Enhancing Satellite Communication Performance with MultiWAN using MPTCP: Implementation and Analysis

Improving Bandwidth Efficiency and Reliability in Satellite Communication

Master's thesis in Computer science and engineering

Algot Axelzon
Lucas Norman

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

Enhancing Satellite Communication Performance with MultiWAN using MPTCP: Implementation and Analysis

Improving Bandwidth Efficiency and Reliability in Satellite Communication

Algot Axelzon

Lucas Norman

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Enhancing Satellite Communication Performance with MultiWAN using MPTCP: Implementation and Analysis

Improving Bandwidth Efficiency and Reliability in Satellite Communication

Algot Axelzon
Lucas Norman

Supervisor: Ahmed Ali-Eldin Hassan, Department of Computer Science and Engineering
Advisor: Martin Löfgren, Satcube AB
Examiner: Romaric Duvignau, Department of Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: *"Satellite in Space"* by The Global Panorama, Courtesy: NASA Goddard Space Flight Center [1].

Typeset in LaTeX
Gothenburg, Sweden 2023

Enhancing Satellite Communication Performance with MultiWAN using MPTCP: Implementation and Analysis

Improving Bandwidth Efficiency and Reliability in Satellite Communication

Algot Axelzon
Lucas Norman

Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In recent years the availability of more than one active internet connection for devices has been increasingly more common. To take advantage of this many actors have looked into the concept of MultiWAN, which is the use of multiple links for internet connection either simultaneously or as a backup. One of the more promising options for this is the Multipath Transmission Control Protocol (MPTCP), which builds upon the traditional TCP protocol. This thesis presents an evaluation study of MPTCP in a satellite communication environment. It also gives insight into the implementation process, with its limitations and possible workarounds. A broad measurement study has been conducted, looking at a wide range of parameters to gauge the performance of MPTCP compared to TCP, combining a number of different internet connection alternatives. The results show that using the MPTCP in its standard configuration can outperform TCP in some circumstances, but also negatively affect the performance in others. From the evaluation study, we can conclude that the results are highly sensitive to the characteristics of the links, which have different degrees of fluctuation over time. The standard MPTCP implementation performs better the more similar the links are but, from our research, we believe it needs further customisation possibilities to handle combining link types of all characteristics.

Keywords: MultiWAN, MPTCP, satellite, communication, satellite communication, Computer, science, computer science, engineering, thesis.

Acknowledgements

We would like to thank our supervisor at Chalmers, Ahmed Ali-Eldin Hassan for his support and guidance. We would also like to thank our company advisor Martin Löfgren for his support together with everyone else at Satcube for their welcoming and helpful attitude.

"Coming together is a beginning. Keeping together is progress. Working together is success."

– Henry Ford

"Simul Fortis Simias"

Algot Axelzon, Gothenburg, 2023-06-26

Lucas Norman, Gothenburg, 2023-06-26

Contents

List of Figures

List of Tables

1

Introduction

As technology continues to evolve and everything becomes more and more connected to the internet, situations where devices often have more than one available internet link to choose from become increasingly common. With this, research about the possibilities of utilising multiple available links simultaneously has grown over the decade. This concept is often referred to as multiple Wide Area Networks (MultiWAN) or multipathing networks. The use of MultiWAN brings many benefits, but the most notable are link aggregation benefits such as increased throughput and higher tolerance to faults such as single-point link failures. The use of MultiWAN so far has seen great potential in data centres where large amounts of data are transferred between endpoints frequently. The same potential has been seen with mobile devices that often have more than one enabled network interface, for example, a cellular 4G connection together with an available WiFi connection. We believe that a similar potential exists in the field of satellite communication, as satellite terminals often reach multiple satellites at once, both of the same and different types.

Satcube is a company based in Gothenburg and Karlstad that focuses on satellite communication. Their main product today, the *Satcube Ku*, is a portable satellite terminal that provides internet access via satellite broadband *"anywhere, quickly and cost-effectively"* [2]. Together with Satcube, we present the use of MultiWAN, where we have utilised numerous internet connections and in turn, made use of the features which MultiWAN makes possible. These internet connections used are of different types, more specifically, low-earth orbit (LEO) or geostationary orbit (GEO) satellites, the local cellular network and a fibre connection used in a simulated environment. The different types of satellites provide different connectivity characteristics. LEO satellites move relative to the Earth and have lower latency, but worse reliability, as the connection switch between satellites as they move. GEO satellites have a locked position above the Earth but are farther away, resulting in a longer latency and slower speeds but a more reliable connection compared to LEO satellites. By using both of these satellite types together we present the possible advantages of aggregating the different benefits of each type. Combining the links makes it possible to use different links depending on availability, cost, bandwidth or latency, or to use multiple links together and aggregate them in order to utilise their combined qualities.

Applying the concept of MultiWAN in Satcube's products can in theory bring many benefits, as well as use cases. An advantage that can come from aggregating multiple

links is the increased reliability and resilience against unavailability. The risk that all links become unavailable at the same time decreases as the number of links increases, even more so if the links are of different types. Another advantage of aggregating multiple links is the increased security in sending data packets through different links and routes. Since only a fraction of packets would be sent over a single link, it would be more difficult for someone with malicious intent to eavesdrop on the communication as they would not as easily be able to observe the whole picture of what is being sent. Furthermore, it makes it possible to use the combined bandwidth of all of the available internet connections together. This opens up new possibilities for the user and improves the overall experience. These advantages: availability, security and performance, are all valuable selling points for Satcube, whose goal is to make internet access more available and secure.

1.1 Goals & Challenges

The main goal of the thesis has been to evaluate the performance of MultiWAN in the context of satellite communication. To achieve this goal, we chose to use the Multipath Transmission Control Protocol (MPTCP) to implement MultiWAN as it builds upon the widely used Transmission Control Protocol (TCP) and acts as an additional transparent layer above it in the network architecture. Based on related research, the protocol has shown promising results. Additionally, Satcube whom we collaborate also expressed an interest in it. This TCP extension is based on RFC 8684 [3], which allows the use of multiple network interfaces for each connection. An implementation of this protocol is developed by a community whose goal is to *"develop, maintain, and improve the Multipath TCP (MPTCP) protocol (v1 / RFC 8684) in the upstream Linux kernel"* [4]. Therefore, the MPTCP implementation is included in the latest versions of the Linux kernel and is actively developed in every updated version.

In order to fulfil our goal, we formulated two research questions as follows:

- How can MPTCP functionality be effectively implemented to support links with varying characteristics, and integrate with satellite terminal hardware with favourable performance?
- To what extent does the implementation of MPTCP perform, compared to standard TCP communication, in terms of factors such as throughput, latency, and reliability, across a range of link characteristics and usage scenarios?

To implement MPTCP so that it could function as intended with the provided satellite terminals, it was necessary to find workarounds for middleboxes that did not support MPTCP. A comparison between the base implementation of MPTCP on its own and MPTCP with added overlay, such as with a Virtual Private Network (VPN), gave insight into the cost of using it in circumstances where it could not work on its own. The performance of MPTCP compared to regular TCP was measured, where the throughput was one of the key interests. The aggregation benefit of MPTCP is formulated as a value in the range between -1 and 2 where:

- 2: Is the perfect summation of the aggregated links throughput.
- 1: Is the throughput of the link with the highest throughput.
- 0: Is the throughput of the link with the lowest throughput.
- -1: Is a throughput of zero.

Other aspects of the benefits of MPTCP will also be covered, such as resistance to link failures and security advantages.

MPTCP, as well as other MultiWAN features, are fairly new concepts that have not yet been widely adopted and implemented. Because of this many middleboxes and endpoints modify or reject data sent using MPTCP communication. One example of this came from previous work at Satcube, where it was found that the modems used in their satellite terminals simply remove the data packets header option that indicates that the peers want to use MPTCP before forwarding the packets, resulting in normal TCP communication. One of the goals of this thesis was to find a solution that is unaffected by interference from middleboxes. This was a notable challenge to achieve in a way that worked in different settings, without significant performance degradation.

The standard version of MPTCP has been shown to perform best when using homogeneous links, i.e. links that share the same characteristics such as bandwidth or latency. The aforementioned aggregation benefit regarding throughput can often reach a value of 1.85, or approximately 90 % of the total summation of the links. However, the benefit quickly becomes less valuable with heterogeneous links, i.e., links with different characteristics from each other. This performance degradation becomes more apparent the bigger the difference between the links is, especially when the throughput of the links differs greatly from what we found. Many alternatives to the default scheduling and congestion control have been studied in recent papers, as well as alternatives to the MPTCP protocol, in regard to mitigating these performance degradation issues. A considerable challenge is to implement a solution that is flexible and can perform well while aggregating links possessing different characteristics.

Further challenges arose from the target environments' limited hardware on which the solution was deployed. These resources are used for other functionalities as well, so the challenge lay in balancing a solution that performed well, while at the same time limiting the resource usage as much as possible.

1.2 Scope of the thesis

As there was a fixed time frame for the project, there was a need to limit the scope of the work that has been done. One of the more obvious limitations was to focus on functionality rather than the interface and user-friendliness since the interest of the project lay in the proof of concept and meticulous evaluation of the working solutions. Further, the focus was directed towards using links with different characteristics to resemble the combinations of typical GEO links, LEO links, cellular network links and so on. Especially the interest in using the combination of GEO and LEO links

and also the combination of two or more GEO links was expressed both by Satcube and by our supervisor at Chalmers. Our preliminary goal with this thesis was to design a working configuration which tries to give a balanced solution, rather than different modes designed for reliability, speed, security or low energy consumption.

If we would have had more time it would be of interest to continue the work by implementing these modes as well, for the possibility of further customisation.

Another thing that we did not have the time to tackle was the problem of not all content servers supporting MPTCP. An interesting solution to explore in the future, as illustrated in Figure 1.1, would be to use an MPTCP-supported proxy server which would make it possible to utilise MPTCP on parts of the path, even when the endpoint does not support it. While this solution would always utilise MPTCP, the proxy is somewhat redundant in the cases where the content server does support MPTCP. A possible "smarter" solution would be to check if the content server does or does not support the MPTCP protocol and then only use the proxy when necessary.

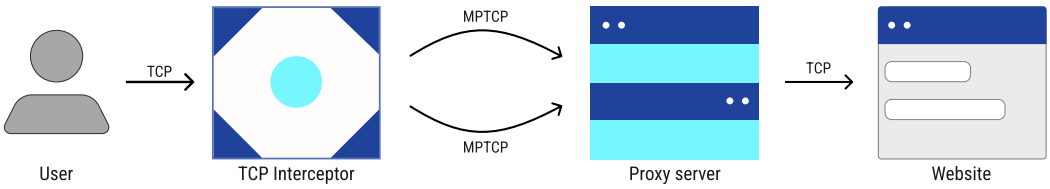


Figure 1.1: An example of how traffic is sent through a proxy server.

We also limited ourselves to only implementing MultiWAN functionality for TCP traffic instead of for more protocols such as UDP. We deemed MPTCP to be the most suitable option in this case as it builds on the TCP protocol, which a large majority of the communication over the internet uses.

Finally, another topic of interest that would have been explored if the time had allowed it is the possibility of reserving a processor core to ensure that the MultiWAN communication always has dedicated resources.

1.3 Approach

To begin with, a strategy for evaluating the different solutions was devised. Different benchmarking software such as Iperf [5], TCPdump [6] and Capinfos [7] was explored and tested to find a suitable strategy that could measure the desired metrics in an adequate way. The metrics evaluated are the following:

- Bandwidth
- Latency
- Resource usage (CPU, RAM, etc.)
- Energy and power consumption
- Packet drop rate
- Jitter

Using these metrics gave a broad picture of how well the solution would perform compared to normal TCP communication and what additional costs it would bring. The evaluation of the initial solution was used as a baseline to be compared against the further developed solution which included an encapsulation method in order to see its effect on the MPTCP communication performance.

Looking at the last two metrics: packet drop rate and jitter together provided an idea of how reliable the solution is. The security of the solutions was also considered, although it was not measured as an exact metric.

As MultiWAN has been of interest to Satcube for some time, previous work on this subject has been done by one of the authors in collaboration with Satcube through a summer internship. The work consisted of implementing MultiWAN on a development board running OpenWrt [8]. Since then, Satcube has replaced the hardware with a new module that has a CPU with 4 cores @ 1.8 GHz and 2 GB of RAM, which is planned to be used in Satcube's next generation of satellite terminals. They have also moved away from OpenWrt, in favour of their own custom Linux distribution created with the Yocto project [9] tools. After devising the evaluation strategy, we had to migrate the previous work from the old platform to the new software and hardware.

The earlier performed work was left off with the realisation that MPTCP options were stripped somewhere in the communication with the GEO satellite terminals. After further investigation, it became clear that the cause of this problem was the modems used in the *Satcube Ku* terminals and the satellite network the modems are a part of. When testing the communication using another modem and against other internet links such as LEO communication using a LEO satellite terminal this problem did not occur, which gives us confidence in this theory.

To work around this, some sort of encapsulation method had to be used where two potential solutions were to use either a GRE tunnel or a WireGuard VPN to wrap the data packets inside additional packets to protect them from header options removal. This solution was implemented only after reaching an equal state of the functionality on the new custom Linux distribution, created using the Yocto project tools, that replaced the old OpenWrt operating system on which previous tests were done.

The initial implementation process was divided into two main steps that were needed to achieve a solution that works with all types of links and for all use cases. The first step was to implement and evaluate the default MPTCP protocol into the Linux kernel in the target environment, following this an encapsulation feature was added to deal with interfering middleboxes such as the GEO satellite terminals that strip part of the headers. Finally, when the evaluation strategy had been devised, the initial solution implemented, and an evaluation of it had been measured as a baseline, the plan was to focus on optimising the solution including by looking at the additional features mentioned in Section 1.2.

The goal improvements can be seen in the order of priority as follows:

1. Use a proxy server to be able to use MPTCP communication on a subpart of the path when endpoint servers do not support it.

2. Scheduler optimised for satellite communication (heterogeneous links).
3. Optimisation of link swapping and choosing primary links, and secondary links.
4. Testing if a proxy server is needed for an MPTCP connection or not.
5. Optimise for different data flow sizes.

1.4 Outline

In Chapter 2, relevant background knowledge is described, as well as details on how the MPTCP protocol is built. Related work is also covered along with a motivation for how this thesis contributes to the research field. In Chapter 3, the method and approach for the work are described together with the tools used for accomplishing our goals with the thesis. In Chapter 4 we present the results of the study while in Chapter 5 we discuss the results and conclude our thesis.

2

Background

The following chapter will provide detail about satellite communication and an overview of the MultiWAN concept. It will also give a description of the TCP protocol, which the MPTCP protocol builds upon, followed by a thorough explanation of the MPTCP protocol itself. Following that, concepts such as *encapsulation* and *proxies* will be discussed, as well as the practical aspect of the work through a description of the Yocto Project. Finally, related work will be described and discussed.

2.1 Satellite Communication

Satellite communication is a form of wireless communication that uses satellites orbiting the Earth. It allows for communication over large distances without the need for infrastructure on the ground. A major benefit of satellite communication is that it can provide Internet service to remote areas that lack the infrastructure for it otherwise. However, it can be less reliable compared to alternatives as it is subject to interference from weather conditions and other sources, which can negatively affect signal quality.

There are many different kinds of satellites that can be categorised by which type of orbit they are placed in. Two types of satellite orbits will be covered in this thesis: geostationary orbit (GEO) and low earth orbit (LEO), which can be seen in Figure 2.1 and Table 2.1.

Table 2.1: Characteristics of the two different satellite link types.

Link type	***Download***	***Upload***	***Latency***
GEO	∼10 Mb/s	∼3 Mb/s	∼350 ms
LEO	∼100 Mb/s	∼20 Mb/s	∼40 ms

As its name describes, GEO satellites are geostationary, which means they are in a fixed position in the sky relative to the surface of Earth. This is due to the satellites orbiting around Earth at the same speed and angle as Earth itself rotates at a radius distance of approximately 36,000 kilometres from the Earth's ground. As they are always in a fixed position relative to the Earth, this gives high reliability when using them, but with the drawback of a largely increased latency and reduced bandwidth compared to alternatives such as LEO satellites, with their closer orbit.

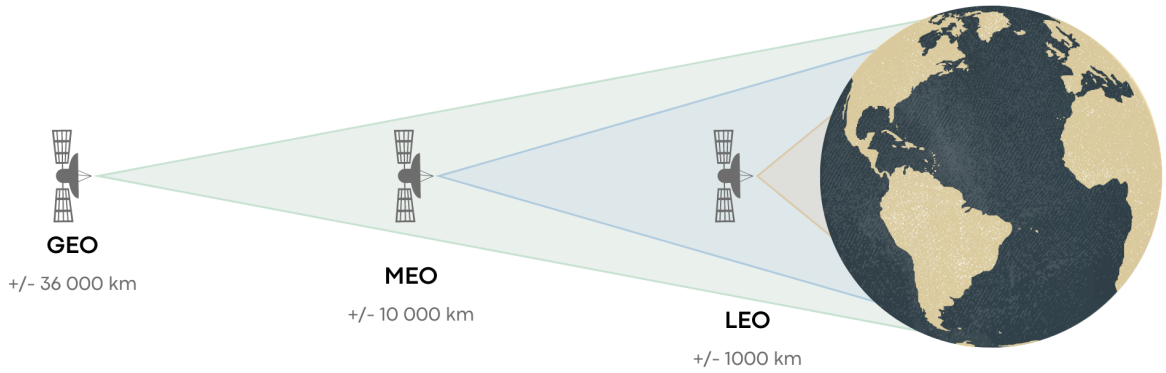


Figure 2.1: Visualisation of distance from Earth for different types of satellites.

LEO satellites, on the other hand, have an orbit much closer to Earth in the range between 80 - 2000 kilometres from the ground, which gives them a lower latency and higher bandwidth in relation to the orbiting distance. The low orbit distance of the LEO satellites means that they need to travel at a velocity of approximately 7.8 kilometres/second, which translates to 28,000 kilometres/hour, to not fall out of orbit. By travelling at this distance, they complete an orbiting period around Earth every 128 minutes with the drawback that LEO satellites have a small field of view, and can only observe and communicate with a fraction of Earth at a time. Therefore, satellite terminals communicating using LEO satellites must continuously switch between satellites as they pass over the terminals, as often as every few minutes, which results in constant connection establishment and termination when using single-link communication.

Satellite terminals, also known as teleports, are the specialised infrastructure used in the telecommunication industry to send and receive signals to and from satellites in space. It is used as the gateway between the terrestrial network and the satellite network, converting signal frequencies as well as modulating and demodulating the signals. An example of this is the KU terminal, which Satcube produces and either rents out or sells, which can be seen in Figure 2.2.

In general, when data reaches a teleport to be transmitted to a satellite, it is in the form of baseband signals, also known as low-frequency signals, which have a limited frequency range. These signals are then modulated onto a higher frequency carrier by the teleport's satellite modem to be suitable for long-distance transmission in space. When data is received from a satellite, the teleport instead demodulates the signals, reversing the process to convert the signals back to their original form as baseband signals.

In addition to the satellite modems, other equipment that is essential to establish a satellite connection include frequency converters and satellite antennas. Antennas are either fixed after the initial positioning, as for when using GEO satellites or are continuously adjusted to track the satellites as they move across the sky, as in the case of LEO satellite communication.

Figure 2.2: Satcube's KU satellite terminal for GEO satellite communication [2].

Due to the characteristics of satellites, the transmission of data between satellites and ground stations is considered to be the bottleneck of the path between endpoints using satellite communication. More specifically, some of the main reasons are the longer latency, caused by the distance from Earth, compared to terrestrial networks, the limited bandwidth of the frequency bands which the satellites compete over, and the limited resources of the satellites.

One of the possible solutions to this bottleneck would be the use of multiple simultaneous connections through the use of a MultiWAN solution, such as the MPTCP protocol. Using reliable links such as GEO satellites as a backup would also improve the experience with LEO communication, as it would remove overlay in the form of the constant need to reestablish connections when switching between LEO satellites as the connection could always fall back to the stable GEO connection during the switch.

2.2 Multiple Wide Area Networks (MultiWAN)

The MultiWAN concept refers to a network approach which allows the use of multiple Wide Area Network (WAN) links. Often, Dual-WAN is associated with this and follows the same concept, but unlike MultiWAN, is restricted to the use of two WAN links. Compared to the more traditional use of one WAN at a time, the use of multiple WANs brings benefits such as increased bandwidth, higher reliability and improved redundancy to a network. It is commonly used in business or enterprise environments where a single WAN may not be sufficient to meet the requirements of the organisation. MultiWAN is usually realised through having multiple links to different internet service providers (ISP) to access the internet while having some redundancy, which can result in being less affected by service outages from the ISPs. With multiple WAN links at hand, they can be configured and utilised simultaneously in various ways in order to achieve a particular desired quality, such as aggregating the links, load balancing or as a failover mechanism for higher robustness.

The implementation of MultiWAN requires suitable network equipment such as routers or firewalls capable of utilising multiple WAN connections, these are often referred to as load-balancing devices, as it is their key benefit. MultiWAN devices distribute the traffic among the WAN connections based on load-balancing algorithms that take in factors such as latency, link capacity, or predefined rules. It is common that different departments of a company are allocated one or more WAN interfaces to prevent traffic-heavy departments from unfairly taking all available capacity, which can affect the other departments negatively by the starvation of resources. It is also common that they include Quality of Service (QoS) capabilities that can prioritise more critical types of traffic over non-critical traffic to ensure that desired service levels and performance are maintained.

There are also a number of DualWAN or MultiWAN routers that have basic fail-over or fall-back mechanisms for broadband users or companies that are concerned about the reliability of their internet connection. These routers will automatically switch to a WAN interface that is online when another fails to prevent loss of connection.

MultiWAN devices also have link aggregating capabilities, take for example, two users that simultaneously want to stream a 3 Mbps video feed. This can't be done on a single 4 Mbps link, but by having a load-balancing router and two available 4 Mbps links, one of the connections could be allocated over to the other available link making it possible. However, one of the limitations of using MultiWAN devices is that a single connection can't be split between links, even if there is more than one available. Because of this, a 3 Mbps stream could not be divided over two available 2 Mbps links even though the total bandwidth is larger.

This is where other multipathing solutions, such as MPTCP, can provide new benefits, such as the possibility to split a connection over multiple links.

2.3 The Transmission Control Protocol (TCP)

The Transmission Control Protocol (TCP) is a vital part of the transport-layer protocol group that has existed and been continuously improved on for decades. It is currently the most suitable alternative when a reliable, in-order, byte-stream service for applications is desired. To achieve this type of communication, TCP uses multiple features such as connection establishments, packet loss detection and retransmissions, segmentation and reordering of packets, as well as flow control and congestion control. The TCP protocol exists in the transport layer and TCP packets have a structure consisting of a payload containing the data and a header part containing the necessary information for the transmission of the packet.

A fundamental aspect of the TCP protocol is the use of sequence numbers that all packet headers contain. The sequence numbers are used to guarantee that all packets sent will be received and delivered in the correct order. To do this a starting number is chosen for the first packet of a communication sequence and that number will then be incremented for every following packet by that peer. Each peer uses their own sequence numbers and sends an acknowledging (ACK) packet as a reply for each packet it receives containing the next expected sequence number of the other peer.

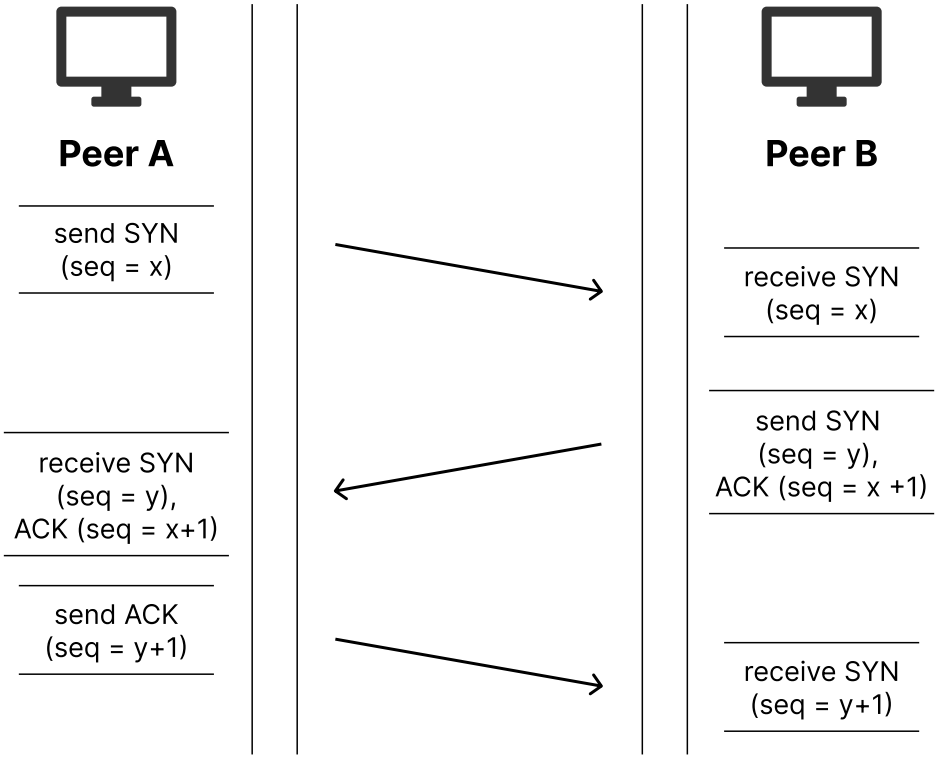


Figure 2.3: A TCP three-way handshake

To ensure the reliability of TCP communication, connections are always established before data transmission can occur. This is to guarantee that both sides of the communication are available and ready to transfer data. To establish a connection,

the TCP protocol uses a "three-way handshake" which can be seen in Figure 2.3. The "three-way handshake" normally starts with peer A sending an SYN message together with a chosen sequence number, peer B responds with an SYN message together with its own chosen sequence number and also acknowledges the first SYN message with an ACK as well as the next expected sequence number of peer A. Finally, peer A replies with an ACK that contains the next expected sequence number of peer B, with this the connection is established and data can be transferred.

Packet loss detection is handled by the aforementioned use of sequence numbers, where a peer can detect that a packet is missing if it receives a packet with sequence number x and a packet with sequence number $x + 2$ but not the packet in between. When the receiver discovers that a packet has been lost it communicates this to the sender by sending an ACK with the sequence number of the lost packet that it expected to receive. The sender then re-sends the packet with that sequence number.

TCP also uses what is known as flow control to regulate the rate at which data is being transferred between the two peers to ensure that the receiver can handle the incoming data. The goal of flow control is to prevent the receiver from being overwhelmed with data that it cannot process or store which could result in packer losses. To implement the flow control TCP uses what is called a sliding window which allows the sender to transmit multiple packets without waiting for an acknowledgement for each packet improving efficiency. In the sliding window mechanism, the sender keeps a window of packets it can send without receiving acknowledgement based on the buffer size of the receiver and the congestion of the network. When the sender transmits a packet it moves the window forward a position. If an acknowledgement for a sent packet is received the window is moved forward a position again and a new packet is transmitted. If an acknowledgement is not received within a time limit, the sender retransmits the packet without moving the window forward hence the sliding window name [10].

Congestion control is a mechanism that controls the rate at which data is being sent over a network to prevent the network from becoming congested. A congested network can result in packet losses and degraded performance for all users on the network. When congestion is detected the sender reduces the sending rate by reducing its transmission window, the window is then increased over time as long the network is not congested. This process is known as congestion avoidance.

TCP congestion control also includes a mechanism for when the network is severely congested known as fast retransmit and recovery. In this mechanism, the receiver sends duplicate ACKs if a packet is lost to inform the sender of the loss. When the sender receives this it immediately retransmits the lost packet instead of waiting for a timeout. This can help quickly recover from a packet loss and keep optimal network performance. There are multiple options in algorithms that TCP can use to achieve congestion control, where the most commonly used is the TCP Vegas algorithm which is a variation of the traditional TCP Reno algorithm.

Even if TCP is a reliable and widely used protocol it has its limitations, where some of the more significant include:

- Limited Real-time data support: TCP was not designed for real-time data transmissions such as audio or video streaming where delays from retransmissions can cause poor performance
- Large overhead: TCP has a larger overhead than other options, this stems from larger headers, connection establishment, retransmissions, congestion control, et cetera.
- Large scale data transfers: The larger overhead of TCP can become a bottleneck in larger data transfers, other issues are the slow start of the TCP and the congestion control algorithm.

Finally, another limitation of TCP is the single-path communication that it uses which is why the MPTCP protocol was created.

2.4 Multipath TCP (MPTCP)

The MPTCP protocol, unlike TCP which is still limited to single-path connections, has the ability to use multiple links simultaneously or as backup links. The possibility to use multiple paths can improve resource usage within the network and in turn, improve user experience through higher throughput and better resilience to network failures. MPTCP provides the capability to simultaneously use multiple paths between peers while maintaining the same type of services as TCP to applications. MPTCP achieves this while operating on the network layer in the same way as the TCP protocol and aims to be transparent to both layers above and below the network layer. This brings additional features on top of TCP, the network architecture when using MPTCP can be seen in Figure 2.4.

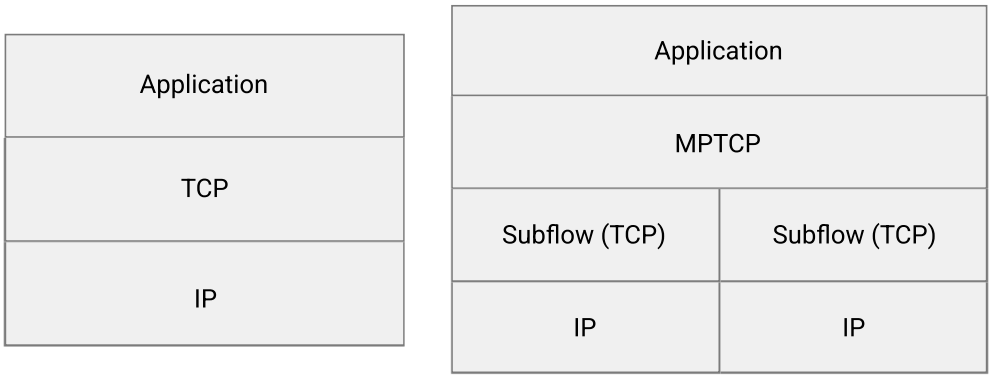


Figure 2.4: Standard TCP vs MPTCP protocol stacks.

The work of MPTCP is an ongoing development by the Internet Engineering Task Force (IETF) which formally started in 2013 with the publishing of the Multipath Specification as an experimental standard in RFC 6824. The RFC 6824 became known as the Version 0 of MPTCP and was replaced by the Version 1 specification in RFC 8684 in 2020. Both versions are still in use and share functionality and version 0 can be used instead of version 1 if one or both endpoints do not support version 1.

Although version 0 performs worse than version 1 and lacks features it is still likely preferred over a regular TCP connection.

2.4.1 Design Choices

During the design phase of the protocol, two key constraints were specified:

- It must be backwards compatible with the current regular TCP.
- It can be assumed that one or both hosts are multihomed and multiaddressed.

To encourage the widespread implementation of MPTCP the creators deemed it important that the design was compatible with regular TCP in all aspects. More specifically, the protocol must function and resemble TCP as much as possible to all external constraints such as firewalls, NATs, and proxies. It must be usable with existing applications without the need for modifications and provide the same services as regular TCP to the applications. The protocol should be able to fall back to regular TCP without user interference if something goes wrong.

For the use of MPTCP between two hosts to be possible, at least one of them needs to be multihomed and multiaddressed, that is that they are connected to more than one network and have more than one IP address available to create paths between. These paths need not be entirely disjoint from each other and can share points, even if they share parts of the route multipathing is useful as it improves resource utilisation and is less vulnerable to a subset of node failures.

2.4.2 Connection Establishment

When there is an attempt to establish an MPTCP connection, the procedure follows the same structure as a TCP connection establishment with a three-way handshake but with an additional option in the header at each step. The SYN, SYN/ACK and ACK messages all carry the MP_CAPABLE option in their headers. This option serves two purposes, it confirms that both peers have support for MPTCP and want to use it, and it is also used for agreeing on how to authenticate and establish additional subflows. The highest version number that each peer support is included in the MP_CAPABLE option from which the peers will agree on the highest version that they both have support for during the handshake procedure. If one of the handshake messages is missing the MP_CAPABLE flag, the MPTCP functionality is aborted and the connection falls back to using normal TCP communication.

When the main subflow has been successfully established, further subflows if available will be added and associated with the MPTCP connection. This is done in the same way as with the main subflow but instead of using the MP_CAPABLE flag the MP_JOIN flag will be used to indicate that the subflow is not the initial and that it should be associated with the main subflow. The initiation procedure of a primary and secondary subflow can be seen in Figure 2.5. During the initiation of the main flow, information about possible endpoints that additional subflows could be established between was shared between the peers. If the set of available IP addresses associated with a multihomed host changes during the connection's

lifetime this information can be shared between the peers without the creation of a new subflow with the ADD_ADDR and REMOVE_ADDR options which adds and removes addresses for potential subflows accordingly.

2.4.3 Data Transfer

The structure of transferring data with an MPTCP implementation can be seen as the MPTCP connection taking a data stream from an application and splitting the packets between its active subflows. Additional control information is included in the packets to guarantee the reassembly and reliable delivery of the data to the recipient, mainly through the Data Sequence Signal (DSS) option of the header. The DSS option provides information about Data Sequence Mapping and Data acknowledgement, both can be included in the same DSS or be provided separately depending on the flags that are set.

The Data Sequence Mapping (DSM) component of the DSS option serves the purpose of mapping the sequence numbers of each subflow, which is the regular sequence numbers of a TCP communication, to the data sequence numbers of the whole MPTCP data stream. This is expressed as a starting sequence number for the subflow and the data level and a length in bytes for which the mapping is valid. After the mapping is processed it will be fixed in that the subflow sequence number is tied to the data sequence number and not allowed to be modified, however, the same mapping can be assigned to multiple subflows for retransmission purposes or for being sent simultaneously for higher reliability. The DSM also includes a checksum for the data covered by the mapping, if checksum functionality has been agreed upon during the connection establishment. The checksum is used to detect modifications of any sort by middleboxes that do not have MPTCP support, if the checksum fails a subflow failure is signalled or alternatively, the connection falls back to regular TCP communication. Enabling checksumming is always recommended as there is

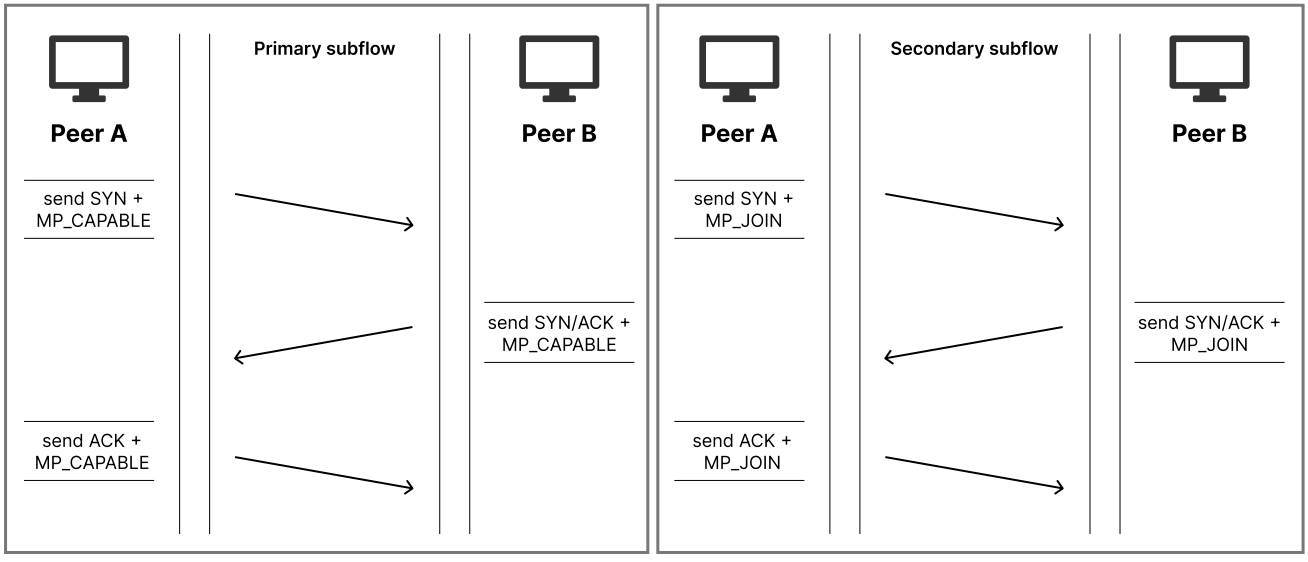


Figure 2.5: The initiation of a primary and secondary subflow.

otherwise a risk that corrupted data is delivered to the application.

The Data Acknowledgement (DATA_ACK) component of the DSS option serves the purpose of providing complete end-to-end reliability. The behaviour of the DATA_ACK acts in the same way as the TCP cumulative ACK in the way that it indicates how much data that has been received without gaps. While the subflow level connections transmit regular ACKs similar to TCP SACKs where the recipient sends ACKs for the packets it has received to indicate which packets need to be retransmitted, DATA_ACKs provide proof that the data and all required MPTCP signalling have been received by the other endpoint.

2.4.4 Connection Termination

To close a regular TCP connection a FIN message is used, however, using MPTCP communication a FIN message closes only the subflow on which the message was transmitted and not the whole MPTCP connection. This functionality serves the purpose of allowing the subflows to operate independently and to have the same appearance as a TCP connection. In the same way, as with TCP communication, the subflow is not fully closed until both peers have ACKed each other's FIN messages. To fully close an MPTCP connection a mechanism referred to as a DATA_FIN is used which indicates that the sender has no more data to send and that the MPTCP connection can therefore be terminated. The DATA_FIN message is indicated by setting the F flag in the DSS option and has the same behaviour as a TCP FIN message but on an MPTCP connection level. The DATA_FIN message will only be responded to with an answering DATA_ACK message when all data transmitted has been successfully received by the other peer. Once the DATA_FIN message has successfully been DATA_ACKed, a regular FIN message should be sent out by both peers on all active subflows to terminate them correctly, finally, it is only when both peers' DATA_FIN messages have been successfully DATA_ACKed that the MPTCP connection can be considered to be terminated.

There exists an additional method for closing an MPTCP connection referred to as a "fast close", which can be seen in Figure 2.6, which acts as a TCP RST signal. To communicate a "fast close" the MP_FASTCLOSE signal is used, this tells the other peer that no more data will be transferred and that the connection will be abruptly closed. The MP_FASTCLOSE signal can be sent in an ACK message with ensures the reliability of the signal or an RST message which does not [3].

2.4.5 Scheduling Algorithms

The Linux kernel implementation of MPTCP includes three different schedulers:

- **Minimum Round-trip Time (Min_RTT) scheduling:** Min_RTT is the default scheduler and schedules the data between the subflows based on the RTT. It will first send packets on the subflow with the lowest RTT until its congestion window is full, it will then move on to the next subflow with the lowest RTT and send packets on that subflow until its congestion window is full.

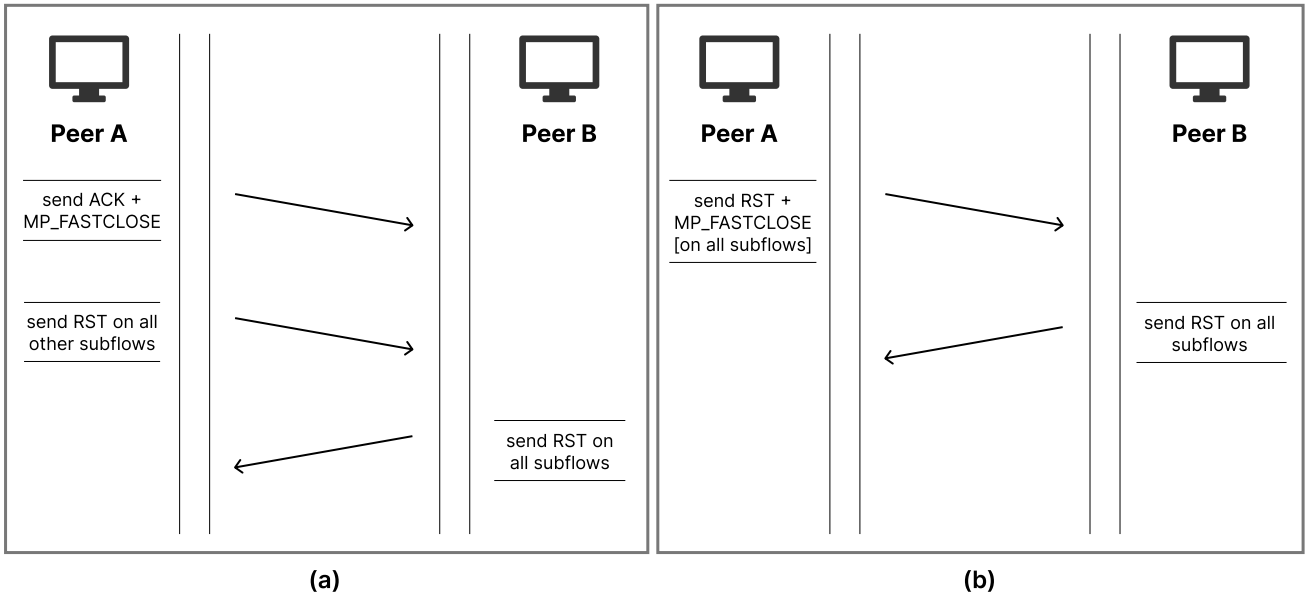


Figure 2.6: MPTCP connection termination through fast close: a) Fast close using an ACK message; b) Fast close using an RST message.

- **Round-robin scheduling:** Schedules the packets in a round-robin fashion where each subflow takes turns in sending out a specified number of segments on each subflow. Round-robin guarantees that the capacity of each subflow is filled and is well-suited for larger data transmission that prioritises available bandwidth.

- **Redundant scheduling:** Schedules the same packets on all active subflows simultaneously, prioritising the redundancy over maximal throughput. Suitable for delay-sensitive transmissions where the loss of packets can be very harmful.

In addition to these three default scheduling schemes, there exist many proposed alternatives that attempt to improve performance based on different environments and situations.

One of the more noteworthy of these is the "Out-of-order Transmission For In-order Arrival" (OTIAS) scheduler. As the name implies the scheduler sends out data packets out of order on the different subflows with the goal of them arriving in order at the destination by estimating the one-way transmission delay for each subflow. The goal of the OTIAS scheduler is to reduce jitter and achieve a higher capacity on each subflow by doing this while reducing the Head of Line (HoL) blocking. The OTIAS scheduler can respond more dynamically to network changes compared to alternative schedulers but has the downside that it performs poorly when the difference between paths is large there are packet losses on the links making its HoL blocking prevention less effective [11].

2.4.6 Congestion Control Algorithms

As part of the design choices, the focus was laid on making MPTCP connections co-exist with normal TCP connections in a way that is fair to all parts. This means that the MPTCP connections will compete for available bandwidth in a way that is neither too aggressive nor too timid. More specifically, the use of multiple paths must not harm users with regular TCP connections at bottlenecks more than a single-link TCP connection would. To achieve this goal, new congestion control algorithms have been proposed and integrated. These can be categorised into loss-based or delay-based algorithms or a hybrid of both. The loss-based algorithms detect congestion by monitoring the number of packets that are being lost or dropped in the network to determine the congestion level and adjust the sending rate accordingly. Delay-based algorithms, on the other hand, determine the congestion of the network by looking at the RTT of the data packets that traverse it. They assume that an increase in RTT is a sign of congestion and reduces the sending rate of data as a consequence. The Linux kernel implementation of MPTCP comes with four different congestion control algorithms as default:

- **Linked Increase Algorithm (LIA):** LIA is a loss-based congestion control algorithm that works similarly to congestion control for TCP where it increases the sending rate incrementally when there is no congestion and halves it when there is congestion individually for each subflow. It keeps track of the congestion with a congestion window for each subflow kept in a linked list. It also enforces fairness to single-link TCP connections by applying a limit to the sending rate for the MPTCP subflows.
- **Opportunistic Linked Increase Algorithm (OLIA):** OLIA is a delay-based congestion control that keeps track of the RTT of the different network flows as well as the overall congestion level of the network. From this, the algorithm approximates a network backlog which is the amount of data that is currently being transferred over the network that has not been acknowledged as received yet. Based on the network backlog, the algorithm adjusts the congestion window of each subflow.
- **Delay-based Congestion Control for MPTCP (wVegas):** The wVegas algorithm is a modified version of the original Vegas congestion control for TCP that has been altered to work with MPTCP communication. By using the RTT of the different subflows it estimates the overall congestion level of the network in the same way as the original Vegas algorithm. An additional function of wVegas compared to Vegas is that it uses a "weight" factor to adjust the sending rate of each subflow based on the available bandwidth of each subflow. The "weight" factor is used to balance the sending rates to prevent one subflow from dominating the available network capacity which can cause unfairness and reduced performance.
- **Balanced Linked Adaption Congestion Control Algorithm (BALIA):** BALIA is a hybrid algorithm that uses both loss-based and delay-based information to detect congestion. It uses a mechanism called "proportional sharing" to allocate the sending rate across multiple subflows. "Proportional sharing"

ensures that the data is transmitted fairly over the subflows by looking at both the RTT and packet loss individually for the subflows and dynamically adjusts the sending rate. BALIA also has a number of features such as adaptive backoff and synchronisation mechanism that makes the congestion control algorithm operate efficiently and effectively.

Many additional algorithms have been proposed and investigated, where the Coupled Congestion Control (CCC) algorithm is one of the more known as it was proposed as an alternative in the MPTCP RFC documentation and is an early experimental implementation for a multipath congestion control algorithm. The CCC algorithm works by coupling each subflows congestion control to achieve a globally coordinated congestion control. Each subflows congestion window is adjusted according to the congestion windows of the other subflows, if a subflow experiences congestion it reduces its own congestion window and signals to the other subflows to do the same. This way the congestion control is coordinated between the subflows and the congestion level of the network gets reduced. The CCC algorithm has its flaws and was mostly proposed as an experimental algorithm to use for evaluation and research purposes in the earlier stages of MPTCP [12].

2.4.7 Benefits & Use Cases

MPTCP can bring many benefits depending on how it is used and the environment it is implemented in. Some of the major possible benefits are summarised here briefly:

- **Increased throughput:** By aggregating the available links, it is possible to sum up their available bandwidth to increase the total throughput of the connection. However, it is worth mentioning that currently, the aggregated throughput is very dependent on the link characteristics and the configuration used and the resulting throughput can vary and sometimes be worse than one of the link's individual throughput.
- **Resilience to failure and connection aliveness:** Similar to other Multi-WAN solutions, the availability of multiple links brings an increased resilience to failure. If one of the links becomes unavailable the option to redirect the traffic to an available link prevents the connection from terminating. For each additional link, this reliability increases making it less likely that the connection dies.
- **Improved network security:** By dividing the data packets of one connection over multiple links it becomes increasingly difficult to intercept the whole transmission and get the whole picture of the message.
- **Transparency to the TCP protocol:** By designing the MPTCP protocol so that it is transparent to TCP it makes the implementation of MPTCP easier. When failure occurs the connection can immediately fall back to using regular TCP communication as well.

Some of the use cases for MPTCP that have shown promise so far include cellular communication, large-scale data centres and potentially satellite communication.

Cellular devices often have more than one active internet link at the same time in the form of both a broadband cellular network such as 4G, and an available WiFi connection. By using MPTCP you could improve the throughput by using both simultaneously and it would also improve the connection aliveness. If the user would be located on the perimeter of the WiFi range, moving in and out of it, it would prevent the connection from being terminated and reestablished continuously as would be the case without using MPTCP. Large-scale data centres transfer extensive amounts of data constantly and could have great benefits from MPTCP to improve the total throughput and also from higher resilience to failure by redirecting the traffic to available paths as soon as a path would be unreachable. Finally, although it has not been thoroughly researched as of today, the potential improvements in satellite communication are of interest. Both for improving more stable satellite alternatives such as GEO satellites by aggregating two or more of them for higher throughput. The option to use a stable but low bandwidth satellite such as GEO as a backup option for when using a higher throughput LEO satellite that is less reliable could bring user experience as it would prevent connection loss when switching between LEO satellites.

2.4.8 Limitations

Although the MPTCP protocol was designed to be transparent to TCP communication to avoid middlebox interference, the communication is still impacted by middleboxes to some extent. A paper was presented by Honda et al. at the Internet Measurement Conference (IMC) 2011 that described the behaviour of middleboxes that can interfere with MPTCP communication [13]. Some of the more noteworthy are as follows:

- Some of the middleboxes replace TCP options with dummy options.
- Some of the middleboxes inspect the payload of TCP segments, sometimes modifying the TCP payload or rejecting out-of-sequence segments.
- Some of the middleboxes drop TCP options that they do not understand.
- Some of the middleboxes change fields of the TCP header needed for MPTCP communication.

A useful tool for detecting middleboxes that modifies the packet headers is called “tracebox” which is an extension to the more well-known network diagnostic tool “traceroute” used to trace the path of traffic through networks. While traceroute only detects IP routers, tracebox can also identify any type of middlebox that alters fields of the network or transport header.

Currently, MPTCP and other MultiWAN solutions still struggle with handling heterogeneous links that have very different characteristics from each other. There exists a lot of research that attempts to negate this but usually, they are not general to all environments. The default Linux kernel implementation of MPTCP performs somewhat poorly currently but by choosing a more suitable scheduler or congestion control algorithm for handling this, improvement can be seen. There also exist

more complex solutions adding an additional layer to the implementation to further improve performance, further detail about this can be seen in the "Related Work" section 2.8.

2.5 Encapsulation, Network Tunnelling and VPNs

Encapsulation is the process of wrapping content inside outer layers of additional information, an real-life example of this process is when a letter is put inside an envelope that gives detail about where the letter should be sent. In computer networks data from the higher layers of the OSI model is wrapped inside outer data in the form of headers and optionally trailers that gives information about how the data should be transmitted between endpoints while at the same time logically separating the different layers and "hides" information of higher up layers from lower level layers. This brings better protection for the inner data, that is the content that the user or application wants to transmit, while the outer layers are there for transmission purposes and often get scaled off and replaced at middle points during transmission. The standard encapsulation process of data when passing through the network layers can be seen in Figure 2.7.

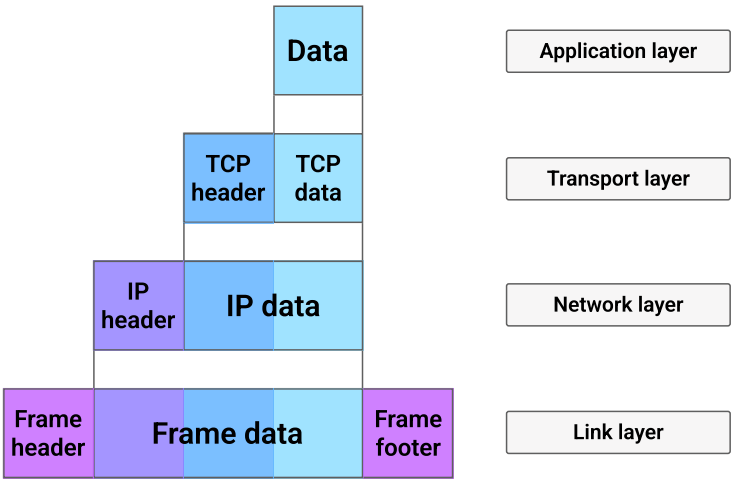


Figure 2.7: The packet encapsulation process

Encapsulation is applied in the concept of tunnelling, which is the process of transferring data between different networks. It involves allowing private network communication to be sent over public networks such as the Internet by encapsulating the data, transmitting it over the public network and then decapsulating the data. Tunnelling in some form is always used when data is traversing different networks but there are also additional tunnelling alternatives for situations such as when further security of the communication is desired or when not all networks support all protocols that are wanted during the communication.

Some of the more commonly used of these additional tunnelling features include:

- Generic Routing Encapsulation (GRE) tunnels: The GRE protocol encapsulate

packets that use one routing protocol inside the packets of another routing protocol. It is often used to set up a point-to-point connection to simplify connections between different networks. The GRE protocol adds two additional headers to each packet, a GRE header to signal that GRE tunnelling is being used as well as an additional IP header which encapsulates the original packet's IP header and payload. Only the routers at each end of the GRE tunnel will inspect the original IP header for routing purposes.

- IP-in-IP tunnels: Similar to GRE tunnels, IP-in-IP tunnels encapsulate IP packets inside other IP packets. Its main use is setting up network routes that otherwise would not be available.
- Secure Shell (SSH) tunnels: As the name implies SSH tunnels add security in the form of encryption to the communication between the endpoints. SSH is often used to remotely access devices, for example when working remotely from home and accessing the company network.

Another networking concept which uses tunnelling and encapsulation is Virtual Private Networks (VPNs). VPN is a technology that gives a safe and encrypted path over a less secure network as it uses tunnelling protocols to route the traffic over a public network. VPN is a way to extend the private network over a public network such as the Internet through a virtualized private network as the name implies. By tunnelling the internet traffic through a VPN it hides the traffic from eavesdroppers that try to look at and modify or steal the data as well as from Internet Service Providers (ISPs) or middleboxes that modify or reject the packets as can be seen in Figure 2.8. When a user communicates with other endpoints through a VPN the real location and IP address of the user are camouflaged and instead, the VPN server's location and IP address are seen as the user's endpoint from an outside perspective. This enables a user to remotely access a private network, for example, when an employee accesses company files on the office network when working from home. Another benefit of using a VPN is that it lets a user access websites or applications that are banned in the user's country by tunnelling the communication through a VPN server in another country where it is allowed. A VPN can also improve the online experience by avoiding internet throttling, which is when the ISP slows down the user's internet connection based on online activities, when streaming or downloading data this can be especially beneficial.

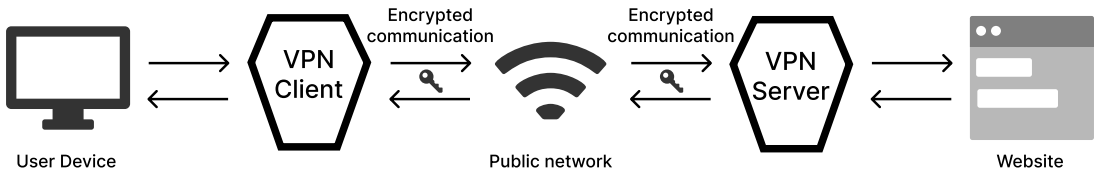


Figure 2.8: Communication over the Internet using a VPN

Wireguard is a relatively new VPN protocol that has gained a lot of popularity. With its lean code base and simple design it challenges the gold standard of VPN protocols, OpenVPN. The fact that it both runs in the Linux kernel and makes use

of extremely high-speed state-of-the-art cryptographic primitives gives the possibility of high speeds. Furthermore, its code base is way smaller than its competitors and one of its goals is for the code to be comprehensively reviewable by single individuals, which in turn minimises its attack surface and makes it easier to find and patch security vulnerabilities [14].

2.6 Proxy Servers

A proxy server, similar to a VPN, acts as an intermediary between endpoints and redirects traffic between them. As it is the proxy server that contacts the endpoint server it is the proxies IP address that is seen in the sender IP field of the traffic, although the actual sender address of the data is usually included on a higher layer depending on the protocol. The purpose of using proxy servers is commonly for anonymity or to work around restrictions such as blocked websites. It can also be used to modify the data in the middle of its path between endpoints, for example, if a user would want to make use of MultiWAN functionality for a subpart of the communication even if the endpoint server does not support it. Although VPNs and proxy servers behave very similarly, the biggest difference between them is that VPN servers automatically encrypt the data and send it through a tunnel while proxies only act as a middleman that forwards the data. When using proxy servers the data flows between the client and server through the proxy but there also exists reversed proxy servers where the data flows between the client and reversed proxy which then forwards the data to the appropriate server. While proxies and reversed proxies seem similar they have different use cases. A reversed proxy appears to clients to be a regular server and is used to provide load-balancing, improve performance or enhance security.

Hypertext Transfer Protocol (HTTP) proxies are designed to redirect traffic looking at the data only at the application layer. They are faster than other proxies as they only need to look at the highest OSI layer but can only be used when accessing websites with HTTP or HTTP Secure (HTTPS) URLs. Another protocol that can be used for proxies is the Socket Secure (SOCKS) protocol which operates at the transport layer of the OSI model. Unlike HTTP proxies, SOCKS proxies are generic and can be used with any type of protocol. One of the key benefits of using SOCKS is that it also offers authentication which can be used to verify the identity of the client device and prevent unauthorised access to the proxy server. A disadvantage of using SOCKS is that each software needs individual configuration to be able to use the SOCKS proxy.

2.7 The Yocto project

In order to implement an MPTCP solution together with Satcube for their development environment the Yocto project is used. It is an open-source build system that makes it easier for developers to create their own custom Linux operative systems for embedded systems. It allows developers to include the specific drivers, kernel

configurations and applications that are needed for their specific hardware and its use.

The structure of a project that uses the Yocto build system can be simplified to roughly be seen as a set of Layers where each Layer contains Recipes. The different Layers are ordered and can be seen as stacked on top of each other where the recipes in each layer can be used either to introduce a new piece of software or to modify another already existing recipe from an underlying layer. This structure of layers and recipes provides a flexible and modular way for developers to customise their operating system by adding, removing and modifying different components in order to meet their specific needs.

One of the main benefits of using Yocto is its modularity which gives the developer the power to pick and choose only the components that are necessary which results in a more optimised system. At the same time, the modularity also makes it possible to easily reuse and share parts that could be useful for other developers.

However, because the developer has so much power in what components will be included and how they will be configured, there is a steep learning curve for understanding how the build system works while it also requires the developer to have a deeper understanding of how a Linux system is structured.

2.8 Related Work

There exist much research that cover various different aspects of MPTCP, that cover various topics such as performance evaluation, scheduler optimisation, solutions for MPTCP to TCP translation, and comparisons of alternatives to MPTCP.

There are many alternatives to using MPTCP that are researched and considered, using other protocols in the network transport layer or by a combination of different solutions.

Fahmi et al. [15] study solutions for MultiWAN routers in the context of public transportation, where two common solutions are splitting the incoming TCP connections into TCP and MPTCP portions by the use of a transparent proxy, or by encapsulating the data inside an MPTCP tunnel. They point out the flaws of these solutions, more notably the use of a proxy can lead to low capacity utilisation and high loss, while a tunnel solution that tunnels TCP connections through an MPTCP tunnel can cause what is known as “TCP meltdown” where a high number of re-transmissions are constantly triggered. “TCP meltdown” is caused by stacking two reliability layers (TCP and MPTCP), each with its own re-transmission mechanism and congestion control. The Authors propose an alternative solution called BOOST that instead uses the QUIC protocol, which takes qualities both from the tunnel and proxy solutions and combines them. However, it eliminates the problems with both proxies and tunnels by instead multiplexing TCP and UDP connections over a single persistent multi-path connection. It also combines multi-path load balancing and scheduling. Short flows are transmitted across a single link to avoid Head of Line blocking, while longer flows are transmitted across multiple paths, utilising

left-over capacity. The authors evaluate BOOST in a simulated environment where they compare it to the aforementioned alternatives and find that it provides better throughput, lower loss and consumes less memory than all MPTCP variants it was compared to. It also provided improved latency and reliability for UDP through redundancy and prioritisation.

Amend et al. [16] present a new multipath framework for heterogeneous links that use MP-DCCP which builds upon the DCCP protocol of the transport layer. In their framework, they make use of a DCCP tunnel which allows both TCP and UDP traffic, the tunnel provides both congestion control and redirection of traffic capabilities. The authors evaluate their proposal by experimenting with different scheduling and congestion control options with the goal to highlight the different benefits. One of the more notable benefits of the framework is its increase in throughput and improved packet latency for UDP traffic.

As stated in subsection 2.4 covering MPTCP, different alternatives have been proposed to the schedulers included in the Linux kernel. Some of the more notable alternatives according to us are briefly described.

Sailer and Hähner [17] propose a scheduler called HTMT that transmits data using the DCCP protocol which is a compromise between the traditional TCP and UDP protocol. The DCCP protocol acts like UDP sending data without reliability but has congestion control functionality like the TCP protocol. The HTMT scheduler is flow-aware which means that there is no need for a reordering functionality since the scheduler only sends data over subflows where the packets do not arrive before the previous packets in the data flow. It also uses an algorithm that considers the latency and throughput of the subflows to choose the most suitable choice. The paper claims that the HTMT outperforms the default MPTCP scheduler MinRTT as well as two other schedulers they compare it to called OITAS and AFMT but the evaluation only covers two asymmetric measurements and one symmetric measurement which is fairly narrow.

In the paper by Xing et al. [18], the optimal transmission strategy for different data flow sizes is investigated where different MPTCP scheduling algorithms are found to be better suited for different flow sizes. For larger flow sizes they found that a fair round-robin scheduling over all subflows is best suited for maximised bandwidth. While for smaller to medium flows, that are often delay-sensitive, it is better to use the redundant scheduler to avoid packet loss or the default Min_RTT scheduler. Finally, for the minuscule flows, the best option is to send the whole flow of data over one subflow only to avoid and overhead from reordering and such completely. They also propose a cross-layer solution named FSA-MPTCP which automatically chooses the best transmission strategy based on the data flow size.

The widespread adoption of MPTCP is still in its early phase and to work around this there are solutions that implement proxies in some way.

Detal et al. [19] look into the option of using MPTCP communication on a subpart of the path when it is not supported on both endpoints. This is done through the use of a middlebox proxy they named MiMBox which according to them is an efficient

protocol converter that transforms the data between TCP and MPTCP and vice versa. They compare it to other common converting options as well as plain IP forwarding and find from their evaluation that it performs better than the other converting options and is close to plain IP forwarding in most situations.

Han et al. [20] suggest a solution for use cases where no endpoints support MPTCP that uses two proxy devices. They propose to place a proxy unit next to each endpoint that transforms the communication between TCP and MPTCP and vice versa. They make use of the SOCKS protocol that is described briefly in subsection 2.6 for the conversion. Additionally, they introduce a routing algorithm they name Proxy Location Selected (PLS) algorithm that finds the best routing option when selecting proxies by using a heartbeat mechanism.

Numerous can see the potential of using MPTCP and from that there are many papers that perform evaluations in different scenarios.

A paper evaluating the performance of MPTCP in the context of cloud workloads in data centres is written by Chaufournier et al. [21] (including Ahmed, our supervisor from Chalmers). More specifically, they investigate the performance of MPTCP in uncongested networks, networks that use multi-tenancy as well as congested networks and networks with link failures. They perform memory-intensive and disk-intensive measurements in private clouds as well as briefly measure throughput performance in public clouds. From their evaluation, they find that the use of MPTCP can bring benefits such as bandwidth improvements, improved resilience and congestion avoidance in some circumstances depending on the bottleneck of the network.

Szilagyi et al. [22] have written a paper where they make a comparison between MPTCP and the more flexible MPT-GRE which instead uses UDP through a GRE tunnel. MPT-GRE is a network layer protocol while MPTCP works in the network layer, it is also not restricted to TCP communication and works with both IPv4 and IPv6. From their evaluation, they found that MPTCP slightly outperforms MPT-GRE but argued that MPT-GRE brings other benefits, mostly in terms of flexibility.

3

Methods

In this chapter, first, the choices made during the thesis work are described and motivated. Following it, the different evaluation architectures are described, and the evaluation metrics are discussed as to why they are chosen and how they are measured. Last, the implementation steps are described.

3.1 Evaluation Architecture

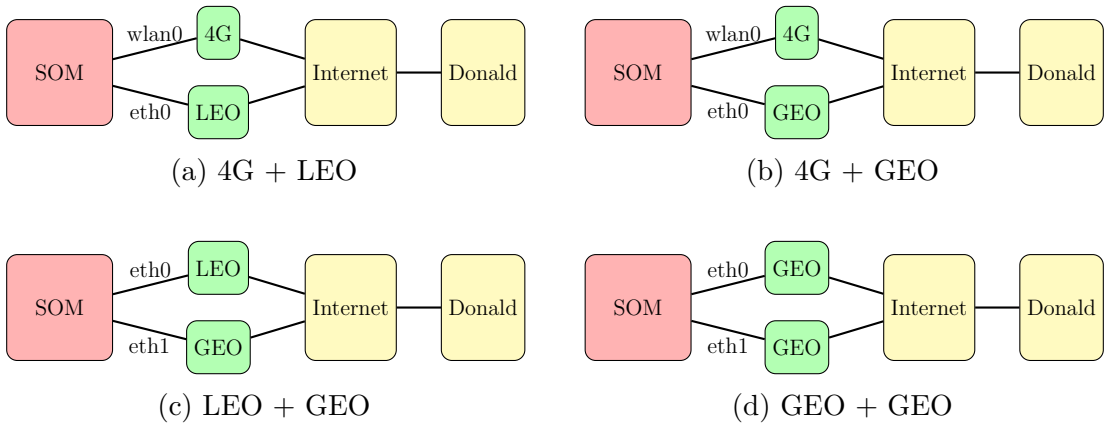
In order to carry out the evaluation, which in practice is to measure the network performance between two endpoints, the two endpoints need to be connected. First and foremost, the two endpoints are the SOM (system on module) running the custom Linux distribution made with the Yocto project, and a VPS (virtual private server) named Donald running Ubuntu 22.04 LTS (GNU/Linux 5.15.0-71-generic x86_64).

The evaluation will be carried out using five different architectures with the common element being that the SOM and Donald are the endpoints. The differences, however, lie in the part of communication between the SOM and Donald, specifically the way the SOM is connected to the internet. As described in section 2.1 the biggest bottleneck in satellite internet communication is the segment of communication from Earth back to Earth via the satellite, or in other words the communication between a satellite terminal at the client's end and the internet. So, in order to evaluate the steps towards the solution to the bottleneck problem the variable between the different architectures is made up of the different satellite types GEO and LEO, as well as mobile cellular 4G.

Combinations of the above-mentioned means of internet access make up four out of the five evaluation architectures and are illustrated in Figure 3.1. In addition to those four architectures, the fifth one is made to be a more controlled environment in contrast to the ever-changing conditions of the GEO, LEO and 4G alternatives. This is made possible with the introduction of two middleboxes running a software called traffic control.

3.1.1 GEO: Satcube Ku

The GEO connection is made with a Satcube Ku terminal via a wired Ethernet connection from its client port. This connection is usually able to maintain a



(a) 4G + LEO

(b) 4G + GEO

(c) LEO + GEO

(d) GEO + GEO

Figure 3.1: Four evaluation architectures with different combinations of LEO, GEO and 4G connections.

download throughput of 10 Mbits/s and an upload throughput of 3 Mbits/s, with a round-trip time between 500 and 1000 ms. It uses the same satellite and does not require any active tracking because the satellite used is GEO stationary.

3.1.2 LEO: Starlink

The LEO connection is made with a Starlink which with an Ethernet adapter enables a wired connection between the SOM and Starlink router. Speeds vary quite much with download throughput noted between 200 and 40 Mbits/s and upload throughput usually around 10 to 30 Mbits/s. The latency is around 40 ms and the Starlink requires the active tracking of the satellite and might break the connection when switching to another satellite.

3.1.3 4G: Cellular 4G

The 4G connection is accessed by the SOM via its wireless interface wlan0 connecting to a WiFi 5 GHz smartphone hotspot which in turn is connected to the internet via cellular 4G. Network performance varies a lot due to different amounts of congestion in the cellular network. The usual download speed is around 20 Mbits/s, upload speed 10 Mbits/s with a latency of around 40 ms. The method of using WiFi to connect to the SOM implies a less stable connection compared to a wired connection using Ethernet. However, with the wireless devices being close to each other without any notable obstacle or interference the bottleneck should not be the WiFi connection. Especially with the fact that the network performance directly between the phone and the internet is not close to saturating the capability of the 5 GHz WiFi connection.

3.1.4 Traffic Control

Because the network conditions are ever-changing for the above-mentioned connection types (GEO, LEO, and 4G) another more controlled environment makes up the last evaluation architecture.

As shown in Figure 3.2 two middleboxes are used between the SOM and an office internet connection in order to limit the upload and download speeds for the two links. Moreover, it is also possible to increase the latency for the network traffic going through the two links. The middleboxes themselves have two Ethernet ports each which are used to connect the SOM and a network switch, again just as it is shown in Figure 3.2.

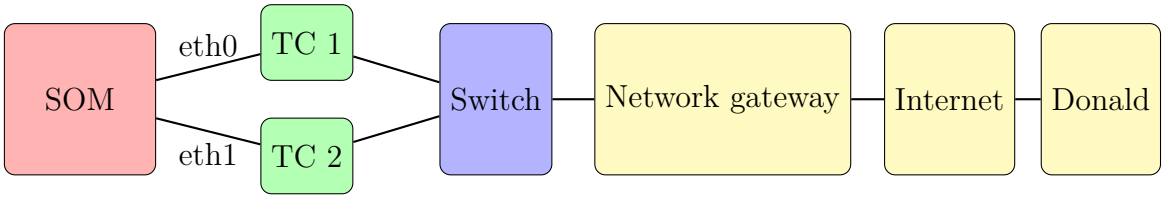


Figure 3.2: The fifth evaluation architecture. The TC 1 and 2 middleboxes are used to limit the throughput and increase the latency between SOM and the internet.

Tc, short for traffic control, is the name of the software that is run on these middleboxes, and it is a tool that can be used to control various aspects of the network traffic on a computer running Linux. In our evaluation, we have used it both to limit the throughput and to increase the latency of a link in order to simulate different link types. The different link types that are simulated in this architecture are listed in Table 3.1.

Table 3.1: The different configurations for the traffic control setting.
Format: Download (Mb/s)/ Upload (Mb/s)/ Latency (ms)

Traffic Control			
Comment	**Protocol**	**Link 1**	**Link 2**
Link reference - 10 / 10 / 10	TCP	10 / 10 / 10	-
Link reference - 50 / 50 / 10	TCP	50 / 50 / 10	-
Homogeneous - 10 / 10 / 10	MPTCP	10 / 10 / 10	10 / 10 / 10
Homogeneous - 50 / 50 / 10	MPTCP	50 / 50 / 10	50 / 50 / 10
Heterogeneous - LEO/GEO mimic	MPTCP	150 / 10 / 40	10 / 5 /500
Heterogeneous - Latency difference	MPTCP	10 / 10 / 10	10 / 10 / 50
Heterogeneous - Throughput difference	MPTCP	10 / 10 / 10	50 / 50 / 10

3.2 Performance Evaluation

Evaluations are made in different iterations of the implementation. Every evaluation measures combinations of different link types and also each link on its own using TCP for comparison. The link types that are used are GEO and LEO satellite links as well as a 4G cellular link, additionally, a stable high-speed fibre connection has been used to modulate different network environments using the network tool **tc** (traffic control).

Two different help scripts were used to carry out the evaluation. The first one uses **sockperf** to measure latency, packet drop rate and jitter while the second script uses **iperf3** to measure bandwidth. Both scripts also use **sar** to measure the CPU, RAM and network card utilisation during the whole test.

An overview of the methods used to evaluate each metric is shown in Table 3.2. And a more detailed description of the different methods and software used in the scripts is found in the following subsections below in this chapter.

Table 3.2: Method used for each metric

Metric	**Method**
Bandwidth	iperf3
Latency	sockperf under-load
CPU utilisation	sar -u
Memory utilisation	sar -r
Network card usage distribution	sar -n
Power consumption	Multimeter
Packet drop rate	sockperf under-load
Jitter	Calculated from sockperf under-load timing logs

3.2.1 Sockperf

Sockperf is an open-source network benchmark utility developed by Mellanox. It can be used to measure various network performance metrics like latency, RX/TX bandwidth and packet loss, using either UDP or TCP. One of the reasons this program is used in the evaluation is the fact that it can measure latency using TCP, which when run with mptcpize instead uses MPTCP. Another reason is that it provides logs that contain all the transmit and receive times for all packets which can be used to calculate the jitter.

Sockperf provides two different methods for measuring latency, ping-pong and under-load. The ping-pong mode only sends one TCP packet at a time and calculates the latency (RTT / 2), while the under-load mode creates a flow of packets and measures the timing for a fraction of the packages in order to simulate the performance under a more realistic network condition. In the evaluation, the under-load mode is used, not only because it simulates a more realistic link utilisation, but also because when using two links with different latency the ordering quality of TCP can cause increased latency if waiting for a packet that took the longer route.

The flag `-tcp-avoid-nodelay` is used because the Linux kernel version(s) tested does not have that socket option implemented for MPTCP. The last part of the client command, namely `&> ${raw_path}/latency`, is used to redirect **stderr** and **stdout**, which prints and summarises the result, to a file. Both commands, on the server and client, include the `mptcpize run` prefix in order to make sockperf use MPTCP sockets.

Server

```
mptcpize run sockperf server -p ${port} --tcp --tcp-avoid-nodelay
```

Client

```
mptcpize run sockperf under-load -i ${ip} -p ${port} --tcp
    --tcp-avoid-nodelay -t $((duration+5))
    --full-log ${raw_path}/jitter &> ${raw_path}/latency
```

Explanation of used sockperf options:

- `server`: Run sockperf in server mode
- `under-load`: Run sockperf in under-load mode
- `-i <ip_address>`: Specify the server ip address
- `-p <port>`: Specify the server port
- `-tcp`: Use TCP, rather than UDP
- `-tcp-avoid-nodelay`: Do not use the TCP_NODELAY socket option
- `-t <duration>`: Specify the test duration
- `-full-log <log_path>`: Write full log to file

3.2.2 Iperf3

Iperf3 is a program developed by ESnet / Lawrence Berkeley National Laboratory that tests a network's maximum bandwidth performance. It is a tool that is used to measure the maximum bandwidth between two hosts in a network. The program can be run in either client mode or server mode. In order to make use of it both modes are needed and thus the network performance is measured between the client and server.

By default the upload speed from the client to the server is measured, but with the `-R` (reverse) flag the client can measure its download speed from the server.

Server

```
mptcpize run iperf3 -s
```

Client

```
mptcpize run iperf3 -c ${ip} -p ${port} -t $((duration+5))
    --logfile ${raw_path}/iperf
```

Explanation of used iperf3 options:

- `-s`: Run iperf3 as a server
- `-c <ip_address>`: Run iperf3 as a client towards server with specified <ip_address>
- `-p <port>`: Specify the server port
- `-t <duration>`: Specify the test duration
- `-R`: Reverse mode. Measure download from server to client.
- `-logfile <log_path>`: Write output log to file

3.2.3 Sar

Sar is a software that can be used to record and observe various activity counters in the Linux operating system. In the evaluation, it is used to measure CPU utilisation, memory usage and network interface card utilisation. The command has two parameters that decide the interval between the measurements and another one that decides how many measurements should be recorded. In the three commands below the interval is 1 s and the number of measurements is set from a common variable in the script.

The output of each command used is saved to a file.

Client

```
sar -u 1 ${duration} > ${raw_path}/cpu
sar -r 1 ${duration} > ${raw_path}/ram
sar -n DEV --iface=eth0,eth1,wlan0 1 ${duration} > ${raw_path}/nic
```

Explanation of used sar options:

- `-u`: Monitor CPU
- `-r`: Monitor RAM
- `-n DEV`: Monitor network devices
- `-iface=<iface_list>`: Specify the wanted network interfaces

3.2.4 Multimeter

In order to measure the power consumption of the system on module (SOM) the current is measured by wiring a multimeter in serial as shown in Figure 3.3. The multimeter has the feature to record the current over a period of time and display the lowest and highest recorded values as well as the average current.

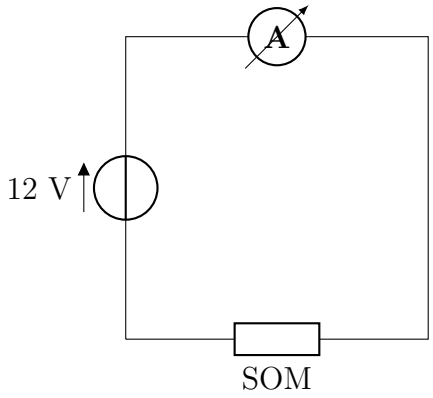


Figure 3.3: Circuit diagram showing how the voltage generator, the ammeter (multimeter) measuring current and SOM are connected in serial.

3.3 Implementation

This section is about the details of the implementation of the work that has been done throughout this thesis. To begin with, the motivations for some of the choices made are described. After that details about different steps in the implementation are described.

3.3.1 Motivation of Choices

Early in the project, it was decided that the protocol to evaluate would be MPTCP. As it is already implemented in the Linux kernel it is relatively easy to set up compared to more customised implementations. Furthermore, for a future potential product for Satcube, it is good to use a protocol that easiest allows for wide adoption by servers and "content providers".

The choice to use the Yocto project over OpenWrt was made first and foremost in order to be compatible with Satcube's development environment, however, another enhancement that comes with Yocto is the ease of adding and controlling what software is included in the OS. This allows for example for better and more thorough optimisations.

As GRE encapsulation was not able to function using the Starlink connection we opted for a more general VPN solution, WireGuard. It is included in the Linux kernel and uses UDP to encapsulate the traffic which is positive as TCP encapsulation would risk TCP meltdown to occur. Moreover, WireGuard is a very simple solution that requires a minimal amount of configuration to get up and running and with some additional routing rules all network interfaces can have their own endpoint in the VPN.

3.3.2 Implementation Iterations

- Iteration 1: This first evaluation is meant to be the most basic setup that makes MPTCP work with multiple subflows. The Linux kernel version is 5.10.
- Iteration 2: Linux kernel is upgraded to version 5.15, which adds congestion control to MPTCP sockets. The plan was to test this on a GEO satellite terminal with an external modem that did not strip MPTCP headers. But due to various reasons, we could not manage to get that modem/connection working at all.
- Iteration 3: Use WireGuard in order to use a normal GEO modem and avoid middlebox interference.

3.3.3 Kernel Build Configurations

As mentioned in the background chapter today's development of MPTCP is made directly into the Linux kernel. This means that kernel versions from 5.6 and onward include an implementation of MPTCP. However, if using a precompiled Linux kernel

the feature might not be included by default. It is the kernel configuration that determines if the feature should be included or not, more specifically it is the following three entries that enable various MPTCP-related features, **CONFIG_MPTCP**, **CONFIG_MPTCP_IPV6** and **CONFIG_INET_MPTCP_DIAG**. The first two are for MPTCP capabilities for IPv4 and IPv6 respectively and the third is to be able to monitor MPTCP sockets for debugging purposes.

3.3.4 Kernel Runtime Configurations

If the MPTCP features are included in the kernel it might or might not be enabled by default. This is because there exists a kernel parameter that decides if MPTCP is enabled or not. This kernel parameter can be changed at runtime by either changing a value in a specific file in the /proc/sys/ subfolder or by using a program **sysctl** which is a more user-friendly interface that under the hood achieves exactly what the first option does. If using sysctl, the parameter that needs to be changed is **net.mptcp.enabled** which must be set to **1**. Or if changing the sysfs directly, change the contents of the file **/proc/sys/net/mptcp/enabled** to **1**.

3.3.5 Convert Sockets from TCP to MPTCP

Most programs that already exist naturally use a normal TCP socket in their source code and are compiled to use that. In order to make use of the MPTCP protocol instead the program needs to use an MPTCP socket instead. The first solution that might come to mind is to simply change the socket type used in the source code for each program and then recompile it. However, this solution would require a lot of work and would increase the code complexity in order to keep support for systems without MPTCP. Instead, there exists another more convenient solution for the problem, something called **mptcpize**. It is a program that can be used to wrap existing programs to create and use MPTCP sockets instead of TCP sockets. This is possible due to the Linux environment variable **LD_PRELOAD** which is used to load libraries before any other library. This makes it possible for users to overwrite specific functions in the C runtime library with modified versions, which is precisely what **mptcpize** does. The function that is used to create sockets, **int socket(int family, int type, int protocol)**, is replaced with one that, if the socket protocol is TCP, instead creates a socket with the MPTCP protocol.

So, to use mptcpize a simple prefix is added to the original command:

```
mptcpize run <program> [program arguments]
```

3.3.6 Allow for Multiple Subflows

With MPTCP available and enabled in the kernel, and the applications changed to use MPTCP sockets it is now possible for programs to utilise the MPTCP protocol. But, in order to fully utilise the advantages of MPTCP some further configurations need to be made in order to make use of the multiple network interfaces.

As described in the background the MPTCP protocol makes use of subflows. A

subflow is like a normal TCP connection between two endpoints, one on your local machine and the other one on the remote host. So, in order for the kernel to create multiple subflows it needs to be informed of the information that defines the local endpoints. This can be done in various ways, with **NetworkManager** version 1.40 or newer, with **mptcpd**, or with **ip mptcp** together with **ip route**. The **NetworkManager** and **mptcpd** approaches require less manual labour and will dynamically adapt to network changes while the **ip mptcp** + **ip rule** approach is more manual but highlights what the different parts of the kernel do. Because of its educational nature, the latter approach will be explained.

The command **ip mptcp** allows for adding, changing and removing endpoints as well as setting different limits and monitoring mptcp subflow activity. The command **ip rule** is used to manipulate rules for the routing policy. While the **ip mptcp** command informs the kernel about the endpoint, i.e. the sender of the TCP packets in a subflow, the **ip rule** command makes sure that the packets from a specific endpoint actually is routed through that network interface.

This example is from the Multipath-TCP website where the following two network interfaces eth0 and eth1 exist.

```
eth0:
  IP-Address: 10.1.1.2,
  Subnet-Mask: 255.255.255.0,
  Gateway: 10.1.1.1

eth1:
  IP-Address: 10.1.2.2,
  Subnet-Mask: 255.255.255.0,
  Gateway: 10.1.2.1

ip mptcp limits set subflow 2 add_addr_accepted 2
ip mptcp endpoint add 10.1.1.2 dev eth0 subflow
ip mptcp endpoint add 10.1.2.2 dev eth1 subflow
```

This creates two different routing tables, that we use based on the source address

```
ip rule add from 10.1.1.2 table 1
ip rule add from 10.1.2.2 table 2
```

Configure the two different routing tables

```
ip route add 10.1.1.0/24 dev eth0 scope link table 1
ip route add default via 10.1.1.1 dev eth0 table 1

ip route add 10.1.2.0/24 dev eth1 scope link table 2
ip route add default via 10.1.2.1 dev eth1 table 2
```

Default route for the selection process of normal internet traffic

```
ip route add default scope global nexthop via 10.1.1.1 dev eth0
```

3.3.7 Encapsulation with WireGuard

With the above configurations, it is possible to create MPTCP connections with multiple subflows. However, to solve the problem of TCP headers being removed by middleboxes in the GEO satellite network, some kind of encapsulation method needs to be configured.

WireGuard is a "fast, modern, secure VPN tunnel" which is a simple way to encapsulate the traffic and fulfil the encapsulation requirement.

A WireGuard tunnel needs to be configured on both the client and the server. One endpoint is configured on the testing server. However, on the client side, one WireGuard interface needs to be added for each network interface which will be used in the MPTCP connection. With some additional routing rules, the traffic from each WireGuard interface is routed through its corresponding real network interface.

3.3.8 Traffic Control

Tc, or traffic control, is a tool that can be used to control various aspects of the network traffic on a computer running Linux. In our evaluation, we have used it both to limit the throughput and to increase the latency of a link in order to simulate different link types.

The TC 1 and 2 "middleboxes" are shown in Figure 3.2 and they are connected to the two Ethernet ports on the SOM. In order to limit both upload and download speed `tc` is used to limit the outgoing traffic on each of the two Ethernet interfaces. Limiting the outgoing traffic towards the SOM will limit the download from the SOM:s point of view. Similarly, limiting the outgoing traffic towards the internet will from the SOM:s point of view limit its upload speed.

To limit the outgoing traffic and increase the latency of a network interface (eth0) the following commands can be used:

```
tc qdisc del dev eth0 root
```

```
tc qdisc replace dev eth0 root handle 1:  tbf rate 30Mbits latency 100ms
burst 1540
```

```
tc qdisc replace dev eth0 parent 1:1 handle 10:  netem delay 20ms
```

4

Results

In this chapter, the results of the evaluation study are highlighted. First, the impact of using versus not using WireGuard is shown where the results are from using the traffic control setting. Then the results from the measurements of 4G/LEO, 4G/GEO, LEO/GEO and GEO/GEO will be shown. Finally, a graph visualising the throughput aggregation benefits when using different combinations of links can be seen at the end of the chapter.

4.1 The impact of using WireGuard

There are various metrics that are affected by the use of encapsulation, in our case the VPN WireGuard. Throughput is neutrally affected because of the encapsulation, that is, the new packet headers take up data that otherwise could be used for the payload. The impact of the effective throughput is shown in Figure 4.1.

With the additional tasks to be performed, including the actual encapsulation, encryption, and decryption of the data sent and received, both memory usage and processor utilisation are affected. Moreover, likely as an effect of higher processor utilisation the power consumption is also affected. These metrics are shown in Figure 4.2, 4.3 and 4.4 respectively.

With these additional tasks to be performed the latency naturally is slightly increased which can be seen in Figure 4.5.

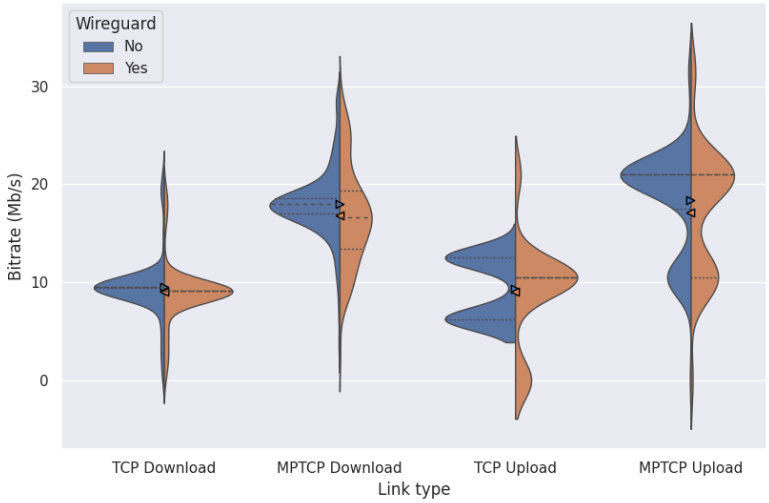


Figure 4.1: Throughput comparison between using and not using WireGuard on traffic control links configured to 10 Mb/s up and down. The median and quarter percentiles are shown as dashed respectively dotted lines inside the violin graphs. The mean throughput is marked with a coloured triangle-shaped marker.

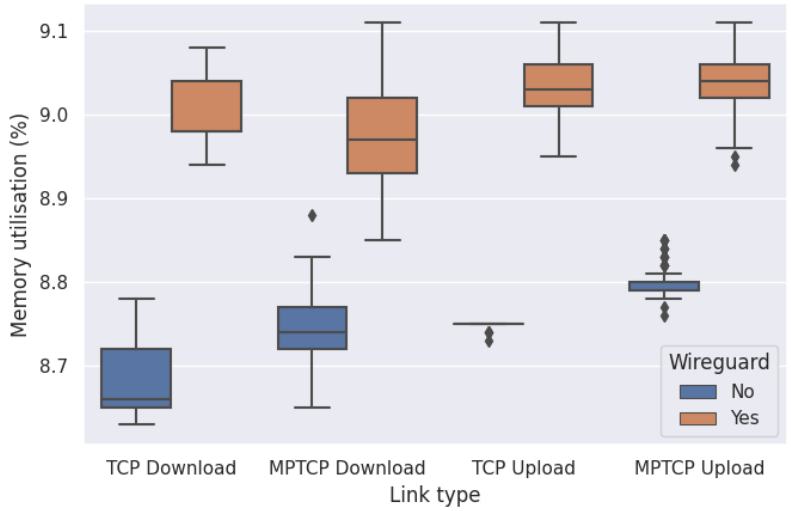


Figure 4.2: Comparison of memory usage between using and not using WireGuard on traffic control links configured to 10 Mb/s up and down.

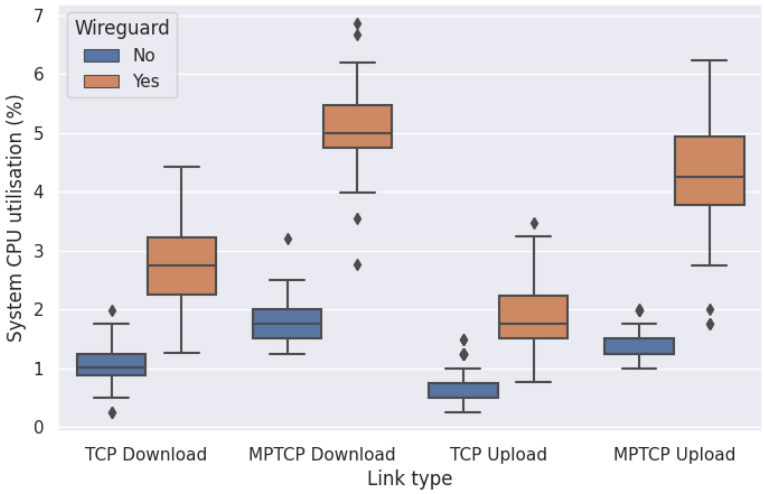


Figure 4.3: Comparison of system CPU utilisation between using and not using WireGuard on traffic control links configured to 10 Mb/s up and down.

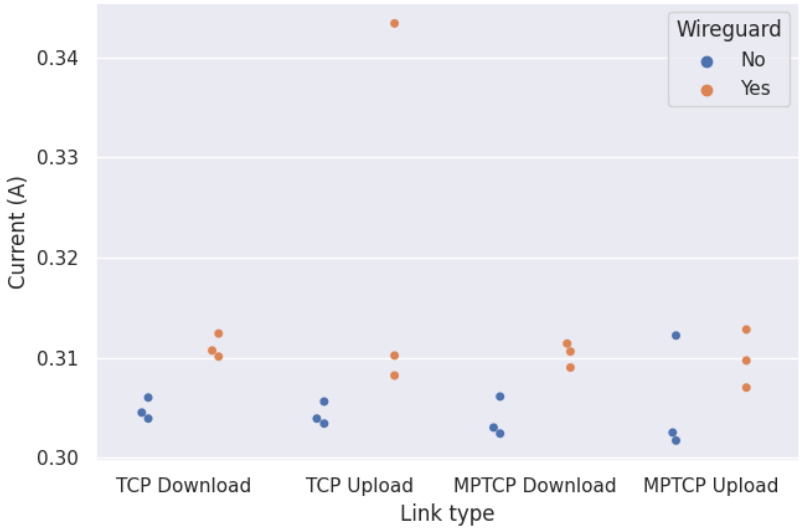


Figure 4.4: Comparison of power consumption between using and not using WireGuard on traffic control links configured to 10 Mb/s up and down. The three points are the maximum, average and minimum recorded value. The current is linear to the power consumption because the voltage is constant at 12 V. The maximum recorded current for TCP Upload with WireGuard is clearly an outlier, its cause could be instrumental error or human error when documenting the value.

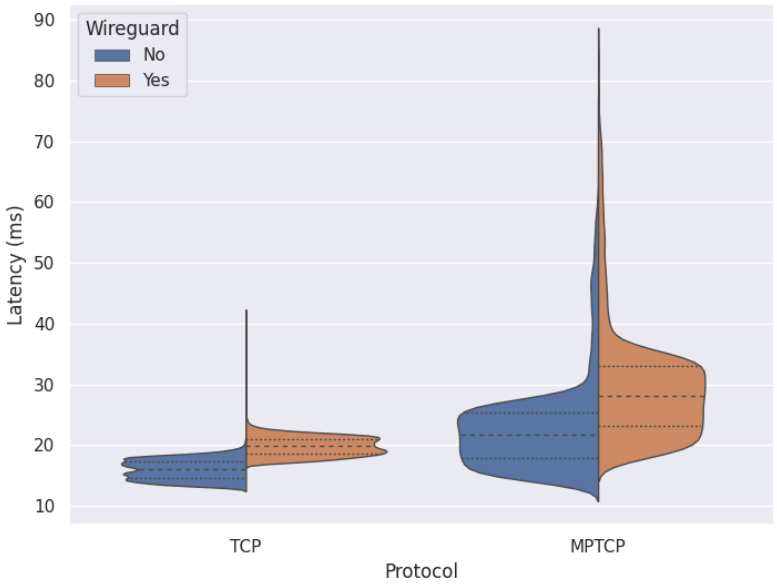


Figure 4.5: Comparison of latency between using and not using WireGuard on traffic control links configured to 10 Mb/s up and down. The median and quarter percentiles are shown as dashed respectively dotted lines inside the violin graphs.

4.2 4G/LEO

Figures 4.6 to 4.9 detail the measurements of combining a 4G cellular link together with an LEO satellite link using MPTCP. For a comparison to the MPTCP aggregation, both links are also measured individually using TCP communication.

Figure 4.6 shows that the average throughput of the aggregated links for both download and upload does not surpass the best single TCP link.

However, Figure 4.7 gives more insight into how the throughput was distributed between the different interfaces, and from that it is visible that in some of the tests the throughput aggregated more successfully than in the average case.

Figure 4.8 and 4.9 show latency as well as the number of dropped packets when measuring the latency. Both the number of dropped packets and the latency increased when using MPTCP.

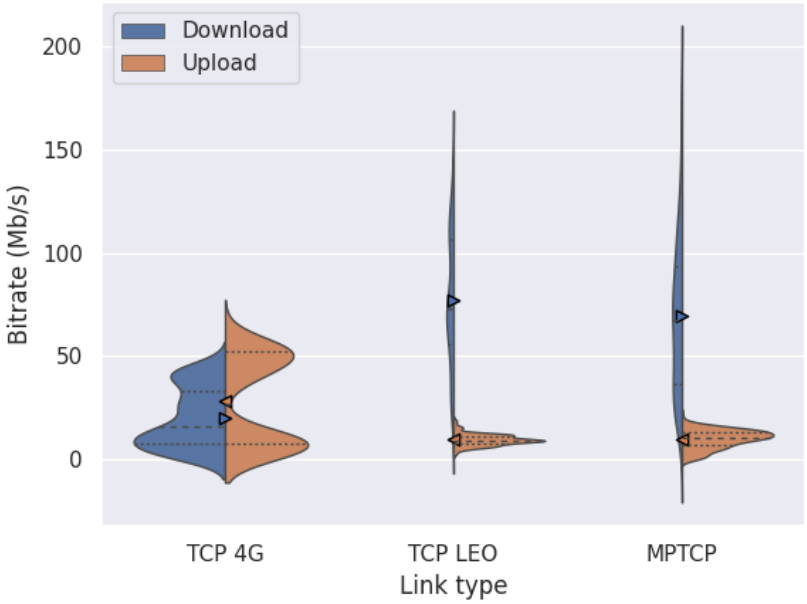
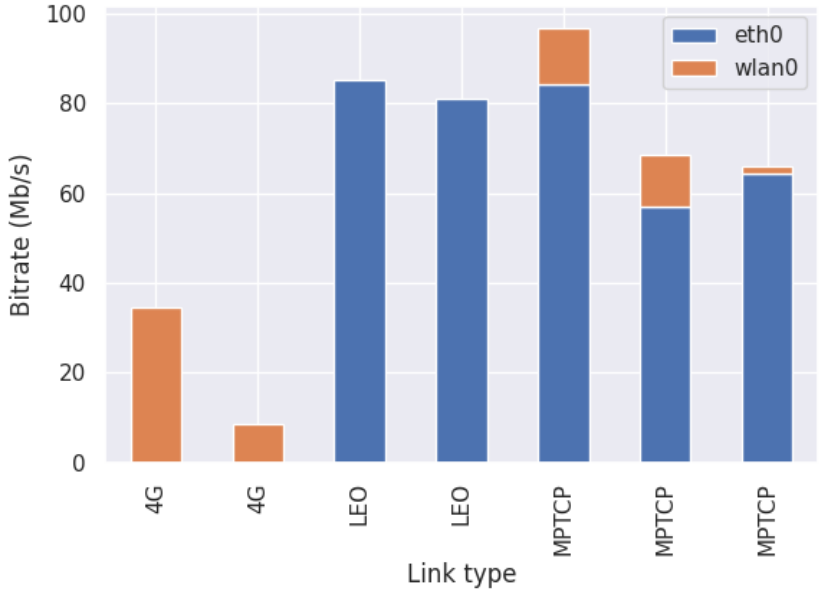
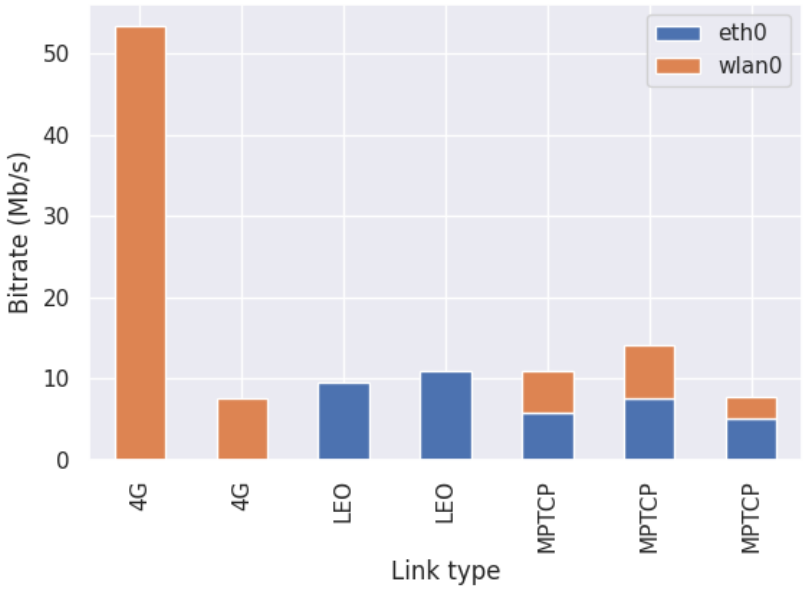


Figure 4.6: Average throughput when measuring download respectively upload using TCP on one 4G link, one LEO link, and using MPTCP with both links. The median and quarter percentiles are shown as dashed respectively dotted lines inside the violin graphs. The mean throughput is marked with a coloured triangle-shaped marker.



(a) Download



(b) Upload

Figure 4.7: Average throughput through each network interface during download (a) and upload (b) for each test. The four first bars are using TCP with only 4G respectively LEO. The last three bars use MPTCP with both links.

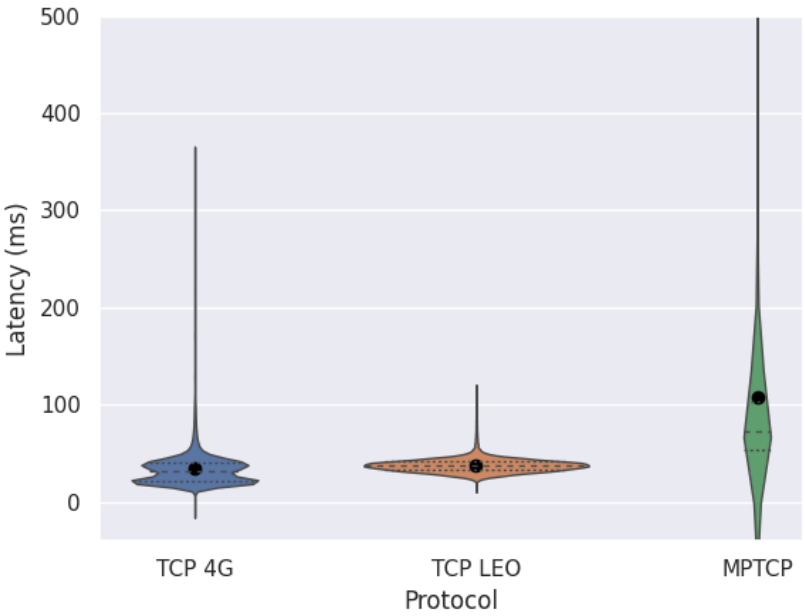


Figure 4.8: Latency comparison between TCP with 4G and LEO, and MPTCP using both links (all also using WireGuard). The median and quarter percentiles are shown as dashed respectively dotted lines inside the violin graphs. The black dot on each graph represents the average measured latency.

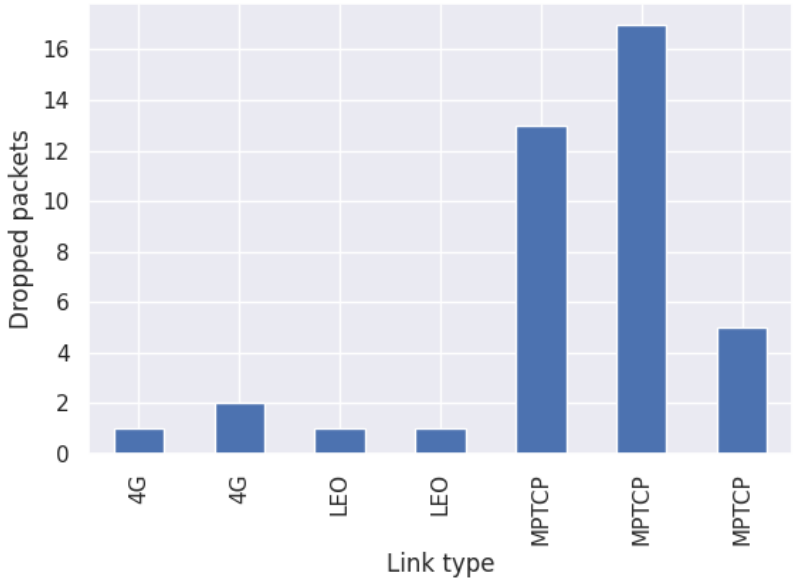


Figure 4.9: Comparison of the number of dropped packets between TCP with 4G and LEO, and MPTCP using both links (all also using WireGuard).

4.3 4G/GEO

Figures 4.10 to 4.13 showcase the performance of combining a GEO satellite link with a 4G cellular link using MPTCP, for a comparison to the MPTCP aggregation, both links are also measured individually using TCP communication.

Figure 4.10 shows that the average throughput of the aggregated MPTCP link is better than the one for TCP GEO. However by looking at Figure 4.11 it is both clear that the throughput of the 4G link varies a lot, and that in some cases the MPTCP link performs almost as well as the two links combined.

Figure 4.11 also shows that the throughput of the more stable link, GEO, is affected more negatively during download compared to during upload.

Further, Figure 4.12 and 4.13 shows for instance that the latency when using MPTCP is much bigger compared to the TCP links. It also shows that the number of dropped packets when measuring the latency is much higher for GEO compared to 4G, and that of MPTCP is even higher than GEO.

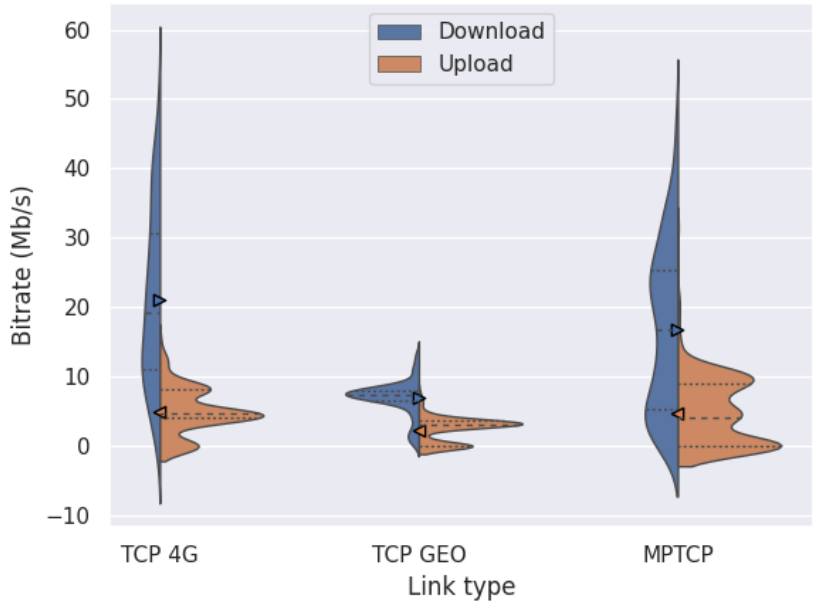
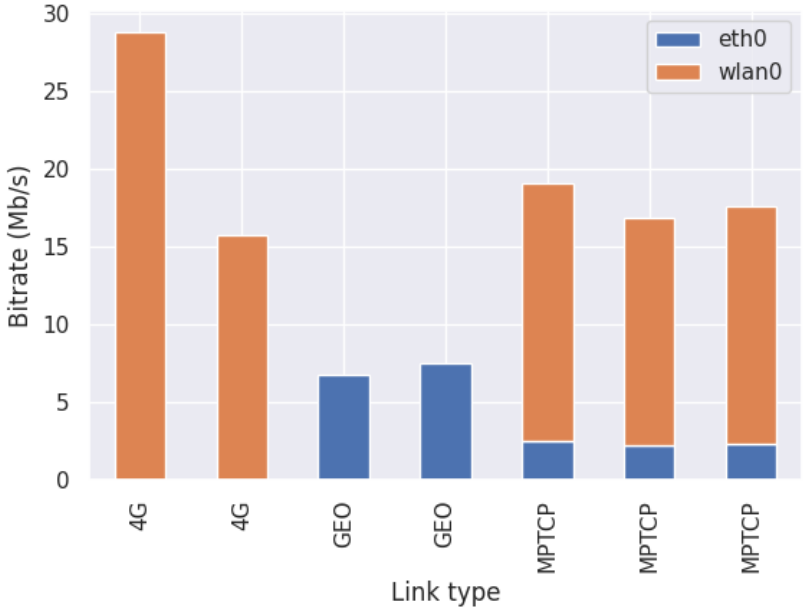
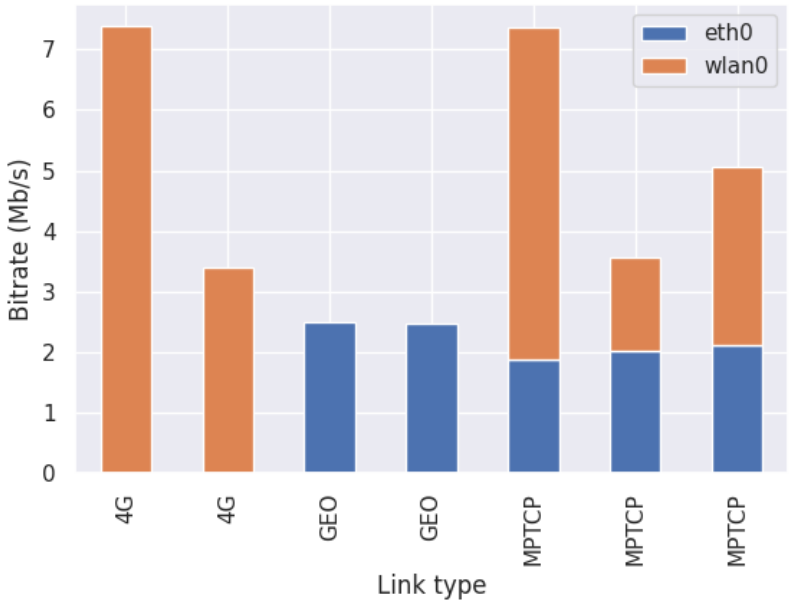


Figure 4.10: Throughput when measuring download respectively upload using TCP on one 4G link, one GEO link, and using MPTCP with both links. The median and quarter percentiles are shown as dashed respectively dotted lines inside the violin graphs. The mean throughput is marked with a coloured triangle-shaped marker.



(a) Download



(b) Upload

Figure 4.11: Average throughput through each network interface during download (a) and upload (b). The four first bars are using TCP with only 4G respectively GEO. The last three bars use MPTCP with both links.

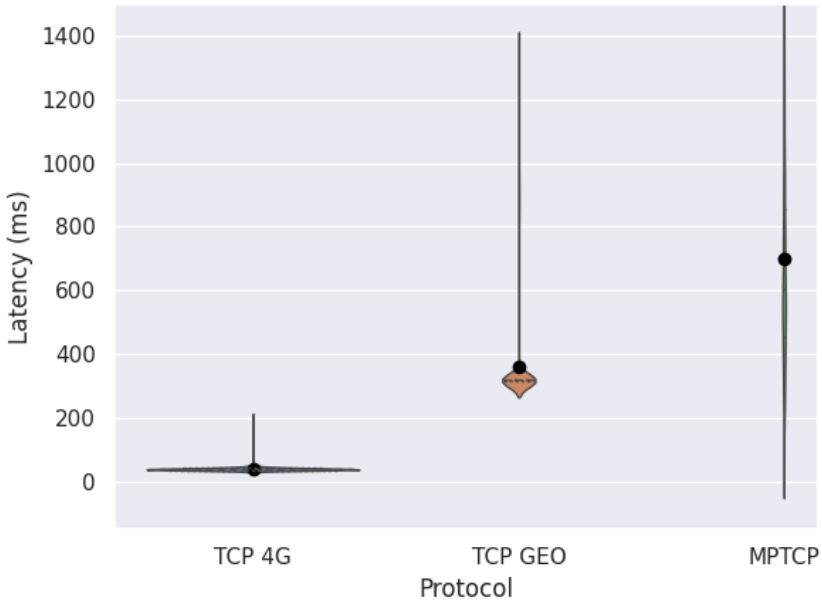


Figure 4.12: Latency comparison between TCP with 4G and GEO, and MPTCP using both links (all also using WireGuard). The median and quarter percentiles are shown as dashed respectively dotted lines inside the violin graphs. The black dot on each graph represents the average measured latency.

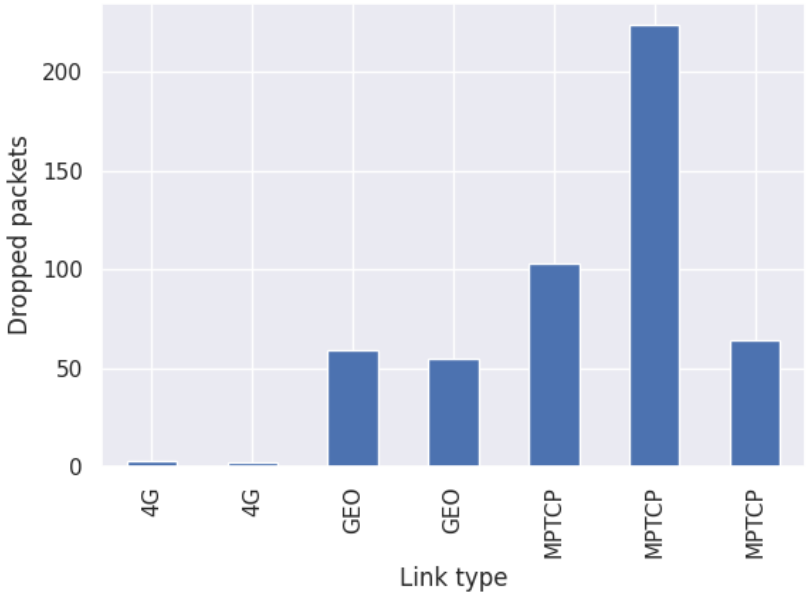


Figure 4.13: Comparison of the number of dropped packets between TCP with 4G and GEO, and MPTCP using both links (all also using WireGuard).

4.4 LEO/GEO

Figures 4.14 to 4.17 present the performance results of combining a GEO satellite link with an LEO satellite link using MPTCP, for a comparison to the MPTCP aggregation, both links are also measured individually using TCP communication.

In Figure 4.14 it is shown that the MPTCP link is performing worse than the LEO link alone. Further, in Figure 4.15 it is visible that in most of the MPTCP tests neither one of the LEO or GEO links performed as well as they did alone.

Similar to the earlier results, the combining of 4G and GEO also results in higher latency and an increased number of dropped packets.

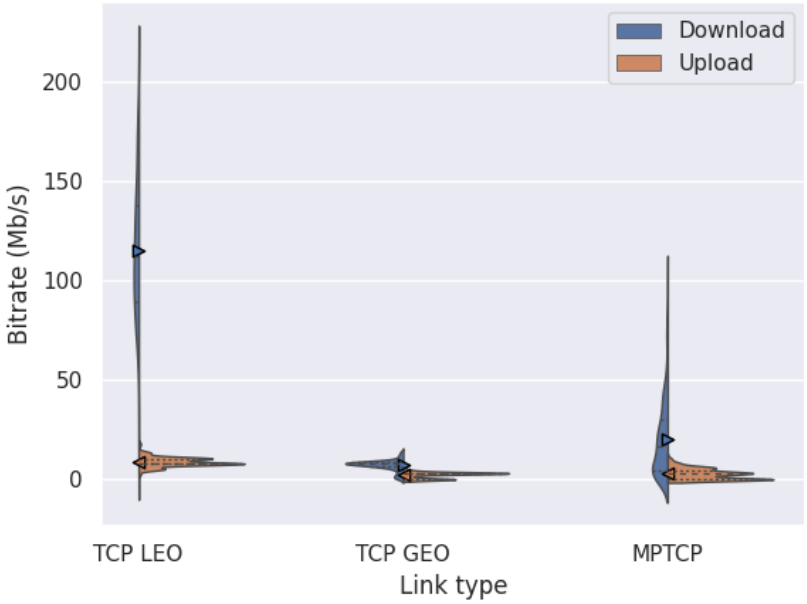
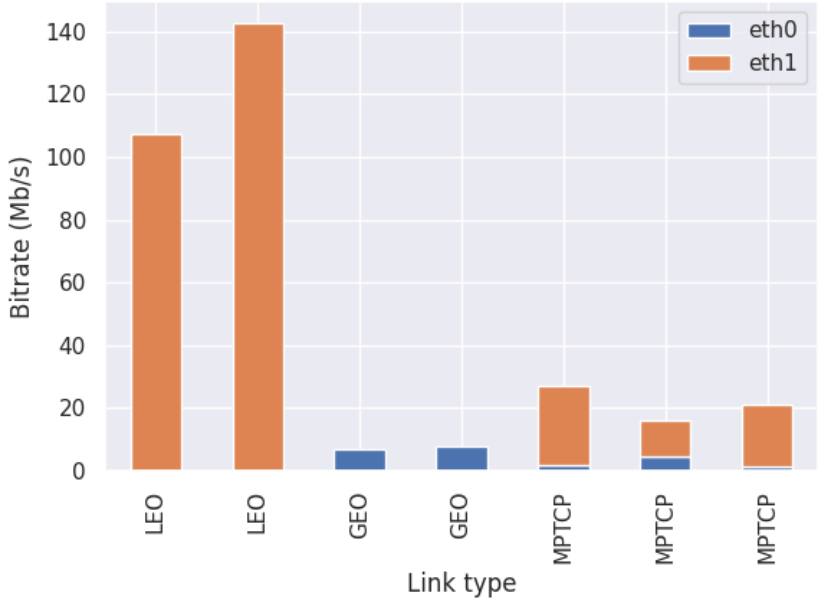
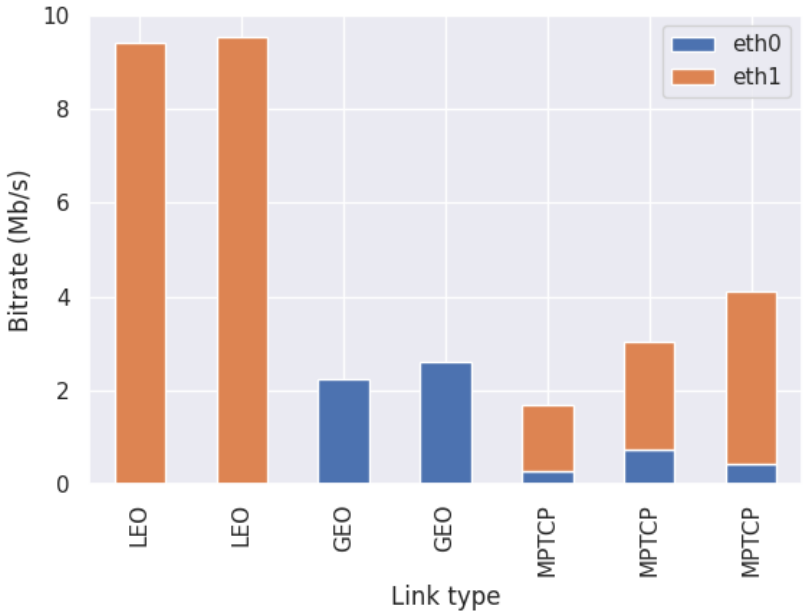


Figure 4.14: Throughput when measuring download respectively upload using TCP on one LEO link, one GEO link, and using MPTCP with both links. The median and quarter percentiles are shown as dashed respectively dotted lines inside the violin graphs. The mean throughput is marked with a coloured triangle-shaped marker.



(a) Download



(b) Upload

Figure 4.15: Average throughput through each network interface during download (a) and upload (b). The four first bars are using TCP with only LEO respectively GEO. The last three bars use MPTCP with both links.

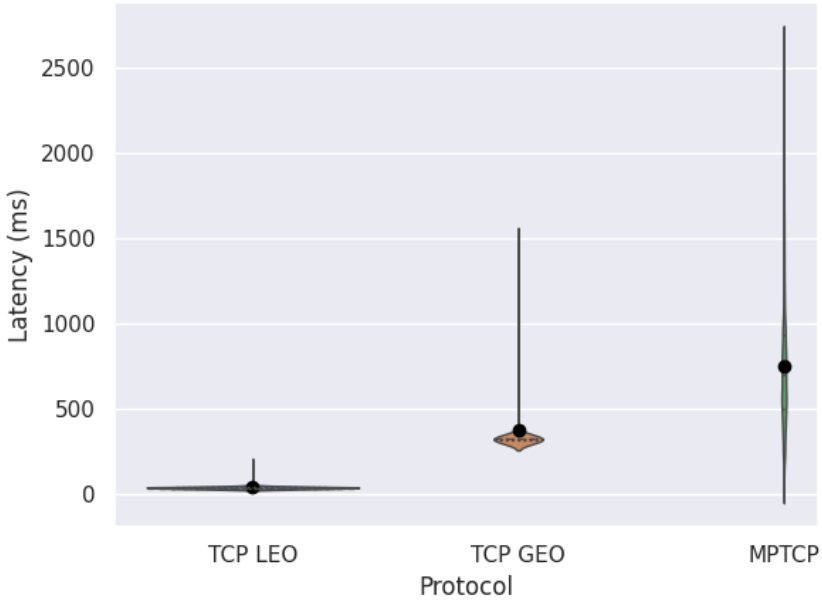


Figure 4.16: Latency comparison between TCP with LEO and GEO, and MPTCP using both links (all also using WireGuard). The median and quarter percentiles are shown as dashed respectively dotted lines inside the violin graphs. The black dot on each graph represents the average measured latency.

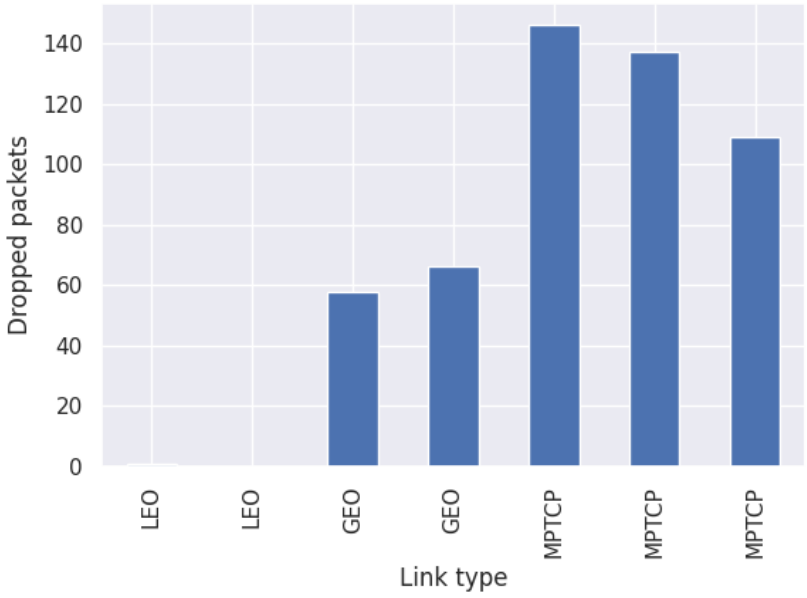


Figure 4.17: Comparison of the number of dropped packets between TCP with LEO and GEO, and MPTCP using both links (all also using WireGuard).

4.5 GEO/GEO

Figures 4.18 to 4.21 present the performance results of combining two GEO satellite links using MPTCP, to compare the MPTCP communication to, both links are also measured individually using standard TCP communication.

In Figure 4.18 and 4.19 it is shown that the upload is aggregated very nicely, while the download is performing worse than the average of the solo links alone.

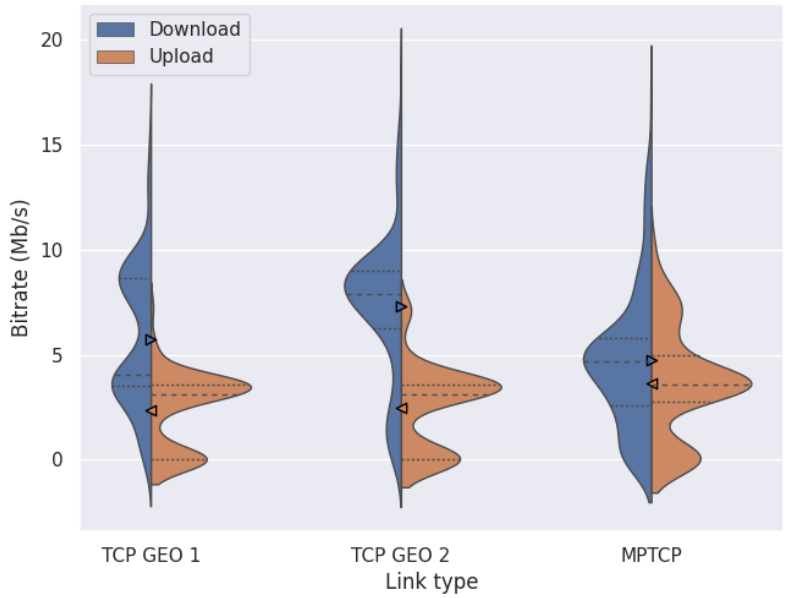
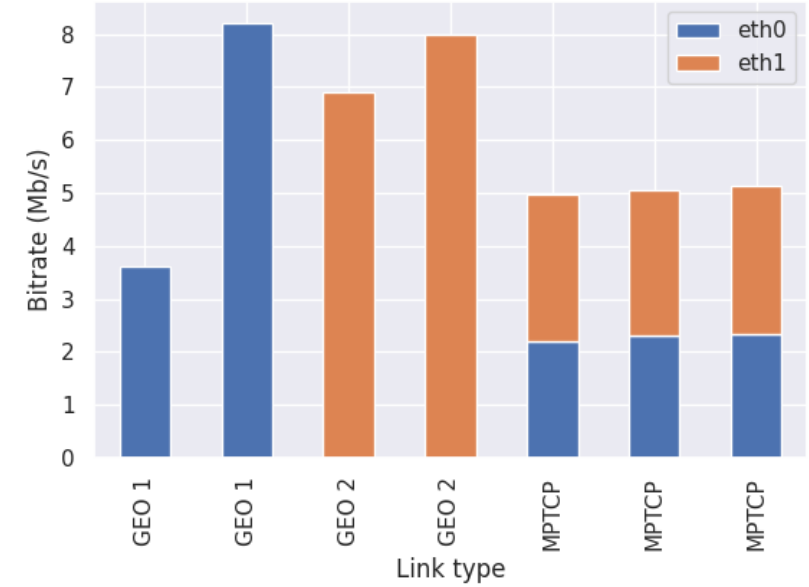
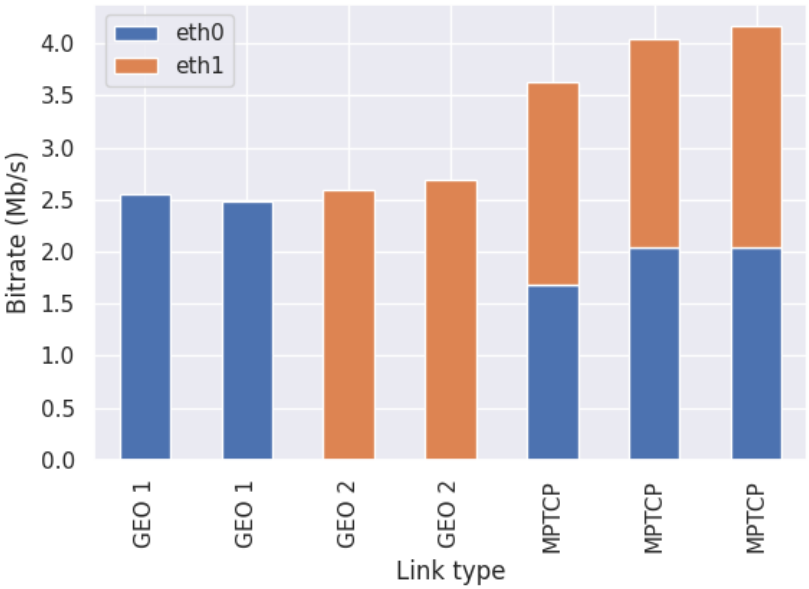


Figure 4.18: Throughput comparison between TCP and MPTCP using two GEO links (with WireGuard). The median and quarter percentiles are shown as dashed respectively dotted lines inside the violin graphs. The mean throughput is marked with a coloured triangle-shaped marker.



(a) Download



(b) Upload

Figure 4.19: Throughput through each network interface during download (a) and upload (b). The four first bars are using TCP with only GEO 1 respectively GEO 2. The last three bars use MPTCP with both links.

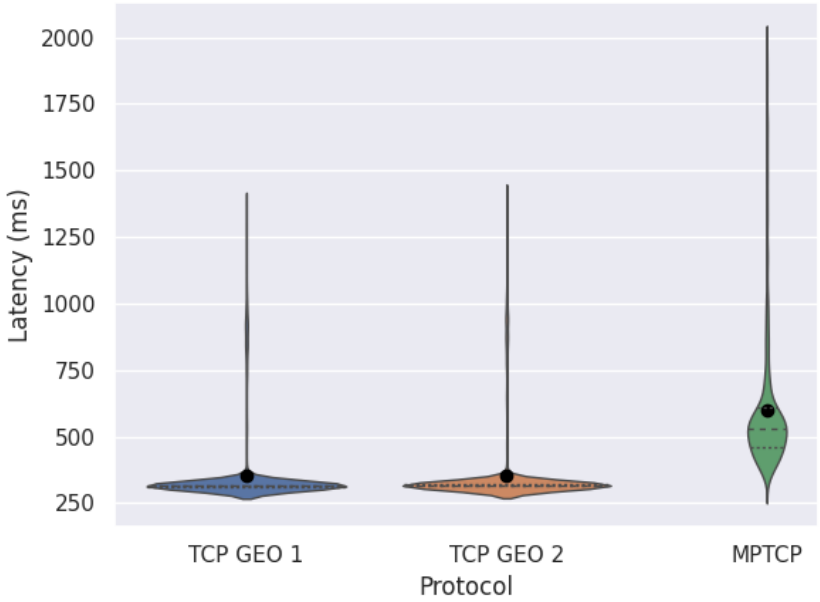


Figure 4.20: Latency comparison between TCP with GEO 1 and GEO 2 and MPTCP using both GEO links (all also using WireGuard). The median and quarter percentiles are shown as dashed respectively dotted lines inside the violin graphs.

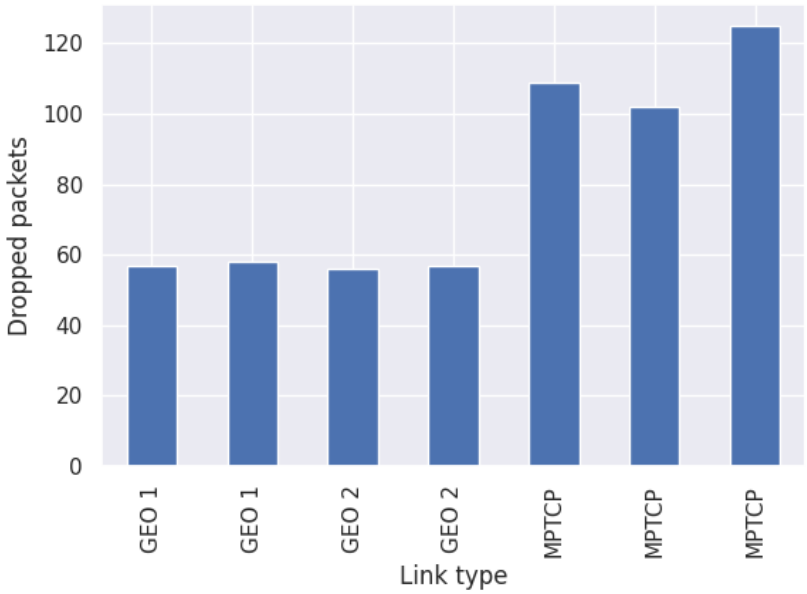


Figure 4.21: Comparison of the number of dropped packets between TCP with GEO 1 and GEO 2, and MPTCP using both GEO links (all also using WireGuard).

4.6 Summary

As a summation of the performance results of this chapter, a graph that visualises the average throughput benefits of the different combinations of link aggregation can be seen in Figure 4.22.

As a recollection from section 1.1 in the Introduction chapter 1, the aggregation benefit is defined as:

- 2: Is the perfect summation of the aggregated links throughput.
- 1: Is the throughput of the link with the highest throughput.
- 0: Is the throughput of the link with the lowest throughput.
- -1: Is a throughput of zero.

The measurements for the traffic controlled fibre connections (**TC 10/10** and **TC 50/50**), of 10/10 Mbits/s and 50/50 Mbits/s each, are included as a comparison. The traffic controlled results can be seen as close to an ideal example of the benefit MPTCP aggregation can bring under fairly optimal circumstances, which in this situation is stable homogeneous links with fairly low latency and bandwidth.

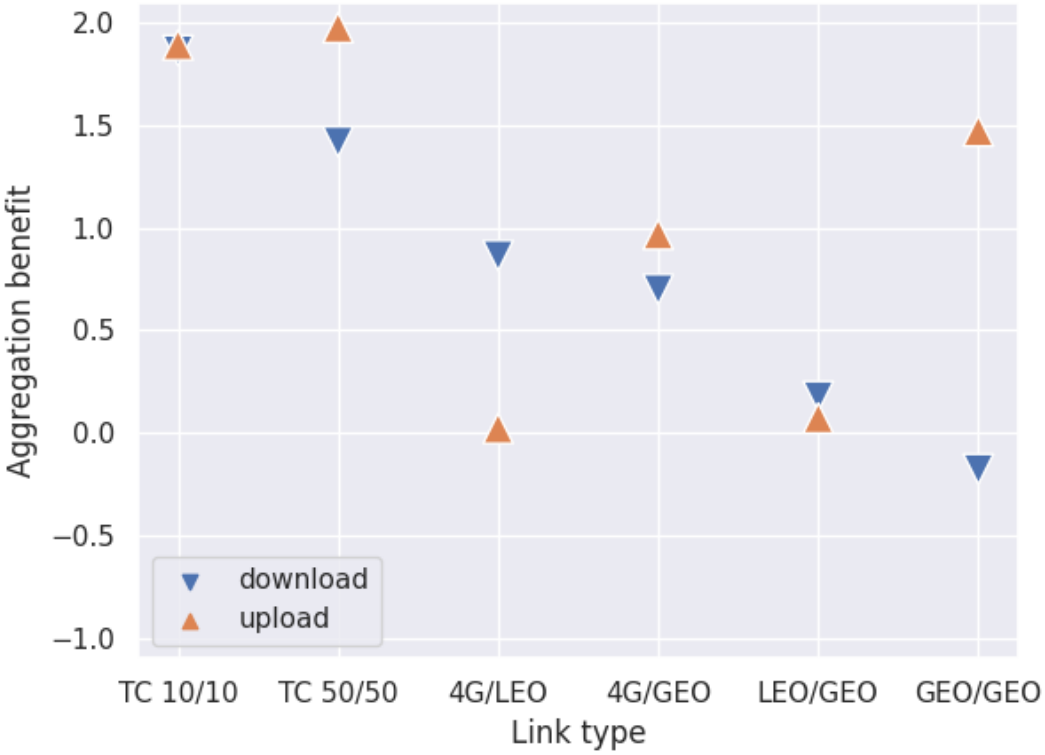


Figure 4.22: The aggregation benefits when using different link types together.

A summary of the average throughput through each link using TCP alone, and together using MPTCP, for the different link combinations can be seen in Table 4.1.

Table 4.1: Summary of tables: average throughput in Mb/s for the different link combinations.

	TCP Link 1		***TCP Link 2***		***MPTCP aggregation***	
Link combination	***Down***	***Up***	***Down***	***Up***	***Down***	***Up***
4G / LEO	20.28	28.07	77.35	9.54	69.6	9.9267
4G / GEO	21.05	4.86	6.915	2.36	16.8	4.7833
LEO / GEO	114.75	8.91	7.17	2.24	20.3333	2.7167
GEO / GEO	5.745	2.375	7.325	2.49	4.7233	3.61

5

Conclusion

In this final chapter, first, the results will be discussed as to what conclusions can be drawn from them. The evaluation study will also be reflected on, looking at what worked well and what could have been done differently to achieve better or more reliable results. Potential future work will be covered, and finally, the thesis report will be summarised and concluded.

5.1 Discussion

The comparison of using an encapsulation layer, in the form of the VPN WireGuard, showed a slight increase overall in the resource usage in the form of energy consumption, as well as CPU and RAM memory usage compared to running it without encapsulation. It also showed that the goodput (useful throughput) was decreased which is a direct effect from the data overhead from the encapsulation. This is an expected outcome, but it is worth noting that this increase was much less severe compared to other encapsulation options, which were briefly trialled, such as the commonly used OpenVPN.

The results of the performance evaluation showed both situations where the use of MPTCP performed better than TCP and situations where it performed worse. Although the measurements were inconsistent, some patterns could be drawn from the results. It was fairly noticeable that the performance results became increasingly worse, compared to normal TCP communication, the more different the links' characteristics were from each other. Similar claims were found when reading related research material in the field. Another realisation was that the performance of MPTCP seemed to compare worse to single-link TCP communication the higher the bandwidth, and the longer the latency was of the available links. However, the measurements in the controlled environment using stable connections, showed higher consistency with both better and more expected results.

The evaluation strategy has for the most part been adequate and working well. However, there was some limitations to our methods that in turn affected the outcome of our results.

The inconsistency of the results can be traced back to several factors, where one of the more significant is the time-varying metrics of the internet connections used during the evaluation which we had very little control over. In addition to this

the number of tests performed for each of the different link combinations was quiet low, which together with the low control of the links resulted in both less reliable and less accurate results. Further, we also want to point out that the evaluation methods and the results they yielded did not take the overhead of measurement into account. Another limitation was that we had no control over the MPTCP scheduler which default behaviour might be unfavourable for the throughput metric, especially for when the aggregated links had very different characteristics like in the case of combining LEO and GEO.

With these limitations in mind there are some aspects of the work that in hindsight we would have done differently.

To start with, we would reduce the number of different link combinations to be tested and perhaps only focus on satellite links. This would allow for us to develop more specialised solutions and optimisations which would be more interesting for the reader while at the same time might also be more useful for Satcube.

We would also put more time and effort into performing better baseline tests for the different link types in order to characterise them. Doing this would allow us to more accurately mimic the links in a controlled environment, which in turn would make it possible to perform more spontaneous development tests with less need of the access to the hardware needed for the real link types. Another change that we would do is to perform longer tests which would be less labour intensive for us while also generating more accurate results.

Last we would also focus on the comparison between TCP and MPTCP rather than the comparison of MPTCP in different stages of our implementation which we planned. This would result in a fewer amount of tests during the development of the implementation and thus leave more time for further development or more meticulous testing.

5.2 Future Work

With additional time, the next step of our work would have been to implement a proxy server that could convert traffic between TCP and MPTCP and vice versa. This would make it so that MPTCP communication could be used on a subpart of the path for situations where the endpoints server lacks support for MPTCP. We believe that this could be useful as the path between Earth and the satellites usually are the bottleneck of satellite communication. If MPTCP could be used in that part of the communication it could make the impact of the bottleneck less severe. It would also make traffic to and from LEO satellites more reliable if a GEO satellite could be connected as a backup when switching between LEO satellites.

For improving link aggregation performance for heterogeneous links it would be interesting to look at alternatives to the default MPTCP schedulers and congestion control algorithms. This would take further work comparing their performance to each other or even attempting to design a new alternative suited for satellite communication. A solution similar to the scheduler Sailer and Hähner [17] proposes

could be of interest, which takes into consideration both flow awareness and the latency and throughput of the available subflows.

We believe there could be potential benefits in implementing a more sophisticated scheduler that adopts different strategies depending on the size of the data flows through communication with the network application layer, similar to what Xing et al. [18] propose in their paper.

5.3 Conclusion

In this thesis, we have performed an evaluation study of MPTCP, combining several types of links with a focus on satellite communication, and measuring the performance over a wide variety of metrics. The solution has been implemented on a development board running a custom Yocto project operating system, mimicking a real-life scenario. An encapsulation layer was added as a mitigation for hardware and middlebox interference, which was evaluated with a focus on resource usage, to give clarity on the cost of using an encapsulation layer as a potential workaround. The performance when using different combinations of links together was measured and compared to the single-handed use of the links separately. Additionally, measurements using simulated link characteristics with the "traffic control" tool were done to give insight into the performance of MPTCP communication in a controlled environment. The result shows signs that indicate that the aggregation benefits decrease depending on the differences in the links' characteristics. The less similarity the links share, the worse MPTCP communication performs compared to regular TCP communication. However, there were also many other parameters that could affect the results, such as the instability and varying performance of the different internet connections. As a conclusion, the study shows that using MultiWAN functionality in the form of MPTCP communication can bring beneficial results under the right circumstances for satellite communication, but that there is still much work that needs to be done on the default configuration to bring consistent and reliable aggregation benefits.

Bibliography

[1] theglobalpanorama, *Satellite in space*, 2014. [Online]. Available: https://openverse.org/image/38e8f67f-f3a8-4dc0-b733-fd857537581c?q=satellite%5C%20in%5C%20space (visited on 06/02/2023).

[2] Satcube, *Satcube*. [Online]. Available: https://satcube.com/about (visited on 12/04/2022).

[3] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch, "Tcp extensions for multipath operation with multiple addresses," RFC Editor, RFC 8684, Mar. 2020. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8684.html.

[4] M. Community, *Home · multipath-tcp/mptcp_net-next wiki · github*. [Online]. Available: https://github.com/multipath-tcp/mptcp_net-next/wiki (visited on 01/18/2023).

[5] I. D. Team, *Iperf website*, 2021. [Online]. Available: https://iperf.fr/ (visited on 01/24/2023).

[6] T. D. Team, *Tcpdump website*, 2021. [Online]. Available: https://www.tcpdump.org/ (visited on 01/24/2023).

[7] W. D. Team, *Capinfos man page*, 2021. [Online]. Available: https://www.wireshark.org/docs/man-pages/capinfos.html (visited on 01/24/2023).

[8] OpenWrt, *[openwrt wiki] welcome to the openwrt project*. [Online]. Available: https://openwrt.org/start (visited on 01/17/2023).

[9] YoctoProject, *Yocto project – it's not an embedded linux distribution – it creates a custom one for you*. [Online]. Available: https://www.yoctoproject.org/ (visited on 01/17/2023).

[10] E. W. Eddy, "Transmission control protocol (tcp)," RFC Editor, RFC 9293, 2022. DOI: 10.17487/RFC9293. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9293.txt.

[11] P. Dong, J. Xie, W. Tang, N. Xiong, H. Zhong, and A. V. Vasilakos, "Performance evaluation of multipath tcp scheduling algorithms," *IEEE Access*, vol. 7, pp. 29 818–29 825, 2019.

[12] D. W. C. Raiciu, "Coupled congestion control for multipath transport protocols," RFC Editor, RFC 6356, 2011. DOI: 10.17487/RFC6356. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6356.html.

[13] M. Honda, Y. Nishida, C. Raiciu, A. Greenhalgh, M. Handley, and H. Tokuda, "Is it still possible to extend tcp?" In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, 2011, pp. 181–194.

[14] J. A. Donenfeld, *Wireguard: Fast, modern, secure vpn tunnel*, 2015. [Online]. Available: https://www.wireguard.com/ (visited on 05/30/2023).

[15] K. Fahmi, D. Leith, S. Kucera, and H. Claussen, "Understanding mptcp in multi-wan routers: Measurements and system design," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, 2021, pp. 132–139. DOI: 10.1109/LCN52139.2021.9524976.

[16] M. Amend, E. Bogenfeld, M. Cvjetkovic, *et al.*, "A framework for multiaccess support for unreliable internet traffic using multipath dccp," in *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, IEEE, 2019, pp. 316–323.

[17] R. Sailer and J. Hähner, "Htmt: High-throughput multipath tunnelling for asymmetric paths," in *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–3. DOI: 10.23919/IFIPNetworking52078.2021.9472781.

[18] Y. Xing, J. Han, K. Xue, J. Liu, M. Pan, and P. Hong, "Mptcp meets big data: Customizing transmission strategy for various data flows," *IEEE Network*, vol. 34, no. 4, pp. 35–41, 2020.

[19] G. Detal, C. Paasch, and O. Bonaventure, "Multipath in the middle (box)," in *Proceedings of the 2013 workshop on Hot topics in middleboxes and network function virtualization*, 2013, pp. 1–6.

[20] J. Han, K. Xue, W. Wei, Y. Xing, J. Liu, and P. Hong, "Transparent multipath: Using double mptcp proxies to enhance transport performance for traditional tcp," *IEEE Network*, vol. 35, no. 5, pp. 181–187, 2021.

[21] L. Chaufournier, A. Ali-Eldin, P. Sharma, P. Shenoy, and D. Towsley, "Performance evaluation of multi-path tcp for data center and cloud workloads," in *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '19, Mumbai, India: Association for Computing Machinery, 2019, pp. 13–24, ISBN: 9781450362399. DOI: 10.1145/3297663.3310295. [Online]. Available: https://doi.org/10.1145/3297663.3310295.

[22] S. Szilágyi, F. Fejes, and R. Katona, "Throughput performance comparison of mpt-gre and mptcp in the fast ethernet ipv4/ipv6 environment," *Journal of Telecommunications and Information Technology*, no. 2, pp. 53–59, 2018.