MASTER THESIS
UNIVERSITY OF TWENTE

Space Internet Congestion Control (SICC)

Optimising loss-based congestion control algorithms for increased throughput over satellite-terrestrial communication networks

Rezfan Pawirotaroeno

November 30, 2022

Supervisors:

Prof.dr.ir. R.M. (Roland) van Rijswijk-Deij (University of Twente)
Dr.ir. B.J. (Bernd) Meijerink (University of Twente)

External examiner:

Prof.dr.ir. A.B.J. (Andre) Kokkeler (University of Twente)

Abstract

Starlink is the new standard for satellite internet to be used in rural areas where traditional internet is not possible. The system is a constellation of many satellites orbiting around the earth in low earth orbit (LEO). A downside of being in LEO is that the satellites have to travel at extremely high speeds resulting in frequent handovers and rerouting between the communicating nodes. The work in this thesis focuses on how the highly dynamic nature of this network affects the performance observed by the end user. To achieve insight into this, we explored the performance over Starlink by performing active TCP throughput measurements of the down-link in the system. Results showed that the congestion control algorithms suffered from periodic packet loss at multiples of 15 seconds. The possible reason behind this is not known, however, it can be presumed to be a result of either the frequent handovers and/or updating the routing tables. This periodic loss results in standard loss-based congestion control algorithms decreasing their congestion window as they presume congestion has taken place when in reality it has not and causes the underutilisation of the available bandwidth in the communication link. As a possible solution to this problem, we developed a custom congestion control algorithm, SICC, based on the open-source code and logic of Cubic. This algorithm is modified to address the problems encountered when communicating over Starlink. It conditionally scales its congestion window depending on the time at which retransmissions take place. The throughput performance of SICC was shown to achieve, on average, almost the same amount of throughput as BBR and be around 40% faster than the average throughput achieved by Cubic. Furthermore, SICC is 100% fairer towards Cubic than BBR over the satellite link, however not as fair as Cubic is towards itself. Additional findings in this research indicate that users perceive a higher throughput during the first 50 seconds of their connection. After this initial period, the user throughput stays at a relatively lower rate for 3.5 minutes before returning to the initial amount of achieved throughput. So far, we were unable to determine the cause of this behaviour. We speculate that this may be deliberate traffic shaping, but leave deeper analysis to future work.

Acknowledgements

While writing my master's thesis, I received lots of support and assistance, which enabled me to develop professionally and personally. On a professional level, I managed to gain an even deeper understanding of the different communication protocols at hand, while on a personal level, I have gained more confidence in my ability to learn, adapt, and overcome problems and challenges thrown my way. I learned not to stress nor be overwhelmed by the tasks that needed to be done but to stay calm, keep my head down and tackle them one at a time.

Firstly, I would like to show my gratitude towards my professor, R.M. van Rijswijk - Deij, whose suggestions formed a clear direction in which my project should proceed. Their expertise in Computer Networking proved essential in establishing the foundational aspects for analysis during the research. Additionally, I would like to thank them for allowing me to do my master's thesis at the University of Twente while working with revolutionary technology.

Secondly, I would like to thank my daily supervisor B.J. Meijerink, who was always quick to respond when I had technical problems during the measurement phase of my thesis. Their knowledge regarding kernel modules in Linux proved invaluable for developing SICC. Furthermore, their suggestions regarding the measurements enabled me also to achieve promising results.

Finally, I would like to thank some of my friends, J.P. Konijn, A. Velkers and R. Kapoor, for supporting me during my entire thesis from a social perspective and providing stimulating discussions regarding the approach in my research. In addition, I would like to thank my mother, S. A. Pawirotaroeno, for enabling me to study and pursue my interests at the University of Twente in the Netherlands. I would not have been able to complete my thesis without them.

Glossary

ACK: *Acknowledgement*

AIMD: *Additive Increase Multiplicative Decrease*

BDP: *Bandwidth Delay Product*

BER: *Bit Error Rate*

CC: *Congestion Control*

CCA: *Congestion Control Algorithm*

CWND: *Congestion Window*

DupACK: *Duplicate Acknowledgement*

FCC: *Federal Communications Commision*

GSO: *Geosynchronous Orbit*

GEO: *Geostationary Orbit*

ISP: *Internet Service Provider*

LEO: *Low-Earth Orbit*

LFN: *Long Fat Network*

LTE: *Long Term Evolution*

MEO: *Medium-Earth Orbit*

OSI: *Open Systems Interconnection*

RTT: *Round-Trip Time*

SACK: *Selective Acknowledgement*

SICC: *Space Internet Congestion Control*

SSTHRESH: *Slow Start Threshold*

TCP: *Transmission Control Protocol*

VM: *Virtual Machine*

Contents

1 Introduction

Nowadays, the internet is an indispensable part of our daily lives as it plays a demanding role in in education, many people their work life and in healthcare. This is why the modern internet is also recognised as a critical infrastructure of society. Because of this, worldwide, different companies and governmental bodies have been attempting to provide a stable, sufficient and secure internet connection as a service to the public [1]. The common way the internet is used is through a combination of traditional copper and modern fibre cables that are primarily placed underground. Furthermore, people can gain access to the internet through locally placed cellular towers in and near populated areas. Financing these projects is quite expensive, which is why these internet cables and infrastructure often run from populated areas to other non-rural areas. This way, it can be ensured that a majority of people living near these areas have access to a decent internet connection. In reality, however, there are also people living in rural areas that have no access to these standard internet connection services from a nearby Internet Service Provider (ISP), nor do they have cell towers nearby for Long Term Evolution (LTE) communication [2]. A possible solution is the use of satellites and gateway stations as a medium for communicating with ISPs and thus gaining access to the internet. This idea has been around for the past few decades, and several companies exist that currently provide internet access through satellite-terrestrial links.

Initially, this was commercially implemented through the form of Geosynchronous Orbit (GSO). These are satellites orbiting the earth at around 35000 km above the globe and can provide a more extensive set of users with internet due to their larger coverage area. The downside of this, however, is that Round Trip Times (RTTs) are more prominent as the signals need to travel a significant distance from the earth to the satellite and back. This would negatively influence the retransmission procedure in case of packet loss because large latencies (around 200ms to 600ms) result in increased waiting times, ultimately lowering the overall throughput over such links [3]. Furthermore, another problem these networks face is sharing the available bandwidth with the users, as one satellite covers a large area of the earth. Having one satellite cover a large area of the earth causes a reduced bandwidth per user, as many users would have to share the communication with a single satellite.

With the rise of internet usage and technological development regarding its capabilities also came an increased need for more throughput [4]. This resulted in GSO internet communication being less favourable as a means for internet communication. A solution to the high RTT problem associated with GSO satellites is to decrease the distance between the earth and the satellites by placing them in an orbit closer to the earth. These satellites are referred to as Low Earth Orbit (LEO) satellites and are located in orbits between 300 and 1600 km above the earth [5]. Due to their lower distance to the planet, there is a significant reduction in RTT between terminals and ground stations. However, the downside to being so relatively close is that the satellites can only cover a comparatively smaller area for communication. Furthermore, they have to travel at higher speeds to maintain orbit. This implies that in order to be able to provide a reliable internet connection, there would have to be several satellites travelling in the same orbit above the users their transmitters and receivers for reliable communication. Implementing such a system in practice would require large amounts of financial support as a large number of satellites would have to be developed. Additionally, all of these would have to be launched into low earth orbit to ultimately provide decent to full internet coverage. One company that is trying to realise this is the American company SpaceX with their project: Starlink.

Starlink is an internet accessibility project started in 2019 with the goal of providing broadband internet access to users through satellite communication. In short, the project consists of different phases where sets of satellites are launched into LEO at around 550 km above the earth's surface. At the moment of writing this thesis, the amount of Starlink satellites in service is around 2650, however the company plans to expand this amount to at least 12000 by 2027 [6]. Although the impact that this system has on the environment is also an essential research direction to consider, this thesis will only focus on its internet connectivity. With this extensive satellite network, SpaceX wishes to provide good internet access to any place where regular internet access would be unavailable (e.g. forests, abandoned islands, war zones, and aeroplanes over the ocean) [7]. According to their Federal Communications Commission (FCC) filings, the Starlink system will use the properties of phased-array beam steering to communicate between satellites and terminals [8]. At the time of writing this thesis, the Starlink internet is commercially available and will be used for the remainder of the experiments discussed in the report.

An important aspect to acknowledge in such a system is its high amount of dynamicity compared to internet from standard terrestrial networks. This is the case because the satellites are flying over terminals and gateways at a speed of around 7.5 kilometres per second resulting in terminals and gateways only communicating with a specific Starlink satellite for a limited period of time before it passes the horizon [9]. Thus at different points, the terminals and gateways have to communicate with different satellite nodes. How this high dynamicity influences the performance perceived at a user terminal will be the foundation of this thesis and will be discussed further in the thesis.

This thesis has the following structure:

The first section of the report, labelled Section 2: **"Background"**, will cover the different concepts and theories required to understand the goals of the report. This is done by first briefly discussing the current state of the art of satellite internet and then covering the basics of the performance metrics in satellite networks.

The "**Related Work**" section will discuss essential findings from other different research papers that revolve around the problems and solutions regarding satellite communication. These findings will form the basis for formulating a problem statement in the following section.

"**Problem Statement & Research Goals**" will formulate a general description of a problem based on the challenges mentioned in the related work from the previous chapter. Consequently, these will be further translated into the main research question and several sub-research questions that will form the goals of this research. The different sub-research questions can be split into three phases that would have to be researched sequentially. The first phase is exploratory, the second is prototyping and the last phase is evaluation.

The next section is thus "**Phase I: Starlink Exploration**" and will first cover the approach for gathering exploratory data. After mentioning how the data will be gathered and analysed, the results will be displayed and discussed before proceeding to the next phase in the following section.

The second phase "**Phase II: Prototyping (SICC)**" will first cover the approach for developing and implementing a prototype based on the findings from the previous section. The results of this prototype after implementation will be shown and discussed.

Then the final phase, "**Phase III: Prototype Evaluation**", will first mention how the developed prototype from the previous phase will be evaluated. The results from this evaluation will be gathered, analysed, shown and discussed.

Afterwards the **"Conclusions"** section of the report will pull together the main points of the thesis and reflect once more on the research questions.

Lastly, the "**Limitations & Future Work**" will identify and acknowledge the limitations of the research and how these reflect on the results. Furthermore, future works will be elaborated upon to suggest future directions for research in the field of satellite internet.

2 Background

This section of the report will provide the reader with the background required to understand the different concepts and goals of the thesis. It consists of two main subsections, the first is a survey of the current major satellite internet service providers. It enables the reader to gain a better perspective regarding the different internet service providers and their capabilities. By doing so, we can ultimately emphasise the relevance of this research. The second subsection will elaborate on the theory behind TCP and bandwidth. This will enable the reader to understand the metrics related to evaluating the performance of a communication network.

2.1 Satellite Internet Survey

This subsection of the report briefly describes the top satellite internet providers as of 2022 according to *www.satelliteinternet.com* [10]. First, details regarding Starlink's history, features and pricing will be explained. Second, Hughesnet will be elaborated upon. Lastly, this subsection will provide a brief overview of Viasat their capabilities and pricing.

2.1.1 Starlink

In 2019, the private space company SpaceX set out to create a complete LEO internet constellation of satellite internet broadband services. For most other companies, this is a complex and expensive challenge to overcome, as sending objects into space is very costly and time-consuming. One of the main challenges that would need to be addressed is the high financial cost of space travel. The transportation mediums would have to be reconstructed/developed after every launch. Fortunately, SpaceX is a company leading in the technology behind reusable rockets [11]. After a successful takeoff and landing of their reusable Falcon 9 rocket, they confirmed their competitive advantage in the field of space travel. With their ability to reuse rockets after every launch, they could send out lots of objects into orbit for a relatively low price. The utilisation of this advantage lead to the Starlink project. Starlink's goal was to provide high-speed broadband satellite internet to users located in rural areas, aviation and marine communication and in developing countries at a lower cost than existing satellite internet services.

By launching objects into orbit at a lower price, SpaceX can cast many satellites into LEO to create an entire constellation for satellites to ultimately provide internet. Since 2019, several thousands of satellites have been launched into low earth orbit every year to expand the constellation and provide internet to additional urban regions. As of 2022, users pay a subscription of around €111,- per month and a singular hardware fee of €600,- [12]. With this subscription, Starlink claims users can achieve speeds ranging from 50 Mbps to 350 Mbps with an expected latency of 20 to 40 ms, depending on the region, the generation of hardware used for the dish, and the amount of traffic at that time.

We argue that due to SpaceX's competitive advantage regarding financing, development and features of the constellation, Starlink will be widely used for satellite internet in the foreseeable future. Do note that this argument does not take into consideration the environmental impact of this system and how it will influence future deployment of the satellites.

2.1.2 Hughesnet

Prior to Starlink, Hughesnet was one of the most widely used providers for satellite internet. It is a satellite internet provider owned by Hughes Network Systems and aims to provide high-speed internet services to users residing in rural areas, business partners, and the military [13]. Hughesnet satellites are classified to be in GSO, thus also have a latency of around 600 ms. As a result of this significant latency, this network suffers when users attempt to use low-latency applications such as gaming, video streaming and conferencing [14].

Regarding the pricing, Hughesnet offers different subscriptions based on the needs of the user. However, the maximum download speed a user can achieve is estimated 25 Mbps, and 3 Mbps upload speed. The amount users have to pay ranges from €70,- to €150,- and are associated with datacaps from 15 GB to 100 GB [15].

2.1.3 ViaSat

Launched in 2012, ViaSat (formerly known as "Exede") attempts to provide internet access to areas where traditional internet was not possible. Viasat has three internet satellites, ViaSat-1, ViaSat-2, and ViaSat-3, in GSO that enable users to connect to the internet using ViaSat dishes.

Regarding the pricing, they provide nationwide satellite internet in the US with speeds ranging from 12 to 100 Mbps and prices ranging from €30,- to €300,- per month depending on the subscription [16]. Similar to Hughesnet, ViaSat internet also contains latencies around 600 ms due to satellites' considerable distance to earth.

2.2 TCP & Throughput

The different providers mentioned in Section 2.1.1 have systems that enable communication with the internet from satellites in space. To enable communication through these complex networks they have to follow a set of standardized protocols. One of these standard protocols that are important for communication is the Transmission Communication Protocol (TCP) [17]. This section of the report will focus on explaining the basics of TCP communication and how associated TCP congestion control algorithms attempt to overcome challenges present in complex networks such as the internet. By doing so, the reader gains more insight into the systems at hand and the performance metrics that are used for this thesis.

2.2.1 TCP Functionality

Similar to terrestrial networks, satellite networks and satellite-terrestrial networks are still reliant on the different layers of the Open Systems Interconnection (OSI) model reference for proper communication [18]. This thesis will primarily focus on the transport layer (layer 4) shown in Fig. 1, as we cannot gain insights into layer 1 (physical), layer 2 (data-link) and layer 3 (networking) parts of the communication with the resources available to us. The layer we can influence however, is the transport layer.

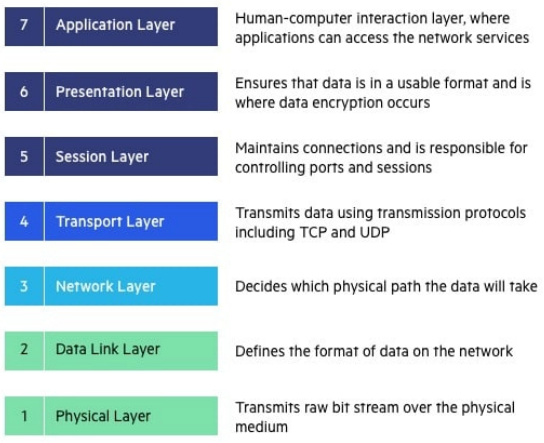

7 Application Layer
Human-computer interaction layer, where applications can access the network services
6 Presentation Layer
Ensures that data is in a usable format and is where data encryption occurs
5 Session Layer
Maintains connections and is responsible for controlling ports and sessions
4 Transport Layer
Transmits data using transmission protocols including TCP and UDP
3 Network Layer
Decides which physical path the data will take
2 Data Link Layer
Defines the format of data on the network
1 Physical Layer
Transmits raw bit stream over the physical medium

Figure 1: OSI layers [19].

Regarding the communication in complex networks the transport layer ensures:

- **Segmentation of data**: Ensuring the data is split into segments that will be sent separately, whenever network capability (data larger than the sender or receiver's buffer) is surpassed.

- **Delivery of proper segment order**: After being broken down into separate parts, the data will be sent from sender to the receiver. However, depending on the network, there is no guarantee that the separated parts will arrive in the same order they were initially after being split. TCP is responsible for ensuring that the arrived packets get reordered into their original order and combining them back into the data it was initially.
- **Multiplexing**: Whenever there are multiple streams coming into and going out of a computer. TCP is also responsible for ensuring that applications are communicating from and to the correct computers.

An important feature here is the reliable delivery of the segmented data. Whenever TCP communication takes place over the internet, the following events occur regarding the sending and receiving of the segmented data:

- TCP connection between two computers that want to communicate is established.
- Each computer tells the other about their window size. This parameter denotes the maximum amount of data one computer can send to another before pausing and waiting for an acknowledgement in return.
- The sender then starts sending data split into segments according to the Maximum Segment Size (MSS). Once the maximum window size is achieved, the computer will stop sending and wait for an acknowledgement for a specified duration of time.
- After waiting the previously mentioned specified duration of time, the TCP sender will see this as packet loss, and retransmit the data.

The TCP layer not only plays a critical role in communication as it provides reliable end-to-end transmissions, but also avoids congestion when the link is experiencing it by adjusting the congestion window (CWND) size of the sender. The congestion window is a variable that the sender maintains and determines how much data will be sent to the receiver. Depending on the RTT and the number of retransmissions, this congestion window will be adjusted accordingly. This adjustment of congestion window also enables the protocol to ensure fairness over the multiple connections flowing through the link. Even though these features exist, communication over the internet comes with additional challenges as it is a very complex system with a large number of dependencies. Two of these are the amount of data one computer can send to another before pausing, and the other being the amount of time one computer has to wait for an acknowledgement before continuing with sending data. These two factors influencing TCP, one being dependent on the RTT and the other being dependent on bandwidth can be combined into a metric called the Bandwidth Delay Product and is described in the following Section 2.2.2.

2.2.2 Bandwidth Delay Product

Bandwidth Delay Product (BDP) is the measure used to estimate how many bits can fill up a link in a given network. It indicates how much data a sender can transmit at a specific time before receiving an acknowledgement from the receiver displayed in Fig. 2 and be expressed using the following function:

$$BDP = Bandwith * Delay \tag{1}$$

The BDP metric is used in understanding the challenges mentioned in the following chapter present in modern internet.

2.2.3 Long Fat Networks (LFNs)

Networks with a large bandwidth delay product are referred to as Long Fat Networks. In LFN's, the "Long" indicates the large distance between the communicating nodes and the "Fat" stands for the links

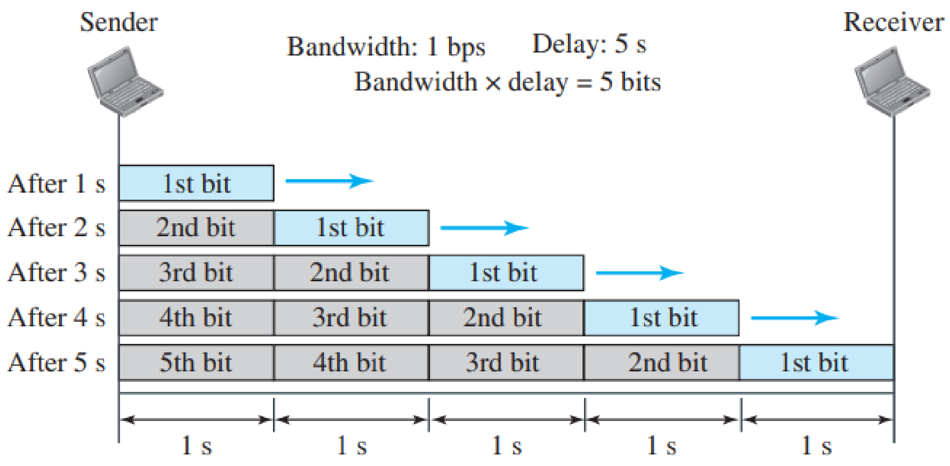


Figure 2: Visualisation Bandwidth Delay Product [20].

between them having large available bandwidth [21]. Within the modern internet, most communications go over these LFN's because routers and links have increased bandwidth as a result of high demand and the large distances between the communicating nodes. Because of this, the sender will potentially have to also spend a relatively long time (∼70-100 ms) waiting for an acknowledgement depending on the RTT before sending more data [22], which results in inefficient use of the link. This issue is referred to as the Bandwidth Delay Problem and thus relates to how much data the sender should transmit depending on the RTT and available bandwidth to efficiently utilise the link.

The part of TCP that can influence the performance for solving for these challenges is Congestion Control (CC). Congestion control algorithms are developed to form the logic for optimising link utilisation depending on the different network factors (e.g. RTT, retransmissions, RWND, CWND). There several different TCP CC algorithms and these will be described in more detail in the next Subsection 2.2.4.

2.2.4 TCP Variants

This subsection of the background will focus on elaborating upon the different Congestion Control Algorithms (CCA's) and how their logic differs from one another regarding the growth and decrease of the CWND. There are many CCAs, however, we will only cover the most widely implemented. These are the older standard CCA Reno, the current Cubic, and the newer variant BBR.

2.2.4.1 TCP Reno

The first popular CCA for TCP CWND scaling was TCP Reno, introduced in 1990 [23, 24]. Although TCP Reno is an outdated protocol, we argue that it is essential for understanding other CCA variants as it contains the most basic functionalities of congestion control. Additionally, Reno's fundamental mechanisms for congestion control are still used in other widely implemented TCP CCAs in todays' internet. Regarding the throughput/CWND growth, the algorithm consists of two phases. Initially, in the slow start phase, TCP Reno will exponentially increase the CWND value until packets are not acknowledged. This will enable Reno to estimate a baseline in which to operate before causing congestion. When packet loss occurs, it will halve the CWND and transition into congestion avoidance mode. When in congestion avoidance mode, Reno will increase its CWND size as long as no retransmissions occur. This increase is done each time an acknowledgement is received from the receiver. When congestion occurs (detected as a packet loss by Reno), link capacity has been reached, and retransmissions take place, Reno will half its congestion window again. A visualisation of the CWND can be seen in Fig. 3. This mechanism for additively increasing the CWND per ACK and decreasing it multiplicatively is referred to as Additive Increase Multiplicative Decrease (AIMD).

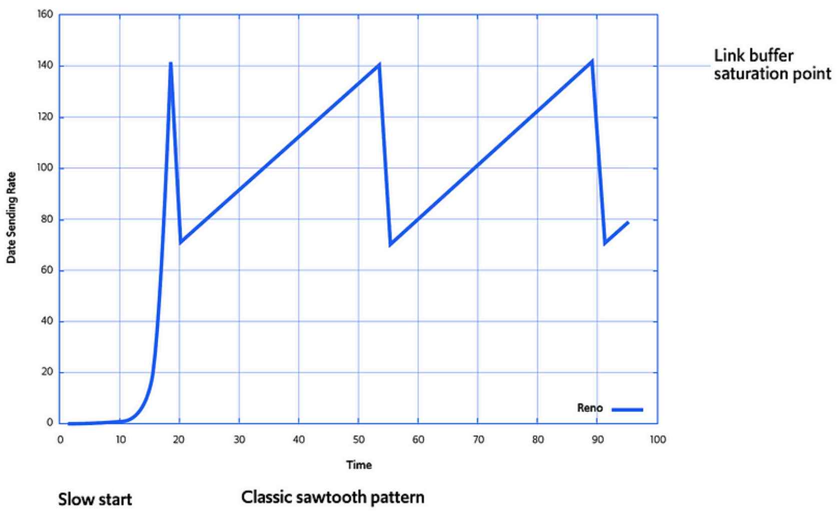


Figure 3: Reno slow start and CWND growth [25].

Furthermore, for handling the previously mentioned packet loss, Reno also implements mechanisms referred to as “Fast Retransmit” and “Fast Recovery”. As mentioned in Section 2.2.1, TCP is responsible for splitting the data into segments. The sender and receiver will both assign sequence numbers to each of these split segments to be able to reassemble them in the correct order on the receiver side. After each time the sender has sent data, the receiver will acknowledge the segments if the sequence numbers are in order. When a packet is lost, the receiver will receive an out-of-order sequence number and keep acknowledging the last sequence number received in the correct order. When the sender observes that multiple duplicate acknowledgements are received for the same sequence number, the sender will know that the packet (segment) containing the correct sequence number was lost and will immediately retransmit that packet. This mechanism is referred to as Fast Retransmit and is visualised in ig. 4 where *Data(10)* is lost and immediately retransmitted from the sender after receiving multiple duplicate acknowledgements for the last correctly ordered sequence number, which is *ACK(9)*. Fast Recovery is the mechanism that Reno uses after Fast Retransmit has taken place. The goal of this mechanism is to quickly update the correctly received sequence numbers at the receiver side, causing the sender not to have to resend the segments that were already sent after the lost segment. This is also seen in Fig.4 where the receiver acknowledges *ACK(19)* after receiving the retransmitted *Data(10)*, letting the sender know that it can start sending from sequence *Data(22)*.

Reno, however, has its limitations, being that CWND decreases multiplicatively for each packet loss in the same congestion window. For example, if two packet losses occurred in one congestion window, CWND would have reduced 4 times (2x 50%), when one decrease in that congestion window would have been sufficient. This issue resulted in the further modifications of this CCA that all attempt to solve these problems.

2.2.4.2 TCP Cubic

After TCP Reno, there were several different largely used TCP algorithms (e.g. NewReno, BIC); however, one of the most widely implemented CCA is TCP Cubic. Since 2006, Cubic has been the standard CCA for many systems on the internet [27] and we argue that lot of computers in the modern internet on the internet still use Cubic, as many of them have a Linux as operating system where Cubic is set as the default CCA for TCP communication [28]. It was designed to achieve high throughput over connections in LFNs, thus attempt to tackle the high BDP problem experienced in the modern internet.

Similar to Reno, before transitioning into congestion avoidance mode, Cubic will first be in slow start mode. Cubic has different slow start modes depending on the specified parameters. These modes are

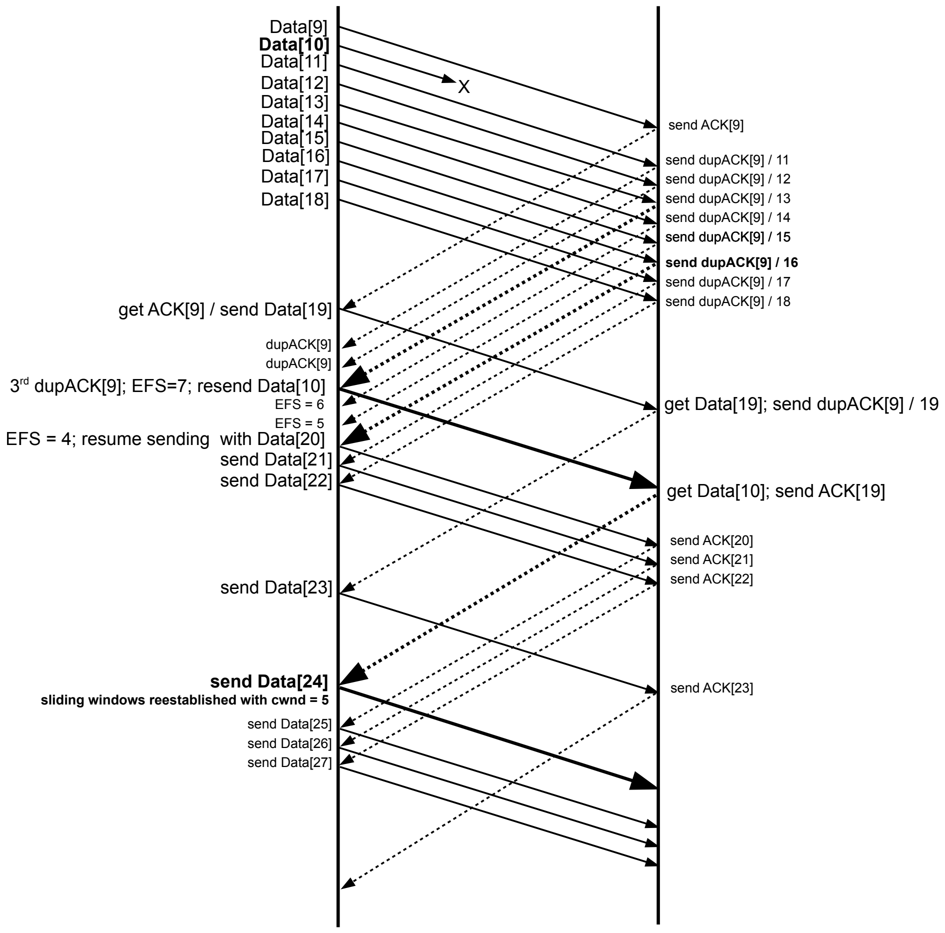
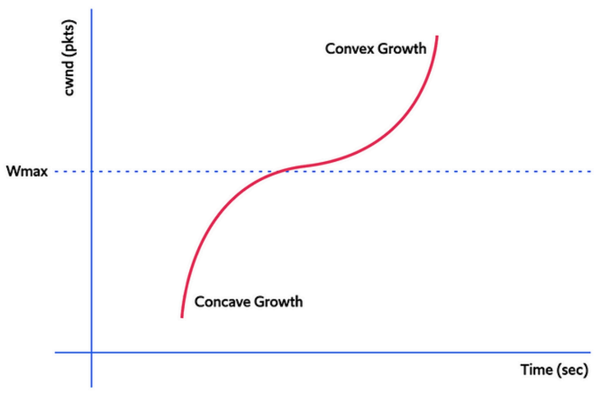


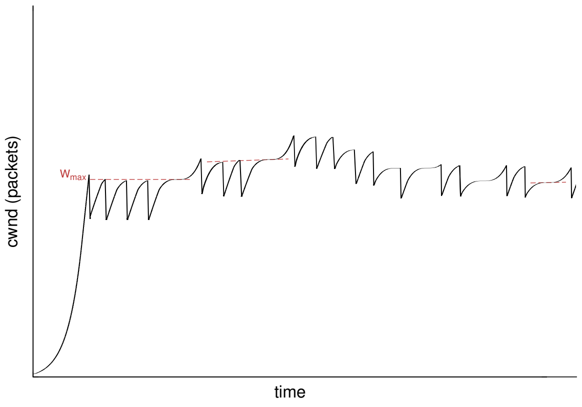
Figure 4: Fast Retransmit and Fast Recovery visualised [26].

standard slow start [29], limited slow start [30] or the Hybrid Slow Start (HyStart) [27]. Since HyStart is best used in LFNs (internet), it is set as the default slow start mode in Cubic. In this mode, Cubic will exit slow start mode prematurely and immediately transition into congestion avoidance mode to increase its CWND. Slow Start Threshold (SSTHRESH) is a value that determines when to exit slow start mode. Cubic will increase until the calculated SSTHRESH is reached. After spending some time in that region, it will start increasing CWND exponentially to search for more bandwidth. This can be described as having a concave growth until SSTHRESH is reached (Wmax) and eventually having convex growth to search for more bandwidth, and is displayed in Fig. 5a.

As seen in Fig. 5a, Cubic uses a 3^{rd} degree polynomial function (hence the name) for achieving its CWND window growth. When a packet is lost, and retransmission takes place, Cubic will also decrease multiplicatively by a pre-specified amount (70% by default). This value of the calculated CWND will be stored at the time the decrease takes place and will be set as the Wmax target. Then Cubic will once again grow its CWND value using the concave profile until the target CWND is reached. After spending some time there, Cubic will use its convex profile to probe for more bandwidth once more. As a result of this, Cubic will ultimately spend the majority of its time between the plateau achieved at the end of the congested event and the region above it. This is visualised in Fig.5. Another important difference between Cubic and other previous TCP congestion control algorithms is that Cubic does not rely on the RTT to increase CWND. The CWND to which Cubic will grow is dependent on the CWND at the time of the previous congestion event. Not relying on RTTs, Cubic ensures more fairness between streams of differing RTTs. Regarding the recognition and transmission of lost packets, Cubic implements the same "Fast Recovery" and "Fast Retransmissions" as that of Reno.



(a) Cubic concave, convex and plateau regions. [25].



(b) Cubic CWND visualised with bandwidth expansion and loss.

Figure 5: TCP Cubic main features.

Even though Cubic performs well in modern networks, it still has some downsides. One implemented feature that results in a downside behind Cubic's logic is that its CWND scaling is only loss based. This means that whenever a loss occurs in the link, the CCA will "think" that congestion has occurred and will decrease the CWND value, thus decreasing the data-sending rate as well. This ultimately decreases Cubic's effectiveness in terms of bandwidth utilisation in lossy links or systems with buffers.

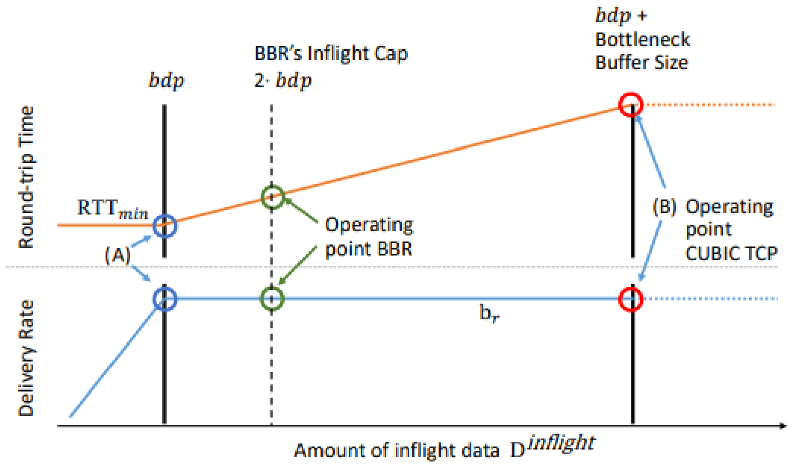
2.2.4.3 TCP BBR

TCP Bottleneck Bandwith and RTT propagation time (BBR), developed by Google in 2016, is one of the newer widely adopted TCP CCA's [31]. BBR attempts to solve the previously mentioned issues that Cubic encountered related to loss-based CWND scaling and the presence of buffers in communicating nodes. Compared to traditional congestion control algorithms, the major difference is that BBR does not adjust its CWND as a result of retransmissions but rather by monitoring both the data rate and RTT. It has two modes; one where it probes for more bandwidth and one where it probes for a minimum RTT.

BBR does so by observing the rate of acknowledgements for their packets, specifically the times between sending packets and receiving acknowledgements for those packets. The CCA is designed to be used in modern systems where latency is largely associated with bufferbloat rather than the route. This is why the developers of BBR argued that network communication performance could better be evaluated using bufferbloat latency rather than congestion event detection through detected packet loss.

The "ideal" throughput (Delivery Rate) can be seen in Fig.6 as the "Operating point BBR" and it is what BBR attempts to achieve. This is defined as the highest delivery rate possible while keeping the RTT to a minimum. In the first probing mode, BBR will attempt to achieve a higher delivery rate while monitoring possible increases in RTT. When BBR sees an RTT increase, it stops increasing the CWND because this increase will only result in filling an intermediate buffer and not increasing the effective delivery rate. In the second probing mode, BBR probes for a minimum RTT by temporarily decreasing the delivery rate. This metric is important for BBR as its value is used to check for filling buffers. By briefly decreasing the bandwidth and monitoring whether a smaller RTT is possible, BBR can determine whether more bandwidth is available by updating its minimum RTT and increasing the delivery rate once again. In most modern networks, BBR does show effective throughput results compared to what its main competitor, Cubic, achieves, but unfortunately, BBR also has its drawbacks.

The downside of using BBR is a result of its different CWND scaling technique compared to that of standard (loss-based) TCP algorithms. Because of BBR's two probing modes and not relying on loss for CWND scaling, BBR tends to prioritise itself when having to share bandwidth in a link with one



bdp + Bottleneck Buffer Size
BBR's Inflight Cap
2· bdp
bdp
Round-trip Time
RTT$_{min}$
(A)
Operating point BBR
(B) Operating point CUBIC TCP
Delivery Rate
b$_r$
Amount of inflight data D^{inflight}

Figure 6: Cubic behavior under probable network scenarios [32].

or multiple non-BBR streams. This unfairness in available bandwidth sharing results in other non-BBR streams having significantly lower throughput performance [33].

3 Related Work

This section of the report will focus on the research that has been done regarding evolution of TCP CCAs for heterogenous networks and the development of protocols for highly dynamic networks, such as satellite networks. Each chapter will briefly summarise the essential topics discussed and found in the papers. These findings will be the foundation for the problem statement in the Section 4.

3.1 TCP Challenges Encountered in Heterogeneous Networks

The modern internet is very complex and consists of, e.g. long-distance wired parts, short-distance wired parts, short-distanced wireless parts and long-distance wireless parts. On a physical layer, networks comprising of these significantly different communicating parts are classified as heterogeneous networks. ”TCP Hybla: a TCP enhancement for heterogeneous networks”, written by Caini et al. [34], is a TCP CCA to increase performance over heterogeneous networks. The authors state that within heterogeneous networks, TCP streams that go over radio and satellite links are significantly disadvantaged compared to wired links as a result of not only their longer RTTs but also the higher bit-error rates. Compared to standard TCP (Tahoe, Reno and NewReno), Hybla attempts to solve the high RTT's negative impact on TCP performance by removing the CCA's dependency on it. This is done through the use of Selective Acknowledgement (SACK), packet spacing and monitoring timestamps to reduce the impact multiple losses would have on a TCP connection. Their ns2-simulations of Hybla in heterogenous (radio and satellite) networks showed better bandwidth utilisation than standard TCP CCAs while remaining fair and friendly towards them.

Another attempt to adapt standard TCP for heterogeneous networks was TCP Westwood. Casetti et al. [35] developed the CCA to increase TCP performance over both wireless and wired networks compared to the standard TCP Reno. Improvements were primarily observed in network connections containing lossy links. The main issue TCP Westwood tries to solve is TCP Reno's inability to distinguish random loss from congestion in heterogeneous networks. This problem results in Reno overreacting to errors by decreasing congestion window size, thus decreasing overall TCP performance. At the time, an essential feature of TCP Westwood compared to other ”TCP lossy link extensions” is that TCP Westwood did not require intermediate proxy nodes nor required inspection of TCP packets. It was a complete end-to-end TCP congestion control, which made it easier to implement in real-world scenarios. Compared to TCP Reno, Westwood monitors the rate of ACKs returning to the sender and estimates the available bandwidth based on these parameters. Whenever three duplicate ACKs are observed, TCP Westwood tries to choose an SSTHRESH and CWND, which results in a bandwidth similar to the estimated bandwidth the sender had at the time of congestion. The developers refer to this mechanism as ”faster recovery”. Experiments showed that Westwood had shown throughputs up to 550% compared to TCP Reno within mixed and wired networks.

Claypool et al. [36] made an insightful comparison regarding measured TCP performance in a heterogeneous network. Initially, they state that realistic assessment of existing heterogeneous (satellite) networks is lacking, as most research results and findings are based on simulation. In their paper they set up an experiment for evaluating the performance of four different TCP CCAs over a commercial Viasat network. Their inspirational testbed is visualised in Fig. 7. The CCAs that are evaluated are Cubic, BBR, Hybla and Performance-oriented Congestion Control (PCC). Regarding the steady-state average bit-rate, they state that the TCP CCAs have more or less the same results. However, the authors suggest that the TCP algorithms differed significantly in their start-up throughputs. Additionally, significant differences in RTT were observed as a result of the protocols filling the buffers, thus queueing up in-flight packets. The researchers defined a “power metric” based on the steady-state throughput and the RTT, and ultimately concluded that PCC showed the best performance, followed by Hybla, and lastly BBR and Cubic showed similar results.

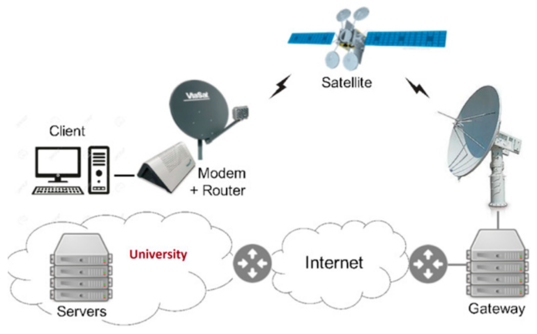


Figure 7: ViaSat TCP performance measurement testbed [36].

3.2 Routing in Highly Dynamic Networks

Besides the challenges TCP faces regarding the wireless aspect present in heterogeneous networks, there are also challenges when looking at the routing in highly dynamic (constantly changing) networks. These are also critical features of satellites in low earth orbit, as they travel at enormous speeds to maintain trajectory.

”Self-learning Congestion Control of MPTCP in Satellites Communications” written by Mai et al. [37], is research that revolves around the usage of Multipath TCP (MPTCP) in LEO satellite networks. They argue that LEO satellites might suffer in performance as frequent handovers have to take place, as a result of the high speed at which the communicating satellites pass by both gateways and user terminals. They state that MPTCP can achieve seamless handovers between satellites, dropping in performance between satellite handovers. Furthermore, they also argue that throughput can be enhanced using MPTCP by utilising a multitude of path transmission mechanisms. For their implementation of MPTCP, the authors used deep reinforcement learning to teach the protocol which congestion control strategy to use for each subflow of the TCP stream. Combined with simulations within ns-3, they used TensorFlow for the agent construction of the Deep Deterministic Policy Gradients (DDPG) agent. Their ns-3 simulations showed increased throughput using the TCP CCA stream selection modification. Lastly, in their conclusion they also state the significance of having a CCA for dealing with the complexities associated with the dynamicity of the network.

”Starlink Analysis” is research done by the Roadmap-5G group from Carinthia University of Applied sciences [38]. In this research paper, they do several experiments regarding the internet performance of a Starlink connection and note their findings. Some of these note the average and maximum download and upload throughputs measured on the Starlink-connected computer over differing durations. Furthermore, they also gathered ping samples to denote the cumulative distribution of the measured latencies and the researchers observed downtime and streaming services were evaluated. Interestingly, the researchers also found jumps in latencies every 15 seconds. Although the exact reason behind these jumps every 15 seconds is unknown, in their summary, they assume that this is the result of the handovers from switching between satellites.

In their paper, ”INTCP: Information-centric TCP for Satellite Networks”, Yin et al. [39] state that TCP is mainly designed for terrestrial networks and will be error prone if used in non-terrestrial (satellite networks). This is also their main motive behind creating their Information-centric Hop-by-Hop TCP protocol (INTCP). INTCP executes hop-by-hop retransmissions and congestion control with the assistance of a request-response and caching model. Hop-by-hop retransmissions enable quick recovery of lost packets in the case of packet loss. This quick recovery is enabled by shorter retransmission delays. Using the INTCP protocol, it is also possible to control both traffic and congestion per hop. Furthermore, by implementing hop-by-hop caching, asynchronous multicast can be implemented and utilized to free up spectrum resources. The performance of this protocol was evaluated within an ns-2 and ns-3 simulation of the Starlink constellation. According to the findings of their simulation, INTCP can reduce one way delays and jitters, while increasing throughput compared to legacy TCP (Reno).

3.3 LEO Satellite Internet Performance

Handley from University College London [40], developed a simulation of the Starlink LEO constellation in their research. The goal was to gain more insights into the latency properties of such a complex communication system. After elaborating upon the way they implemented the Starlink LEO simulation, they summarise interesting takeaways from their results. The first is that their simulations suggest that delay-based CCAs such as BBR may not perform as well compared to its performance on standard terrestrial networks due to frequent reordering. They argue that the frequent reordering of the packets through the LEO constellation can be predictable as long as queues do not build up. Secondly, they say that the network will be less susceptible to failures, as packets can quickly be rerouted through other satellites, resulting in minimal impact on communication performance. Lastly, they mention that routing schemes must be updated more frequently than standard networks. Centralised load-dependent routing schemes (e.g. Local Distribution Router (LDR) & B4) are too slow as they only update their routing tables every minute.

Also evaluated in an Iridium-like satellite simulation, Cao et al.'s paper "Congestion Control & Routing over Satellite Network" [41], focuses on solving several networking issues that will be encountered in LEO satellite networks; being network routing optimisation and network congestion control. They mention that inefficient congestion control algorithms can result in considerable queuing delays, which significantly decreases internet performance. This is why they propose using a self-developed, stable congestion controller to improve performance over TCP/ Active Queue Management (AQM) networks. They propose using a fuzzy logic Proportional-Integral (PI) controller to achieve good performance for handling congestion in TCP networks that require Active Queue Management. However, their suggestion was only validated using simulations. To solve the second LEO network problem related to routing, they developed a routing system that uses Cross Entropy (CE) called CEAARS. Also through simulation, they compared it to existing routing protocols that use Distance Vectors (DV) and Link-state (LS) and showed promising results in terms of convergence speed of finding efficient paths.

Another research paper that summarises several challenges mentioned in the previous chapters is done by Wang et al. [42]. Their article "Aggressive Congestion Control Mechanism for Space Systems" states the relevance of reliable communication. The researchers argue that satellite systems are associated with higher bit error rates (BER), longer time delays and asymmetric channels in the space environment. This results in standard TCP CCAs underperforming. Ultimately, they developed a more "aggressive" algorithm through TCP modifications, which showed increased throughput utilisation. Their protocol focused on adjusting the fast start procedure to be more aggressive and makes use of network feedback to determine actual packet loss and maintain the congestion window size. Using simulations they compared their TCP protocol to existing standards such as TCP Tahoe, TCP Reno, and TCPW. Their results confirmed that standard TCP protocols suffered from the high BER. Furthermore, they also emphasise the use of Delay Tolerant Networks (DTNs), a computer network architecture that attempts to solve issues related to the challenges present in non-wired/heterogeneous networks. However, they only mention its advantages in GEO and Medium Earth Orbit (MEO) systems as these suffer mainly from large delays.

4 Problem Statement & Research Goals

This part of the thesis will first formulate a clear motive and goals for the research through a problem statement in Section 4.1 and formulated research questions in the proceeding Section 4.2.

4.1 Problem Statement

The related work regarding the performance of LEO satellite internet systems indicated that many researchers acknowledge that communication performance through satellite links will significantly influence the performance of TCP in a negative manner [34, 35, 37, 39, 42, 41]. The recurring reason behind this is the presence of wireless links resulting in higher BER. Furthermore, some researchers also suggest that at the networking layer, the performance of LEO satellite networks will be negatively impacted due to the constant rerouting and handovers of the communicating nodes [40, 41]; however, most focus on improving the performance in the TCP layer. A possible reason for this is that researchers have less insight into these systems' network layer protocols. Some suggest different CCAs [34, 35, 37, 41, 39, 42], however all of these seem to be only evaluated using ns-2 and ns-3 simulations. Claypool et al. [36] state the importance of actual real-life evaluations of satellite internet systems in preference to exclusively evaluating them through simulations. His measurement looked at CWND and throughput over a ViaSat satellite network, nonetheless based on the knowledge regarding internet providers (Section 2.1), we argue the relevance of evaluating the real-time performance of the Starlink system as it will most likely be a widely used mechanism for satellite internet communication. To the best of our knowledge, the only research done regarding the real-time performance of Starlink internet is done by the researchers at Carinthia University of Applied Sciences [38] at the time of writing this thesis. However, their findings only denote the measured RTTs and throughput in different regions for different services, with no deeper focus on e.g. packet loss and the CCAs in those systems and how these relate to the measured throughput.

4.2 Research Questions

Because, to the best of our knowledge, little to no extensive research has been done regarding the real-time performance of the internet through Starlink, we make this the central motive of this thesis. This lack of knowledge can be defined as a problem and can be broken down into several research questions of which the answers ultimately provide us with more insights into the behaviour and possible complications present in such a system.

The main research question that would have to be answered is:

How is the throughput performance of the 2022 Starlink internet system affected by its highly dynamic communication nodes?

To answer the main research question, several preliminary steps must be taken in the form of sub-questions that need to be answered. The first set of sub-questions relates to exploring and analysing the system's behaviour. Afterwards, the second set of sub-questions relates to solving possible problems encountered in the first set. Lastly, the final stage of sub-questions revolves around evaluating the proposed solution from the second set of sub-questions.

The first set of sub-questions consist of:

How can the performance of satellite networks be defined and measured?

How do the most common TCP CCAs compare to each other regarding their throughput performance over Starlink satellite networks?

What notable observations can be made when evaluating the throughput performance of the Starlink satellite networks?

In the second phase of the thesis, the question revolves around solving the observed problems in the initial phase;

How to increase throughput performance over Starlink satellite networks?

How can the proposed solution be developed in a way that it can implemented and tested in a real network?

And the final phase revolves around evaluating the developed solution for the satellite network based on the standard requirements of TCP CCA; these questions are:

How does the performance in terms of throughput of the solution compare to the existing standards?

How does the performance in terms of convergence and fairness of the solution compare to the existing TCP standards?

5 Phase I: Starlink Exploration

This section of the report will focus on the exploratory phase of this research. The approach will be elaborated upon, followed by the measurement results and completed by the discussion. The findings of this phase will form the basis of the motive behind the second phase of the thesis.

5.1 Approach

The approach will first discuss the measurement setup for gathering the relevant metrics describing the behaviour of the Starlink system. Then it will discuss the tools that will be used for enabling the measurement, followed by the mathematics and statistics that will be used for analysis.

5.1.1 Measurement setup

To observe and evaluate the performance of Starlink internet, the University of Twente purchased a Starlink dish combined with a basic user-subscription. At the top of the Vrijhof building located at the University of Twente in Enschede, The Netherlands (52° 14'35.9124"N 6°51'12.528"E), a circular Generation 1 Starlink dish has been placed. Its location on the building can be seen in Fig. 8. Do note that the specifications for these antennas can vary significantly depending on the generation of the dish, which is why the specifications are mentioned in more detail in Table 1.

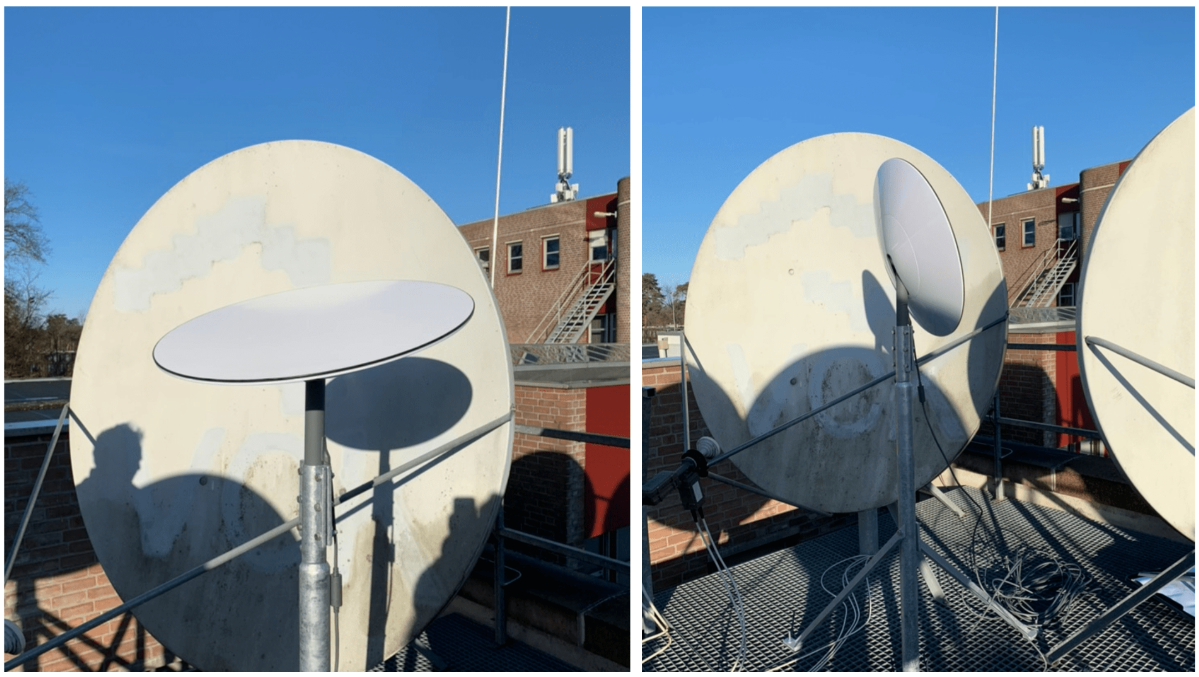


Figure 8: Starlink dish located at the Vrijhof building at the University of Twente, Enschede.

The Starlink dish has Ethernet (wired) connection to a server that is also located on campus at the University of Twente and can be accessed through a Secure Shell (SSH) connection. This will be referred to as the ***Starlink server***, and can be used to send and receive data over the wireless Starlink link (Satellite to user terminal) for the measurements. Information regarding the Starlink server's specification can be found in Table 2.

An important metric for evaluating the performance of an internet connection is measuring the throughput (upload and download speeds) of the Starlink wireless link. To gather metrics related to the throughput over the Starlink satellite link, another server is required from which packets can be sent and received. This will be referred to as the ***Measurement server***. To decrease the chances of artefacts within the measurements, one would ideally keep the number of intermediate hops between the sender and the

Description	***Specification***
Dish generation	Gen 1
PoE IEE Standard	IEEE 802.3bt
TX Frequency	14.0-14.5 Ghz
RX Frequency	10.7-12.7 GHz
Diameter	60 cm
Operating temperature dish	$-30°C$ to $40°C$
Router & Power Supply operating temperature	$10°C$ to $30°C$
Input Voltage	100 to 240V
Input Current	2.5A max (50-60 Hz)

Table 1: Starlink Dish Specifications

Description	***Specification***
Architecture	x86_64
Vendor ID	GenuineIntel
Model Name	Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz
Operating System	Debian GNU/Linux 11 (bullseye)
Kernel	Linux 5.10.0-9-amd64

Table 2: Starlink server specifications.

receiver to a minimum when doing measurements. To do so, we must looked at the nearest Starlink gateways that will most likely be used for communication with the dish. The nearest gateways to the Starlink Dish are located in Aerzen (52°03'39.6"N 9°19'41.6"E), Usingen (50°19'47.7"N 8°28'14.8"E) in Germany and Gravelines (50°59'22.7"N 2°09'28.4"E) located in France, as visualised in Fig. 9. Do note that these locations are based on information retrieved from a third-party website [43] as Starlink itself has not disclosed official information regarding the locations of its base stations.

Description	***Specification***
Architecture	x86_64
Vendor ID	GenuineIntel
Model Name	Intel Core Processor
Operating System	Debian GNU/Linux 11 (bullseye)
Kernel	Linux 5.10.0-13-cloud-amd64

Table 3: Measurement server specifications.

For measuring and observing the performance of the Starlink Dish, a server from a Cloud Provider located in Frankfurt, Germany, will be used, of which the hardware and operating specifications can be found in Table 3. This is a decent location for the server as it is near the Starlink gateways to ensure fewer intermediate hops between nodes when doing the internet measurements. Additionally, from initial traceroute measurements it appeared that Starlink communication took place going through the Deutscher Commercial Internet Exchange (DE-CIX) in Frankfurt. This observation is why we also ensured that the chosen measurement server is connected to this exchange point.

An overview of how the Starlink server and measurement server are connected to each other can be seen in Fig. 10. Do note that this setup assumes that the wireless link between the Starlink dish and the satellite will be the bottleneck within the communicating links, which enables us to do a throughput measurement for this part in the link. This is a valid assumption, as most modern wired links can achieve speeds that are larger than Starlink's promised 100 Mbps download speeds (average user subscription download speed

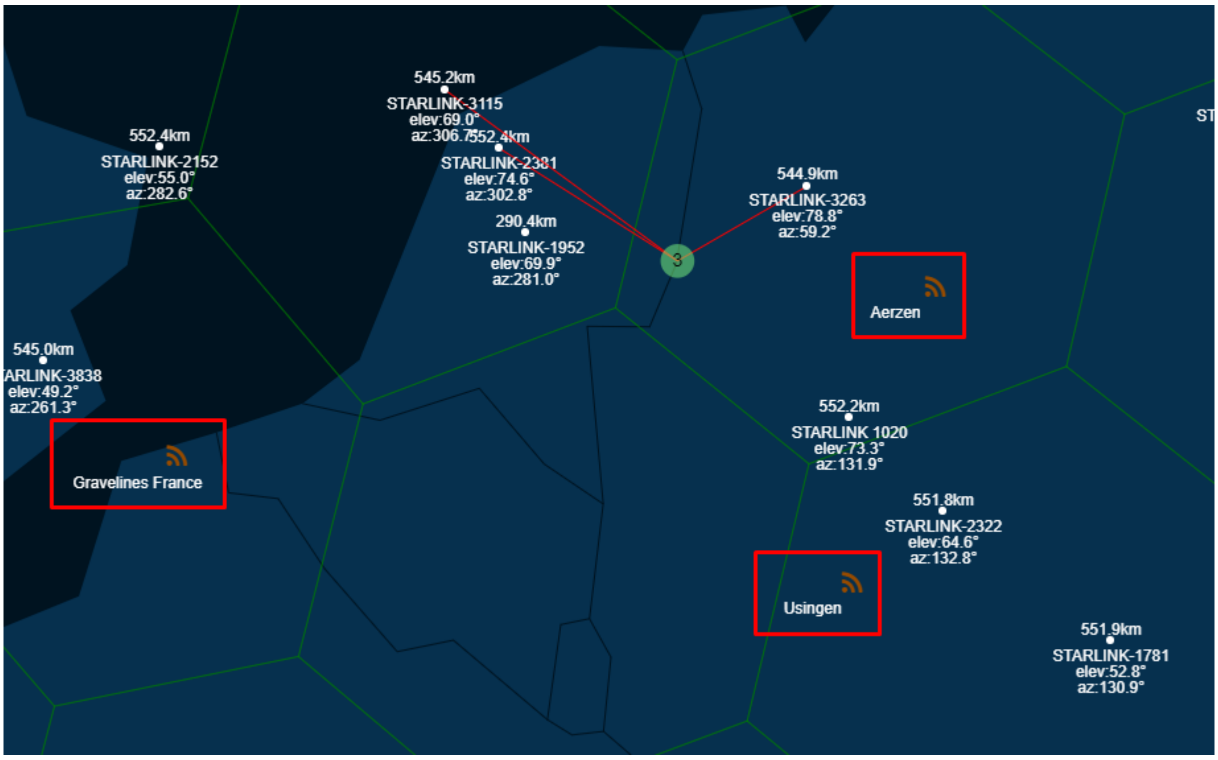


Figure 9: Nearby Starlink gateways. [43].

[12]). Additionally, we ensure that the measurement server has a bandwidth capacity of around 500 Mbps to have enough bandwidth for the Starlink throughput measurements.

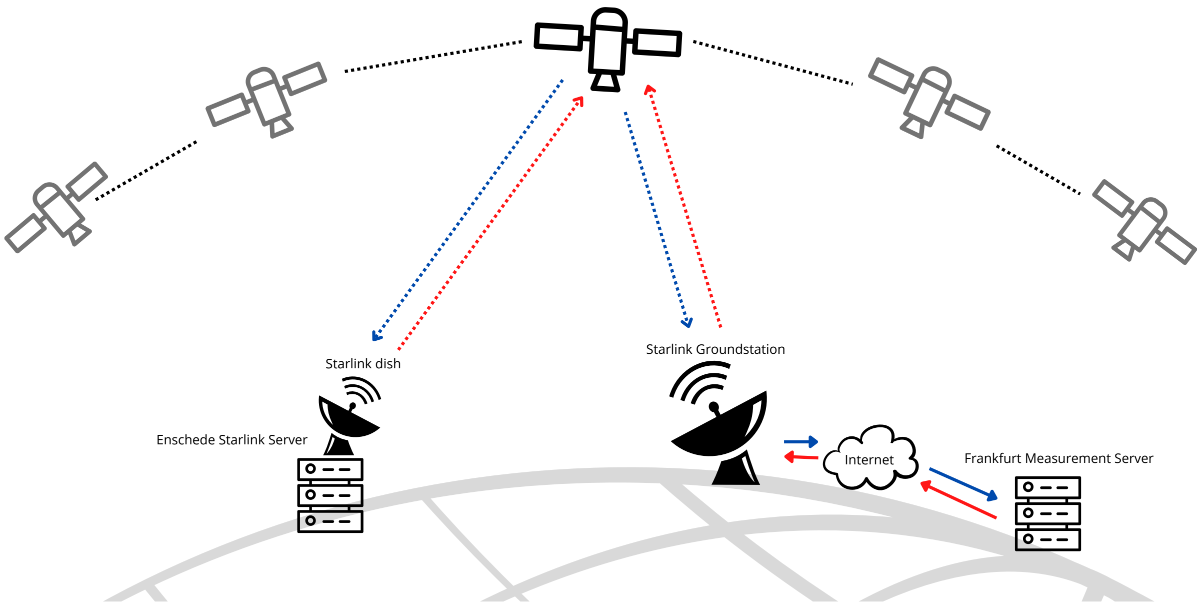


Figure 10: Measurement setup.

5.1.2 Tools & Analysis

The iperf3 [44] tool will be used to measure the estimated download speed over the satellite links (from the measurement server and Starlink server). Throughput consists of both upload speed and download speed. However, for this research, we will only look at the download speed. The reasoning behind this is that the download speed is of more relevance for most users based on their (daily) activities on the internet. Iperf3 is an active measurement tool that can be used for measuring both TCP and UDP traffic performance by generating corresponding flows to load the network. By doing so, the tool enables the monitoring and logging of relevant parameters related to the throughput such as congestion window size, bandwidth, RTT and retransmissions (packet loss). Furthermore, iperf3 also enables the user to specify measurement parameters such as timing, duration, congestion control protocol, ports, amount of flows and interval time between samples.

For gathering the relevant metrics relating to the download speed of the Starlink system, iperf3 will be installed and utilised on both the Starlink server and the measurement server. Using the tool, we can specify the Starlink server to be the receiver and the measurement server to be the sender. Ultimately, parsing the tool's logs at the receiver side will provide an overview of the throughput of the measurement, and the logs from the sender side will give insight into the CWND and the retransmissions.

Different CCAs will be used to look at the throughput and its parameters. This is because the modern internet also consists of various CCAs, each having other mechanisms for estimating the available bandwidth and dealing with possible congestion. Even though there exists a large variety of these CCAs, we will only look at 3 to narrow the scope of the research. These algorithms are:

- ***TCP Reno***; because it was the former standard congestion control algorithm. Additionally, as mentioned in Section 2.2.4.1, as it is a basic TCP CCA, it enables us to see the linear growth of the CWND and adjustments made according to the perceived packet loss.
- ***TCP Cubic***; because the congestion control algorithm is current (widely implemented) standard in modern computers (default CCA in most Linux-based machines). Evaluating this CCA will provide us with insights to how users' internet performance will be affected when using a Starlink internet connection, as a lot of the servers they will be accessing data from will be Linux-based with default TCP settings.
- ***TCP BBR***; because next to CUBIC, it is a relatively new CCA and is widely implemented by Google servers. Regular internet users will most likely also make use of these servers that have BBR when accessing Google services, e.g. Youtube, Google Search, Gmail etc...

Between measurements, the CCA at the sender side (measurement server) will be updated using the *sysctl* to modify the *net.ipv4.tcp_congestion_control* parameters. To gather throughput information on the different CCAs, the duration of the iperf3 measurement will be set to 10 minutes (600 seconds). We argue that this will give us enough exploratory insight regarding the throughput behaviour of the system. The interval at which iperf3 checks and logs the number of retransmissions will be set to 0.1 seconds. We do this to monitor the retransmissions accurately in the time. Iperf3 will return a .json formatted file for each measurement and these will be parsed and visualised using matplotlib and seaborn in python3.

5.1.2.1 Moving Average & Linear Interpolation

The previously mentioned approach will be applied in two scenarios. Firstly, to observe the behaviour of the throughput parameters over a particular measurement and secondly to follow the throughput over a longer duration. To analyse how the throughput performance behaves over a more extended period, we combine individual 10-minute measurements over 24 hours. This amount would be enough to give us insight into the possible day and night cycles of the available bandwidth in the connection. Do note that for this measurement only the CCAs Cubic and BBR will be looked at as these are widely used. The throughput of Reno over a 24-hour is not of great significance as it would not be as representative of average internet usage as compared to the other two. Furthermore, to decrease the chances of errors corrupting the data, the duration of the throughput will be measured in samples of 10 minutes with 5-minute gaps in between. This is done using the *crontab* tool installed on both measurement and Starlink server. At the times of the measurement, the timestamps will also be logged. This is executed in samples instead of one continuous (24-hour) measurement, as errors may result in the JSON file becoming corrupted and thus not being able to be parsed for analysis.

Regarding the legibility of the 24-hour throughput measurement, it can become difficult to interpret the throughput values as they can fluctuate significantly over the duration of the measurement. To solve this, the statistical technique referred to as the moving average will be used. This technique is commonly utilised for smoothing data that is inherently prone to large fluctuations. By taking the moving average over the series of data, one is able to create less noisy graphs from which it is easier to see increases and decreases in the data. The equation for the simple moving average is given as follows:

$$SMA_k = \frac{p_n - k + 1 + p_n - k + 2 ... + p_n}{k} \tag{2}$$

Which can be further simplified to:

$$SMA_k = \frac{1}{k} \sum_{i=n-k+1}^{n} p_i \tag{3}$$

- Where k denotes the number of entries over which the moving average will be taken.
- and n is the total number of data-point entries.
- p_i denotes the value at index i.

Another consideration is that we will measure the throughput discretely and not continuously, however we will visualise it as if it were continuous. To compensate for the missing throughput gaps, linear interpolation (Equation 4) will automatically be implemented in the line plotting code after parsing the data points for the measured throughput.

Linear interpolation is defined as follows:

$$y = f(x) = y_1 + \frac{(y_2 - y_1)(x - x_1)}{x_2 - x_1} \tag{4}$$

- Where x_1 (time since measurement started) and y_1 (measured throughput) denote the values before the gap.
- and x_2 (time since measurement started) and y_2 (measured throughput) denote the values after the gap.
- x denotes the point to which interpolation will be done.
- y denotes the interpolated value.

5.1.2.2 Autocorrelation Function

Next to the 24-hour throughput measurement, it is also important to look at the retransmissions that take place over the Starlink connection, because the amount of retransmissions determines the CWND growth for a majority of the loss-based CCAs (Cubic & Reno). Furthermore, measuring the periodicity of the loss events, the time between recurring retransmissions must be observed and analysed. After using iperf3 to collect the data regarding the retransmissions at specific times, its information will be processed using the Pearson autocorrelation function. With this function the signal will be compared with a delayed version of itself, from which the correlation will be output as an integer number between 0 and 1. By doing so we are able to observe higher and lower correlations at different delays of the signal, which may assist in determining higher periodic retransmissions. Within the data processing code, the autocorrelation method will be used for analysing the retransmissions in the time domain. For each specified delay (ranging from -600 seconds to 600 seconds), the Pearson correlation will be calculated between the retransmissions and a delayed/shifted version of itself. The Pearson coefficient is a method for determining how variables are correlated with each other and is defined as follows:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i^2 - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i^2 - \bar{y})^2}} \tag{5}$$

- Where n denotes the total sample size.
- x_i denotes the value of the retransmission at a given time.
- y_i denotes the value of the retransmission after being delayed a specific value.

5.2 Results & Discussion

This section of the report will discuss the exploration of the thesis. The first part will cover the observed throughput over time, whereas the following section will cover the CWND & retransmissions of the different congestion control protocol. Lastly, the final section will display and discuss the measured retransmissions.

5.2.1 Throughput Over Time

The results and discussion of the 24-hour sample are displayed below. The measurement metrics that comply with the graph results are elaborated upon in the discussion.

5.2.1.1 Results

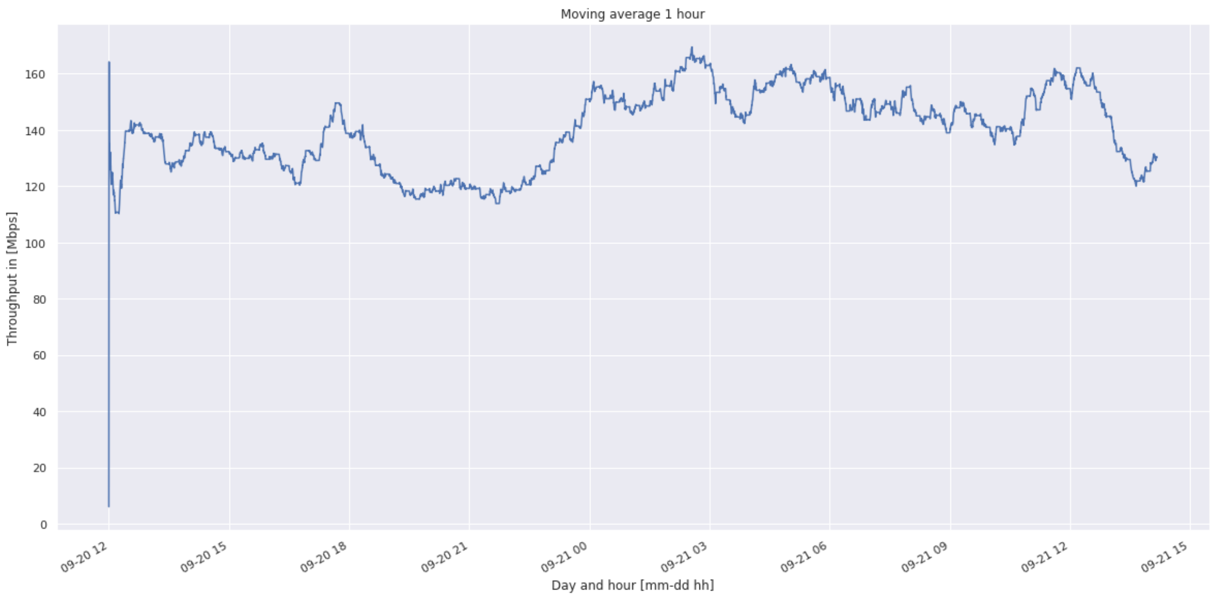


Figure 11: Sampled & interpolated throughput of Cubic over a single day taken with 1 hour moving average.

5.2.1.2 Discussion

Fig 11 shows the average throughput measured at the receiver while the congestion control algorithm at the sender side was Cubic. The measurement lasted around 24 hours from the 20th of September to the

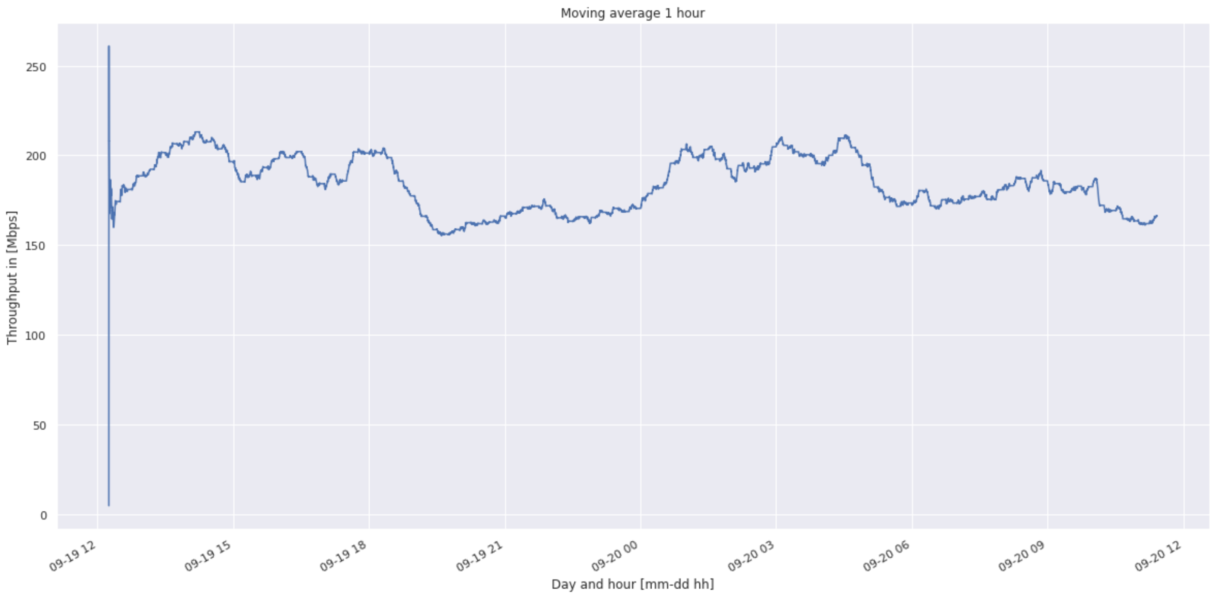


Figure 12: Sampled & interpolated throughput of BBR over a single day taken with 1 hour moving average.

21st of September 2022. The results for this measurement show significant increases and decreases in the measured throughput at some times during the measurement day. For example, between 18:00 and 23:00, there seems to be a significant drop in measured throughput. A possible reason for this is many users end their work day (9 am to 5 pm), arriving home and using the internet for entertainment services such as streaming. If many users do this, less bandwidth will be available per user. After 23:00, many users may go to bed, thus not utilising the capacity available in the Starlink system. This results in more bandwidth being available. Therefore iperf3 can send more data, resulting in larger observed bandwidth from this point. Another interesting finding is at the start of the measurement (09-20 12). There seems to be a significant spike in throughput as compared to the rest of the measurement, reaching a throughput larger than 160 Mbps. This is an artefact that results from the moving average window not having any values to take the average over in the beginning.

Regarding the observed throughput over 24 hours from BBR displayed in Fig.12, we observe a similar significant throughput decrease between 18:00 and 23:00. The reasoning behind this is similar to Cubic. BBR does seem to have a more significant average throughput than Cubic when comparing these two 24-hour samples. However, it is impossible to say that BBR consistently achieves larger available bandwidth utilisation based on a comparison of Fig. 11 with Fig. 12, as these measurements are not taken simultaneously. Possible measurement artefacts resulting in the significant difference may be the difference in weather during these measurements. Additionally, similar to Cubic, BBR also shows a significant spike in observed throughput in the beginning of the measurement (09-19 12), similar to that of Cubic, this is also an artefact resulting from the rolling mean window at the beginning.

5.2.2 CWND & Retransmissions

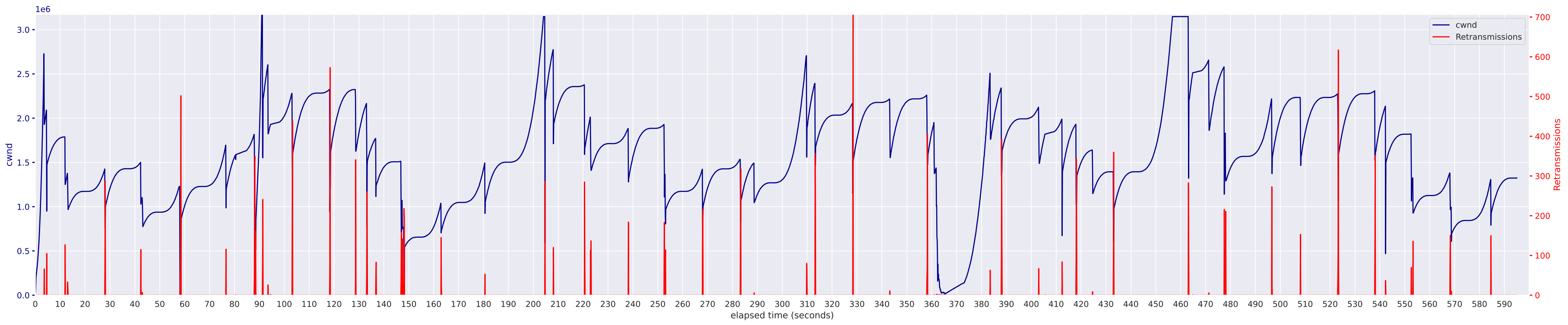


Figure 13: Cubic CWND value and retransmissions plotted over time.

Fig. 13 shows how the CWND of Cubic evolves over time of a single 600-second measurement. Cubic's slow start at the beginning of the measurement shows no abnormal behaviour. The CWND value seems to exponentially ramp up to a value of around *2.6 * 1e6* Bytes before experiencing packet loss. This results in the CWND being multiplicatively decreased (*0.7), setting the CWND at the time of loss as the Wmax value to which the CWND will grow in an exponentially decreasing manner. At brief moments, Cubic can achieve a significantly large CWND, which is larger than *2.6 * 1e6* Bytes. These can be found at around elapsed time values of 90, 205, and from 460 to 480 seconds. However, the CWND does not stay large as it re-scales as a result of packet loss. Furthermore, around 360 seconds, there seems to be a significantly smaller value of CWND. This is the result of the CWND constantly re-scaling (multiplicatively decreasing) as a result of very small but sequential retransmissions over a 5 to 7-second period. Possible causes for this phenomenon could be the weather or an unknown object obstructing the view of the sky perceived from the dish. Without applying any methods for periodicity analysis, it can be observed that recurring retransmissions take place at the same intervals. Examples of these are around 28 seconds to 43 seconds and at 43 seconds and 58 seconds. There seem to be predictable moments at which retransmissions take place (around every 15 seconds). This is an abnormal phenomenon, as true congestion would only occur when one of the intermediate nodes becomes congested. These would then be moments when a certain CWND value is reached, however, this is clearly not the case. It is unclear what exactly causes the periodic packet loss; however, possible reasons can be the updating of the routing tables in the satellites or handovers taking place. These relate to layer 2 and layer 3 of the communication, into which we have no insight in the given setup (seen in Fig. 1).

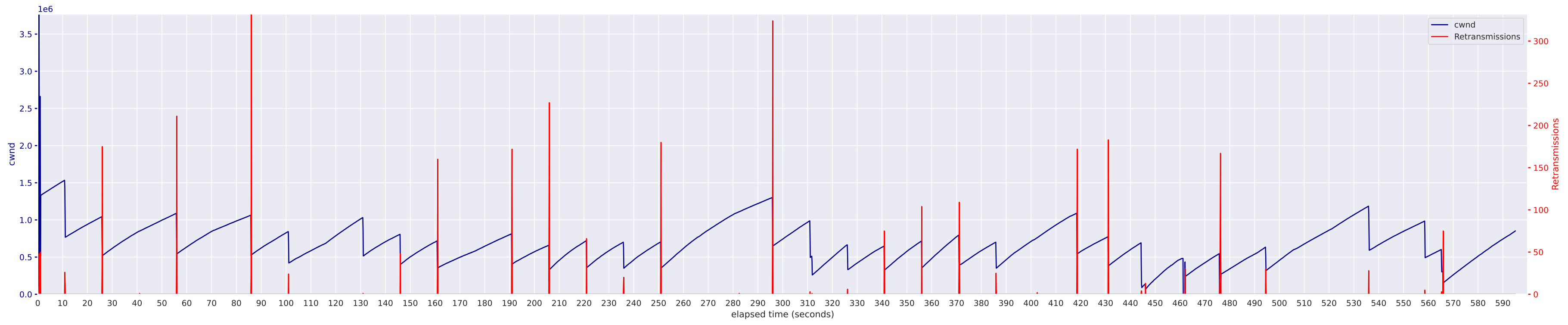


Figure 14: Reno CWND value and retransmissions plotted over time.

Fig. 14 shows how Reno's congestion window value CWND evolves over time. Slow Start seems to quickly ramp up the CWND to around *2.6 * 1e6* Bytes. Then packets are lost, and retransmissions take place. Hence Reno transitions into congestion avoidance mode, where it increases the CWND linearly. When packet loss occurs, it seems to halve the CWND and restarts the growth. The highest CWND value achieved seems to be around *3.6 * 1e6* Bytes; however, this value is only achieved during the slow start phase, where the CCA attempts to estimate the baseline CWND from which to work from. Afterwards, this value is never achieved again and those achieved are significantly lower (around *1.0 * 1e6* Bytes). Interestingly, there also seems to be a moment where the CWND becomes significantly low at around 445 seconds. This is a result of small sequential amounts of packet loss. Similar to Cubic, a reason behind this could be an unknown object obstructing the dish's view of the sky or the artefacts introduced by the weather introducing additional packet loss. Also, like Cubic, without applying any methods for periodicity analysis, it can also be observed that recurring retransmissions take place at the same intervals. From 190 seconds to 251 seconds, there seem to be retransmissions at specific intervals of 15 seconds. These findings suggest that other loss-based CCAs will also suffer from these periodic losses that do not seem to result from actual congestion. More insight into these periodic retransmissions will be given in Section 5.2.3.

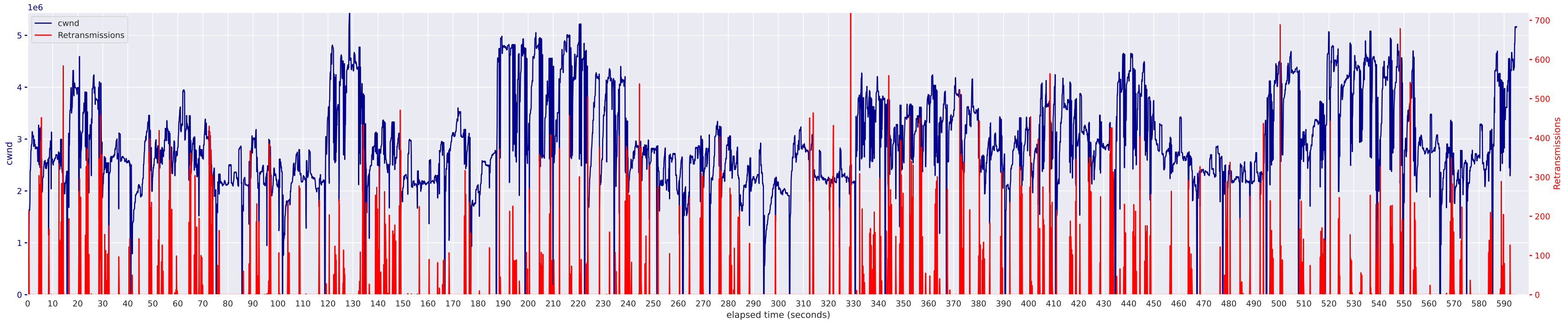


Figure 15: BBR CWND value and retransmissions plotted over time.

Fig. 15 shows the CWND achieved by BBR and contains compared to the Fig. 13 and Fig. 14 very large values for the congestion window, along with frequent retransmissions. During the slow start phase, BBR seems to achieve a CWND of around *3.0 * 1e6* Bytes. However, BBR will increase its CWND to achieve as much throughput as possible once in congestion avoidance mode, ignoring the packet loss. This indicates that BBR is behaving as expected. Furthermore, the retransmissions do not seem to have the same amount of interval time in between. It is unknown whether retransmissions within 15-second intervals are not observable as a result of a large number of retransmissions or whether these do not occur with BBR based on these results. At some points, we do observe that BBR's CWND drops significantly (to around 0) for a brief moment before quickly recovering. These moments can be found at 86 to 101 seconds and 101 to around 117. Another example is 166 to 181 seconds. A possible cause for this is a change in RTT, which BBR's CWND scaling is dependent on. This was originally observed by the Starlink Analysis paper [38]. In order to confirm this, we generate Fig. 15 once more with RTT as well.

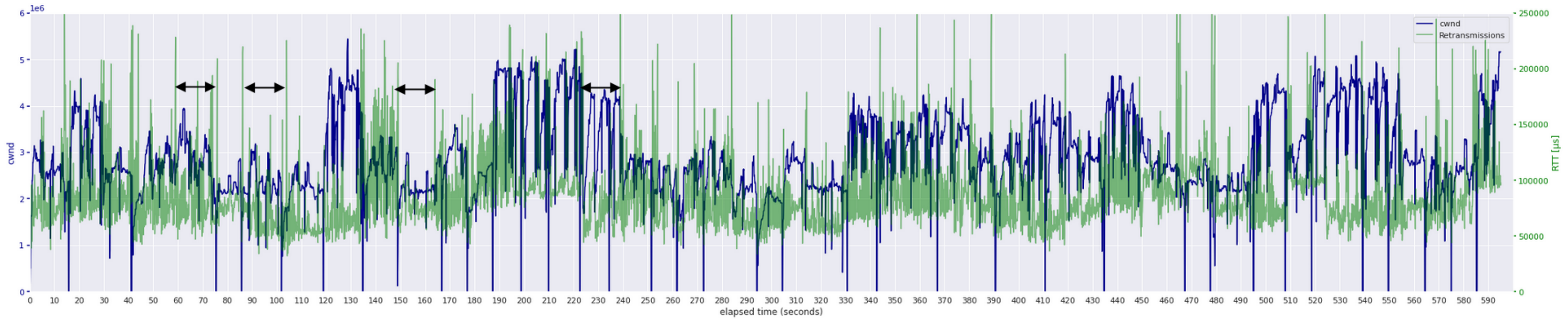


Figure 16: BBR CWND value and RTT plotted over time.

When looking at Fig. 16, we see that both the CWND in bytes and the RTT in µseconds. Although from the retransmissions, it could not clearly be seen that there was a 15-second periodicity, we can see abnormal behaviour from the RTT as well. At multiples of 15 seconds, there seems to be a sudden increase in RTT. This is denoted by the black arrows at 60 seconds to 75 seconds, 85 seconds to 100 seconds and again at 150 seconds to 165 seconds. Do note that there are more of these spikes present in the figure; however, these are not identified by the arrows. It is unknown what exactly causes these spikes are in RTT. Possible reasons are that the satellites their routing tables get updated when the gateway force pushes the updated tables (Network layer - OSI layer 3) or it could be a side effect of the Time Division Multiple Access (TDMA) mechanism implemented in Starlink (Data-link layer - OSI layer 2).

5.2.3 Autocorrelation Results & Discussion

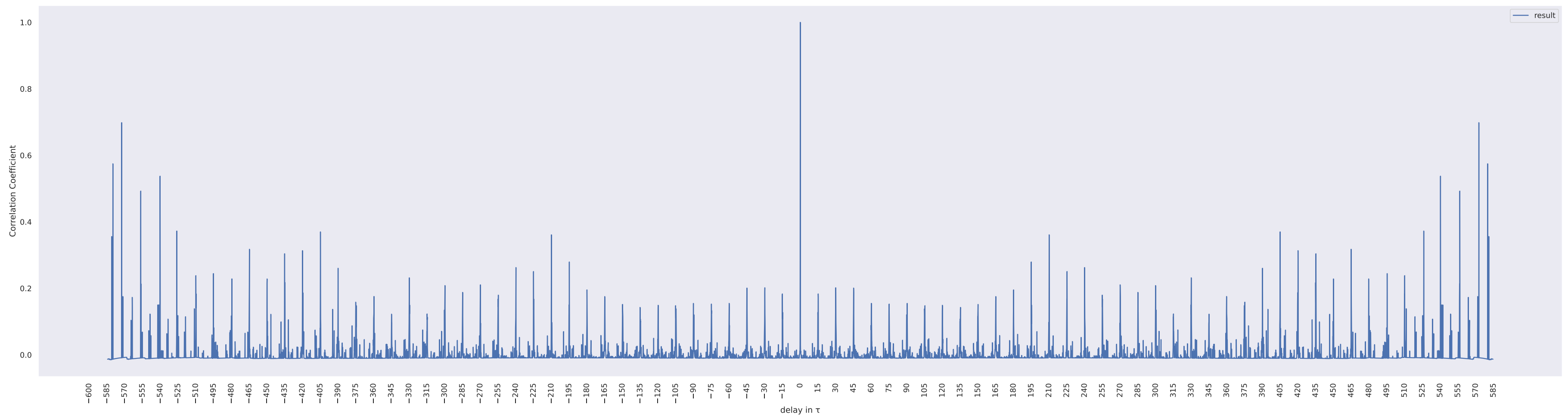


Figure 17: Cubic autocorrelation of the retransmissions.

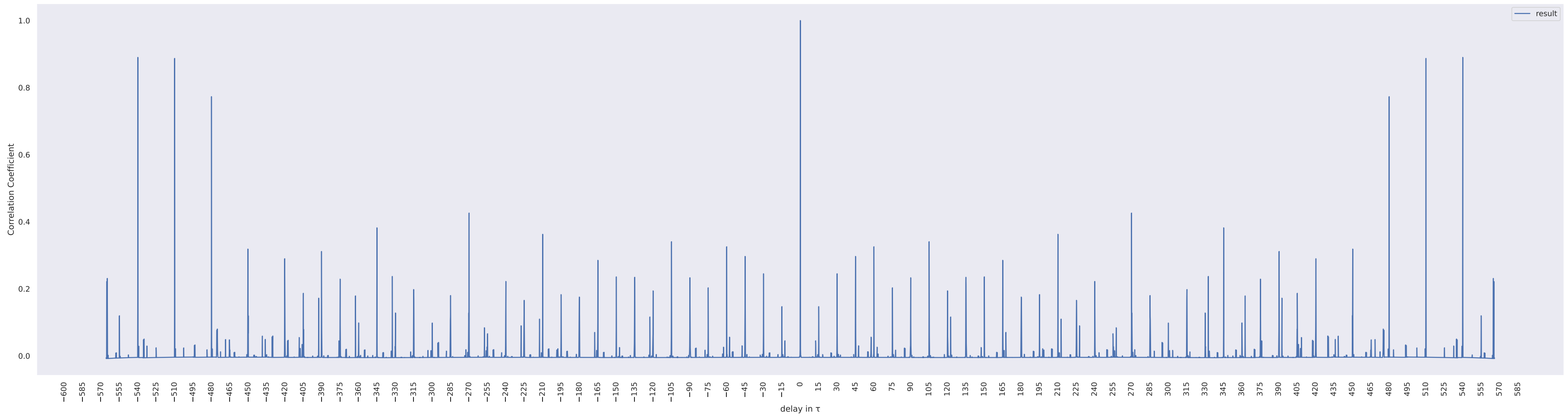


Figure 18: Reno autocorrelation of the retransmissions.

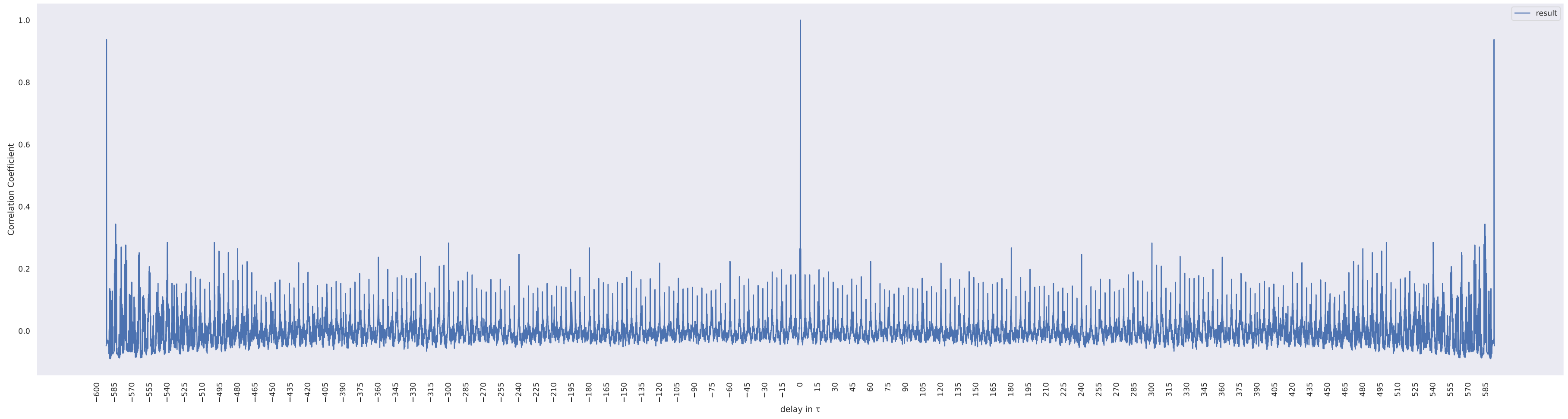


Figure 19: BBR autocorrelation of the retransmissions.

In Fig. 17, we can see at a delay of 0 seconds that there is a Pearson correlation of 1, which is expected as the signal is compared to itself. In addition to this significant spike, for a majority, there seem to be low correlations, other than multiples of 15 seconds. This further strengthens the argument about the recurrence of packet loss at multiples of this 15 second period. Slightly higher than average correlations can be found at a delay of, e.g. 210 and 405. The reason behind this is the number of retransmissions. When the amount of retransmissions overlaps with another large amount during a specific delay, the Pearson correlation output is higher than the others. This predictable packet loss is an abnormal observation and influences how Cubic increases its CWND, affecting the possible throughput measured at the receiver.

This observation can also be made for the other loss-based CCA, Reno displayed in Fig. 18. Here it is possible also to see similar higher correlations at multiples of 15 seconds.

Furthermore, the Pearson correlation graph of BBR in Fig. 19, shows no higher correlations at multiples of 15 seconds. A possible reason for this is due to the large number of retransmissions taking place in BBR. This large number of retransmissions results from BBR constantly trying to find more bandwidth within the link while ignoring the number of retransmissions.

These findings confirm that the default congestion control algorithm, Cubic, that is used in a lot of computers suffers from these recurring moments of packet loss and retransmissions. As a result, the congestion control algorithm “thinks” that congestion is taking place when in reality, no actual congestion is taking place at these 15-second intervals. This ultimately results the available bandwidth being underutilized, which is not ideal for the system’s end users, as their achievable download speed will be less than actually possible.

6 Phase II: Prototyping (SICC)

During phase I, we determined that loss-based CCAs (Reno & Cubic) suffer from periodic packet loss and showed how it resulted in the CWND decreasing, thus also reducing the download throughput measured at the receiver side. During the second phase, we will develop a CCA based on the prior observed inefficiencies. This section of the report will first describe the approach for developing the problem's solution, followed by results that test the developed solution. Lastly, we will discuss the results.

6.1 Approach

The most deployable and practical way to modify the standard TCP protocol that the communicating computers use is by developing a loadable kernel module. These modules enable users to load and unload logic rules in code into the kernel on demand and can thus be used to modify the rules for TCP communication. In the approach, we first define a set of requirements that the TCP kernel module will have. Then we proceed with describing the implementation of the basic CCA rules in the kernel module, followed by implementing the features that solve the issues observed over the Starlink communication link.

6.1.1 TCP CCA Development Requirements

Since TCP Cubic is the de facto standard for most TCP protocols in many computers in the current internet [45], it can be assumed that most people using the Starlink system for internet access also suffer from the observed decrease in performance. From the measurement results, it showed that one of the primary decreases in throughput performance was the result of the Cubic's congestion windows decreasing in size unnecessarily. In conventional/terrestrial networks, they only decrease in size when congestion occurs somewhere throughout the network. These observations lead to the need for a new TCP congestion control protocol that will solve the inefficiencies caused by the congestion window decrease as a result of a non-congestion event. The solutions to these inefficiencies can be redefined as the requirements for the proposed congestion control algorithm.

The ***Space Internet Congestion Control (SICC)*** protocol will be developed based on several different requirements. The first is that the protocol will achieve throughput speeds at least as fast as that of Cubic but ideally faster. Furthermore, as mentioned in Section 2.2.4.3, Cubic's main competitor in the modern internet, BBR, tends to decrease Cubic's bandwidth usage on a link to be able to use more for itself when different streams are present. One can argue that this is relatively unfair for the servers communicating with Cubic as its congestion control because BBR inherently decreases their performance. Thus, regarding fairness, SICC should be fairer to Cubic compared to BBR.

Second, SICC's congestion window should not decrease due to expected (15-second) retransmissions. It will recognise the retransmissions that are not a result of congestion and will, as a result, not reduce its congestion window but keep the CWND value at its current size and along with the growth mode (convex or concave profile) that it is in. Some aspects of the TCP congestion control based on Cubic will be briefly discussed in Section 6.1.2. With aspects is meant the logic behind the congestion control algorithm in given scenarios.

Third, we acknowledge that part of its communication takes place over a wireless link, which is prone to errors in the physical layer. Because these also result in congestion window decreases, SICC will need a mechanism for not falling victim to random loss (high BER) and keep throughput as high as possible.

And lastly, to implement SICC in existing servers for real-life measurement testing and utilisation, it will be developed as a Linux kernel module. Linux kernel modules allow users to download, compile and insert kernel modules into their Linux systems. By doing so, they can, in our case, change the congestion control algorithm rules for the system in which they loaded the kernel module and specified that the system should use it. A Linux kernel can consist of several different modules that interact with hardware and applications through the kernel core as visualised in Fig. 20.

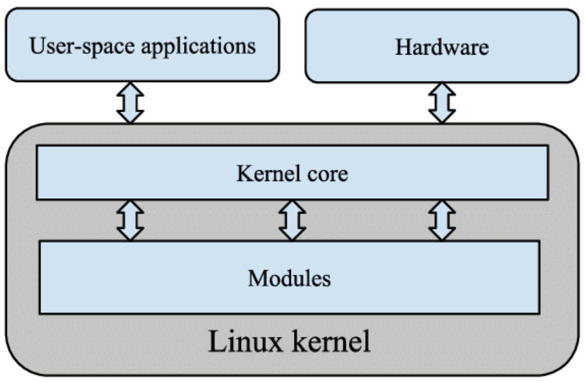


Figure 20: Linux kernel module and how it interacts to its environment [46].

6.1.2 TCP default Parameters

In order to avoid conflict with Cubic (and other possible loss-based TCP CCA) streams possibly present on the Starlink connection, SICC's algorithm will largely be based on that of Cubic. This will ensure that SICC will behave similarly to Cubic in different scenarios, which mainly ensures fairness between heterogeneous streams. These functionalities are based on Cubic's RFC [47] along with their research paper written by Ha et al. [27]. Their implementation will be described in more detail in the following paragraphs.

6.1.2.1 Window Increase Function

SICC will increase its congestion window at the sender side after every acknowledgement (ACK) sent back from the receiver. Whenever a retransmission takes place, the value of the congestion window (CWND) at that time is stored under variable $\mathrm{W_{max}}$. However, in the case of duplicate ACKs, (i.e. a congestion event), SICC will also use the equation below for its window growth function.

$$W_{SICC}(t) = C(t-K)^3 + W_{max} \tag{6}$$

In Eq. 6, C denotes a fixed constant, a Cubic parameter that determines the aggressiveness in networks with a relatively large BDP. And t denotes the time elapsed since the beginning of the congestion event. Lastly, K denotes the period that Eq. 6 takes in order increase the current value of CWND to the $\mathrm{W_{max}}$ (when no other congestion events occur) and can be described as:

$$K = \sqrt[3]{\frac{W_{max}\beta}{C}} \tag{7}$$

In Eq. 7, β describes the value of the multiplicative decrease factor. Which is used to determine how much the value of the CWND will decrease in the case of retransmissions. Whenever a re-transmission takes place, SICC will reduce its CWND to:

$$W_{SICC}(0) = W_{max} * \beta \tag{8}$$

When this happens, similar to Cubic, SICC enters congestion avoidance mode. After the retransmission of the lost packets has taken place, it will compute the rate that CWND will increase during the following

RTT period using Eq. 6 after receiving an ACK. The calculated value of $W_{\text{SICC}}(t+RTT)$ is set as the target for the CWND.

From this point SICC will be transition into different profiles depending on the value of the CWND. These are:

1. **TCP-Friendly region**: Where SICC tries to achieve atleast the same throughput as standard TCP.
2. **Concave region**: When not in TCP friendly mode AND CWND $<$ W$_{\text{max}}$.
3. **Convex region**: When not in TCP friendly mode AND CWND $>$ W$_{\text{max}}$.

Each of these modes will be discussed in more detail in the following paragraphs.

6.1.2.2 TCP-Friendly Region

Standard TCP congestion control algorithms such as Reno, still perform well under links and networks that do not face the large bandwidth delay product. In networks that have e.g. shorter delays (RTTs) and/or smaller amounts of bandwidth, standard TCP (Reno) is thus still preferred. In scenarios with short RTT and$/or$ small bandwidths, like Cubic, SICC will perform in a similar way.

Upon receiving an ACK while in congestion avoidance mode, SICC must first know what TCP region it is in. Depending on the region, its additive increase and multiplicative decrease values must be adjusted.

$$\text{AVG } W_{aimd} = \sqrt{\frac{\alpha_{aimd} * (1+\beta_{aimd})}{2*(1-\beta_{aimd} * p)}} \tag{9}$$

But can also be described according to the research paper [27] with the RTT as:

$$\text{AVG } W_{aimd} = \frac{1}{RTT}\sqrt{\frac{\alpha_{aimd}}{2}\frac{2-\beta_{aimd}}{\beta_{aimd}}\frac{1}{p}} \tag{10}$$

Using the above equation, the protocol will estimate that value that CWND should have in TCP friendly mode, depending on the value of α and β. From this [27] noted that the value of α should be equal to $\frac{3\beta}{2-\beta}$, which enables the protocol to calculated the desired CWND value:

$$W_{est}(t) = W_{max}(1-\beta) + 3\frac{\beta}{2-\beta}\frac{t}{RTT} \tag{11}$$

6.1.2.3 Concave and Convex Profiles

Concave and convex window adjustment profile are also implemented in SICC as they provide more stability in networks that are under high utilization [27]. When SICC is in congestion avoidance mode and it receives an ACK while also not being in the TCP-friendly region and CWND is smaller than W$_{\text{max}}$, the congestion control will transition into the Concave region where the CWND will be incremented using:

$$CWND = \frac{W_{SICC}(t+RTT) - CWND}{CWND} \tag{12}$$

Where $W_{SICC}(t+RTT)$ is calculated using Eq. 6.

When SICC is in congestion avoidance mode and it receives an ACK while also not being the TCP-friendly region and CWND is larger than or equal to W_{max}, then the SICC congestion control algorithm will also be in the convex region. In this mode, similar to Cubic, SICC will try to exponentially increase CWND to see if there is more available bandwdith on the link. In this region, SICC hopes to find a new, larger value for W_{max}. For this region Eq. 12 is also used.

6.1.3 TCP Starlink Modification

The previous chapters discuss the logical foundation of what SICC is built. However, these are similar to that of Cubic. What sets the SICC congestion control protocol apart from Cubic is how its multiplicative decrease is triggered for CWND scaling.

From the observations, the value of the congestion window CWND was hindered by the recurring retransmissions every 15 seconds. As a result, the congestion control algorithm would "think" that there is congestion is present, thus decreasing the value of CWND, resulting in lower overall throughput. Additionally, because the Starlink link is wireless, i.e. lossy pipe, random loss also frequently occurs.

Normally, when packet loss has occurred, and transmissions take place:

- W_{max} = CWND; the window size is saved before reduction (W_{max} = CWND).
- sshtresh = CWND * β; new slow-start threshold is set.
- sshresh = max(sshtresh, 2); ensure that maximum segment size (MSS) is at least 2.
- CWND = CWND * β; multiplicative decrease to ensure CWND reduction.

One way of ensuring that the CWND retains its value during the expected retransmissions is by conditionally manipulating the value used for its multiplicative decrease β and maintaining the current mode the congestion control algorithm is in. Whenever expected retransmission occurs (at around 15 seconds from the previous retransmission) in SICC, the congestion window, CWND, should not decrease and stay at its current rate. One way of doing this is by adjusting the multiplicative decrease variable value β to 1 instead of the default value of 0.7.

Within the kernel module C code, an array of fixed size is created as a mechanism for checking whether the retransmissions are a result of congestion or a result of the unknown 15 seconds. In order to check whether the retransmissions are expected, the timestamp is required and is thus an important metric to monitor. Fortunately, it is possible to acquire this metric in kernel modules by using the jiffies variable from the kernel. Since the booting of the system, the kernel maintains a global variable called jiffies, where it stores the number of CPU ticks. In computer systems, this tick rate is associated with the CPU in Hz and can be defined as:

$$jiffies = seconds * Hz \tag{13}$$

From which the seconds can be derived by reformulating the equation as:

$$seconds = \frac{jiffies}{Hz} \tag{14}$$

Initially, an array of fixed size will be created, of which all values at the indices will be filled with a value of 0. Whenever retransmissions take place, the time in seconds will be retrieved from the system, and the value difference will be checked with the existing values in the array. After completing the check, the value will be stored at the last index of the array. Whenever a new retransmission takes place, the specific time at that moment will be checked with existing values in the array once more. Afterwards, all values existing in the array will be moved one index lower and the smallest index will be removed. Thereafter, the previously acquired time value will be stored at the last index of the array. Every time a new time value is added to the final index, the system will check with the existing entries within the array whether it differs 15 seconds from any of the existing entries in the array.

This logic can also be denoted mathematically. Let an array of length n exist, where i denotes the index in the array of which the indexes go from i= 0 to i = n-1. Let $f(t)$ denote an amount of retransmissions taking place at time t. If $f(t)$ is larger than 0, the following equation will be checked:

$$t_{n-1} - t_{n-(i+1)} \text{ for } i = 1,2,3,....,n-1 \tag{15}$$

This fixes one part of the issue related to the congestion window scaling as a result of expected (periodic) losses. However, another major issue with using the default Cubic TCP as a CCA over the satellite-terrestrial link is the nature of the connection being wireless at some links. This results in additional random losses, which are not a result of congestion but physical errors within the wireless connection. The congestion window should not decrease due to these wireless losses because congestion has still not occurred in reality. Therefore, there is still at least the same amount of bandwidth available. Within the TCP protocol itself, there is no way of determining whether retransmissions that take place are a result of congestion or as a result of physical errors. We suggest classifying recurrent losses as actual congestion and singular random losses as physical errors. Fortunately, this can also be implemented with the β scaling factor. Similar to the expected retransmission solution, the same array containing retransmitted time values can be used for additionally checking whether the stored time values are recurrent or not. However, we argue that keeping, in this case, the multiplicative β scaling factor as 1 would result in a significant decrease in performance regarding the convergence and fairness if multiple streams are present in the connection. We argue that β should only scale the congestion window by 90% rather than 100% or the default 70%. This conditional logic and how it relates to searching the entries in the array can be formulated using the equation:

$$\text{if } f(t) > 0, \text{ then } \begin{cases} \beta = 1, \text{ if } t_{n-1} - t_{n-(i+1)} = 15, \text{ for } i = 1,2,3,...,n-1 \\ \beta = 0.7, \text{ if } t_{n-1} - t_{n-(i+1)} = 1, \text{ for } i = 1,2,3,...,n-1 \\ \beta = 0.9, \text{ otherwise} \end{cases} \tag{16}$$

After this has completed, the time slots stored within the array will be updated, following:

$$t_{n-i} \implies t_{n-(i+1)}, \text{ for } i = 1,2,3,4,...,n-1 \tag{17}$$

This results in the congestion window being inherently less aggressive regarding its multiplicative decrease. Furthermore, it indicates that convergence between different SICC communications will be lower compared to Cubic and BBR. In other words, the fairness of the bandwidth of the streams using SICC will most like converge more slowly as compared to Cubic and BBR.

Pseudocode for the logic described in Eq. 16 and Eq. 17 can be found in Algorithm 1.

The logic and functionality mentioned in the previous chapters are combined within the C code of the kernel module, of which an overview can be given in the next chapter.

Algorithm 1 SICC ssthresh recalculation function

Ensure: Function is triggered when retransmissions have taken place.

```
$time \leftarrow \frac{jiffies}{Hz}$                         ▷ Get current timestamp in seconds
$*ca \leftarrow inet - ca$                       ▷ Passes current TCP stream parameters into struct
for for $i = 1$, $i{+}{+}$, while $i < n$ do

    $diff \leftarrow time - array[n-i]$
    if $diff$ = 15 then
        $\beta = 1$                                 ▷ Periodic loss, so no adjusting CWND
        exit loop
    else if $diff < 2$ then
        $\beta = 0.7$                               ▷ Recurring loss, so probable congestion
        exit loop
    else
        $\beta = 0.9$                               ▷ Most likely loss due to physical error
        loop for next entry
    end if

end for
```

6.1.4 SICC Algorithm Overview

The TCP communication default parameters of SICC, combined with the multiplicative decrease modifications form the code for the TCP kernel module. An overview can be found in Fig. 21.

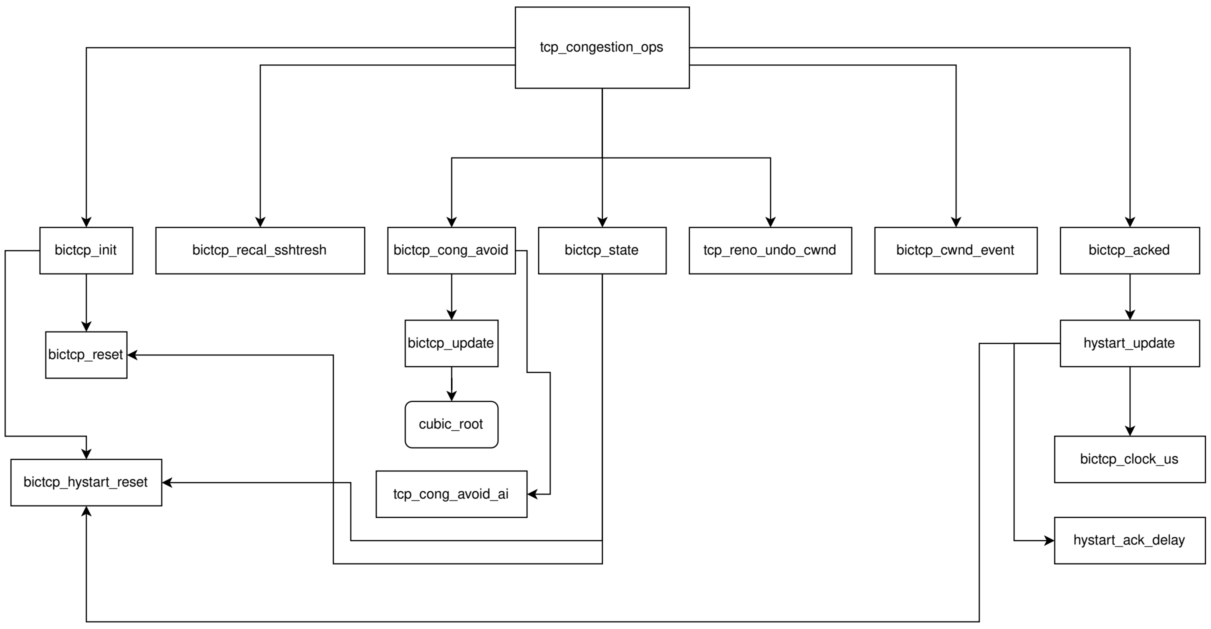


Figure 21: SICC kernel module function overview

Each of these main functions from figure 21 are based on Cubic's code [27], which is based on BIC's [48] implementation (hence the names) for TCP and have the following functionalities:

- ***tcp_congestion_ops***: The rest of TCP will call to access the congestion algorithm to know what to do. This is standard for all TCP congestion control algorithms.

- ***bictcp_init***: Initializes the variables. Also resets private variables to their initial values and handles conditional hystart reset and initial slow start threshold.
- ***bictcp_cong_avoid***: Function for handling increasing congestion window. It calculates the expected value of the congestion window using the RTT and checks it with its current value of the congestion window.
- ***bictcp_set_state***: Handles timeouts; if a certain time threshold is exceeded, it calls the reset function.
- ***bictcp_undo_cwnd***: Function that max value between previous largest value for the congestion window and its current value.
- ***bictcp_acked***: Function for monitoring the delays. In the case of a timeout, this value would be reset.
- ***bictcp_recalc_ssthresh***: Function that is called when retransmissions take place and multiplicatively decreases congestion window.

SICC will be evaluated similar to the the way the standard CCAs were evaluated, described in Section 5.1.1. Where the measurent setup from Fig. 10 and SICC is deployed at the measurement server. At the receiver we specify a download duration for 600 seconds, logging the CWND, retransmissions and throughput every 0.1 seconds. These parameters are chosen as they will provide enough insight into the implementation of SICC in real-life.

6.2 Results & Discussion

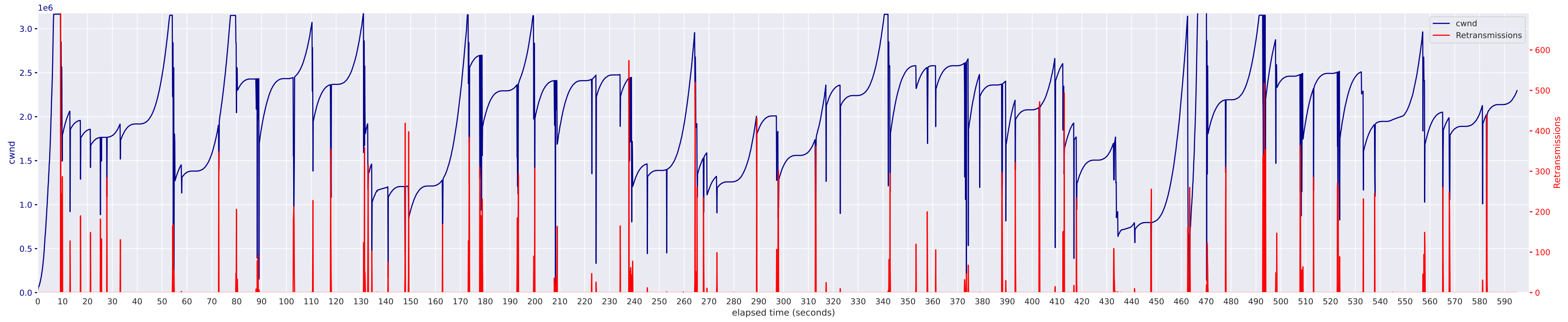


Figure 22: SICC CWND value and retransmissions plotted over time.

The CWND from SICC displayed in Fig. 22 shows that its value does not decrease at multiples of 15 seconds unless there are sequential retransmissions taking place. The mode that SICC is in when the CWND recovers from the expected retransmission stays the same. This can, for example, be seen at time 74 seconds. At this amount of elapsed time, we see that periodic (15-second) retransmission takes place. This results in the CWND dropping but then immediately recovering to its current value. Additionally, we observe that the protocol stays in congestion avoidance mode while maintaining its convex growth profile. As a result, we observe that SICC was able to gain a sizeable CWND value(3 * 1e6 Bytes), which enabled the server to transmit a lot more, thus making more efficient use of the available bandwidth. Furthermore, we see, for example, at 140 seconds and 225 seconds, random transmission losses resulting from physical errors. Here we observe that the CWND does not entirely decrease with 70%, but only 90%. And lastly, to reduce the CWND in the case of actual congestion, we see that SICC responds as desired. For example, at 180 seconds and at 210 seconds, we observe sequential retransmission taking place as a result of packet loss. In these cases, it can be seen that the CWND decreased by the standardized 70%. In conclusion, we can say that SICC works as expected and counters the described problems observed in the Starlink communication link. How SICC performs compared to the two most widely adopted protocols (Cubic & BBR) will be discussed in the following chapter.

7 Phase III: Prototype Evaluation

After proving that the developed prototype SICC complies with the previously mentioned requirements, we want to evaluate its performance compared to the two most used standards, which are Cubic and BBR. Reno will not be used in comparison as it is not as widely used nowadays. This phase consists of the Approach, where we describe the measurement setup, tools, and relevant parameters used during the evaluation, followed by the Results and the Discussion of these findings.

7.1 Approach

The approach for evaluating the performance of SICC is divided into two sections. The first will describe the setup for measuring the throughput of SICC compared to Cubic and BBR. The second section will describe the fairness of the three different protocols when having to share the available bandwidth in a connection.

7.1.1 Measurement setup throughput

Whether SICC performs better, worse or the same as Cubic and BBR in terms of achievable download throughput will now have to be evaluated. To be able to make a proper comparison between the throughput of two congestion control algorithms, it is essential to take into consideration that the measured throughput can vary significantly depending on the time of the measurement. This is due to several factors such as the weather, network usage, used satellites, and available gateways directly influencing the performance at that specific time. Furthermore, it is impossible to measure the maximum achievable download throughput of two different TCP congestion control algorithms simultaneously, as the available bandwidth will have to be shared with these two streams. Depending on the algorithms used, the execution of two active throughput measurements can result in artefacts during the measurement.

The proposed solution is to take the throughput measurement samples consecutively while switching between the TCP congestion algorithm in between measurements. To achieve this, iperf3 can also be used to calculate and estimate the available throughput actively. Similar to the first experiment, for measuring the download throughput of the Starlink dish, the Starlink server will be set up as the receiver during the measurement, and the measurement server located in Frankfurt will be used as the sender during the measurement (setup from Fig. 10). In order to take a measurement where consecutive samples are being taken, timing is relevant. At the sender side and the receiver side, different events will have to be triggered related to opening the ports at the sender side at specific times and connecting to those specified ports from the receiver side. Additionally, the duration of the measurement determines the timing at which the protocols will switch, and the measurement has to be taken again. An overview of the sequential events for gathering throughput metrics for comparison can be seen in Fig. 23.

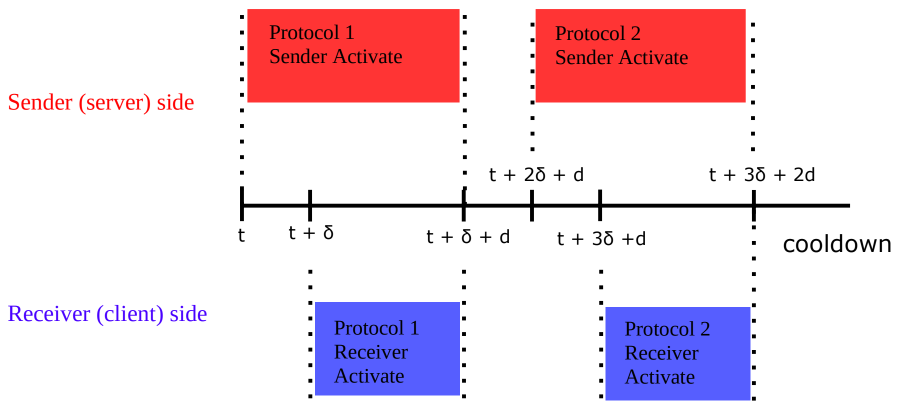


Figure 23: Event timeline for multi-protocol throughput comparison sampling.

In Fig. 23 at timestamp t, the sender side will open an iperf3 connection port for sending the data, at timestamp $t+\delta$ the receiver will try to connect to the iperf3 of the sender. Do note that the *sender activate*, and *receiver activate* cannot be triggered exactly at the same time, as this will result in the receiver trying to connect to the sender while the sender has not finished setting up an open port for the iperf3 connection; hence the additional delay δ. The receiver will download data from the sender that has the first congestion control algorithm set for the measurement. The duration of this download is denoted by d, thus the download will persist until $t+\delta+d$. After the throughput measurement is complete, both the sender and receiver close their connection, and the measurement data is stored. Because the sender's connection is closed, it enables us to change the congestion control algorithm to the second protocol. This is done at $t+2\delta+d$. Here the sender will switch congestion control algorithm and opens its port for an iperf3 connection. After δ time, the receiver will once again activate and receive packets from the sender, from which packets will be downloaded and the throughput will be calculated, estimated and stored by iperf3. Again this will be done for a duration of d, thus end at $t+3\delta+2d$ and the data for the second throughput measurement will be stored. Afterwards, the connection is terminated, and the measurement does nothing until called again.

Regarding the measurement parameters, t will be specified to trigger every *15 minutes* for period of around 6 hours, the δ parameter will be set to *5 seconds*, and the logging intervals will be set to *1 second*. Furthermore, the duration of the measurement will be set to *295 seconds*. We argue that these metrics will provide use with enough samples to gain an overview of download throughput of the CCAs. The comparisons that will be made regarding the throughput download performance are SICC against Cubic and SICC against BBR. Results will be gathered and will be showed in a histogram in the results from which we can gain insight into the distribution of the observed download speeds.

7.1.2 Measurement setup fairness

The second important measurement metric at which SICC is evaluated is its fairness. As mentioned earlier, fairness is an essential factor in determining how the available bandwidth is shared amongst the available streams. Similar to the throughput measurement explained in subsection 5.1.2, iperf3 will be used to gather the relevant metrics in the scenario described in Fig. 10. In this setup, two different ports will be used at both server and client side. Every *15 minutes* for a total duration of around 24 hours, the ports at the server side will open an iperf3 connection and after a delay of *5 seconds* the client will connect to the two ports on the server with two separate TCP streams. This will enable the measurement server (sender) sending two streams of data to the Starlink server (receiver). From this multi-throughput measurement we are able to measure how the bandwidth is shared amongst multiple streams during the measurement by looking at their achieved throughput at receiver side.

The fairness between two streams will be measured for the following CCAs:

- ***Cubic vs Cubic***; This will enable us to determine how the TCP CCA standard Cubic shares the bandwidth with itself over a Starlink connection. Here we expect the congestion control algorithms to fairly share the available down-link bandwidth with each other.
- ***BBR vs BBR***; Since it can also be expected that different BBR streams will most likely also compete with each other for the achievable bandwidth, we argue that it is important to also gain insight into how fair BBR is with itself over the Starlink connection. Because the streams will be homogeneous, we expect the bandwidth to be fairly amongst the two streams.
- ***BBR vs Cubic***; BBR and Cubic are two very commonly used TCP CCAs and is a primary motive for observing how they compare to each other in terms of fairness. Similar to terrestrial networks, we expect the BBR to be unfair towards Cubic (explained in Section 2.2.4.3). This measurement will also enable us to gain more insight into how quantitatively unfair BBR is towards Cubic by looking at the ratio between their achieved throughputs.

- ***SICC vs Cubic***; This fairness measurement is important to observe how fair the developed CCA, SICC, compares to the widely used standard, as SICC will most likely also have to compete against Cubic streams for available bandwidth. For this measurement we expect SICC to be slightly unfair towards Cubic, as its CWND does not decrease when expected packet loss occurs, nor decrease as much as Cubic when random packet loss takes place. This results in SICC being able to achieve a higher throughput during the measurement. However, we also argue that SICC will be more fair towards Cubic compared to BBR, as a lot of its TCP logic is also based on that of Cubic (congestion avoidance modes & tcp-friendly mode).
- ***SICC vs BBR***; Since BBR is also widely used, we argue that it is also relevant to look at SICC's fairness toward BBR. Doing so enables us to gain insight into how the available bandwidth is shared when these two streams would download simultaneously. Here we expect BBR to also be unfair towards SICC, as BBR will increase CWND, while ignoring packet loss, and SICC will (similar to Cubic) react to this packet loss and decrease CWND, thus utilising less of the available bandwidth.

Do note that for the CCA fairness comparison, we do not compare SICC with itself as we expect it to behave similarly to Cubic as SICC's sharing logic in the code is the same as Cubic. Because these throughput measurements will be done repeatedly over a long duration, different throughput samples will be generated. For analysis, the 95% confidence interval will be looked at for the different sampled throughputs for each CCA. Doing so will enable us to make a statement regarding the confidence of the actual download throughput mean. For additional interpretation, we will also denote the middle (mean) between these two boundaries. By doing so, we are able to state with 95% confidence that the actual throughput mean lies within those boundaries based on the acquired sample. The formula for calculating the confidence interval defined as:

$$\text{Confidence range} = \bar{x} \pm t * \frac{s}{\sqrt{n}} \tag{18}$$

- Where $\bar{x}$ denotes the sample mean.
- t denotes the t variable taken from the statistical t table.
- s denotes the standard deviation.
- n denotes the sample size.

This will be implemented in the code used to parse and visualise the acquired data.

7.2 Results & Discussion

First the results of the throughput comparison will be shown and discussed. Afterward, we will show the results from comparing the fairness of the different CCA protocols, which is also followed by a discussion.

7.2.1 Throughput Distribution

The histograms displayed in Fig. 24 show the distribution of the sampled throughputs of SICC and Cubic. These results are based on the measured throughputs from 36 5-minute samples from both Cubic and SICC (72 total). These measurements were taken from 2022-06-13 13:18 *(yyyy-MM-dd HH:mm)* to 2022-06-14 06:17. The average measured download throughput that Cubic achieved during this period is 118 Mbps with a standard deviation of 52 Mbps. During the same period, SICC was able to achieve an average throughput of 162 Mbps with a standard deviation of 61 Mbps. These results confirm that SICC handles the problems (recurring retransmissions & high BER) present in the Starlink connection better, thus achieving a throughput that is, on average, 38% faster than Cubic.

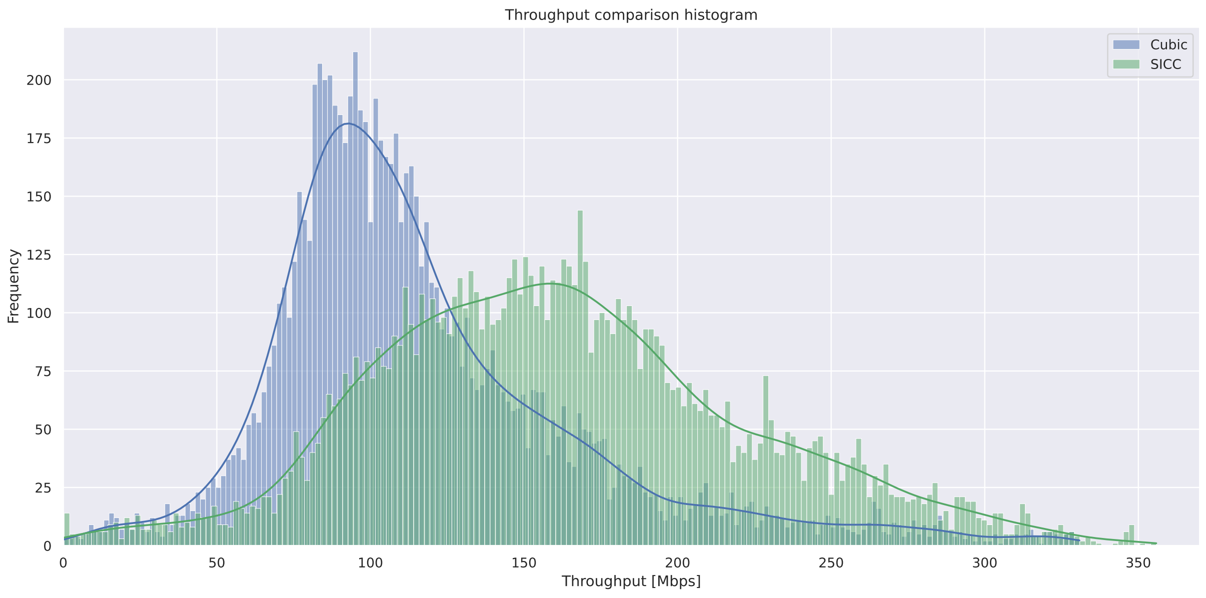


Figure 24: SICC vs Cubic throughput distribution.

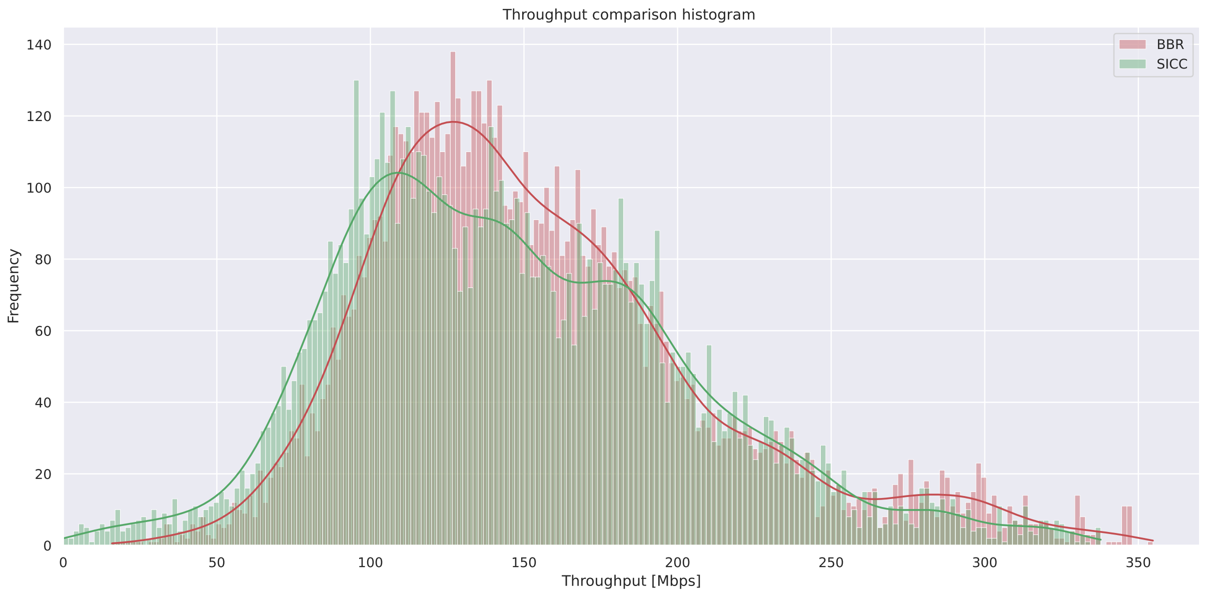


Figure 25: SICC vs BBR throughput distribution.

The histograms displayed in Fig. 25 show the distribution of the sampled throughputs of SICC and BBR. These results are based on throughput measurements from 32 5-minute samples from both SICC and BBR (64 total). These measurements were taken from 2022-06-18 13:33 *(yyyy-MM-dd HH:mm)* to 2022-06-19 00:23. For this measurement we observed that BBR was able to achieve an average throughput of 155 Mbps along with a standard deviation of 56 Mbps. During this same period, SICC was able to gain an average throughput of 148 Mbps along with a standard deviation of 57 Mbps. These results show that SICC almost achieves the same throughput as BBR while being a loss-based CCA. BBR seems to be slightly 5% faster on average compared to SICC.

These performance results from SICC show promise as it achieves significantly more download throughput than Cubic, and almost as much as BBR. However, another important metric that defines “good” TCP CCAs is the fairness that will be discussed in the following chapter.

7.2.2 CCA Fairness Comparison

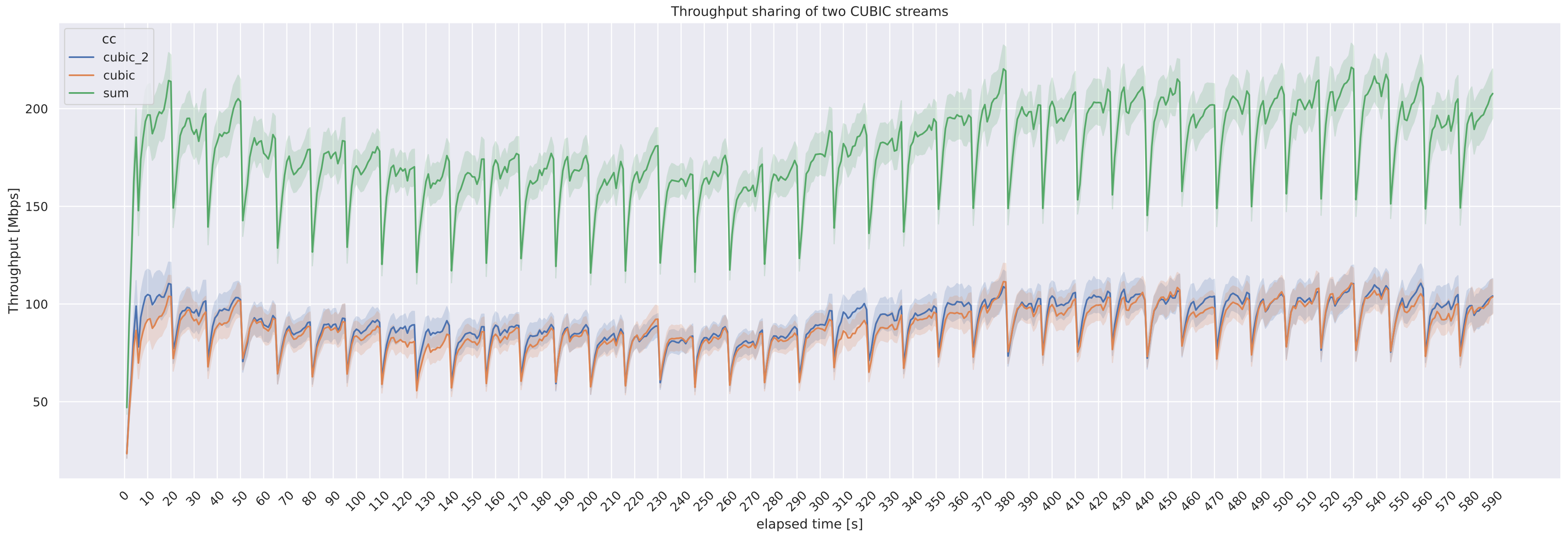


Figure 26: Fairness Cubic vs Cubic congestion algorithm.

The results from two Cubic streams competing for the available bandwidth is depicted in Fig. 26. The 95% intervals displayed in this graph consist of 190 measurement samples, of which 95 samples are from the first Cubic stream, and the other 95 samples are from the second Cubic stream. This sample is the throughput measured from 2022-10-05 09:45 *(yyyy-MM-dd HH:mm)* to 2022-10-06 09:15. The measured average shows that the two Cubic streams more or less share the available bandwidth equally, thus achieving around the same throughput around 95 Mbps. This was expected for Cubic as they would both react the same way to the packet loss experienced in the link. Additionally, the figure also shows that a majority of the streams seem to suffer from the 15-second interval retransmissions. This is said because consistent drops in throughput can be seen in periods of 15 seconds in between when looking at both the streams individually and their sum. Examples of this can be seen starting at an elapsed time of around 5 seconds, then again at 20 seconds and so forth with 15-second intervals. Because this is observable on in the averages and confidence intervals, it means that a majority of the Cubic streams experienced this drop at the same time since the start of the measurement. A reason for this notable observation is that the cycle at which the recurring 15-second retransmissions occur has aligned with the cycles at which the measurements are being taken. This indicates that some Starlink gateways and/or the Starlink satellites have synchronised clocks with a specific (possible layer 2 or layer 3) protocol being triggered at expected times that result in packet loss every 15 seconds. Since the measurement also follows a cycle (triggered every 15 minutes), these losses in packets that result in a drop in throughput are observed for most of the samples. Another interesting observation is that the total throughput starts at around 190 Mbps, then proceeds to go down at around 50 seconds, reaching an average throughput of around 165 Mbps, then proceeds to go upward again at around 270 seconds, ultimately reaching a throughput of around 200 Mbps. The reason behind this is not known, however, a possible reason is that the Starlink connection prioritises users for the first 50 seconds of their usage by giving them more available bandwidth, then proceeds to throttle their bandwidth, and if the system notices that the user is still requiring a lot of bandwidth, they allocate more again to the user after around 3.5 minutes.

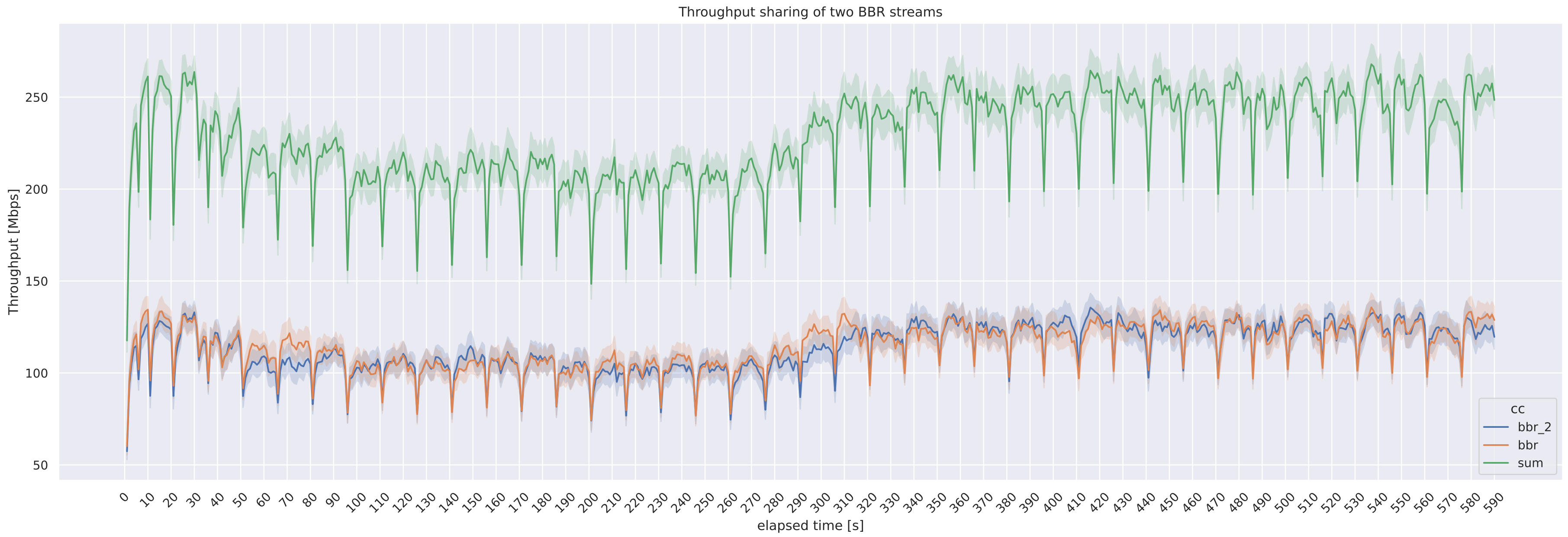


Figure 27: Fairness BBR vs BBR congestion algorithm.

The graph displayed in Fig. 27 shows how two BBR streams share the available bandwidth in the same communication link. These results are based on a total of 180 samples, of which 90 are from the first BBR stream and the other 90 are from the second BBR stream. Two competing BBR streams seem to be fair towards each other regarding the sharing of the available bandwidth, achieving both around 125 Mbps. The measurement was taken from 2022-10-06 09:45 *(yyyy-MM-dd HH:mm)* until 2022-10-07 08:15. Similar to the results of two competing Cubic streams from Fig. 26, we also observe drops in measured throughput for BBR at moments with 15 second intervals. The reasoning behind this is similar to that described earlier. However, we observe that the BBR throughput samples recover more quickly than those of Cubic. Possible reason behind this is due to BBR quickly ramping up its bandwidth usage while ignoring packet loss as described in paragraph 2.2.4.3. For this measurement, we also observe that the throughput during the first elapsed 50 seconds, then decreasing for around 3.5 minutes before increasing again. The possible reasoning behind this is similar to the one described earlier when discussing Fig. 26.

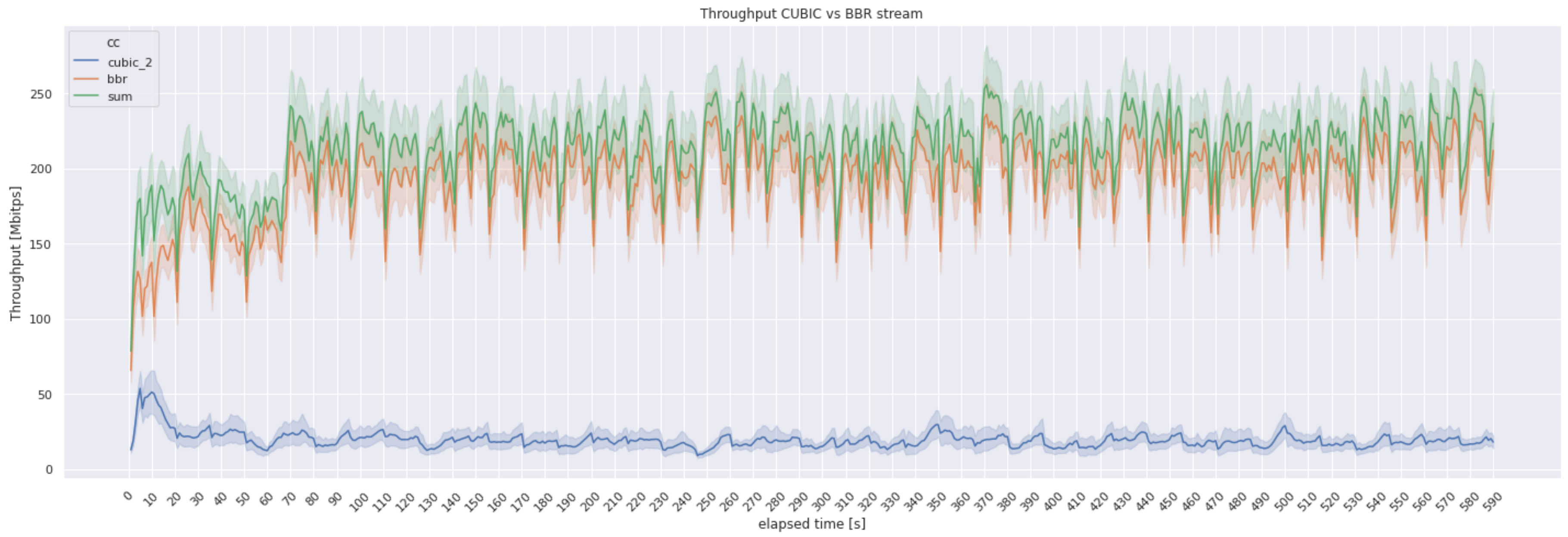


Figure 28: Fairness Cubic vs BBR congestion algorithm.

When looking at a Cubic stream competing against a BBR stream, we also see, as expected, that the BBR stream seems to dominate the available bandwidth in Fig. 28. The measurement was taken from 2022-06-28 21:15 *(yyyy-MM-dd HH:mm)* until 2022-06-29 07:15. Do note that this figure is the result of 82 measurements, of which 42 are Cubic samples, and the other 42 samples are from BBR. The 95% confidence intervals show that the actual expected average of the Cubic stream is roughly around 25 Mbps. In contrast, the BBR streams are around 200 Mbps when the available bandwidth (sum) seems to be around 225 Mbps. This means that BBR takes up around 90% of the available bandwidth, whereas Cubic only manages to achieve 10% of the achievable throughput, which is quite unfair but expected. As mentioned earlier, the Cubic CCA responds to packet loss, whereas BBR does not. BBR purposely introduces packet loss by attempting to fully utilise the bandwidth in the link, resulting in a significant decrease achieved by Cubic. Furthermore, we also observe the drops in both sum and BBR throughput at the 15-second intervals here. We do not see a decrease in measured throughput after 50 seconds for these results. A possible reason for this is the time at which this measurement was taken compared to the other more recent measurements.

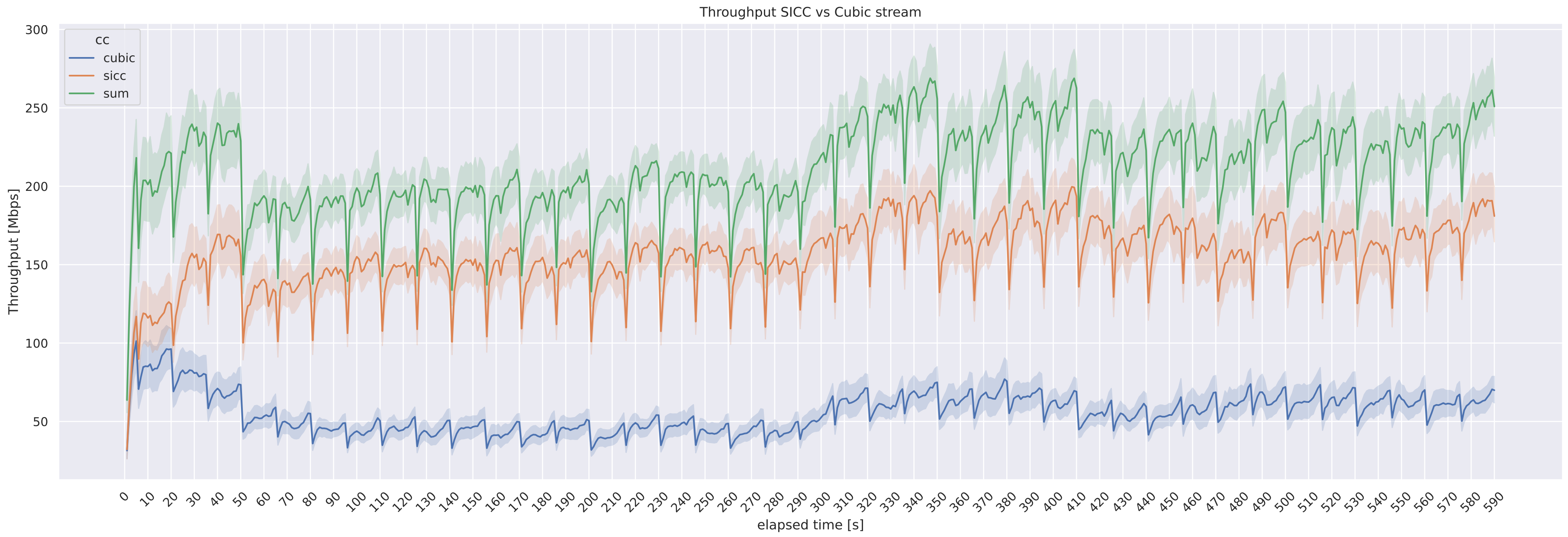


Figure 29: Fairness SICC vs Cubic congestion algorithm.

Fig. 29 shows how SICC compares to Cubic in terms of TCP fairness. This measurement consists of 114 samples in total, of which 57 are the throughput samples from SICC and the other 57 samples are from Cubic. The measurement was taken from 2022-10-03 18:15 *(yyyy-MM-dd HH:mm)* until 2022-10-04 08:15. Similar to the previous figures, we here we again observe the recurring drops in throughput at 15 second intervals. In the figure, it can be seen that SICC is more fair towards Cubic regarding sharing the available bandwidth for the throughput. The observed throughput from Cubic is seems to be aronud 50 Mbps, which is double the amount of throughput Cubic could achieve compared to when it competed against BBR in Fig. 28. The reason behind this is a result of SICC handling packet loss slightly different than Cubic. In the figure it can be seen that SICC recovers faster than Cubic during these frequent drops in throughput. The total achievable throughput seems to be at around 220 Mbps as well. Lastly, for these results we also observe a relatively significant drop in achievable throughput at around the 50 second mark lasting around 3.5 minutes before ramping up again and achieving more total bandwidth. Both streams seem to suffer from this.

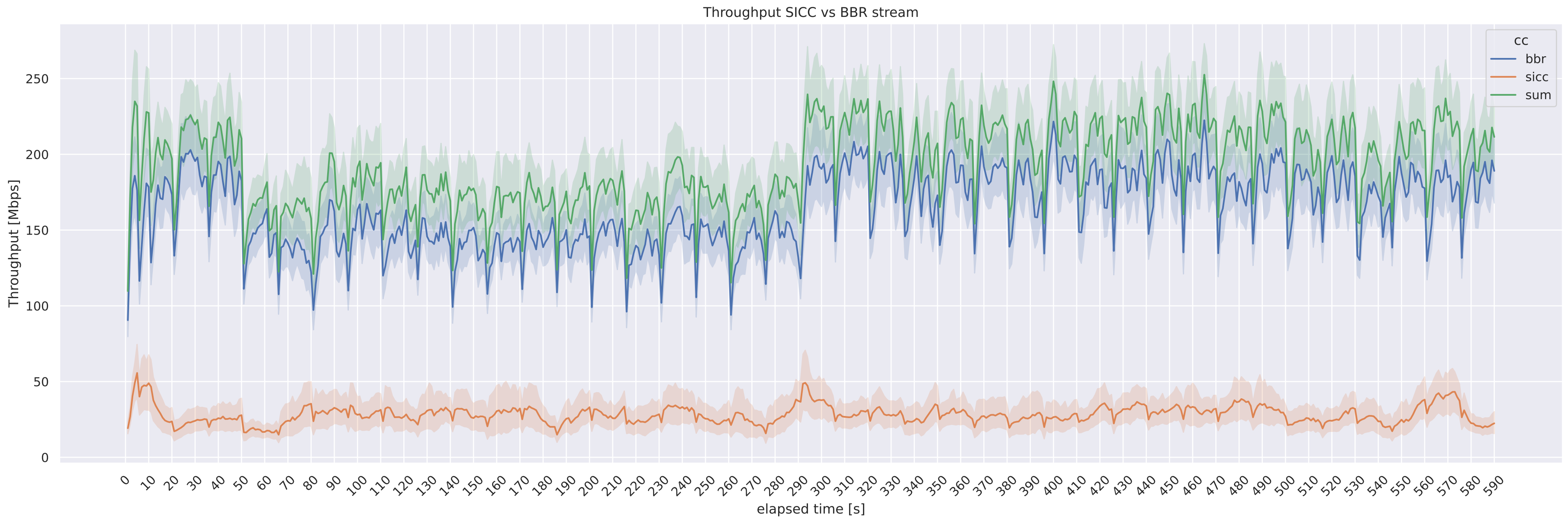


Figure 30: Fairness SICC vs BBR congestion algorithm.

Lastly, SICC competing against BBR is depicted in Fig. 30. This measurement is taken from 2022-10-04 08:45 *(yyyy-MM-dd HH:mm)* until 2022-10-04 18:00. The total amount of throughput samples this graph is based on is 76, of which 38 are the measured throughput from SICC, and the other 38 are from BBR. Similar to figure 28, BBR seems to dominate the available throughput, resulting in SICC only receiving a small amount. This amount SICC achieves seems to be similar to that of Cubic, which is around 25 Mbps, whereas BBR seems to achieve close to the maximum available bandwidth (around 220 Mbps). For this measurement, there is also an observable drop in achievable throughput at around the 50-second mark, also lasting around 3.5 minutes before becoming its original size.

What these results further indicate will be summarised in the Conclusion of this thesis.

8 Conclusions

With the arrival of Starlink, a promising LEO Satellite Internet provider, also come questions regarding the performance of this system using protocols that were originally developed to be applied in static (terrestrial) networks. To the best of our knowledge, related work in the field of LEO satellite internet performance was mainly limited to ns-2 and ns-3 simulations. Hardly any research has been done regarding the measurable performance of the real-life Starlink internet. This lack of knowledge is the main motive behind this thesis as it attempts to scrutinise the impact of the highly dynamic nature of the Starlink system on the performance received by the end-users.

The research done in this thesis was divided into three main phases. The first focused on exploring the system, the second focused on prototyping a solution to the problems observed initially, and the final phase focused on evaluating the proposed solution. Together the findings of this research ultimately provide the reader with insight into the issues computers experience communicating over a Starlink connection.

To evaluate the Starlink internet's throughput performance, we used a setup with a receiving server connected to a Starlink generation 1 dish communicating with the different Starlink satellites to a server located in Frankfurt. The Frankfurt server functions as the sender, which we can use to actively measure and estimate the parameters related to the achievable download throughput at the receiver. We mainly focus on the download throughput as it is, in many cases, the most relevant metric for standard internet users.

During the exploration phase, we evaluated the performance of three different CCAs (Reno, Cubic & BBR) by measuring the CWND and packet loss during communication over Starlink. By looking at the retransmissions as a result of packet loss, it was observed that there were periodic amounts of packet loss at 15-second intervals. Through autocorrelation of the retransmissions, we confirmed this for loss-based CCAs (Reno & Cubic). These frequent retransmissions influence the achievable download throughput from the user, as the CCA will keep the CWND small as it believes the link is under congestion when in reality, it is not. For BBR, it was challenging to observe the presence of these 15-second intervals as the data samples contained large amounts of retransmissions. However, when looking at the BBR streams' RTT, we observed sudden spikes in it at 15-second intervals.

In the prototyping phase, we attempted to develop a loss-based protocol based on the current TCP CCA standard, Cubic. This CCA is called "Space Internet Congestion Control" (SICC) and is developed as a loadable kernel module for Debian-based computers. Built on the open-source Cubic code and thus containing a lot of the features of Cubic itself, such as its slow-start and congestion avoidance mode, SICC is further modified to solve the problems experienced in Starlink. This is done through conditional rescaling of the CWND. The protocol uses timing parameters acquired from the CPU ticks to log the times when retransmissions occur. If retransmissions take place at a 15-second interval from a previous retransmission, SICC's CWND will retain its value and growth profile. Furthermore, to compensate for the high BER experienced in the wireless links between Satellite and earth terminals, SICC will only decrease CWND by 90%. This will result in the CWND staying at a large value, thus enabling the user to utilise more of the available bandwidth. For the detection of true congestion, we implemented the use of monitoring for consecutive packet loss. If SICC observes consecutive packet loss, it will decrease the CWND by 70% to avoid congestion and/or make room for other TCP streams on the link.

During the final phase of the thesis we focused on evaluating the performance of SICC with two widely implemented CCA algorithms, being Cubic and BBR. The achievable download throughput measurement data showed that SICC was able to achieve, on average, almost 40% higher download throughput, compared to Cubic while almost matching the average throughput of BBR. During the evaluation of the fairness we looked at the behavior of two TCP streams at the same time and observed the ratio between their achieved throughputs. By using a confidence interval along with the average as an estimator for the region in which the possible true throughput lied during the measurement, we also observed that all CCAs seem to suffer from the 15-second packet loss intervals, of which Cubic the most. BBR and SICC their throughput recovers faster than Cubic. In terms of fairness SICC is 100% more fair towards Cubic than BBR, while BBR significantly unfair towards both Cubic and SICC. This is due to BBR not being a loss-based algorithm, thus not reacting to packet loss the same way SICC and Cubic do. Additional measurement findings show that when utilising the bandwidth of a Starlink connection, for all different measured CCAs, the user will experience a comparatively large amount for the first 50 seconds of the measurement before decreasing. This decrease lasts for around 3.5 minutes before increasing bandwidth utilisation again.

The findings in this thesis proved that standard CCAs suffer from unknown mechanisms present in the Starlink system. This suggests additional research into the cause of the large amounts recurring of packet loss present in the system. Future suggestions will be mentioned in the following chapter as a possible guide for further development.

9 Limitations and Future Work

This section of the report will summarise the limitations present in this thesis and provide an overview of possible future directions for this research.

The limitations to take into consideration when interpreting this research are:

- ***Weather dependency***; The weather at the time of the measurements can heavily influence the acquired metrics. The weather at the Starlink dish (terminal) and Gateway were not logged for this research. This means that we do not know how the weather influenced measurements.
- ***Dishy no issues assumption***; The measurements executed with a Gen 1 Satellite dish of, which we assumed had no issues in its hardware or firmware. Although not likely, we cannot ignore the slight possibility that these findings are limited to only our Starlink dish.
- ***Bottleneck assumption***; For this research, a limitation we do not have control or insight into is the intermediate links between the communicating nodes and their capacity. Because these measurements were done in a real scenario, we assume that the intermediate links are not the bottleneck, but the wireless Starlink connection is. This is a valid assumption as most modern wired connections in the internet have large capacities exceeding the bandwidth offered by the Starlink service.
- ***Limited Starlink Research***; Because Starlink is a relatively new system, not a lot of scientific research has been done regarding it. A result of this is that some statements relating to e.g. the pricing and achievable throughputs, are taken from information websites instead of research papers (as there exist none comparing these). The scientific statements in this thesis however, are based on valid research papers and RFCs.

Possible future directions that can be looked at regarding the Starlink:

- ***Weather dependency***; Because the influence of the weather is unknown, it is an exciting research direction to take. Observing the weather while evaluating Starlink's performance allows one to gain insight into how these two relate to each other. Additionally, it is possible to analyse the effect of Solar winds on Starlink's performance.
- ***Other CCAs***; Another exciting direction to take is analysing other CCAs in Starlink and how they perform in terms of throughput. There exist a large variety of CCAs, of which some claim to be specifically developed for Satellite Networks (Hybla and Westwood). It is interesting to see how other non-loss-based CCAs perform over Starlink (e.g PCC & BBRv2).
- ***Lower layer protocols***; Only at the Transport layer can we observe, test and modify protocols relating to the communication of Starlink. No insight is present in the lower layers (physical, data link and network). However, if these were to be made public, researchers could conduct more analysis.
- ***Newer generation Starlink***; At the moment, Starlink is still in its early stages. The measurements of these are done using a generation 1 Starlink dish. However, as of October 2022, Starlink has already released their second generation of dishes. How these perform compared to a generation 1 dish is also interesting. Furthermore, Starlink plans on implementing inter-satellite links in the future. Once these become active, it would be interesting to observe the system's performance as well.

References

[1] Barry Leiner et al. "A Brief History of the Internet". In: *Computer Communication Review* 39 (Oct. 2009), pp. 22–31. DOI: 10.1145/1629607.1629613.

[2] Irune Ruiz-Martínez and Javier Esparcia. "Internet Access in Rural Areas: Brake or Stimulus as Post-Covid-19 Opportunity?" In: *Sustainability* 12.22 (Nov. 18, 2020), p. 9619. ISSN: 2071-1050. DOI: 10.3390/su12229619. URL: https://www.mdpi.com/2071-1050/12/22/9619 (visited on 10/03/2022).

[3] Jean-Francois Castet and Joseph Saleh. "Geosynchronous communication satellite reliability: statistical data analysis and modeling". In: Jan. 2009, pp. 431–431. DOI: 10.1049/cp.2009.1196.

[4] Max Roser, Hannah Ritchie, and Esteban Ortiz-Ospina. "Internet". In: *Our World in Data* (2015). https://ourworldindata.org/internet.

[5] Inigo del Portillo Barrios, Bruce Cameron, and Edward Crawley. "A technical comparison of three low earth orbit satellite constellation systems to provide global broadband". In: *Acta Astronautica* 159 (Mar. 2019). DOI: 10.1016/j.actaastro.2019.03.040.

[6] Stephen Clark. *Live coverage: SpaceX rocket, Starlink satellites launch from pad 39A – Spaceflight Now*. URL: https://spaceflightnow.com/2022/05/18/falcon-9-starlink-4-18-live-coverage/ (visited on 10/03/2022).

[7] Xavier Bracke. *SpaceX's Starlink: Everything you need to know*. Sept. 2022. URL: https://mercuron.eu/en/kennis/blog-en/spacexs-starlink-everything-you-need-to-know/.

[8] Evelyn Arevalo. *SpaceX's newest FCC filing reveals plans to 'enhance' Starlink broadband with a 'slightly smaller antenna'*. June 2021. URL: https://www.tesmanian.com/blogs/tesmanian-blog/next-dish.

[9] Andrew May published. *Low Earth orbit: Definition, theory and facts*. Space.com. May 30, 2022. URL: https://www.space.com/low-earth-orbit (visited on 10/06/2022).

[10] *Best Satellite Internet Providers of 2022*. SatelliteInternet.com. URL: https://www.satelliteinternet.com/ (visited on 10/03/2022).

[11] Justyna Matuszak. *How was the SpaceX Falcon 9 reusable rocket built?* KnowHow. Jan. 11, 2022. URL: https://knowhow.distrelec.com/defence-aerospace-and-marine/how-was-the-spacex-falcon-9-reusable-rocket-built/ (visited on 10/03/2022).

[12] *What is Starlink? Everything you need to know about Elon Musk's satellite internet service*. ZDNET. URL: https://www.zdnet.com/home-and-office/networking/starlink-satellite-internet-how-does-it-work-cost-features-speed/ (visited on 10/03/2022).

[13] Mark Holmes. *Hughes Collaborates with TCS in Military/Government Market - Via Satellite -*. Via Satellite. Nov. 10, 2011. URL: https://www.satellitetoday.com/uncategorized/2011/11/10/hughes-collaborates-with-tcs-in-militarygovernment-market/ (visited on 10/03/2022).

[14] Alan Weissberger. *PCMag Study: Starlink speed and latency top satellite Internet from Hughes and Viasat's Exede*. Technology Blog. Nov. 5, 2020. URL: https://techblog.comsoc.org/2020/11/04/pcmag-study-starlink-speed-and-latency-top-satellite-internet-from-hughes-and-viasats-exede/ (visited on 10/03/2022).

[15] *2022 HughesNet Internet Review — HughesNet plans*. SatelliteInternet.com. URL: https://www.satelliteinternet.com/providers/hughesnet/ (visited on 10/04/2022).

[16] *2022 Viasat Internet Review — Viasat plans and more*. SatelliteInternet.com. URL: https://www.satelliteinternet.com/providers/viasat/ (visited on 10/04/2022).

[17] Tonny Ondeng. "TCP/IP Technology". In: (Aug. 2017).

[18] Yadong Li et al. "Research based on OSI model". In: *2011 IEEE 3rd International Conference on Communication Software and Networks*. 2011, pp. 554–557. DOI: 10.1109/ICCSN.2011.6014631.

[19] *What is OSI Model — 7 Layers Explained — Imperva*. Learning Center. URL: `https://www.imperva.com/learn/application-security/osi-model/` (visited on 10/07/2022).

[20] Shannon. *Chapter 3,4 & 6 1-TRANSMISSION IMPAIRMENT 2-DATA TRANSMISSION & MODES - ppt download*. URL: `https://slideplayer.com/slide/13176659/` (visited on 10/07/2022).

[21] Yuetsu Kodama et al. "Realization of a stable network flow with high performance communication in high bandwidth-delay product network". In: (Jan. 2004).

[22] Phillipa Sessini and Anirban Mahanti. "Observations on Round-Trip Times of TCP Connections". In: (), p. 7.

[23] *19 TCP Reno and Congestion Management — An Introduction to Computer Networks, desktop edition 2.0.9*. URL: `https://intronetworks.cs.luc.edu/current/html/reno.html` (visited on 10/10/2022).

[24] Vern Paxson, Mark Allman, and W. Richard Stevens. *TCP Congestion Control*. Request for Comments RFC 2581. Num Pages: 14. Internet Engineering Task Force, Apr. 1, 1999. DOI: `10.17487/RFC2581`. URL: `https://datatracker.ietf.org/doc/rfc2581` (visited on 10/11/2022).

[25] Vincent Nyangaresi, Solomon Ogara, and Silvance Abeka. "Security Evaluation of the Adaptive Congestion Control Algorithms for Virtual Data Center Communication". In: *European Academic Research* 4 (Nov. 2016), pp. 7049–7071.

[26] *19 TCP Reno and Congestion Management — An Introduction to Computer Networks, desktop edition 2.0.9*. URL: `https://intronetworks.cs.luc.edu/current/html/reno.html` (visited on 10/11/2022).

[27] Sangtae Ha, Injong Rhee, and Lisong Xu. "CUBIC: a new TCP-friendly high-speed TCP variant". In: *ACM SIGOPS Operating Systems Review* 42.5 (July 2008), pp. 64–74. ISSN: 0163-5980. DOI: `10.1145/1400097.1400105`. URL: `https://dl.acm.org/doi/10.1145/1400097.1400105` (visited on 09/13/2022).

[28] Moritz Michael Geist. "Overview of TCP Congestion Control Algorithms". In: (2019). In collab. with Chair Of Network Architectures. Medium: PDF Publisher: Chair of Network Architectures and Services, Department of Computer Science, Technical University of Munich. DOI: `10.2313/NET-2019-06-1_03`. URL: `https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2019-06-1/NET-2019-06-1_03.pdf` (visited on 11/15/2022).

[29] Ethan Blanton, Vern Paxson, and Mark Allman. *TCP Congestion Control*. Request for Comments RFC 5681. Num Pages: 18. Internet Engineering Task Force, Sept. 2009. DOI: `10.17487/RFC5681`. URL: `https://datatracker.ietf.org/doc/rfc5681` (visited on 10/10/2022).

[30] Sally Floyd. *Limited Slow-Start for TCP with Large Congestion Windows*. Request for Comments RFC 3742. Num Pages: 7. Internet Engineering Task Force, Mar. 2004. DOI: `10.17487/RFC3742`. URL: `https://datatracker.ietf.org/doc/rfc3742` (visited on 10/10/2022).

[31] Mario Hock, Roland Bless, and Martina Zitterbart. "Experimental evaluation of BBR congestion control". In: Oct. 2017, pp. 1–10. DOI: `10.1109/ICNP.2017.8117540`.

[32] Mario Hock, Roland Bless, and Martina Zitterbart. "Experimental evaluation of BBR congestion control". In: *2017 IEEE 25th International Conference on Network Protocols (ICNP)*. 2017 IEEE 25th International Conference on Network Protocols (ICNP). Toronto, ON: IEEE, Oct. 2017, pp. 1–10. ISBN: 978-1-5090-6501-1. DOI: `10.1109/ICNP.2017.8117540`. URL: `http://ieeexplore.ieee.org/document/8117540/` (visited on 10/11/2022).

[33] Yi Cao et al. "When to use and when not to use BBR: An empirical analysis and evaluation study". In: *Proceedings of the Internet Measurement Conference*. IMC '19: ACM Internet Measurement Conference. Amsterdam Netherlands: ACM, Oct. 21, 2019, pp. 130–136. ISBN: 978-1-4503-6948-0. DOI: `10.1145/3355369.3355579`. URL: `https://dl.acm.org/doi/10.1145/3355369.3355579` (visited on 10/11/2022).

[34] Carlo Caini and Rosario Firrincieli. "TCP Hybla: a TCP Enhancement for heterogeneous networks". In: *International Journal of Satellite Communications and Networking* 22 (Sept. 1, 2004), pp. 547–566. DOI: `10.1002/sat.799`.

[35] Dzmitry Kliazovich, Fabrizio Granelli, and Daniele Miorandi. "TCP Westwood plus Enhancement in High-Speed Long-Distance Networks". In: vol. 2. July 1, 2006, pp. 710–715. DOI: 10.1109/ICC.2006.254791.

[36] Saahil Claypool, Jae Chung, and Mark Claypool. "Comparison of TCP Congestion Control Performance over a Satellite Network". In: *Passive and Active Measurement*. Ed. by Oliver Hohlfeld, Andra Lutu, and Dave Levin. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2021, pp. 499–512. ISBN: 978-3-030-72582-2. DOI: 10.1007/978-3-030-72582-2_29.

[37] Tianle Mai et al. "Self-learning Congestion Control of MPTCP in Satellites Communications". In: *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*. 2019 15th International Wireless Communications Mobile Computing Conference (IWCMC). ISSN: 2376-6506. June 2019, pp. 775–780. DOI: 10.1109/IWCMC.2019.8766465.

[38] *Starlink Analysis*. URL: https://forschung.fh-kaernten.at/roadmap-5g/files/2021/07/Starlink-Analysis.pdf.

[39] Jinyu Yin et al. "INTCP: Information-centric TCP for Satellite Network". In: *arXiv:2112.14916 [cs]* (Dec. 29, 2021). arXiv: 2112.14916. URL: http://arxiv.org/abs/2112.14916 (visited on 01/10/2022).

[40] Mark Handley. "Delay is Not an Option: Low Latency Routing in Space". In: Nov. 2018, pp. 85–91. DOI: 10.1145/3286062.3286075.

[41] Jinhua Cao. "Congestion control and routing over satellite networks". Publication Title: Ph.D. Thesis ADS Bibcode: 2010PhDT.......163C. PhD thesis. Jan. 1, 2010. URL: https://ui.adsabs.harvard.edu/abs/2010PhDT.......163C (visited on 01/10/2022).

[42] Jingjing Wang et al. "Aggressive Congestion Control Mechanism for Space Systems". In: *IEEE Aerospace and Electronic Systems Magazine* 31.3 (Mar. 2016), pp. 28–33. ISSN: 0885-8985. DOI: 10.1109/MAES.2016.150117. arXiv: 1701.00234. URL: http://arxiv.org/abs/1701.00234 (visited on 02/10/2022).

[43] *Starlink Satellite Tracker*. URL: https://satellitemap.space/.

[44] Vivien GUEANT. *Iperf - the ultimate speed test tool for TCP, UDP and SCTPTEST the limits of your network + internet neutrality test*. URL: https://iperf.fr/.

[45] Philip. *TCP congestion control algorithms comparison*. Aug. 2020. URL: https://www.speedguide.net/articles/tcp-congestion-control-algorithms-comparison-7423.

[46] Ilja Zakharov et al. "Modeling Environment for Static Verification of Linux Kernel Modules". In: vol. 41. Jan. 2014. ISBN: 978-3-662-46822-7. DOI: 10.1007/978-3-662-46823-4_32.

[47] I. Rhee et al. *Cubic for fast long-distance networks*. Feb. 1970. URL: https://www.rfc-editor.org/rfc/rfc8312.

[48] Lisong Xu, K. Harfoush, and Injong Rhee. "Binary increase congestion control (BIC) for fast long-distance networks". In: *IEEE INFOCOM 2004*. Vol. 4. 2004, 2514–2524 vol.4. DOI: 10.1109/INFCOM.2004.1354672.