDM specific parameters.

Parameter	Value	Description
T_{SLOT}	9 µs	Time slot duration
T_{SIFS}	10 µs	Short IFS, time between response frames
T_{DIFS}	28 µs	DCF IFS, time medium must be idle before transmission of a frame is permitted. Equal to $T_{SIFS} + (2T_{SLOT})$
T_{SYM}	4 µs	Transmission duration of OFDM symbol
$T_{PREAMBLE}$	16 µs	Transmission duration of IEEE 802.11g ERP-OFDM PPDU PLCP Preamble, see figure B.1
T_{SIGNAL}	4 µs	Transmission duration of IEEE 802.11g ERP-OFDM PPDU Signal part, see figure B.1
$L_{SERVICE}$	16 bits	See figure B.1
L_{TAIL}	6 bits	See figure B.1
T_{EX}	6 µs	Signal extension at the end of a transmission
L_{RTS}	20·8 bits	Length of a RTS
L_{CTS}	14·8 bits	Length of a CTS
L_{ACK}	14·8 bits	Length of an ACK
CW_{min}	0	Min contention window expressed in time slots
CW_{max}	15	Max contention window expressed in time slots, assuming no prior failed transmissions.

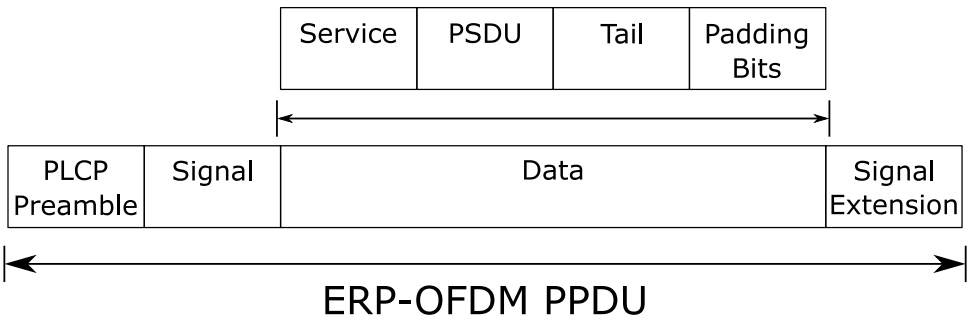


Figure B.1: Simplified illustration of the ERP-OFDM PPDU (Protocol data unit)frame format. The subcomponents of some of the fields have been omitted, since they are irrelevant for this project.

Since the method used for transmitting data is OFDM, the maximum achievable data

rate at the physical layer, can be calculated using equation 15, 16 and 17, together with the data present in column 2-4 in table 8.

$$N_{CBPOS} = N_{BPSC} \cdot N_{SD} \tag{15}$$

$$N_{DBPOS} = N_{CBPOS} \cdot R \tag{16}$$

$$PHY_{DR} = \frac{N_{DBPOS}}{T_{SYM}} \tag{17}$$

Table 8: Achievable data rates of 802.11g ERP-OFDM, from the perspective of the physical layer in respect to the OSI model.

Modulation	**Coded bits per sub-carrier** N_{BPSC}	**Data sub-carriers** N_{SD}	**Coding Rate** R	**Coded bits per OFDM symbol** N_{CBPOS}	**Data bits per OFDM symbol** N_{DBPOS}	**PHY layer data rate (b/µs)**
BPSK	1	48	1/2	48	24	6
BPSK	1	48	3/4	48	36	9
QPSK	2	48	1/2	96	48	12
QPSK	2	48	3/4	96	72	18
16-QAM	4	48	1/2	192	96	24
16-QAM	4	48	3/4	192	144	36
64-QAM	6	48	2/3	288	192	48
64-QAM	6	48	3/4	288	216	54

However, the maximum achievable data rate at the physical layer is not equal to the data rate when viewed from the perspective of the higher layers. The reason for this is because every layer adds extra overhead, in the form of headers, trailers, control packets etc., which consumes some of the available capacity.

To calculate the available data rate for a higher layer, the time required to send different types of packets must first be known. This can be calculated using equation 18 and the parameters present in table 7 and table 8.

$$T_{DATA} = T_{PREAMBLE} + T_{SIGNAL} + T_{SYM}\left(\frac{L_{SERVICE}+L_{DATA}+L_{TAIL}+L_{PADDING}}{N_{DBPOS}}\right) + T_{EX} \tag{18}$$

Here L_{DATA} is the amount of data bits that needs to be sent, where data refers to the PSDU field in figure B.1, while $L_{PADDING}$ is the amount of padding bits needed to fill the excess space in the last OFDM symbol. Table 9 illustrates the time it takes to transmit different types of packet, at different physical data rates. For the $T_{Data500B}$ column it

is actually $500B$ data plus $92B$ headers, while the $T_{Datamax}$ column represents $1408B$ data plus $92B$ headers, for a total packet size of $1500B$, which is the max size of a non-fragmented Ethernet frame in OMNeT++. The rest of the B.A.T.M.A.N. specific packets uses their default size plus the relevant headers.

Table 9: Transmission duration for control packets and the T_{DATA} part, calculated using equation 18, and the data from table 7 and table 8.

PHY data rate (b/µs)	T_{RTS} (20B) (µs)	T_{CTS} (14B) (µs)	T_{ACK} (14B) (µs)	T_{ELP} (16B) (µs)	T_{OGMv2} (48B) (µs)	$T_{ELPProbe}$ (200B) (µs)	$T_{ELPProbe}$ (16B) (µs)	$T_{ELPProbe}$ (205B) (µs)	T_{Data} (500B) (µs)	T_{Data} (1408B) (µs)
6	58	50	50	102	142	346	102	354	822	2030
9	50	42	42	78	106	242	78	246	558	1362
12	42	38	38	66	86	186	66	190	426	1030
18	38	34	34	54	66	134	54	138	294	694
24	34	34	34	46	58	106	46	110	226	530
36	34	30	30	42	46	82	42	82	162	362
48	30	30	30	38	42	66	38	70	126	278
54	30	30	30	38	42	62	38	66	118	250

The average total time for transmitting either an OGMv2, broadcast ELP, ELP-probe or data packet is then calculated using respectively equation 19, 20, 21 or 22.

$$OGMv2 = BO_{mean} + DIFS + T_{OGMv2} + T_{prop} \tag{19}$$

$$ELP = BO_{mean} + DIFS + T_{ELP} + T_{prop} \tag{20}$$

$$ELPProbe = BO_{mean} + DIFS + 3 \cdot SIFS + T_{RTS} + T_{CTS} + T_{ELPProbe} + T_{ACK} + 4 \cdot T_{prop} \tag{21}$$

$$Data = BO_{mean} + DIFS + 3 \cdot SIFS + T_{RTS} + T_{CTS} + T_{Data} + T_{ACK} + 4 \cdot T_{prop} \tag{22}$$

Common for all types of packets are that they need to factor in the back-off period (BO), a DIFS and the signal propagation time ($T_{prop} = \frac{DistanceBetweenNodes}{SpeedOfLight}$). Whereas it is only the unicast packets such as ELP-probes and data, which needs to factor in a RTS/CTS and an ACK plus a SIFS between each part of the four-way handshake. The time from the equations are shown in table 7 and 9.

Table 10: Full duration of different types of packets' transmission, calculated using equation 19, 20, 21 or 22, and the data from table 7 and table 9.

PHY data rate (b/µs)	$OGMv2$ (48B) (µs)	ELP (16B) (µs)	$ELPProbe$ (200B) (µs)	$ELPProbe$ (16B) (µs)	$ELPProbe$ (205B) (µs)	$Data$ (500B) (µs)	$Data$ (1408B) (µs)
6	238	198	631.5	387.5	639.5	1107.5	2315.5
9	202	174	503.5	339.5	507.5	819.5	1623.5
12	182	162	431.5	311.5	435.5	671.5	1275.5
18	162	150	367.5	287.5	371.5	527.5	927.5
24	154	142	335.5	275.5	339.5	455.5	759.5
36	142	138	303.5	263.5	303.5	383.5	583.5
48	138	134	283.5	255.5	287.5	343.5	495.5
54	138	134	279.5	255.5	283.5	335.5	467.5

Table 10 shows the theoretical average full transmission time of the different types of packets, that is present in the simulation environment, for different physical bitrates.

C Appendix: Bitrate estimation

C.1 Theoretical calculation of double packet train estimation

For the theoretical approach to calculating the estimation of the double packet train, equation 21 from appendix B is used. Here it is assumed that two ELP-probes are sent back-to-back and that the receiving node timestamps both ELP-probes once they are fully received. The receiving node, then knows the time it took to transmit a single ELP-probe and the size of the payload, and can therefore estimate the bitrate used. Figure C.1 shows this estimation for varying ELP-probe payload sizes and at different underlying physical bitrates.

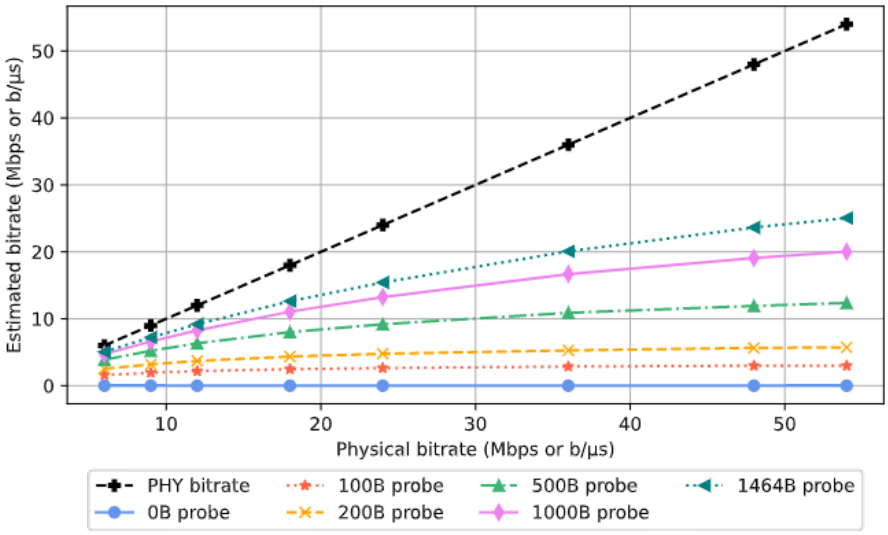


Figure C.1: Theoretical packet train estimation in a 2 node setup with varying physical bitrate and probe sizes.

This figure is created to compare with figure 6.4 in section 6.1.4.

C.2 Effect of ELP-probe size in TAPT

It is observed in the simulation that the TAPT method for estimating the bitrate results in an estimation which is slightly higher than the actual physical bitrate. The estimation gives $\approx 61.33 Mbit/s$ whereas the underlying bitrate used is $54 Mbit/s$. This is because the physical layer used in the simulation is 802.11g using OFDM, which does not transmit every bit that a packet consists of sequentially, but rather transmits a number of bits in parallel in OFDM symbols. Different physical bitrates allows for different amount of data bits per OFDM symbol, for more info see 8 in appendix B. This result in the last OFDM symbol for a packet can be either filled up with actual data, just have a single bit of data or somewhere in between, depending on the packet size. Since the time it takes to transmit a OFDM symbol is constant, it ends up being important how full the last OFDM

symbol is, to get the most accurate estimation of the physical bitrate. Using equation 21 from appendix B, and dividing the size of the third ELP-probe with the time difference ($\Delta T1$) between the first two ELP-probes of $16B$ and subtracting this from the time it takes to send the third ELP-probe ($\Delta T2$), results in the estimated physical bitrate using the TAPT method. calculation shown in equation 23.

$$EstimatedBitrate = \frac{ThirdELPPayloadSize}{\Delta T2 - \Delta T1} \tag{23}$$

By trying out different sizes of the third ELP-probe, it is found that the size resulting in the best bitrate estimation is an ELP-probe of $205B$ instead of the $200B$ which B.A.T.M.A.N. normally uses. Using this size should yield an estimate close to $54Mbit/s$, and trying it out in the simulator results in an estimate of $\approx 53999kbit/s$.

C.3 Expanded Link State View

For the TAPT method to be usable and work, the reception time measurements are done on the receiving node, and the two probes must both be successfully received. If the assumption is that the bitrate is symmetric on a link, then by calculating the bitrate from the received packets should also be the same bitrate for that link in the other direction. But if this is not assumed the estimation made at the receiving nodes are only for one direction and must be shared back to the transmitting node in order for the transmitting node to know what the bitrate is towards the receiving node.

This information sharing between neighbouring nodes will be used for more than just the bitrate estimation and increases the local area a node possess information about. This information provides an expanded view of each link's state and is as such named ELSV. The main concept for sharing this information is to use the ELP-probes that by default gets padded with $184B$ and instead use these $184B$ for information sharing, see figure G.1 in appendix G for the ELP probe's packet frame. This makes the ELSV almost without further overhead, except for that the probes are now periodic and not conditional as previously mentioned. Alternatively, the ELSV could be piggybacked on unicast messages and ELP-probes so that the overhead becomes even less, however, as this is more of a proof-of-concept, the simplicity of implementing it only for ELP-probes is chosen. The bitrate estimation is a single uint32_t that is shared back to the node that is sending ELP-probes on that link.

The packet loss is an amount of packets received shared back to the node sending them, so that the sending node can compare that against the amount sent, and thereby estimate the packet loss. Furthermore, if the size of the window of sent and received packets can vary, this should also be shared. This can be sent as a uint32_t for the amount, and a uint32_t for the window size.

The usage requires one value for direct neighbours total usage by themselves, this is sent as a double. Then for considering RTS/CTS usage by a node two hops away causing indirect usage of a nodes channel, individual usage of a link is shared as a double so that

the direct neighbour can collect these and send a list of all the indirect usages back to a node. This list contains pairs of doubles and MAC addresses. This also makes the sharing of indirect usage caused due to RTS/CTS a two-step process, where the first step sends information to direct neighbours, whom then makes a list and share this again. An example is shown in figure C.2, where node **A**'s total usage is determined by its own usage(known by itself), its neighbours' total usage shared back as a single value, and **C**, **D** and **E**'s usage first shared to **B** whom then collects these and send all to interference as a list.

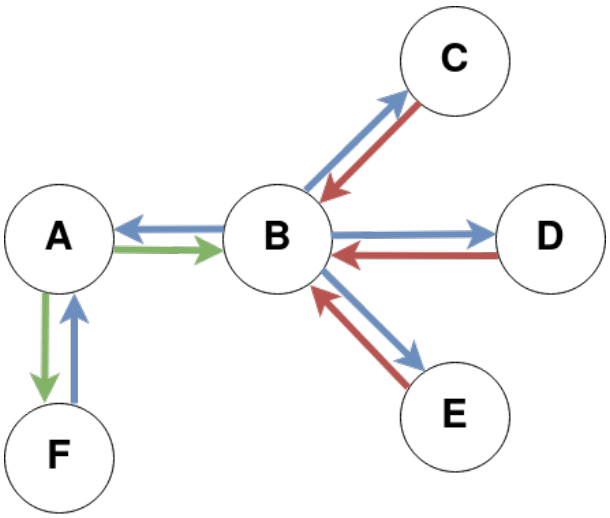


Figure C.2: Example of a network where the usage of node **A** is determined by its own outgoing links direct usage(green), neighbours outgoing links direct usage(blue) and neighbours neighbours links to direct neighbours links usage(red).

The ELP-probe is filled with the most significant pairs of MAC addresses and single link usages. The amount of pairs depends on the amount of neighbours or the remaining amount of room in the ELP-probe, with each pair being a 48 bit address and a double.

The bitrate and packet loss collectively use 12 bytes in each probe, then for usage a single double of 8 bytes is also in each probe. Then for every individual usage ratio pair uses 14 bytes in each probe. So if the payload is 189 bytes, there is room for 12 pairs.

Figure C.3 shows an overview of all the data a single ELP-probe from node **B** to **A** would contain, along with colours to show where this information would come from.

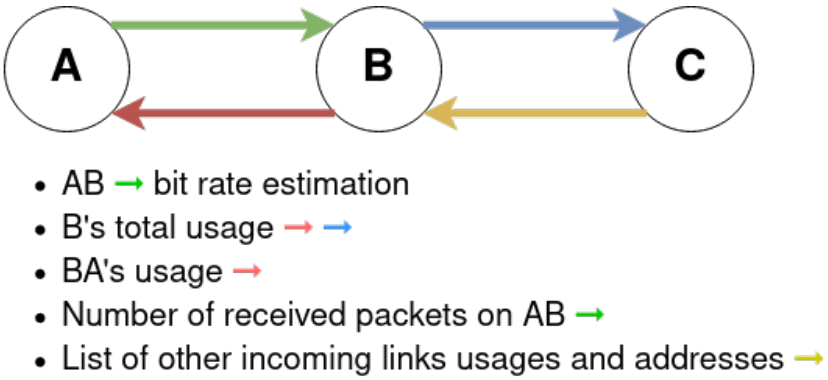


Figure C.3: Sketch of what an ELP-probe from B to A contains, and colour coded where what link that information comes from.

Generally, for all integer values an uint32_t is used and for floating point values a double is used. This could perhaps be optimised so that bits are not wasted on unneeded precision. However, for the simulator this is not so important as it is possible to send more data than the packet can contain, for example it is not considered for usage if there are too many pairs, as it would still function in the simulator and such a scenario is never constructed in this project, but they could occur in other scenarios.

A side note to the information sharing ELSV provides, is that the information will always be slightly outdated due to the propagation and processing time between nodes.

C.4 OGM self interference and propagation jitter

One lesson learned from section 6.1's implementation and simulations is that self interference from an OGM stream forking is an issue. To illustrate this, figure C.4 is made, with a more detailed description of why this is the case.

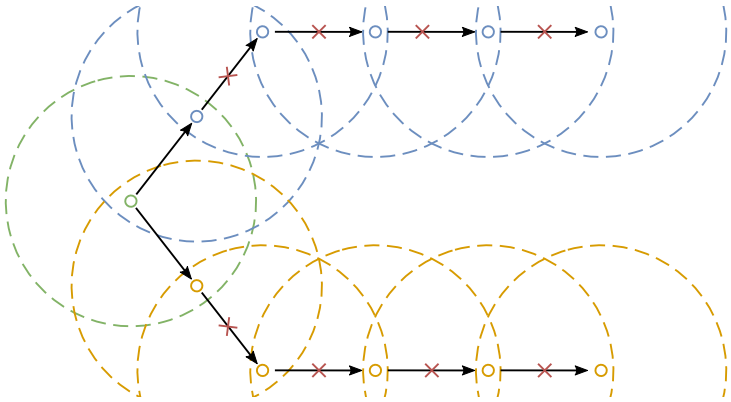


Figure C.4: The green node broadcasts its OGM in the direction of the arrows. When the nodes on the top path (blue) and the bottom path (yellow) retransmits the OGM packet, there is a high risk of interference resulting in the OGM being dropped at any of the links with a red cross, due to the two paths being outside each other's communication range but within each other's interference range. Furthermore, since the packet is broadcast, it is impossible to tell whether or not it is received due to the lack of ACK.

This causes nodes to have routes that become invalid, often due to the small amount of OGM packets arriving at them. To what extent this is an issue in a real physical setup is hard to say, as it is much more unlikely that things happen at the exact same time as they do in the simulator, as scheduling and other randomness are present here.

In order to deal with this self interference, a small random jitter uniformly distributed between $0ms$ and $2ms$ is inserted before an OGM packet is forwarded. Doing this eliminated almost all cases of this happening, however, it is worth remarking that it comes at the cost of a higher propagation time and potentially a larger delay in a node's discovery period, as each hop would on average takes $1ms$ longer.

The cost of this jitter is for this project even further considered to be worth it as it is already considered to have a jitter that allows a node to see more potential routes. As only the first time an OGM is observed, it will be forwarded, where previously this would be OGM's arriving via the shortest paths. The jitter's random nature allows other potentially longer routes to be observed by a node, which is considered an advantage, since the routes with the highest throughput metric is not necessarily the shortest. Compared to B.A.T.M.A.N.'s original assumptions that the shortest path is the best according to their bitrate routing metric.

C.5 Baseline scenarios

To test out different methods of estimating the bitrate described in section 6.1 three simulation scenarios are defined. Common for all scenarios are the physical placement of the nodes, which are arranged in a ten by ten grid, with a separation of $150m$ between neighbouring nodes in each of the cardinal directions. Due to the configuration of the nodes' radios, which are shown in table 11, and the chosen path loss model, each node can communicate with nodes within a radius of $\approx 176m$, while they can interfere with

nodes up to $\approx 843m$ away. This setup is illustrated in figure C.5, where each node is depicted as a green or red dot while the communication range for the upper left corner node is shown as a blue circle and the interference range is shown as a dashed blue circle.

Table 11: Simulation parameters

Parameter	**Value**
Transmission power	1 mW
Operation mode	802.11 g(erp)
PHY data rate	54 Mbit/s
Wi-Fi channel	1
Centre frequency	2.412GHz
Bandwidth	20 MHz
RTS/CTS threshold	0 B
Background noise	-110 dBm
min reception power	-85 dBm
min interference power	-110 dBm
max Queue Size	64

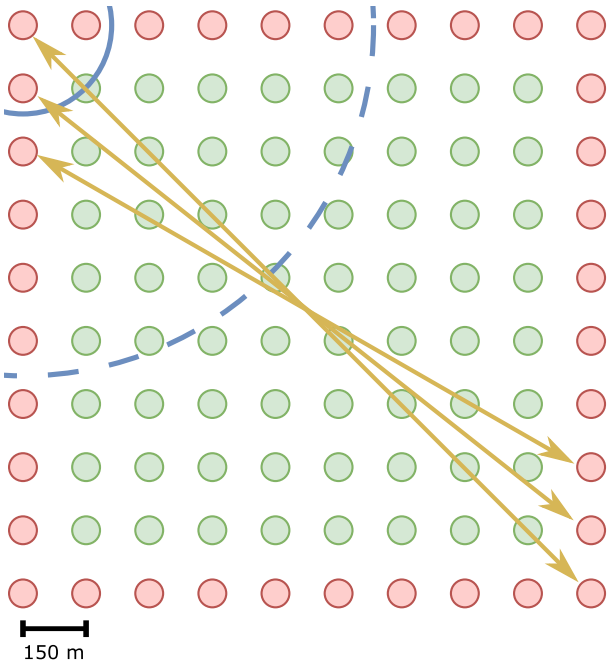


Figure C.5: Illustration of node arrangement in grid. Green and red nodes generate random destinations for periodic traffic in scenario 1 and 3. In scenario 2 and 3 the red nodes generate Pareto distributed traffic towards the opposite red edge notes.

Scenario 1 Random Traffic: In this scenario, every $2000ms$, each of the 100 nodes picks another node at random and schedules a UDP data packet towards this aforementioned node. Every UDP data packet consists of $500B$ of data and is transmitted towards the next best hop, according to the B.A.T.M.A.N. protocol, when the medium is idle.

In this scenario, every node is therefore responsible for transmitting their own generated packets, and the packets that it needs to forward for their neighbours.

Since the nodes pick another node at random for the destination, the average length of a route is $\frac{2}{3}10 = 6.67$ hops, as mentioned in section 5.2.5.

Scenario 2 Cross Traffic: In this scenario the destination is constant and corresponds to the edge node opposite of the transmitter. Figure C.5 shows a couple of pairs between transmitting edge nodes and their destinations as yellow arrows. Here the edge nodes, which generate traffic, are the nodes shown as red circles.

The traffic generated by the edge nodes is UDP data packets of $1408B$, however, these are generated according to a Shifted Pareto distribution with a shape value of 15 and scale value of $5000ms$, this gives a mean generation time of $5353ms$.

In this scenario, the average length of the routes, if the packets travels by the shortest path, is 13.56 hops.

Scenario 3 Combination Traffic: In this scenario, the underlying traffic pattern is the same as in scenario 1, however, here the traffic from scenario 2 is added to the network as well.

D Appendix: Usage and packet success rate estimation

Figure D.1 shows all the nodes and their current estimation of both the usage ratio and the PSR, 3600s into a simulation. Inside each node is the usage ratio estimation that it currently uses at the time, the more red a node is, the higher the usage. While the current PSR towards its neighbours is shown with the arrows, where the dark blue arrows have a PSR of ≈ 1, and teal arrows are around ≈ 0.75.

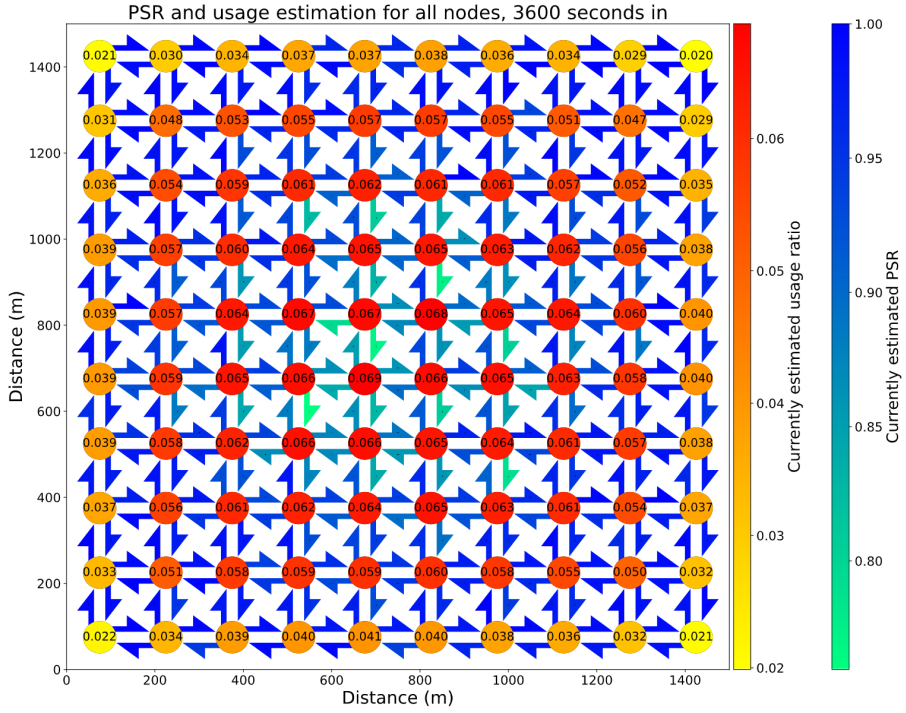


Figure D.1: Insight into how the nodes perceive the usage and PSR towards their neighbours, at 3600s. In this simulation, all nodes use an underlying physical bitrate of 54$Mbit/s$.

It is worth noting that the PSR on some link pairs are different depending on the direction.

E Appendix: Random Traffic figures

These figures are for the evaluation of individual nodes' performance in baseline scenario 1. The first two figures, figure E.1 and figure E.2, are referenced in section 6.2.4 during the evaluation of the BUS metric. The last figure E.3 is referenced in section 8 during the evaluation of both the BUS metric and the SWAP method.

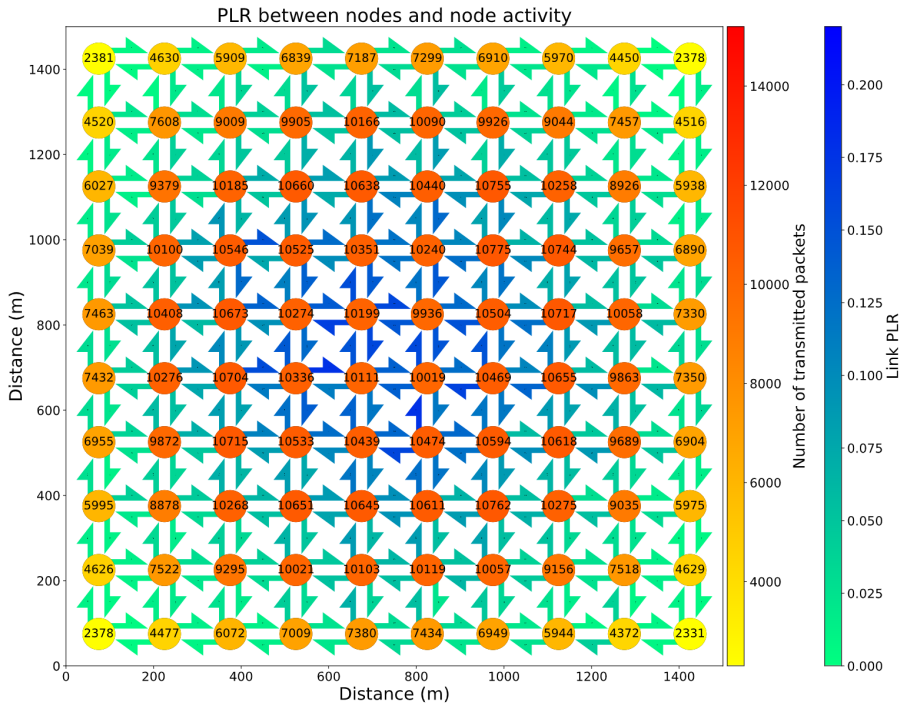


Figure E.1: Basline scenario 1 with original B.A.T.M.A.N.'s *throughput override*, where all nodes use a physical bitrate of $54Mbit/s$.

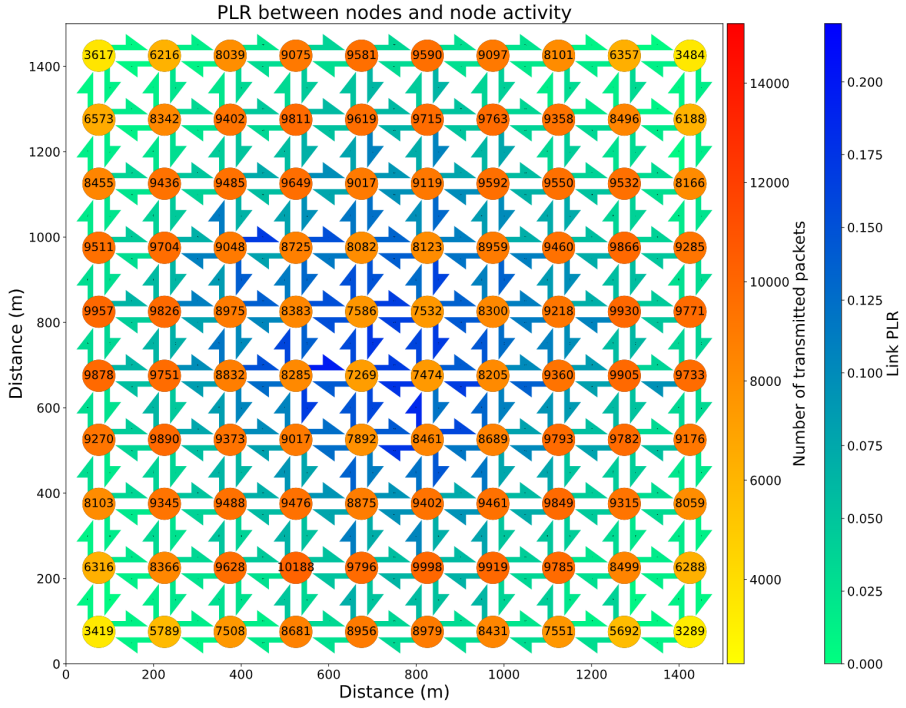


Figure E.2: Baseline scenario 1 with the BUS metric, where all nodes use a physical bitrate of $54Mbit/s$.

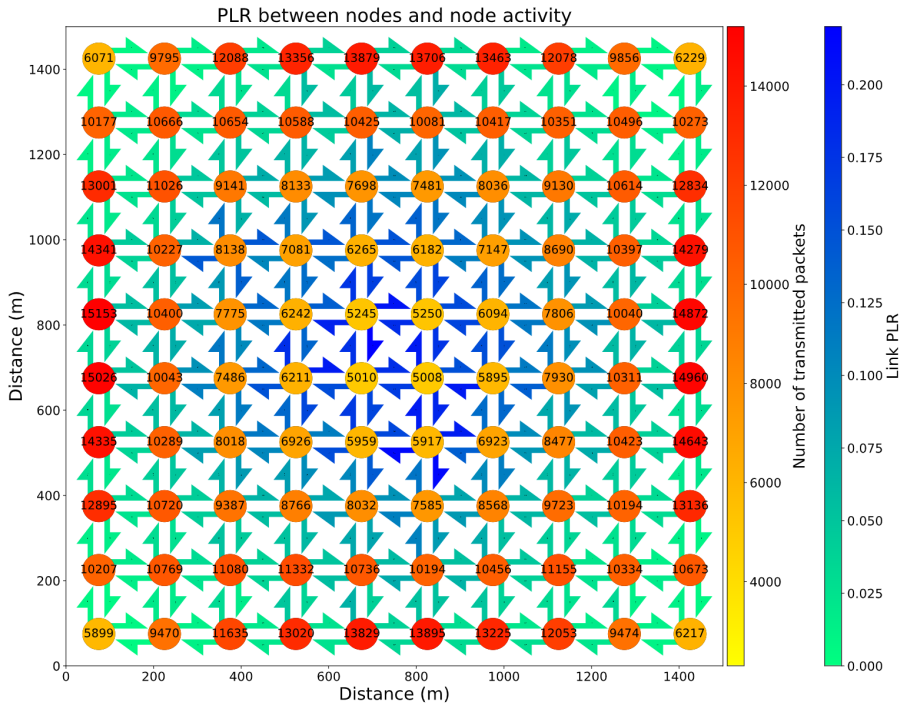


Figure E.3: Baseline scenario 1 with the BUS metric and SWAP, where all nodes use a physical bitrate of $54Mbit/s$.

F Appendix: Results

The following figure F.1 and figure F.2 are referenced in section 8 during the evaluation of individual nodes' performance in baseline scenario 1, for one specific realisation of a network of heterogeneous nodes.

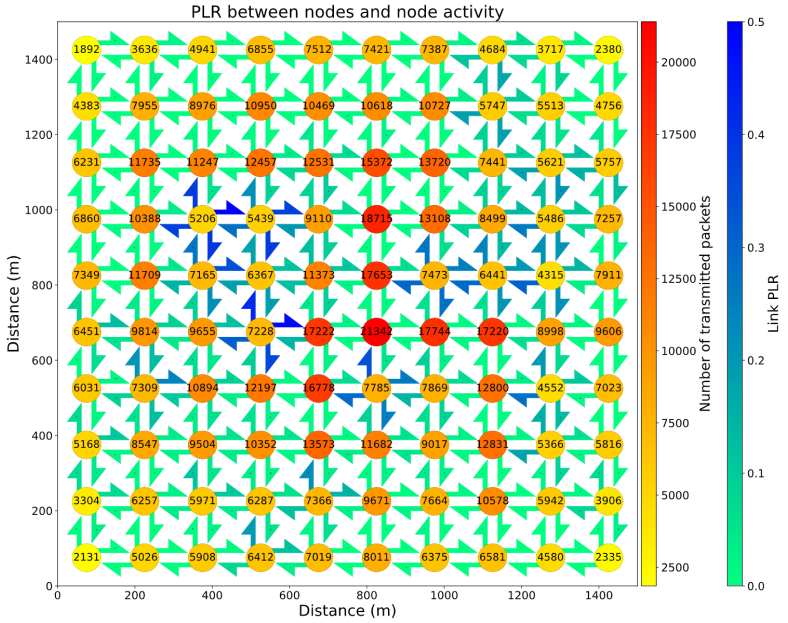


Figure F.1: Basline scenario 1 with original B.A.T.M.A.N.'s *throughput override*, where the nodes use a physical bitrate as in figure 8.4 in section 8.3.2.

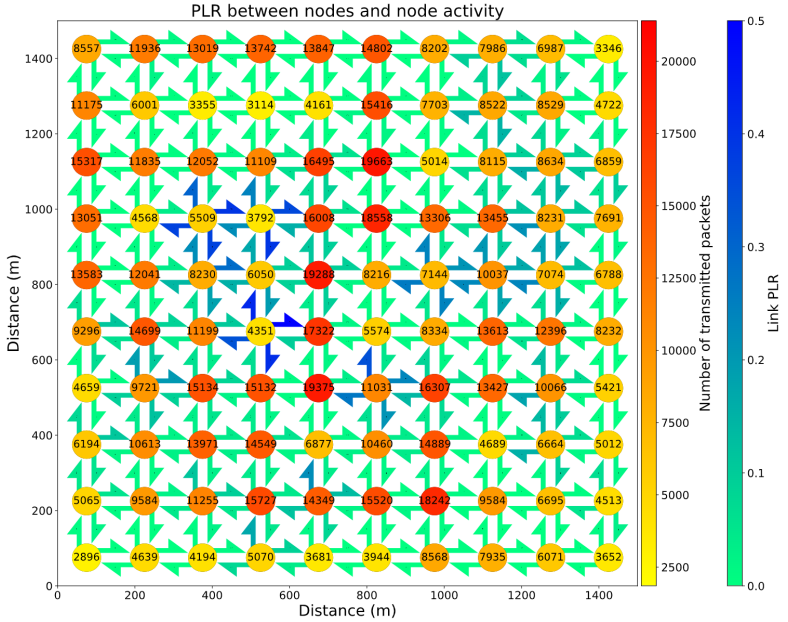


Figure F.2: Baseline scenario 1 with the BUS metric and SWAP, where the nodes use a physical bitrate as in figure 8.4 in section 8.3.2.

G Appendix: Encapcualtion and B.A.T.M.A.N. packet frames

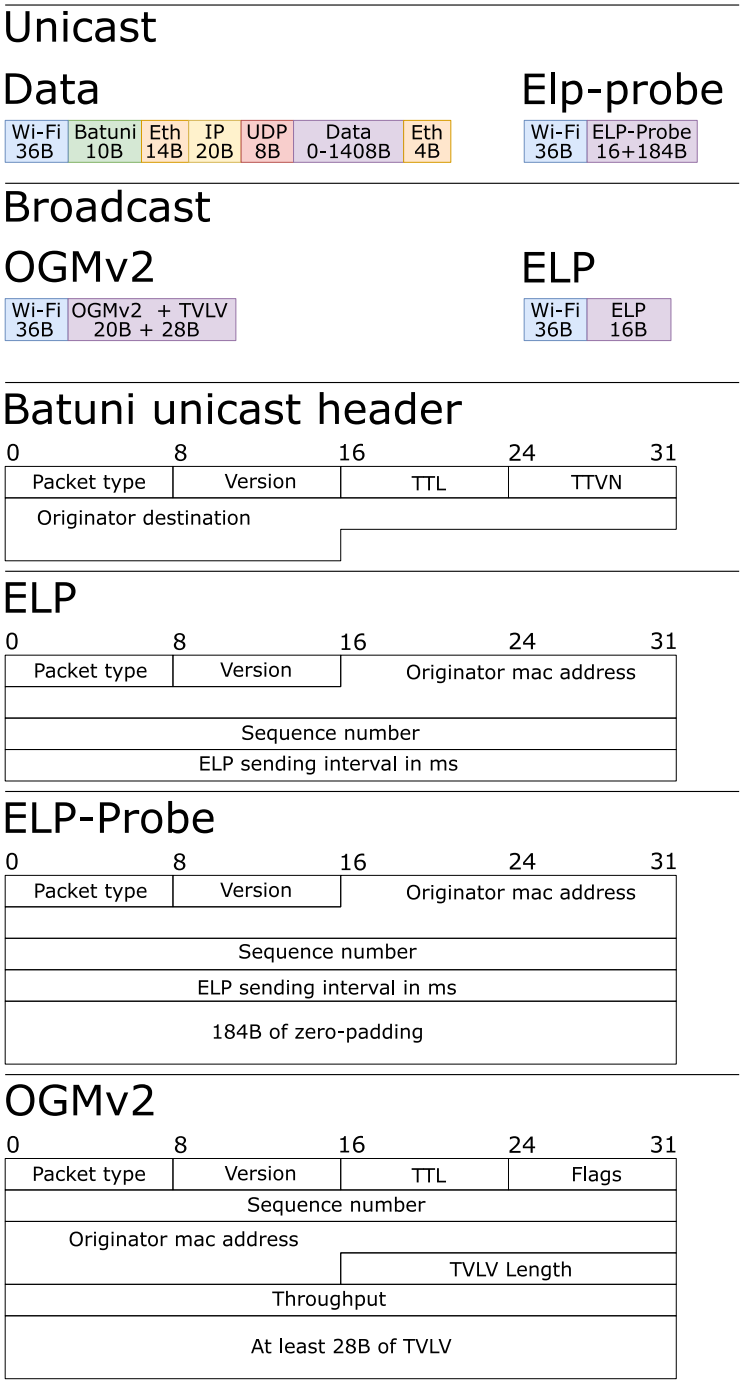


Figure G.1: The top of the figure shows how different types of packets are encapsulated, while the bottom shows the original B.A.T.M.A.N.'s packet formats.