DEGREE PROJECT IN COMMUNICATION SYSTEMS, SECOND LEVEL
STOCKHOLM, SWEDEN 2015

5G user satisfaction enabled by FASP

Evaluating the performance of Aspera's FASP

PATRIK HAGERNÄS

KTH ROYAL INSTITUTE OF TECHNOLOGY
INFORMATION AND COMMUNICATION TECHNOLOGY

5G user satisfaction enabled by FASP

Evaluating the performance of Aspera's FASP

Patrik Hagernäs

2015-08-28

Master’s Thesis

Examiner and Academic Adviser
Gerald Q. Maguire Jr.

Industrial advisers
Annikki Welin & Tomas Thyni

KTH Royal Institute of Technology
School of Information and Communication Technology (ICT)
Department of Communication Systems
SE-100 44 Stockholm, Sweden

Abstract

With Ericsson's goal to have optimal user experience at 5G's 2020 release, it is very important to optimize transport protocols and techniques to manage the increasing amount of data traffic. Additionally, it will be important to manage handovers between very high speed 5G networks and older networks. Today most of the traffic is video on demand and the amount of this kind of traffic is expected to increase. Moreover, the current amount of data traffic will increase by an order of magnitude over the next few years. This thesis focuses on radio access networks and the difficulties they face in delivering high speed data traffic.

This thesis analyzes one of the most used TCP protocols, CUBIC, as well as a new transport protocol developed by Aspera, called the Fast and Secure Protocol. Aspera's FASP is a new transport protocol that promises full link utilization. FASP is built upon UDP and uses advanced round trip time measurements and queuing delay to detect the available bandwidth between two communicating hosts.

This thesis project also provides methods to realize experiments to assess the limitations of transport protocols. These experiments are conducted in an environment that resembles the upcoming 5G radio access network. Results have shown that both delay and packet loss affect TCP more than we expected and that high packet loss is devastating. In contrast, Aspera's FASP is very resistant to both delay and packet loss. These results and analysis provide a foundation upon which others can build.

Keywords TCP, 5G, emulated radio access network, handover, FASP, CUBIC TCP, packet loss

Sammanfattning

Med Ericssons mål att ha optimal användarupplevelse vid släppet av 5G år 2020 är det oerhört viktigt att optimera transportprotokoll och tekniker för att hantera den ökande mängden datatrafik. En annan viktig aspekt kan vara att hantera överlämningar mellan 5G nätverk och äldre radionätverk. Idag är den största trafiken streamad video och prognoser visar att den sortens trafik bara kommer att öka. Prognoserna visar också att all trafik kommer att öka mångfaldigt de närmaste åren. Denna thesis kommer att fokusera på svårigheterna just inom radionätverk.

Denna thesis kommer att analysera ett av vårt mest använda transportprotokoll CUBIC TCP, den kommer också att analysera ett helt nytt transportprotokoll utvecklat av Aspera, Fast and Secure Protocol. Aspera lovar fullt utnyttjande av den mellanliggande länken. FASP är byggt ovanpå UDP och använder avancerade tur- och returtidsmätningar för att använda all outnyttjad bandbredd.

Denna thesis visar även hur man kan göra experiment för att hitta begränsningar i transportprotokoll. Alla dessa experiment kommer utförs i en miljö som efterliknar det nya 5G-nätverket. Resultatet visar att både förlora paket tillsammans med en hög fördröjning påverkar mycket mer än väntat och att frekvent förlora paket är förödande för TCP. Asperas FASP är i motsats mycket tålig mot både paketförlust och hög fördröjning. Detta resultat och denna analys lägger en grund var andra kan arbeta vidare.

Nyckelord TCP, 5G, överlämningar, FASP, CUBIC TCP, tappade paket, radioaccessnät

Acknowledgments

I would like to thank Annikki Welin at Ericsson for making this thesis possible and providing me with everything that I have needed to complete this thesis. Annikki has been an excellent supervisor helping me with all the questions that have emerged during the thesis project. It has been very easy to conduct the project at Ericsson her guidance of. Thank you for the great help and most of all your availability.

I would like to thank Professor Gerald Q. Maguire Jr. for being an excellent academic supervisor and examiner. Thank you for all the interesting questions and answers about my subject and problems.

I would like to thank Tomas Thyni for his great technical help and input regarding the lab equipment and conducting the tests.

I would like to thank Ericsson in general for all the equipment I have been using for this thesis project and the location I have utilized.

I would like to thank Aspera for giving me a research license to use FASP. Without this license, this project would not have been possible.

Stockholm, August 2015
Patrik Hagernäs

Table of contents

List of Figures

List of Tables

List of acronyms and abbreviations

ACK	Acknowledgment
AIMD	Additive Increase/Multiplicative Decrease
ascp	Aspera SCP
BDP	Bandwidth Delay Product
BIC	Binary Increase Congestion
bwm-ng	Bandwidth Monitor Next Generation
CDF	Cumulative Distribution Function
cwnd	Congestion Window
DACK	Duplicate Acknowledge
DCCP	Datagram Congestion Control Protocol
disk-IO	disk-Input/Output
ECN	Explicit Congestion Notification
FASP	Fast And Secure Protocol
HDD	hard disk drive
ITU	International Telecommunication Union
LAN	Local Area Network
LFN	Long Fat Network
LTE	Long Term Evolution
MTU	Maximum Transport Unit
RMSS	Receiver Maximum Segment Size
rwnd	Receive window
SCP	Secure Copy
SMSS	Sender Maximum Segment Size
SSH	Secure Shell
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UDT	UDP-based Data Transfer Protocol
UE	User Equipment

1 Introduction

The world's population and the number of smartphone subscriptions are increasing every day. Additionally, the data traffic per subscription is increasing at an enormous rate. The networks and systems today do not meet the requirements needed to satisfy the expected future customer needs. Transport Control Protocol (TCP) is currently the main protocol used for this traffic, hence network utilization is very low because this protocol treats all packet loss as an indication of congestion, as TCP reduces the source's sending rate to avoid a congestion collapse of the network. The future 5G network will have high bandwidth capacity and low latency (below 1 ms from the user device to the base station [1]), but it still utilizes a wireless communication link and bit errors and losses will occur. As a result, TCP is a bad choice of transport protocol, hence there is a need for improvements in this transport protocol or there is a need to change to a new transport protocol that will better utilize the new 5G networks.

Switching from one radio access network to another is called a handover. Technically this means that the communication between the user's radio equipped device and a base station ends and another base station is used for subsequent communication with the user's radio equipped device. When using a radio access network handovers are inevitable due to mobility and changes in signal strength. Numerous researchers have shown that TCP does not adapt to the new network conditions in a satisfying manner when a handover occurs from a network with very high bandwidth to a network with low bandwidth [2]. Similarly, when going from a network with low bandwidth to a network with high bandwidth, TCP does not utilize the newly available high bandwidth as fast as it should.

TCP was originally designed to transport data over a relatively low data rate network* where losses primarily occurred due to congestion. However, in reality random losses due to non-zero error rates of the link occur even in the absence of congestion. This thesis addresses the problem faced by a transport protocol utilizing very high speed links with non-zero random packet losses that are not due to congestion. The aim is to improve the performance of devices utilizing a future 5G network.

This thesis will focus on Aspera's Fast and Secure Protocol (FASP) and compare its performance to one of the most used TCP versions (specifically CUBIC TCP, see Section 2.2.4.2). This thesis is part of a collaborative project in which another thesis project focuses on Network Coding using Steinwurf's Kodo library [3] and a third thesis project focuses on Forwarding Error Correction. The work that has been conducted in collaboration with these other students will be pointed out throughout this thesis.

1.1 Background

This thesis concerns the transport layer of the seven layer OSI model [4]. The most well-known and widely used transport protocol is the Transmission Control Protocol (TCP) that was originally designed in the 1970's. There have been many improvements to TCP and different versions of this protocol have been proposed (see Section 2.2.4). The biggest problem with TCP in the context of a radio access network is that TCP thinks that every packet loss is an indicator of congestion in the network, as described in RFC 5681 [5].

* Typical links in the 1970's supported only 56 kbps to 1.5 Mbps.

1.2 Problem definition

This section describes the problems that will be addressed in this thesis project in detail. After reading this section, the reader should understand why these problems exist. In order to fully understand this initial chapter, the reader may need to read Chapter 2.

In a radio access network, there are many obstacles to overcome. This thesis will investigate two problems. The first problem arises when the user equipment (UE) switches between two base stations. This process is called a handover. If the UE is currently communicating via a base station which in turn is connected to a network with relatively low bandwidth, then following a handover to a base station connected to a network with relatively high bandwidth the increased bandwidth will not be utilized. This is because the transport protocol does not realize that this increased bandwidth is now available. The transport protocols used today do not have any mechanism for dealing with large changes in bandwidth. A similar problem arises when going from a high bandwidth network to a low bandwidth network, as the throughput does not adapt to the available link bandwidth as rapidly as it should to optimize the user's perceived Quality of Service (QoS).

Figure 1-1 illustrates a handover scenario where a UE switches from a high bandwidth network with low delay to a network with lower bandwidth and higher delay. The reason for this handover might be due to movement or changes in the radio link.

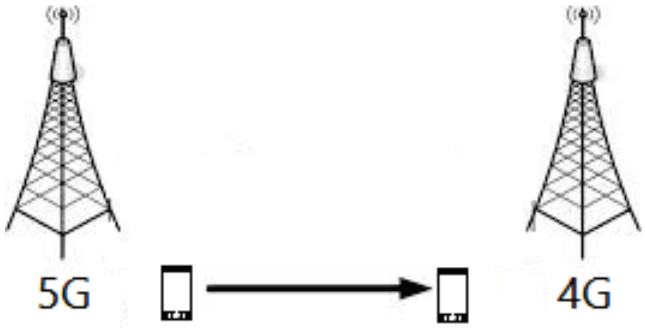

Figure 1-1: Handover scenario

The second problem arises from the instability nature of network links, especially in Radio Access Network. Losses will occur naturally, but TCP does not handle these losses well. Lakhsman and Madhow [6] conclude that this problem arises commonly and that there are many improvements that can be made in this area (such as better congestion avoidance algorithms based on round-trip delay estimates and the use of selective acknowledgement when random losses occur).

1.3 Purpose

The expected network architecture of the upcoming 5G technology has a very important impact on this thesis project. If the problems that this thesis examines are not resolved, then the user's experience with 5G will not be as good as it should be. Hopefully, in the near future 5G will be the major radio access network technology used around the world, with an expected deployment in 2020. If these 5G networks works as the IMT-2020 requirements [7] from International Telecommunication Union (ITU) [8] states, with higher data volumes, more connected devices,

higher data rate, better UEs, increased UE battery life, and reduced latency [9], then the world will benefit as every user will experience better Quality of Experience. This thesis was initially proposed by Ericsson; hence, it should also be beneficial for Ericsson. In particular, Ericsson would like to know how to solve these problems or at least know which direction to move in order to solve these problems. As Ericsson's vision is to be a leader in the telecommunications industry, this is an important step.

1.4 Goals

The goal of this thesis project is to help Ericsson and others developing 5G technologies by exploring how the ITU IMT-2020 requirements of 5G could be met. More specifically, there is a need to identify what protocols should be used and what additional technology may need to be implemented. The project's goal has been divided into the following three sub-goals:

1. Compare plain TCP and the current TCP implementations with forward error correction, network coding, and Aspera's FASP. This will provide the basis for the subsequent work.
2. Identify problems and identify where improvements could be made. The focus will mostly be on problems when using Aspera's FASP technology in a radio access network.
3. Propose a potential solution. Existing technology as well as combinations of different technologies and entirely new ideas will inspire this solution.

1.5 Research Methodology

Most of the reports that the background section built on were based on simulations, hence there was a strong motivation to do experiments on actual hardware. These experiments will be conducted in a test bed that should be able to act as a 5G network in terms of latency, capacity, handovers, and packet losses. More detailed information can be found in Chapter 3.

1.6 Delimitations

This thesis project will not seek to optimize the hardware and software in order to reach ITU's 5G requirements[7], but it will rather utilize existing hardware and software running the latest Linux kernel. The emphasis will be to examine the difference between TCP and FASP in performance in this emulated 5G environment. Some optimizations will be made, but not at all that may be needed to reach the ITU's 5G requirements.

This thesis will not alter any of Aspera's FASP code in an attempt to improve it to work better in the test bed nor with the scenarios specified in Section 3.2.2.

Most of the research that is done with high bandwidth links and high latencies uses jumbo frames where the MTU size is increased to 9000 bytes and therefore fewer packets needs to be in flight at a given time to achieve a given throughput. The advantages of jumbo frames is seen as an end system optimization for high-speed TCP [10].

1.7 Structure of the thesis

This chapter describes the problem definition in detail to enable the reader to understand why this thesis project was worth putting effort into. Chapter 2 gives the reader the necessary background of the field and introduces the reader to all the concepts necessary to fully understand this thesis. Chapter 2 also introduces many techniques and protocols that partially solve the stated problem, but might not be optimal in our context. Section 2.5.5 describes the methodology that enables the

testing and to acquire the results documented in Chapter 4.2. Chapter 4 describes the experiments and how they were conducted. Chapter 5 gives an analysis of the results from the experiments and discusses these results. The thesis ends with conclusions, a discussion of the limitations of this work, suggestions for future work, and reflections in chapter 6.

2 Background

This chapter will provide the reader all the necessary background information to understand the experiments and the corresponding result. The reader is introduced to all the protocols used as well as understanding the testing tools. At the end of the chapter the reader is also introduced to related work.

2.1 Transport and application layer protocols

This thesis's main focus is to enable applications running on devices connected via future radio access networks to fully utilize the large amount of bandwidth that will be available via the radio link. Ericsson believes that indoors and in dense environments 10 Gbit/s should be generally available[1]. Moreover, the 5G network will have both *higher bandwidth* and *lower delay* than previous radio networks. Because of this, devices will need to make better utilization of the available link bandwidth than TCP does today – otherwise many of the aims of ITU's vision for these 5G networks will not be realized.

This thesis only considers the transport layer and its interaction with the application layer. The key metric of performance that is considered is the time taken to deliver an entire file to an application. To achieve high speed file transfers Aspera's FASP operates at the application layer and uses User Datagram Protocol (UDP – see Section 2.3) as its underlying transport protocol, for further details see Section 2.5.

2.2 TCP

TCP has many properties, but the two properties that are most interesting for this thesis are that TCP ensures that all of the data sent from the sender actually arrives at the receiver and that the data is delivered in the correct order to the application. TCP also tries to utilize all of the available bandwidth, but aims to avoid congestion and to be fair to other TCP flows. These aims are achieved by using a sliding congestion window technique, not to be confused with TCP's flow control window. The use of a flow control window is a mechanism for the receiver to control how much data will be in transit at any given time, as the receiver does not want its incoming buffers to overflow due to receiving more data than the receive is prepared to handle. When TCP decides how much the sender can transmit, it considers both the sliding congestion window and the flow control window and utilizes the smaller of these. There are two variations of the sliding congestion window algorithm. These variants differ in how they handle buffers, acknowledgements, and retransmissions. The first algorithm is the *Go-Back-N* algorithm. This algorithm is the simplest and it controls the flow by allowing only one segment to be in transit at any given time. However when the roundtrip delay is large, this is not an effective algorithm, while it might work well in a Local Area Network (LAN) environment. The other variant is the *Selective Repeat* algorithm. This algorithm is more complex, but usually more effective. The crucial part of this algorithm is deciding how many segments should be transmitted before waiting for them to be acknowledged.

One of TCP's big limitations is that when the path between the sender and receiver has a large Bandwidth Delay Product (BDP) [11], there is a need to have more outstanding bytes in transit than permitted by maximum window size which is limited by the 16 bit maximum window size field in the TCP header. This field's size limits TCP's maximum window size to at most 64 Kbytes. Fortunately, RFC 1323 [12] introduced the TCP Windows Scaling option in order to overcome this problem. This option enables up to 2^{30} bytes to be in transit.

2.2.1 Connection establishment and termination

TCP is a connection-oriented protocol, which means that the sender and receiver have an ongoing connection between when the TCP connection is established and when it is terminated. A three-way handshake is required to establish a connection and another four-way handshake is used to terminate a TCP connection. Note that there is a transaction extension to TCP that allows data to be sent *without* the requirement for a persistent connection, see T/TCP in RFC 1644 [13].

Connection establishment is done with a three-way handshake. Following this handshake, there is a full-duplex session between the two parties. The sender initiates contact with a TCP packet in which the SYN bit is set in the TCP header. In this TCP header the sender also sets the sequence number to a random value *A*. The receiver then responds with a TCP packet with the SYN bit set and the ACK bit set. The header contains an acknowledgment set to *A+1* and its own sequence number set to a new random value *B*. Finally the sender responds with a TCP packet with the ACK bit set and the sequence number is set to *A+1* and the acknowledgement number set to *B+1*. Once this handshake is complete a TCP connection exists between the two parties and the data transfer phase begins [14].

TCP connection termination is also done with a handshake, but this is a four-way handshake and each party terminates independently of the other party. This means the connection can be half-open, thus only one party can send data and the other party simply listens and sends acknowledgements. When a party wants to end its TCP connection a FIN packet is sent. The other party responds with an ACK packet that terminates the connection in that direction. The same can be done by the other party. This results in a four-way handshake, but in practice this is very often shortened to a three-way handshake by combining the first ACK packet with the second FIN packet [14].

2.2.2 Congestion control

This section explains the fundamental elements of TCP's congestion control mechanisms. Different implementations, improvements, and additional congestion control mechanisms will be explained in the subsections of this section.

In TCP, all data is sent in segments. Each segment consists of a contiguous sequence of bytes received from the application layer at the TCP sender. A number of concepts are fundamental to understanding TCP, these are listed below:

Segment	A segment consists of a TCP header and some amount of application layer data. The acknowledgement number can be carried (piggyback) in the TCP header of a segment being sent in the opposite direction. The segment's size limits how much data that can be sent in a given segment.
Sender Maximum Segment Size (SMSS)	SMSS the largest sized data segment that the sender can transmit. This size is based on many factors, such as: MTU, path MTU discovery[15], and Receiver Maximum Segment Size.
Receiver Maximum Segment Size(RMSS)	RMSS is the maximum sized segment that the receiver can handle.

Receiver Window (rwnd)	rwnd specifies how many octets that the receiver is willing to accept. This can be limited by how much memory the receiver can allocate for buffering of this TCP connection.
Congestion Window (cwnd)	The congestion window specifies how much data can be in transit before segments are acknowledged. – M. Allman, et al. [5] specified that "*at any given time, a TCP MUST NOT send data with a sequence number higher than the sum of the highest acknowledged sequence number and the minimum of cwnd and rwnd*", where cwnd is the congestion window and rwnd is the receiver window.
Initial Window	The initial congestion window after the three way connection establishment handshake.
Slow start threshold (ssthresh)	The slow start threshold ends the Slow Start phase.
Flight Size	Flight Size [5] is the amount of data sent, but not yet acknowledged.

TCP treats lost segments as a warning of congestion in the network. This can obviously be incorrect, especially when there are non-optimal link conditions. In all links, random bit errors can occur and these may lead to packet losses, even when there is no congestion. Because the basic version of TCP always treats these losses as if there are congestion, different mechanisms have been introduced to handle congestion. These will be described in the following subsections.

2.2.2.1 Slow Start

At the beginning of a TCP connection or in some versions of TCP after a bandwidth bottleneck has been passed, the protocol uses the *Slow Start* algorithm. According to this algorithm, the congestion window (cwnd) is doubled until a loss occur. This is an exponentially fast way of reaching the bandwidth limit of the link, hence the name of the algorithm could be considered ironic. When a loss occurs the congestion window is halved and the connection then enters the Additive Increase/Multiplicative Decrease (AIMD) phase described in Section 2.2.2.2

2.2.2.2 Additive Increase/Multiplicative Decrease

AIMD is the basis of TCP's congestion control mechanisms. AIMD is used with variations in both TCP Reno and Tahoe. When using AIMD, the congestion control algorithm slowly increases the congestion window and as soon as a loss occurs it decreases the window size multiplicatively, typically halving it. An advantage of this algorithm is that if there are multiple TCP flows passing over the same link they will sooner or later each get a fair share of the bandwidth, thus achieving TCP Fairness (see Section 2.2.3).

2.2.2.3 Fast Retransmit Fast Recovery

A standard enhancement of TCP exploits the concept of duplicate acknowledgements (DACKs). The fast retransmit fast recovery algorithm speeds up retransmission by avoiding waiting for a time out to occur. If a loss occurs, then the next (expected) segment that is acknowledgement will continue to be the previously received segment, thus generating the first DACK. After two more DACKs, the fast retransmission algorithm kicks in. First the algorithm sets the ssthresh to no more than $max(flight\ size/2,\ 2 \cdot SMSS)$ as described in RFC 5681 [5]. The lost segment is retransmitted and the cwnd is set to ssthresh $+ 3 \cdot$ SMSS. This is done because the receiver has buffered three

segments, although we do not yet know which three segments. For every DACK, cwnd is incremented by SMSS. This will not work very well when there are multiple losses in a single cwnd, but there are other ways to improve this case, which can be found in the RFC TCP Congestion Control [5].

After the retransmission of the lost segment is acknowledged, Fast Recovery allows cwnd to be ramped up faster. This is not done by using the Slow Start algorithm (previously described in Section 2.2.2.1), but rather using only an additive increase. This leads to a saw tooth like flow as shown by TCP Reno in Figure 5-3.

2.2.3 TCP Fairness

When multiple TCP connections share a common link, it is important that one of these TCP connections does not take all the available bandwidth, starving the other connections of bandwidth. TCP Fairness is achieved when each of the *N* connections sharing a link each take no more than $1/N$ of the available bandwidth. This is achieved by the AIMD function in TCP (explained in Section 2.2.2.2). According to the basic theory of AIMD, any number of TCP connections will eventually adapt to each other.

2.2.4 TCP improvements

This section briefly describes techniques used to improve the original TCP protocol. Most of these techniques aim to reduce the negative effects of AIMD described in Section 2.2.2.2.

2.2.4.1 New Reno

New Reno improves the Fast Recovery phase in a way that every time the sender receives a DACK it sends a new segment at the end of the congestion window. Thus, solving the problem of Fast Retransmit Fast Recovery only retransmitting one segment for each Round-Trip Time (RTT). In this version of Fast Recovery, the highest unacknowledged segment sequence number is saved. Once that segment is acknowledged TCP will continue with congestion avoidance.

2.2.4.2 CUBIC TCP

The TCP implementation in the current Linux kernel uses CUBIC [5]. Therefore, many UE's uses this implementation, since Android runs on a Linux kernel. This algorithm works well on a Long Fat Network (LFN)[*] as does TCP Binary Increase Congestion (BIC), but it is less aggressive than BIC as it uses a cubic function ($W_{cubic} = C \times (T - K)^3 + W_{max}$, where C is a constant and $C = \sqrt[3]{\frac{W_{max}\beta}{C}}$) to decide how large the congestion window will be. This algorithm theoretically behaves as shown in Figure 2-1.

[*] A network with a high BDP.

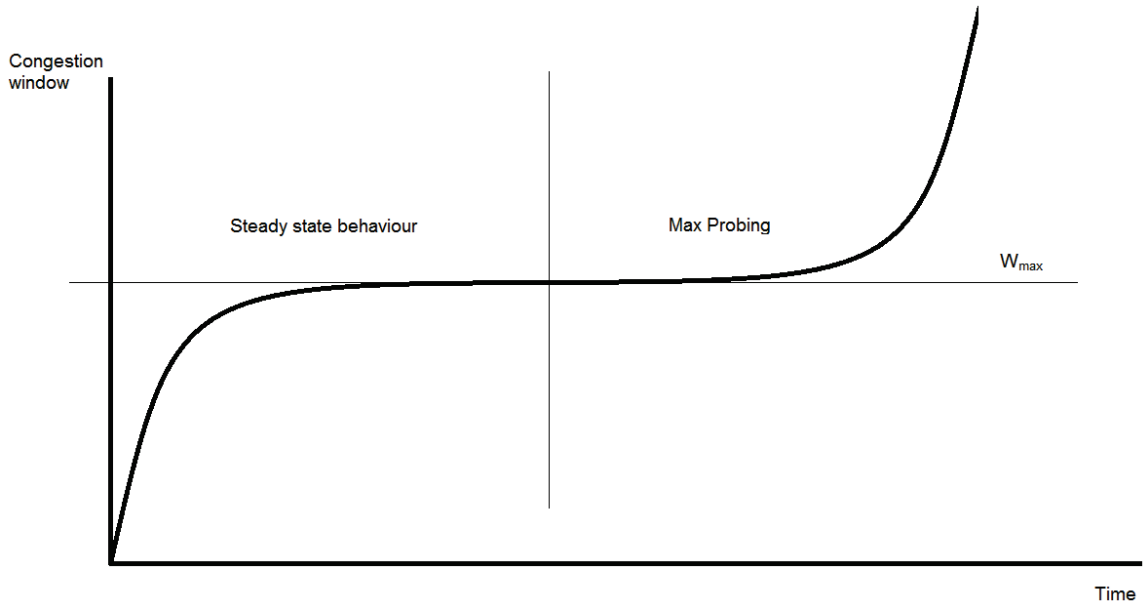

Figure 2-1: CUBIC TCP theoretical growth

2.2.4.3 Compound TCP

Compound TCP is a Microsoft properietary TCP implementation. Compound TCP is used in their Windows TCP stack. A difference from standard TCP is that this implementation uses estimates of the queuing delay to check for congestion in the network. The algorithm actually uses two windows; one AIMD window and one queuing delay window and the actual sliding window is the sum of these.

2.3 User Datagram Protocol

The most commonly used protocol for connectionless transport, such as video and audio streams and video games, is User Datagram Protocol (UDP). There are many applications that use UDP with additional control mechanisms to provide exactly the amount of control desired. For instance in a video stream the data does not need to arrive in order nor need it all arrive, as losses are okay and simply lower the media quality. The application is responsible for buffering, reordering, and dealing with missing datagrams. However, the video stream might need to be connection-oriented, but this can be implemented on top of UDP.

The UDP header contains only four fields, as shown in Figure 2-2. The source UDP port field is optional, but necessary if the sender wants a response of some sort. The destination port specifies which application the packet should be delivered to as this application must open a UDP socket that listens on this port. The length of the entire UDP segment and the use of the optional UDP checksum are described below.

0 15 16 31

Source port	Destination port
Length	Checksum
data octets …	

Figure 2-2: UDP header. Four fields of information each 16 bits long, which makes the header 8 byte in total.

The UDP checksum is the only control mechanism in UDP and it ensures that the data does not have errors or has been misrouted. The checksum is computed over a combination of a pseudo header (shown in Figure 2-3), used only in the checksum computation, and the UDP datagram itself. The sender creates a checksum – if needed, otherwise it put a zero in the checksum field (as the checksum is optional). When the datagram is received, and if the checksum field is non-zero, the receiver computes the checksum. The pseudo header contains the source IP address and the destination IP address, and it also contains the length of the entire UDP datagram, and the protocol field from the IP header [16].

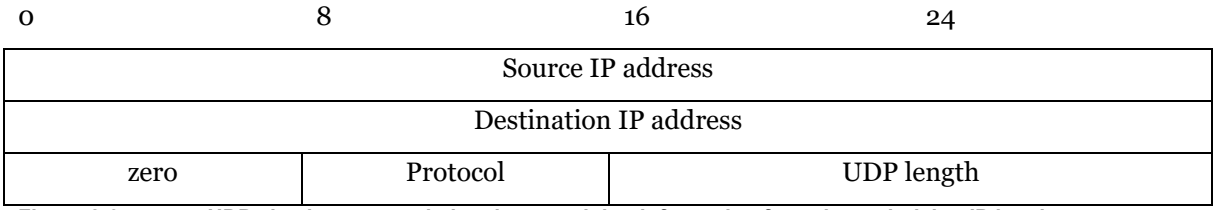

Figure 2-3: UDP checksum pseudo header containing information from the underlying IP header.

A sender of UDP datagrams does not know if the receiver actually receives the datagrams or whether they were lost on the way. A receiver waits for incoming datagrams on a specific port. This part must be known to the sender, but how the sender knows the destination port number is outside the specifications of UDP.

In summary, UDP is a very light weighted protocol that runs on top of IP. It adds very little overhead, primarily simply enough to provide multiplexing and demultiplexing via the UDP port numbers. It has no mechanisms to ensure network fairness, congestion control, or data reliability and thus is optimal for very fast transport of datagrams. Many protocols are built on top of UDP as described in the following subsection.

2.4 Running protocols on top of UDP

Many protocols have been built on top of UDP, as UDP is implemented by nearly all network components, hence nothing new needs to be implemented within the networks. As a result many protocols have been built on top of UDP to add congestion control and support reliability. In many cases, the protocol is implemented entirely in the application layer. One of the most common protocols of this form is the UDP-based Data Transfer Protocol (UDT). UDT provides connections, reliability, and congestion control. Another is the Datagram Congestion Control Protocol (DCCP) [17], which uses Explicit Congestion Notification (ECN) [18] and ECN Nonce[19].

2.5 Aspera's FASP

This section provides information about Aspera's FASP, the information given comes from Aspera's own explanations and details about the protocol. Unfortunately, there is limited public information about the protocol, but there is some information disclosed in one of Aspera's patents [20]. Additional information has been taken from Aspera's advertisements for FASP as this is a commercial product, hence it should be read with caution.

2.5.1 Bulk data

FASP is for bulk transport of data, hence the order of the packets is not relevant only that the entire file is correctly delivered. In practice this means that if the first packet is lost the receiver will need to see that it is retransmitted by the sender and then placed into the proper position in the file.

Aspera also has a streaming protocol where the byte order is crucial, but this thesis has only tested the bulk data transfer protocol.

2.5.2 UDP

Aspera's FASP is built on top of UDP as the underlying transport protocol and therefore it might experience UDP's limitations, such as no link aggregation. FASP is implemented in the application layer of the OSI model and the implementations are only needed in the end hosts. A disadvantage of this could be that the network equipment does not support any of the extra processing of the packets; hence all of the work is done by the end host's processor. This can potentially put a heavy load on the processor. Since this thesis project is focusing on high speed radio access networks, this means that all this extra processing occurs in UE's processor. As these UEs are typically small hand held devices this additional processing may incur more cost than there is processing capacity for or could drain the device's battery – hence these factors need to be evaluated in a future project.

2.5.3 Queuing delay

FASP does not use packet loss as congestion symptom as TCP does but rather takes advantage of small variations in the queuing delay of the packets. Figure 2-4 shows that it is the competing traffic that creates these queuing delays (and delay variance). This is a much better method of estimating congestion than using packet loss as symptom because loss is only experienced after a queue in the router is full and the router has started to drop packets. The idea of exploiting the queuing delay information is really simple. When a router has received more packets within a timespan than it can send, queuing delays occur as packets are placed in a queue and remain there until either the packet can be forwarded or the packet is dropped.

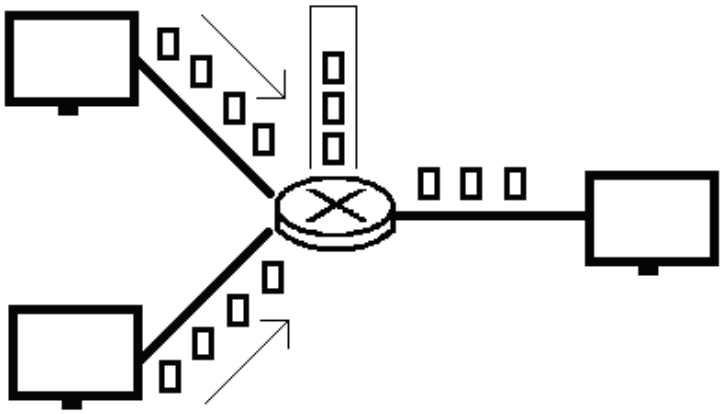

Figure 2-4: Queuing delay

Using queuing delay as a symptom of congestion requires a really good RTT estimation algorithm in order to notice these small alterations in RTT and to react to them by either lowering the sender's transmission rate when delay increase or increasing the sender's transmission rate when the delay decreases. Aspera claim that they have an RTT estimation algorithm that is very good and estimates RTT to near perfection. This algorithms follows Van Jacobson's theory as described in his paper "Congestion Avoidance and Control" [21]. Unfortunately, this algorithm is not detailed any further since this is the heart of Aspera's FASP implementation. However, Aspera claims that FASP achieves total TCP fairness claim since it uses Van Jacobson's theory.

2.5.4 Disk-IO and CPU scheduling

One problem when sending packets at very high rates is that the time between accessing the CPU is much larger than the time a process can utilize the CPU. This becomes a problem when the CPU only processes one packet each time it is allocated in the CPU. For example, this might happen when the system is heavy loaded or disk-Input/Output (disk-IO) subsystem is not reading the data fast enough for the processor to handle more than one packet. FASP attempts to group packets into batches whose size is calculated to be able to fully utilize the CPU when the application is allocating the CPU. Aspera claims that this increases a server's maximum packet injection rate by a lot.

Aspera warns that disk-IO can grow quite heavy very quickly and the use of high speed disks is necessary to achieve higher injection rates and therefore higher throughput.

2.5.5 Related work

This section describes some related work. It might be interesting for readers to consider other solutions and other techniques.

2.5.5.1 FASPstream

Aspera has developed a specialized streaming transport protocol called FASPstream [22] and it would be interesting to examine that protocol in order to understand how streaming services could benefit from it. Aspera is claiming that FASPstream achieves as good a utilization as FASP and can be used, for example, to live stream sports programs. There is a great demand[23] for this kind of service and if Aspera is correct that its utilization is as good as FASP, then this would be somewhat groundbreaking for live stream services.

2.5.5.2 Google QUIC

Google has developed a new transport protocol called QUIC [23]. QUIC has properties similar to Aspera's FASP. However unlike FASP, QUIC does not claim to be independent of packet loss. It seems that QUIC still uses loss as a symptom of congestion. The protocol has several mechanisms to overcome random packet loss, for instance through Forward Error Correction (FEC). QUIC has the advantage that the source code of the implementation is available for everyone [24].

2.5.6 Summary

This chapter has introduced the reader to numerous concepts that are important to understand for the rest of this thesis, specifically TCP, UDP, and Aspera's FASP. TCP and FASP will be compared in Chapter 4. Note that the major reason to choose CUBIC TCP is because it is the standard transport protocol in Linux and therefore also in Android. As Android has the major market share for smartphone operating systems this is the major potential future application of the results of this thesis project.

3 Methodology

The purpose of this chapter is to provide an overview of the research method used in this thesis. Section 3.1 describes the research process. Section 3.2 describes the test bed. Section 3.3 focuses on data reliability and validity. This project started with a literature study to obtain all of the information necessary to do the analysis of the data collected in the planned experiments and to understand the results. The project required implementing both a suitable test bed and all of the necessary software and tools. Using this test bed, a series of tests were performed (see Chapter 4) and an analysis of the results is presented in Chapter 0.

3.1 Research Process

This research process focuses on collecting empirical evidence by conducting two different sets of tests. The first set of tests using TCP provides a performance baseline and the other set using Aspera's FASP enables us to compare the performance of FASP to TCP. The baseline uses today's most popular TCP implementation and shows what happens today in the handover scenarios described in the problem description. These test sets are described in Section 4.1. The result from these two test sets will be compared and discussed in Section 4.2.

3.2 Test bed design

This section describes the test bed's hardware and the software, as well as the configuration parameters used to conduct the two sets of tests.

3.2.1 Hardware

The test bed consists of three computers: client, network, and server. The details of these computers can be found in Table 3-1. These hardware choices were in part made by Ericsson before the project began and remained unchanged. Unfortunately, some of the equipment, for instance the hard disk drive (HDD) was not optimal for the test sets and testing showed that a RAMDisk was needed to be used to remove the bottleneck that the HDD caused. All of the subsequent tests were performed using a 32 GB RAMDisk. The network cables were shielded CAT6A to ensure that the tests were not compromised by the cables.

Table 3-1: Hardware details, note that the HDD is not used, but rather a 32 GB RAMDisk was used.

Subsystem	Description
Processor	Intel Xeon E5-1630 v3 3.8 GHz
RAM	DDR4 64 GB @ 2133 MHz
Network adapter	Myricom-10G-PCIE2-8C2-2T
HDD	SATA 6 GB/s, 7200 RPM, 64 MB cache

3.2.2 Software

Both the server and client computers ran Ubuntu with the latest stable Linux kernel, as further described in Table 3-1. This decision was made because this was a familiar operating system and it was relatively easy to install the software needed for testing. The alternative was to use FreeBSD not only in the network computer but also in the hosts, but after realizing that the testing software would run perfectly on top of a Linux kernel this alternative was rejected.

Table 3-2: Software details of client & server

Client & Server Software	Description
OS	Ubuntu 14.04.1
Linux kernel	3.16.0-37-generic

It was natural to choose FreeBSD as operating system for the network computer, since the most widely used traffic shaping system is *dummynet* and this software is built-into FreeBSD. The latest stable version of FreeBSD was used.

Table 3-3: Software details of network

Network software	
OS	FreeBSD 10.1-STABLE
IPFW version	Built-in
Dummynet	Built-in

3.2.2.1 Dummynet

Dummynet provide pipes that can be configured to shape the traffic going through the computer's interfaces. The details of the 12 configurations of pipes used during testing can be found in Table 3-4. The values in Table 3-4 were obtained by analyzing tests made by Ericsson in North America and Cumulative Distribution Function (CDF) graphs were made to extract these values in an end-to-end connection. The values correspond to 50% in the CDF graphs and it is questionable what packet loss rate that comes from congestion in the network and what packet loss rate comes from random packet loss. The conclusion that is drawn is that extracting the values from 50% assumes that this packet loss rate is due to random packet loss in the network. Note that the value 0.06% that is used is an assumption of the packet loss rate that is due to random packet loss and the test results is therefore comparable with a network that has 0.06% random packet loss. Note that the Aspera's FASP tests were made with 0.13% packet loss which is around **twice** the packet loss rate (of 0.06%) used with our TCP tests.

Table 3-4: Configurations of Dummynet pipes used for testing

Pipe number	Bandwidth Mbit/s	Packet loss %	Delay ms
1	10 000	0	74
2	10 000	0.06	20
3	10 000	0.06	74
4	1 000	0	92
5	1 000	0.06	20
6	1 000	0.06	92
7	400	0	92
8	400	0.06	20
9	400	0.06	92
10	100	0	92
11	100	0.06	20

12	100	0.06	92

Handovers in the tests were achieved by switching (using a command script) between these pipes during an ongoing file transfer.

3.2.2.2 Measurement software

All of the project's measurement software and tools are described below. To decide which tools to use a lot of testing and implementations were made and the final decisions was based upon what best suited the project scenario. While working on the test bed and deciding which software to use a lot of tests were made using iPerf [25]. The software is used to transmit data as well as measure how long it takes to do a transfer.

To transmit a file between the two end-hosts, the selected software needed to provide reliable transport. In addition, it was desirable that this software was widely used today by a lot of end users and that it was likely that this software would continue to be used in the future. The file transfer software that was finally chosen was Secure Copy (scp [26]). The SCP program utilizes Secure Shell (SSH) to secure the data being transmitted. This was an important property of the software since most of the applications that are responsible for a large share of internet traffic include some sort of security, as for example Netflix and Youtube applications do. As this project is concerned with end users' satisfaction, it is important that the tests are comparable to what the end user would experience their future 5G network usage.

To be able to measure the traffic reception the Linux tool Bandwidth Monitor Next Generation (bwm-ng [27]) was used. The program provides a good and stable measurement of the bandwidth. The program takes snapshots every 500 millisecond of the received bandwidth and therefore the output was not very detailed for short time periods. The program was also able to measure disk-IO which was used when setting up the test bed to ensure that disk-IO was not a bottleneck.

3.2.3 Configurable parameters

Even when running natively Ubuntu, the kernel's parameters are not optimized for 10 Gbit/s throughput (as such interfaces are still rather rare in typical users' configurations). This section provides some information about all of the changes that were made in the Linux kernel to achieve high network performance.

The computers UEFI was changed to disable hyper threading, as hyper threading can in some cases cause packets to unnecessary be transmitted out of order. Additionally, CPU frequency scaling was turned off so that the processors ran at their maximum frequency all the time.

Changes to the file */etc/sysctl.conf* were extensive and a lot of preliminary testing was made to optimize these parameters in a brute-force trial and error kind of manner. Table 3-5 shows all the parameters altered for TCP and UDP. Following this table are some of the arguments motivating these altered values.

Table 3-5: TCP & UDP parameters in /etc/sysctl.conf

Parameter name	Initial Value (min, default, maximum)
net.ipv4.udp_rmem_min	8192
net.ipv4.udp_wmem_min	8192
net.ipv4.tcp_rmem	4096 87380 16777216
net.ipv4.tcp_wmem	4096 87380 16777216
net.core.optmem_max	40960
net.core.wmem_default	536870912
net.core.rmem_default	536870912
net.core.rmem_max	536870912
net.core.wmem_max	536870912
net.core.netdev_max_backlog	250000
vm.swapiness	10

The **net.ipv4.udp_rmem_min** and **net.ipv4.udp_wmem_min** ensure that the buffer for UDP is at least 8192 bytes. If this value is too low it may lead to low performance. The same is true for the TCP buffer, **net.ipv4.tcp_rmem** and **-wmem** where the values are minimum, default, and maximum; where the maximum is set to a 16 Mbyte (which experienced was sufficient). The value of **net.core.optmem_max** is the maximum amount of optional memory buffers and was set thru trial and error. The values of **net.core.wmem_default, -rmem_default, -rmem_max** and **-wmem_max** were set to a high value - as these control the size of the socket's receive buffer**.** The value of **net.core.netdev_max_backlog** increases the queue length used by the processor and via testing this was revealed to be a sufficient value. Finally, **vm.swapiness** was set to 10 to decrease the swapping of memory to disk.

3.3 Assessing reliability and validity of the data collected

This section explains why the data that was collected via the tests are both reliable and valid.

3.3.1 Reliability

The main difference between this thesis project's experiments and other similar experiments is that these tests used actual hardware running the protocols, while most similar evaluations utilized simulations. The results from those simulations and calculations are mainly *theoretical* results, as they are based upon the model used by the simulation. In contrast, the results reported here are actual results anyone could measure when testing the protocols tested with the hardware that is available today.

3.3.2 Validity

The tests in this thesis are relying on several things: (1) that the hardware works correct, (2) that the software works correct, and (3) that the traffic shaping is providing the correct emulated link behavior. The traffic shaping has three components: bandwidth, packet loss, and delay. All of these parts are thoroughly investigated below.

- The hardware is very hard to validate in any other way see that it performs as well as the tests require. In our case the hardware is able to send and receive sufficient packets to saturate a 10 Gbit/s link.
- To validate the software there are three parts that need function: iPerf3, bwm-ng, and the kernel software that implements the transport protocol. When iPerf3 was tested with UDP until 10 Gbytes has been successfully transferred (without any firewall in the network computer) bwm-ng reports that 10 Gbytes have been received. When iPerf3 is running with TCP and the transfer rate is reported as 9.57 Gbit/s, the total amount of data sent in under 120 seconds is exactly the same as bwm-ng reports.
- To ensure that the traffic shaping in the network computer works as it should, three simple tests were made. To test that the delay is correct a simple ping command is executed. To test the bandwidth limitation Iperf3 was used with UDP, to see that if the bandwidth is set to 1 000 Mbit/s, then Iperf3 transfers 1 000 Mbit/s, but as soon as the bandwidth is increased the network starts to drop packets. The TCP loss percentage was calculated based upon how much data was send by the server computer divided by how much data was actually received at the client computer.

4 Running tests

The project's main task was to run a number of tests to evaluate whether Aspera's FASP enables user satisfaction in a 5G radio environment better than CUBIC TCP. For all of these tests the MTU was set to 1500 bytes, because this is still the most common MTU used in Internet and probably will remain so for the near future. There are numerous papers showing the benefits of using jumbo frames and these benefits can probably have been gained in these tests using jumbo frames. However, because this thesis focuses on the end users experience and not all of these users will be utilizing larger MTU even in the future, an MTU of 1500 was used. The tests are divided into three scenarios as defined in the following section.

4.1 Test sets

The test sets consists of a number of scenarios. The first scenario considers a UE connected via a 5G network, as emulated using pipe number three in Table 3-4. After 40 seconds a handover occurs where the UE is switched to another network with lower speed and less bandwidth, emulated as pipe number twelve in Table 3-4. The second scenario considers a UE connected via a low speed network with limited bandwidth, as emulated by pipe twelve in Table 3-4. After 40 seconds a handover occurs where this UE is switched to a 5G network, emulated using pipe three in Table 3-4. The third scenario considers a UE experiencing two handovers. First from pipe number nine and then after 40 seconds to pipe number twelve and after yet another 40 seconds switched to pipe number six. These scenarios are summarized in Table 4-1.

Table 4-1: The three scenarios, where the pipe numbers refer to Table 3-4.

Scenario number	Handover(s)
1	Pipe three → pipe twelve
2	Pipe twelve → pipe three
3	Pipe nine → pipe twelve → Pipe six

4.2 I/O problems

Achieving high throughput is very demanding of the underlying hardware and this became a problem even when reading and writing to memory (i.e., to/from the RAMDisk). The test bed was limited by the mechanical HDD once the reading or writing rate was above 2 Gbit/s, as seen in Figure 4-1.

As noted earlier, the solution to the limited throughput of the HDD was to create a RAMDisk. Fortunately, this was not a problem in the test bed as both the client and the server had 64 GB of RAM available. A 32 GB RAMDisk was sufficiently large to send and receive a 30 GB file. As seen in Figure 4-1, the throughput went up to a maximum of 9.8 Gbit/s which is accepted as the maximum transfer speed. These tests was created with SCP transfers of very large files between the end hosts.

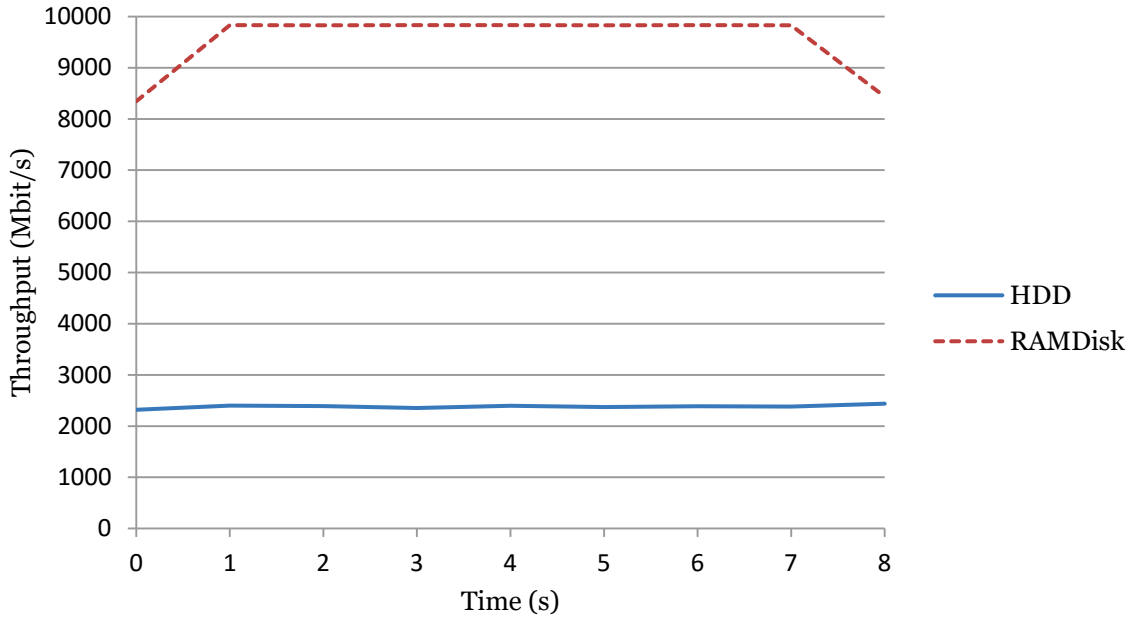

Figure 4-1: **Throughput with HDD as the ultimate destination versus a RAMDisk as the ultimate destination.**

4.3 Processing problem

When too many packets are arriving and need to be processed faster than the processor can handle them, then a processing bottleneck is created. A source of this difficulty is the interrupt scheduling process. The problem was solved by Myricom for TCP packets in the hardware interface, but unfortunately not for UDP packets. This bottleneck limited UDP throughput to a maximum of 7.3 Gbit/s. This was solved by binding the interrupt scheduler to a dedicated processor core. This is further investigated in Victor Johansson's thesis [28].

4.4 Baseline

The baseline tests consisting of performing the transmission using TCP Reno, CUBIC TCP, and UDP as the underlying transport protocol. These tests constitute the baseline to which Aspera's FASP tests will be compared and analyzed.

4.5 Aspera's FASP

To test Aspera's FASP, their licensed program is used to transfer a 30 Gbit file in each of the three test scenarios. During these tests everything remains the same as in the earlier tests except instead of using SCP, the transfer utilizes Aspera's FASP program called Aspera SCP (ASCP). ASCP has numerous parameters that can be set to initialize a transfer. The implementation of Aspera's FASP is easy if one follows their installation guide and recommendations, but to achieve the desired high utilization for 5G is harder. Most of the implementation effort (beyond implementation of the test bed) was done attempting to achieve high utilization in a 5G (like) environment. The difficulty of achieving high utilization occurred due to the lack of processing power and inability to prepare and send sufficiently many packets per second. Since the MTU was 1 500 bytes, a lot of processing power was necessary and it seemed that the extra preparation and calculation that Aspera's FASP is doing was too much for the processor used in the test bed. The attempts to improve the utilization of the

implementation followed a white paper written by Aspera and Intel entitle “Big Data Technologies for Ultra-High-Speed Data Transfer in Life Sciences” [29]. This white paper states that using a processor that enables Intel Integrated I/O and Intel Data Direct I/O Technology (Intel DDIO) the results presented in the paper should be achieved. Specifically a throughput of around 4 Gbit/s* should be achieved. According to Aspera the test bed is sufficient and the ASCP transfer parameters are what they should be. Figure 4-2 is a snippet from the ASCP log file. This snippet shows the transfer parameters. Notice that in this test a limit of 3 Gbit/s is set because the goal of the test was only to achieve greater than 2 Gbit/s throughput. The second and third rows of the ASCP transfer log (Figure 4-3) shows that the average transfer rate was below 1.88 Gbit/s. This is in contrast to the white paper and the statements by researchers at Aspera why that the transfer rate should be around 4 Gbit/s.

```
LOG FASP Session Params uuid=fedf157c-95e7-492c-bc15-e836ddec6c93 userid=0 user="server"
targetrate=3000000000 minrate=0 rate_policy=fair cipher=none resume=0 create=0 ovr=1
times=0 precalc=no mf=0 mf_path=- mf_suffix=.aspera-inprogress partial_file_suffix=
files_encrypt=no files_decrypt=no file_csum=none dgram_sz=0 prepostcmd=- tcp_mode=no
rtt_auto=yes cookie="-" vl_proto_ver=1  peer_vl_proto_ver=1 vl_local=10000000000
vlink_remote=0 vl_sess_id=1107 srcbase=- rd_sz=0 wr_sz=0 cluster_num_nodes=1
cluster_node_id=0 range=0-0 keepalive=no test_login=no proxy_ip=- net_rc_alg=alg_delay
exclude_older/newer_than=0/0
```

Figure 4-2: ASCP transfer parameters from log

```
LOG FASP Session Statistics [Sender] id=fedf157c-95e7-492c-bc15-e836ddec6c93 delay=13ms
rex_delay=44ms solicited_rex=1.88% unsolicited_rex=0.00% ave_xmit_rate 1.88Gbps
effective=98.12% effective_rate=1.85Gbps (detail: good_blks 73949162 bl_total 75367059 bl_orig
73949163 bl_rex 1417896 dup_last_blks 1) (sndr ctl: sent 477 rcvd 477 lost 0 lost 0.00%) (rcvr ctl:
sent 79505 rcvd 79505 lost 0 lost 0.00%) (rex  ctl: sent 11842 rcvd 11842 lost 0 lost 0.00%)
(progress: tx_bytes 107374182400 file_bytes 107374182400 tx_time 477809833) (estimate
transfer size: 0 bytes)  sess->instru.num_rex_xfer_not_found 0
sess->instru.num_rex_abrtxfer_not_found 0
```

Figure 4-3: ASCP transfer statistics from log

Despite many emails back and forth to and from researchers at Aspera this problem seemed to be a mystery to them. A conclusion might be that the paper did not present results that are generally valid. This lack of utilizations is obviously something that Aspera needs to keep working on before 10 Gbit/s networks can be utilized. This means that later versions of Aspera’s software need to handle calculations and preparations better. Most users of Aspera’s FASP uses jumbo frames which eliminates this problem. Using jumbo frames the test bed was able to achieve transfer rates of up to 9.8 Gbit/s. However, users of the Internet cannot simply demand that their Internet service provider enable jumbo frames, hence they are stuck with the standard MTU size of 1500 bytes.

* Note that Intel reports in the white paper that they needed to use 3-4 streams (hence spreading the load over multiple CPUs) to achive a 10 Gbit/s throughput.S

4.6 Increasing datagram size

By altering the ASCP transfer option for datagram size that is by default detected by ASCP itself as the path MTU, the maximum transfer speed was improved to above 6 Gbit/s. This alteration changes the maximum throughput from 1.8 Gbit/s to 6 Gbit/s and might be explained with the decreased number of computations the protocol does on each datagram, thus lowering the systems workload. By using larger pieces of data, less computations are necessary because surely the computations are per datagram and not determined by the size of the data itself. The datagram size is set to the maximum allowed by the protocol [30], 10 000 bytes, that is a bit larger than the Jumbo Frame MTU size of 9 000 bytes. As mentioned in the previous section, Aspera's FASP has no trouble utilizing a 10 Gbit/s link using Jumbo Frames. What happens when ASCP sends these large datagrams is that they get fragmented before transmitted. The problem with IP fragmentation that is relevant in this thesis is the basic principle of IP fragmentation that if one fragment is lost the entire segment is considered lost. This makes the random packet loss rate larger than it actually is with a factor of how many fragments is needed per segment. In a protocol like TCP, this is merely devastating. Why this does not have a bad effect on Aspera's FASP is because the protocol is not that affected by packet loss. In this situation the fragmentation only increases the throughput. Note that there are many other problems regarding fragmentation that is not at all considered in this thesis.

5 Analysis

In this chapter, the results of the tests are presented and discussed. In Section 5.1 the baseline results are presented, analyzed, and discussed. Section 5.2 presents, discusses, and analyzes the Aspera's FASP results. Section 0 looks at the reliability and validity of the results. The chapter ends with a general discussion of the results.

5.1 Baseline results

In the literature study of TCP, it seemed that TCP's bad behavior was exaggerated and TCP's congestion control seemed to be underrated, but as soon as testing started the results showed that TCP's bad behavior was worse than expected and TCP's congestion control seemed to only work *against* fully utilizing the available bandwidth. Figure 5-1 shows that both TCP Reno and CUBIC TCP suffer greatly from losses. While none of these protocols utilized the 10 Gbit/s bandwidth, CUBIC TCP was much better in that it was almost always fast as TCP Reno. This is neither surprising nor interesting because CUBIC TCP *should* outperform TCP Reno, especially with the relatively long delay of the link in scenario 1 (74 ms and 92 ms). Note that CUBIC TCP does *not* utilize the bandwidth as well as it could. Moreover, in the event of a loss the utilizable of the available bandwidth is clearly poor. One could argue that the utilization would be better for CUBIC TCP if only there had been more time before the bandwidth was decreased due to the handover. However, this is incorrect because 40 seconds is quite long when considering the download of a web page or a buffered video stream. Given the very high available data rates the time to adapt is way too long. Even after a very long time neither version of TCP makes good utilization of the available bandwidth, due to the non-zero packet loss rate.

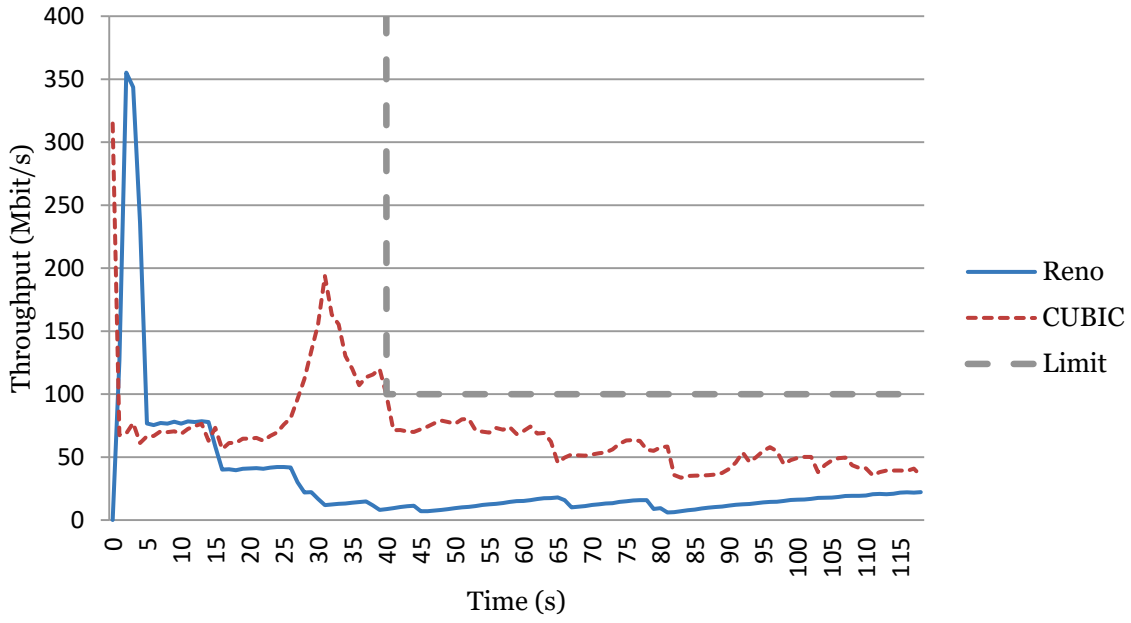

Figure 5-1: **Throughput of TCP Reno and CUBIC TCP in scenario 1. The available bandwidth is 10 000 Mbit/s with 74 ms RTT the first 40 seconds and then 100 Mbit/s with 92 ms RTT. The loss rate is 0.06%.**

Figure 5-2 shows the throughput with a reversed handover from that of Figure 5-1. The only thing that is remarkable with this result is that the TCP Reno transfer is very lucky that a loss does not occur for the first fifteen seconds. Otherwise the CUBIC TCP utilization is about twice as fast as the Reno TCP utilization.

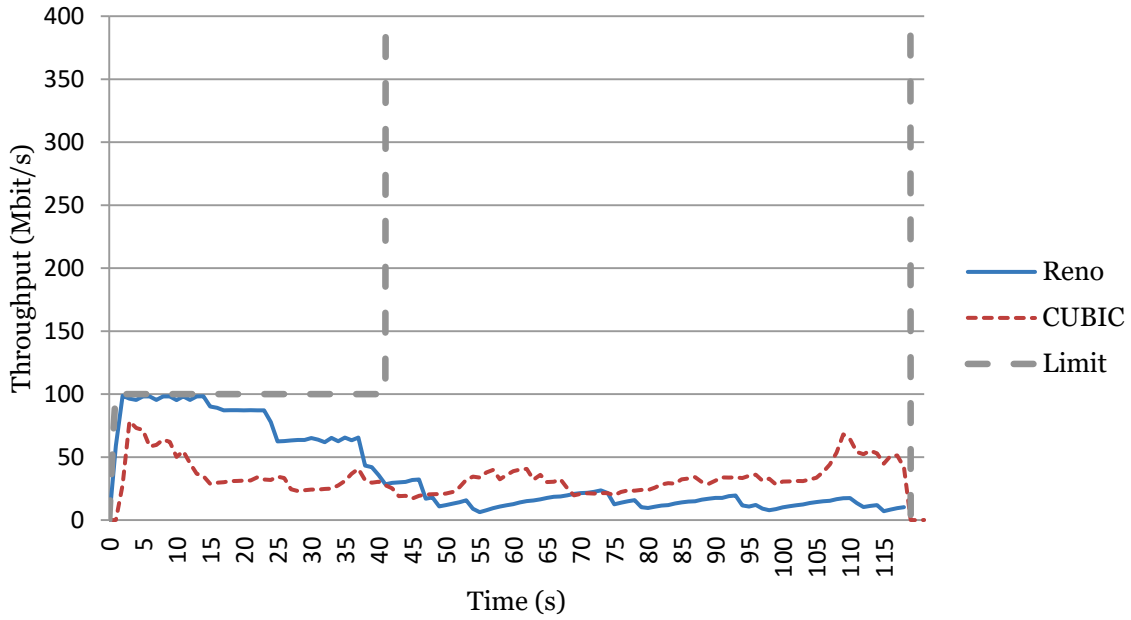

Figure 5-2: **Throughput of TCP Reno and CUBIC TCP in scenario 2. The available bandwidth is 100 Mbit/s with 92 ms RTT the first 40 seconds and then 10 000 Mbit/s with 74 ms RTT. The loss rate is 0.06%.**

In Figure 5-3 the results are as expected. What can be noted is that the handovers did not affect the throughput, but instead it is the devastating effect of random losses that limits the throughput for both protocols.

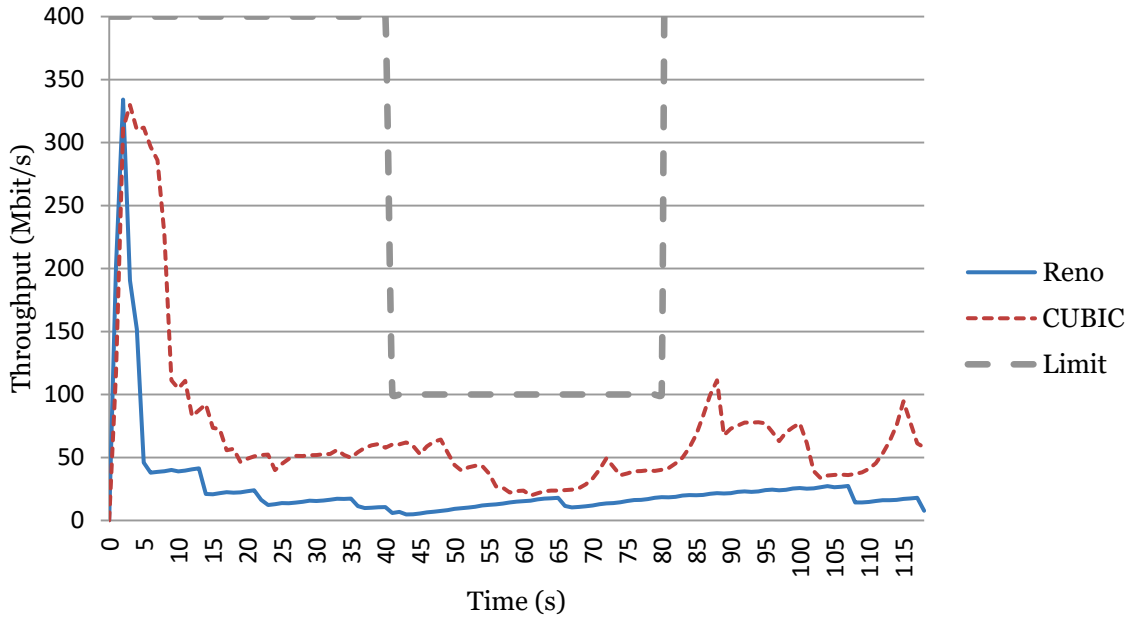

Figure 5-3: **Throughput of TCP Reno and CUBIC TCP in scenario 3. The available bandwidth is 400 Mbit/s the first 40 seconds and then 100 Mbit/s for 40 seconds and then changed to 1 000 Mbit/s. The loss rate is 0.06% and the RTT is 92 ms.**

5.2 Aspera's FASP results

In the literature study all of Aspera's own results showed the throughput of FASP was extraordinary and almost fully utilized the link's available bandwidth in all scenarios. Although they achieved full 10 Gbit/sec link utilization when using jumbo frames, they only achieved about 4 Gbit/s throughput over a 10 Gbit/s bandwidth when using a MTU of 1 500 bytes. Aspera's FASP manages to transfer data at 6 Gbit/s, as shown in Figure 5-4.

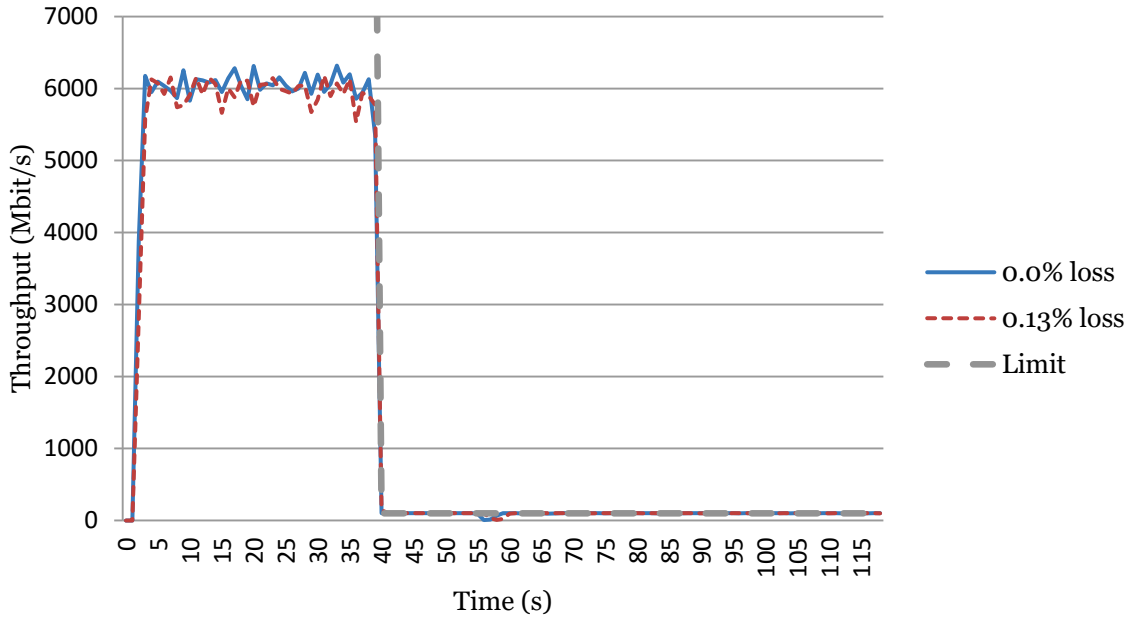

Figure 5-4: **Throughput of Aspera's FASP protocol with and without 0.13% packet loss in scenario 1. The available bandwidth is 10 000 Mbit/s with 74 ms RTT the first 40 seconds and then 100 Mbit/s with 92 ms RTT. The loss rate is 0.06%.**

Figure 5-5 shows that Aspera's FASP utilization is extraordinary even after a handover to a network with lower bandwidth and higher delay. The drop that both graphs show between 55 and 60 seconds is a phenomenon quite common. This behavior occurs from time to time as shown in Figure 5-7. In Figure 5-8 the drops are very frequent and it looks as if the protocol is reacting to congestion or the limited bandwidth, but it does so much more aggressive when the bandwidth limit is 400 Mbit/s and 1000 Mbit/s. These drops might also occur because of the test bed, but neither the processor nor the memory was fully loaded at the time. The tests were performed with clean reboots between them. This behavior is not analyzed further in this thesis, but it might be interesting for Aspera or others to investigate it in the future.

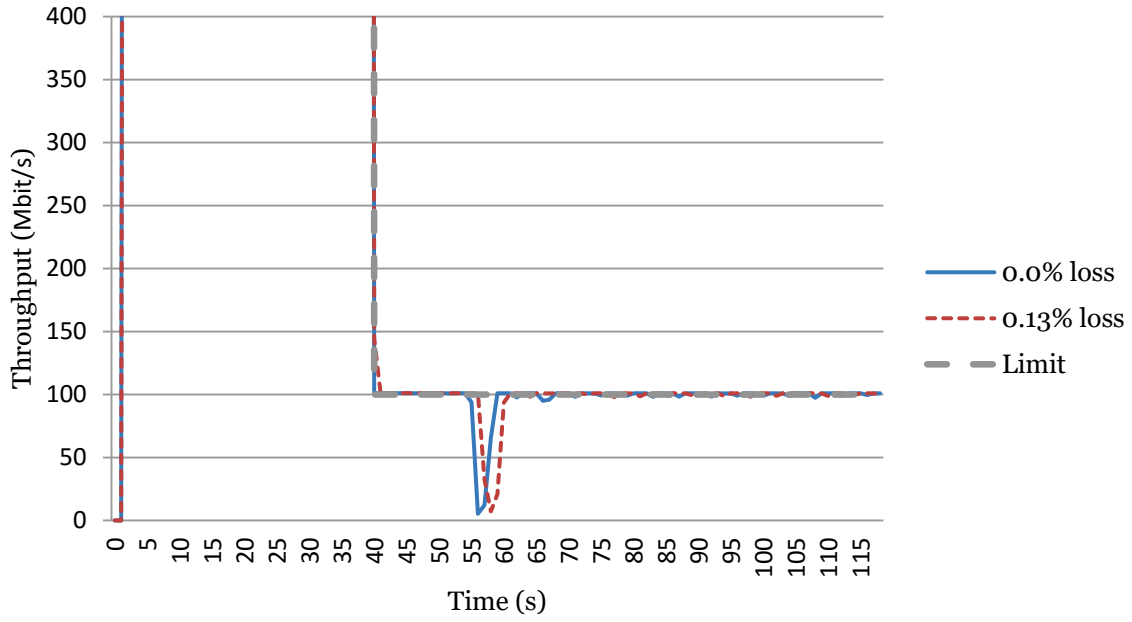

Figure 5-5: **Zoom of the throughput of Aspera's FASP with and without 0.13% packet loss in scenario 1.**

In scenario 2 the expected result is shown except for a delay when the handover occurs. About five seconds after the handover occurs Aspera's FASP reacts to it and increases its bandwidth to the link's maximum bandwidth. This delay is also apparent in scenario 3 (see Figure 5-8). This delayed reaction is somewhat difficult to explain and it had not been shown in any previous Aspera tests. It seems that either Aspera is unaware of it or this weakness is known, but has not been disclosed as it lacks a solution. Further analysis of this delay can be found in Section 5.3.

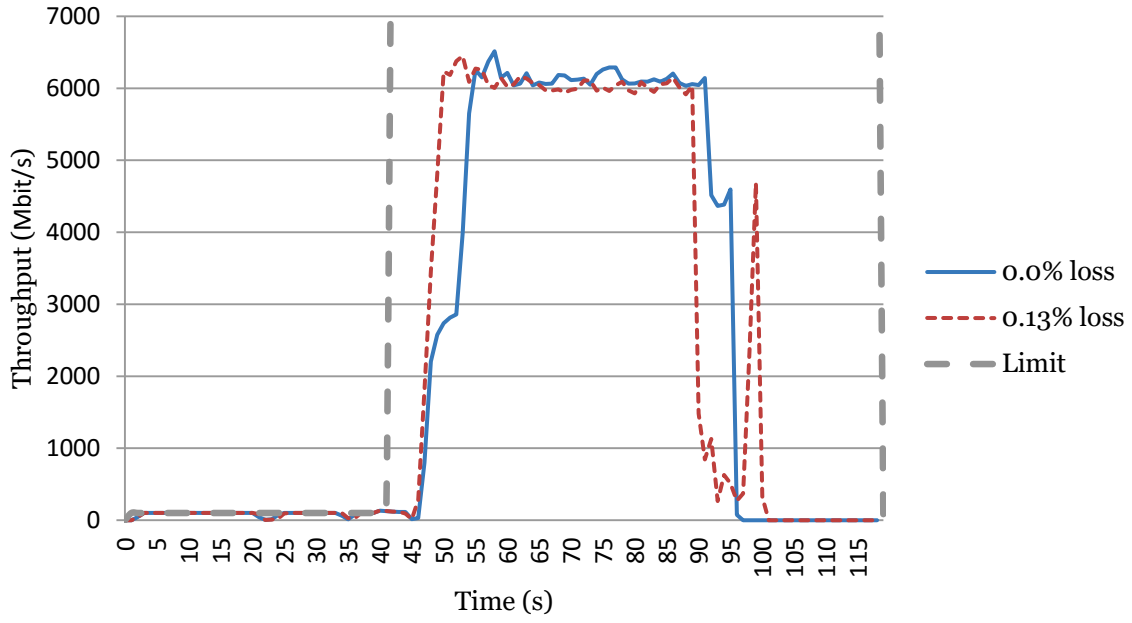

Figure 5-6: **Throughput of Aspera's FASP protocol with and without 0.13% packet loss in scenario 2. The available bandwidth is 100 Mbit/s with 92 ms RTT the first 40 seconds and then 10 000 Mbit/s with 74 ms RTT. The loss rate is 0.06%.**

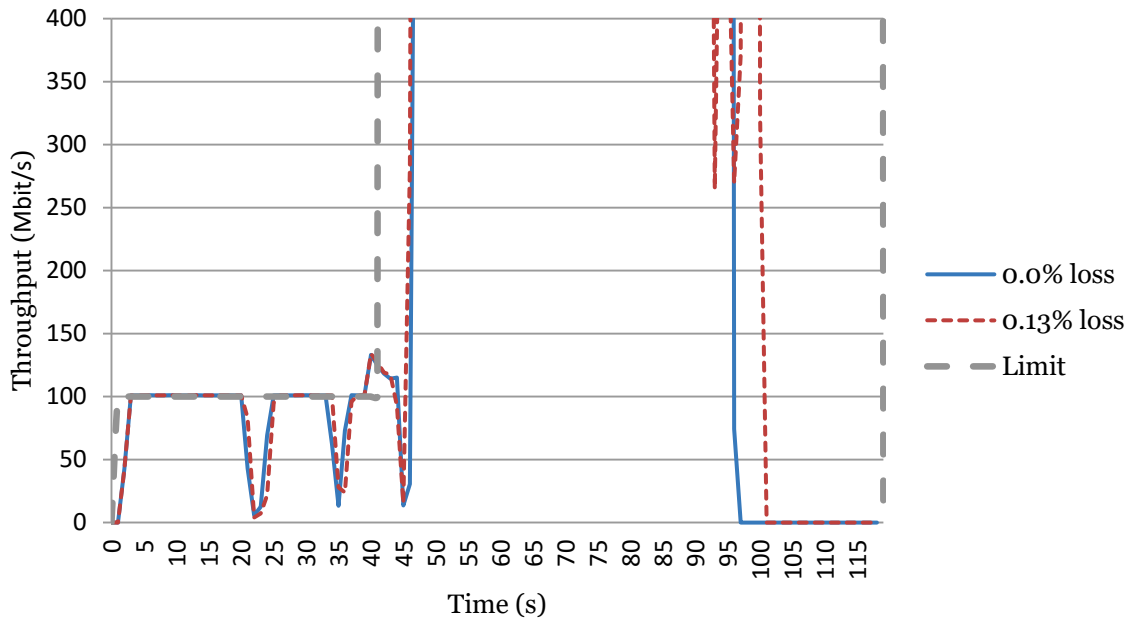

Figure 5-7: **Zoom of the throughput of Aspera's FASP protocol with and without 0.13% packet loss in scenario 2.**

Figure 5-8 shows Aspera's FASP in scenario 3. As before it handles the handovers really well, except for the five seconds delay in the second handover.

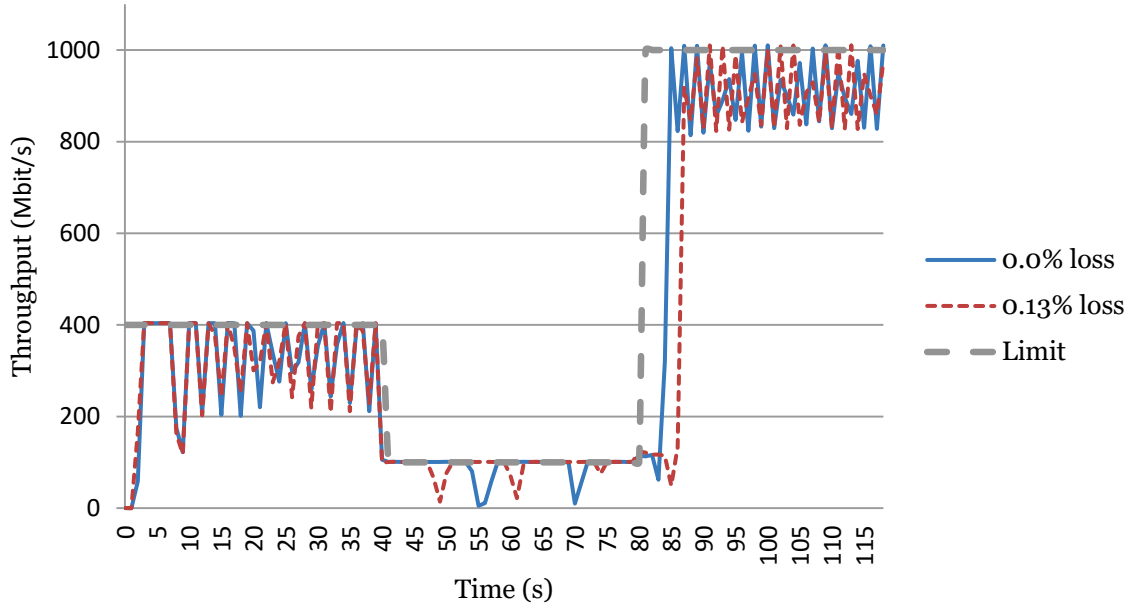

Figure 5-8: **Throughput of Aspera's FASP with and without 0.13% packet loss in scenario 3. The available bandwidth is 400 Mbit/s the first 40 seconds and then 100 Mbit/s for 40 seconds and then changed to 1 000 Mbit/s. The loss rate is 0.06% and the RTT is 92 ms.**

5.3 Potential explanation for delay when adapting to handover

Figure 5-7 showed an approximately 5 second delay before Aspera's FASP increased the transfer rate. This delay might be caused by the how Aspera's FASP probes for available bandwidth.

In order to examine in more detail how ascp is probing for bandwidth a test was made using Aspera's Connect Demo [31] to transfer a 1 GB file over a link with a specified (to the program) maximum transfer rate of 100 Mbit/s. Figure 5-9 shows a plot of the number of UDP datagrams sent from the server to the client between each of the UDP datagrams sent from the client to the server over the entire period of the file transfer. The initial delay of 2.046480 seconds before any UDP datagrams are sent is due to the TCP communication taking place between the client and server. The client computer is located at KTH Royal Institute of Technology in the Department of Communication Systems of the School of Information and Communication Technology. This computer is physically in Kista, Stockholm, Sweden The server machine has IP address 198.23.89.123 and is located in San Jose, California, USA. The path MTU is 1492 Bytes and the RTT is ~176 ms (with a starting estimate of the RTT being 174 ms).

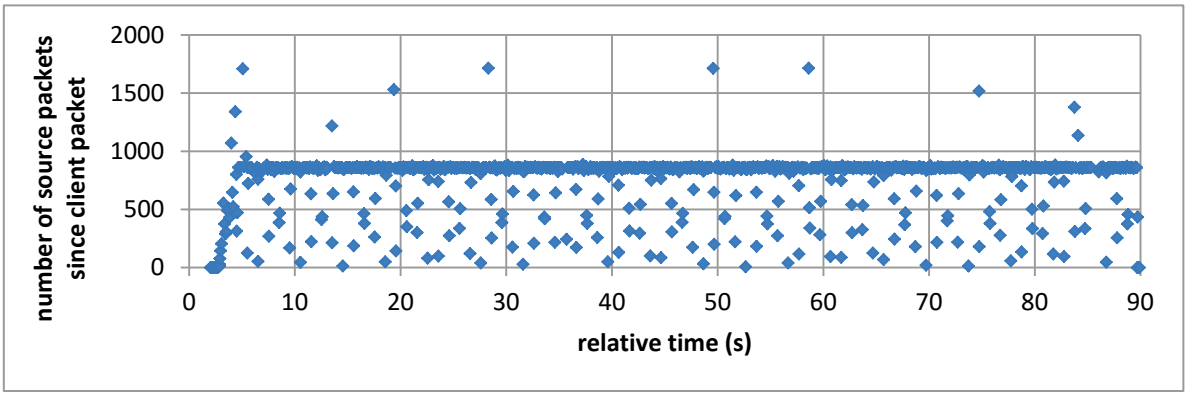

Figure 5-9: **Aspera Connect Demo transfer of 1Gbytes with a specified maximum transfer rate of 100 Mbit/s showing number of source packets since last client packet over the entire file transfer time.**

Figure 5-10 shows the first 25+ seconds of samples from the previous plot. An initial impression from this plot of burst lengths suggests that at least four different processes are being used to select the UDP datagram burst length. It seems that variations of the burst length are used to probe the link to determine the currently available bandwidth. The first process probes rarely but with long bursts, see the points above 1 000 packets since the last client packet. The second is a steady process that has slow start properties up to which stays around 800 packets since last client packet. The third and fourth are processes that probes in a sine wave manner, and therefore utilize a variety of shorter bursts. Notable is that since the two sine waves are 180 degrees out of phase, the sum of the two bursts are always the same, around 800 packets since the last client packet. The period of the sine waves are 8.3 seconds which may explain delays in adapting to the newly available bandwidth after handovers of up to ~16 seconds (assuming that the statistics from roughly two periods are used to estimate that there is a change in bandwidth).

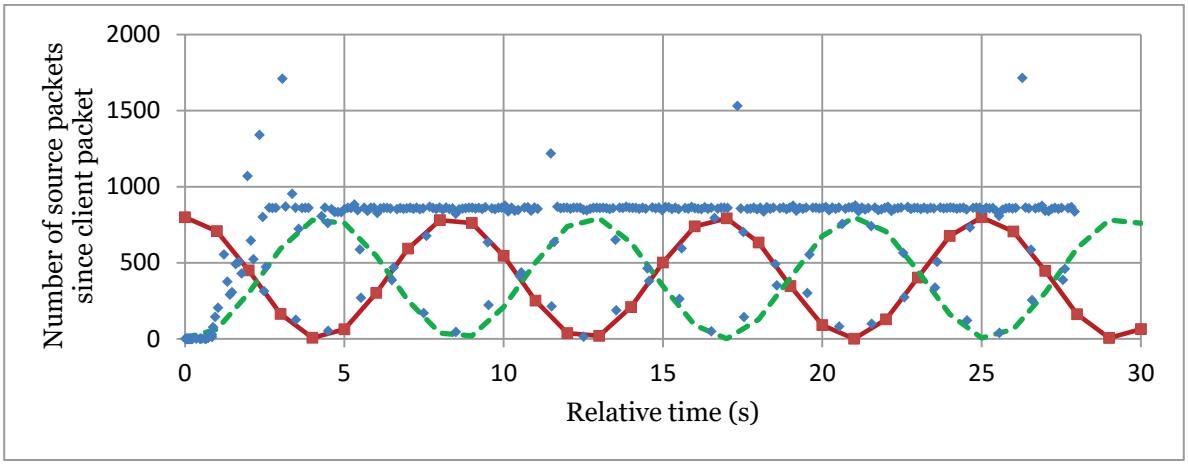

Figure 5-10: **Aspera Connect Demo transfer of 1Gbytes with a specified maximum transfer rate of 100 Mbit/s showing number of source packets since last client packet over the first tens of seconds of the file transfer time.**

5.4 Reliability and Validity Analysis

As was mentioned in Section 3.3, the test bed is refreshed between every test, meaning that all of the computers are rebooted, all caches cleared, and all the software restarted. For these reasons, the test data seems both reliable and valid. The analysis of the data is reliable and valid because of the simple nature of the analysis. All of the previous discussion about the results concern results that can easily be observed in the different figures.

5.5 Discussion

The results obtained and the analysis are at the same time uninteresting and interesting. The TCP results are bad as was expected. This known poor behavior of TCP was part of the reason for this thesis project. Aspera's FASP results are in general good, but the aggressiveness of the response to congestion and unexpected drops in utilization were not expected from a protocol that aims to fully utilize the available bandwidth. The delay after a handover before increasing utilization of the higher bandwidth is unexpected and something that potential users should be aware of. This behavior might not be what a user expects when frequently switching between two networks with very different amounts of available bandwidth. Despite this, Aspera's FASP utilizes the bandwidth better, hence creating better end user experience than TCP, independent of whether it is Reno TCP or CUBIC TCP.

In all of the baseline tests it is clear that packet loss is the source of the problem and not the link delay. This is the reason that TCP in general behaves poorly, because a packet loss is managed as a symptom of congestion rather than a simple randomly dropped packet. Handovers in either direction are not a problem since the packet losses rarely allow TCP to get close to the available bandwidth limit.

The maximum throughput of Aspera's FASP was around 6.3 Gbit/s, which is not that bad considering all of the extra processing and calculation that needs to be done. This limit is even more understandable in light of the pure UDP throughput results of 7.3 Gbit/s as reported by Victor Johansson [28]. He found that this limitation was due to the network interface not doing interrupt coalescing for UDP traffic, while it did so for TCP traffic.

From the results of these tests, it is obvious that Aspera's FASP throughput is nearly independent of packet loss. This feature should encourage others to aggressively exploit queuing delays for congestion detection rather than basing the decision on packet losses.

Because Aspera's FASP is a licensed product, it is not available for use by small independent programs that might benefit by using it. Therefore there is a great need for a similar protocol that is free and open source. Such a new protocol might be produced in cooperation with some of the larger companies[*] by defining such a protocol and releasing it as an open standard.

[*] Such as those companies that would greatly benefit from higher utilization of the available bandwidth either of their links or of their devices.

6 Conclusions and Future work

This chapter provides some general conclusions, as well as describing some of the limitations faced by the project. Additionally, some future work is suggested. The chapter concludes with some reflections on the project with respect to sustainability.

6.1 Conclusions

In regards to the first sub-goal, TCP has been compared to FASP in this thesis, while the other comparisons are done in the two parallel thesis projects (of which one has been completed - [28]). In Section 0 the problems with both TCP and FASP has been investigated and analyzed based upon tests in a test bed to examine scenarios that are thought to be representative of a high bandwidth radio access network. Issues and problems have been investigated in both TCP and FASP as the second goal stated. Hence the first two sub-goals have been met.

The third goal (to propose a new solution) has not been met, but a direction has been proposed as will be further discussed in the Section 6.3. I regret that no implementation of a new solution was tried, but there was only so much that could be done within the project's limited duration.

In this project one major obstacle was somewhat overwhelming, that was all of the difficulties that the three thesis project students faced when setting the parameters of the software and it was hard to achieve very high performance in the test bed. This primarily occurred due to the large number of parameters and the complex interactions between the hardware and software when trying to operate stably over 10 Gbit/s network interfaces. The difficulty of actually using 10 Gbit/s network interfaces and achieving stable high speed operation explains why almost every paper that was read during the literature study used simulation rather than tests using real hardware and software. One of the rewards of doing emulation is that the results are more likely to be correct and closer to what a user will actually experience. However, this is only worthwhile if there is sufficient time to optimize the operation of the system with real hardware.

When testing new protocols and implementing new techniques simulation might be the best choice, but when the focus is on predicting the performance that will experienced in the future by a user, emulation makes more sense. This is especially true because there might be many things that have been overlooked in the preparatory phase of the project, but their importance will become clear once testing begin using real hardware.

Future projects should be more efficient as the test bed is already set up. Moreover, several different types of file transfer softwares have already been tested, hence the process of deploying and testing yet another file transfer program and protocol should not be so time consuming.

6.2 Limitations

An obvious limitation of this thesis project is of course the test bed itself. The network hardware is typical of current 10 Gbit/s network interfaces and it is expected that these interfaces will improve over the next several years – off-loading more of the processing from the computer. Additionally, the test bed used desktop class computers and not one of today's smartphones, as should be expected since the thesis project's focus is on users of future 5G networks and there will be several generations of smartphones released before these networks are deployed. It should be obvious that today's standard smartphones are unsuitable for the very high bandwidths utilized in the test bed, hence it is infeasible to evaluate future battery power consumption due to the doing of extra processing that FASP and other new protocols require using existing smartphones.

One of the goals of the project was to analyze Aspera's FASP more in detail. Unfortunately, this was simply not possible because of the license and all the trade secrets about FASP's mechanisms. This is of course understandable because Aspera is a commercial company and the value of their technology is based on preserving these trade secrets. However, additional testing (such as that described in Section 5.3) would allow further analysis of this protocol.

Fortunately, this project did not experience any other major limitations and the results are clear and valid: *FASP should be used rather than using TCP when doing large file transfers via high bandwidth links – even in a radio access network context with realistic error rates and delays*.

6.3 Future work

Clearly an area of future investigation is to examine the drops and delays in the Aspera's FASP transfers, as mentioned in Section 5.2. This is clearly a problem of interest to Aspera, but might also be interesting for others who want to develop similar protocols.

It would be interesting to test Aspera's streaming protocol to see how it behaves in the same scenarios used for the tests in this thesis project. It would be especially interesting to see if it has the same issues FASP has. Aspera's streaming protocol was not researched in this project because only FASP was available at the time of this thesis project.

The availability of an open standard transport protocol for such high bandwidth radio access networks would enable normal end users to experience the benefits of these future 5G networks. Clearly the performance of Aspera's FASP indicates that substantial improvements are possible. It would be in Ericsson's interest to enable 5G users to achieve these high levels of link utilization.

To compare Aspera's FASP with Google QUIC would be interesting and to see when Aspera's FASP is better and when Google QUIC is better. It would also be interesting to see the difference between the protocols in terms of how they are modeled and how they are implemented. In the radio network environment there are system limitations, especially in the UE that one protocol might handle better than the other, whereas this would be interesting when choosing between the two protocols.

Since the extra processing occurs in the UE's processor, there is a need for experiments to quantify this processing versus the UE's processing capacity. Moreover, there is a need to evaluate the battery drain on the device's battery in comparison with the battery power required to transfer a given sized file with CUBIC TCP. However, as noted in the previous section above, it would be premature to do this with today's smartphones, but perhaps it could be done via power accurate simulations of future smartphones.

The drops in throughput shown in Section 5.2 are surely interesting for Aspera and the reason they have not found these issues in their own results is unclear. It might be cause of the lack of understanding or that this behavior was not observed previously. The reason behind these drops needs more analysis and might be interesting for anyone who develops a protocol that is similar to Aspera's FASP.

6.4 Reflections

The most obvious ethical aspect of this work was properly observing the licensing limitations when analyzing licensed software owned by another company. While one can observe the performance of this software operating under different conditions and in different scenarios, one is limited to black-box testing, as it would not be ethical or legal to reverse engineer the code directly from the object version of the code provided under license from Aspera. However, based upon the observations report in this thesis it is clearly ethical to propose the development of an open standard protocol

that could achieve comparable performance. Moreover, such an open standard would benefit all users.

Successfully developing an open standard protocol that would have similar utilization as Aspera's FASP would potentially have a huge impact on society since this would increase the user satisfaction of all 5G radio access network users. The thought of what our smartphones and other UEs could do with such high utilization of a 10 Gbit/s radio access network is overwhelming.

Alternatively, others could license Aspera's FASP technology and incorporate it into their products. While this would also have a beneficial impact on the user, experience from earlier Internet protocol standards would suggest that this path will not have as high nor as rapid an impact as an open standard would.

References

[1] Erik Dahlman, Gunnar Mildh, Parkvall Stefan, Janne Peisa, Joachim Sachs, and Yngve Selén, '5G radio access', *Ericsson Review*, vol. 2014, no. 6, p. 7, Jun. 2014.

[2] Mattias Ronquist, 'TCP Reaction to Rapid Changes of the Link Characteristics due to Handover in a Mobile Environment', Master's thesis, KTH Royal Institute of Technology, Teleinformatics, Stockholm, Sweden, 1999 [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-95138

[3] 'Network Coding Kodo library.' [Online]. Available: http://steinwurf.com/kodo/

[4] J.D. Day and H. Zimmermann, 'The OSI reference model', *Proceedings of the IEEE*, vol. 71, no. 12, pp. 1334–1340, Dec. 1983. DOI: 10.1109/PROC.1983.12775

[5] M. Allman, V. Paxson, and E. Blanton, 'TCP Congestion Control', *Internet Request for Comments*, vol. RFC 5681 (Draft Standard), Sep. 2009 [Online]. Available: http://www.rfc-editor.org/rfc/rfc5681.txt

[6] T.V. Lakshman and U. Madhow, 'The performance of TCP/IP for networks with high bandwidth-delay products and random loss', *IEEE/ACM Transactions on Networking*, vol. 5, no. 3, pp. 336–350, Jun. 1997. DOI: 10.1109/90.611099

[7] 'ITU towards "IMT for 2020 and beyond"', *ITU*. [Online]. Available: http://www.itu.int/en/ITU-R/study-groups/rsg5/rwp5d/imt-2020/Pages/default.aspx. [Accessed: 02-Jul-2015]

[8] 'ITU: Committed to connecting the world', *ITU*. [Online]. Available: http://www.itu.int/en/Pages/default.aspx. [Accessed: 02-Jul-2015]

[9] Afif Osseiran, Federico Boccardi, Volker Braun, Katsutoshi Kusume, Patrick Marsch, Michal Maternia, Olav Queseth, Malte Schellmann, Hans Schotten, Hidekazu Taoka, Hugo Tullberg, Mikko A. Uusitalo, Bogdan Timus, and Mikael Fallgren, 'Scenarios for 5G mobile and wireless communications: the vision of the METIS project', *IEEE Communications Magazine*, vol. 52, no. 5, pp. 26–35, May 2014. DOI: 10.1109/MCOM.2014.6815890

[10] J.S. Chase, A.J. Gallatin, and K.G. Yocum, 'End system optimizations for high-speed TCP', *IEEE Communications Magazine*, vol. 39, no. 4, pp. 68–74, Apr. 2001. DOI: 10.1109/35.917506

[11] V. Jacobson and R. T. Braden, 'TCP extensions for long-delay paths', *Internet Request for Comments*, vol. RFC 1072 (Historic), Oct. 1988 [Online]. Available: http://www.rfc-editor.org/rfc/rfc1072.txt

[12] V. Jacobson, R. Braden, and D. Borman, 'TCP Extensions for High Performance', *Internet Request for Comments*, vol. RFC 1323 (Proposed Standard), May 1992 [Online]. Available: http://www.rfc-editor.org/rfc/rfc1323.txt

[13] R. Braden, 'T/TCP – TCP Extensions for Transactions Functional Specification', *Internet Request for Comments*, vol. RFC 1644 (Historic), Jul. 1994 [Online]. Available: http://www.rfc-editor.org/rfc/rfc1644.txt

[14] J. Postel, 'Transmission Control Protocol.' [Online]. Available: http://tools.ietf.org/html/rfc793. [Accessed: 28-Jan-2015]

[15] J. C. Mogul and S. E. Deering, 'Path MTU discovery', *Internet Request for Comments*, vol. RFC 1191 (Draft Standard), Nov. 1990 [Online]. Available: http://www.rfc-editor.org/rfc/rfc1191.txt

[16] J. Postel, 'User Datagram Protocol', *Internet Request for Comments*, vol. RFC 768 (INTERNET STANDARD), Aug. 1980 [Online]. Available: http://www.rfc-editor.org/rfc/rfc768.txt

[17] E. Kohler, M. Handley, and S. Floyd, 'Datagram Congestion Control Protocol (DCCP)', RFC Editor, RFC4340, Mar. 2006 [Online]. Available: https://www.rfc-editor.org/info/rfc4340. [Accessed: 03-Jul-2015]

[18] K. Ramakrishnan, S. Floyd, and D. Black, 'The Addition of Explicit Congestion Notification (ECN) to IP', RFC Editor, RFC3168, Sep. 2001 [Online]. Available: https://www.rfc-editor.org/info/rfc3168. [Accessed: 03-Jul-2015]

[19] N. Spring, D. Wetherall, and D. Ely, 'Robust Explicit Congestion Notification (ECN) Signaling with Nonces', RFC Editor, RFC3540, Jun. 2003 [Online]. Available:https:/www.rfc-editor.org/info/rfc3540. [Accessed: 03-Jul-2015]

[20] Michelle Christine Munson and Serban Simu, 'Method and system for reliable data transfer', US8583977 B2, 02-Jul-2012 [Online]. Available: https://www.google.com/patents/US8583977. [Accessed: 10-Aug-2015]

[21] V. Jacobson, 'Congestion Avoidance and Control', in *Symposium Proceedings on Communications Architectures and Protocols*, New York, NY, USA, 1988, pp. 314–329 [Online]. DOI: 10.1145/52324.52356

[22] ‘FASPSTREAM API l Aspera Technology.’ [Online]. Available: http://asperasoft.com/technology/transport/faspstream-api/. [Accessed: 10-Aug-2015]
[23] ‘QUIC, a multiplexed stream transport over UDP - The Chromium Projects.’ [Online]. Available: https://www.chromium.org/quic. [Accessed: 16-Aug-2015]
[24] ‘quic - chromium/src/net - Git at Google.’ [Online]. Available: https://chromium.googlesource.com/chromium/src/net/+/master/quic/. [Accessed: 26-Aug-2015]
[25] ‘iPerf - The TCP, UDP and SCTP network bandwidth measurement tool.’ [Online]. Available: https://iperf.fr/. [Accessed: 16-Aug-2015]
[26] ‘Example syntax for Secure Copy (scp).’ [Online]. Available: http://www.hypexr.org/linux_scp_help.php. [Accessed: 16-Aug-2015]
[27] ‘bwm-ng(1) - Linux man page.’ [Online]. Available: http://linux.die.net/man/1/bwm-ng. [Accessed: 16-Aug-2015]
[28] Victor Johansson, ‘Enhancing user satisfaction in 5G networks using Network Coding’, Master’s thesis, KTH Royal Institute of Technology, School of Information and Communication Technology (ICT), Radio Systems Laboratory (RS Lab), Stockholm, Sweden, 2015 [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-171210
[29] Intel Corporation, ‘Big Data Technologies for Ultra-High-Speed Data Transfer in Life Sciences’, Intel Corporation, White paper 0413/KP/SS [Online]. Available: http://asperasoft.com/fileadmin/media/Asperasoft.com/Resources/White_Papers/Big_Data_Life_Sciences_AsperaWP.pdf. [Accessed: 10-Aug-2015]
[30] ‘Aspera Aspera Ascp User Guide 3.5.2.’ [Online]. Available: http://download.asperasoft.com/download/docs/ascp/3.5.2/html/index.html#dita/ascp_usage.html. [Accessed: 16-Aug-2015]
[31] ‘Aspera - High-speed file transfer software.’ [Online]. Available: http://demo.asperasoft.com/. [Accessed: 26-Aug-2015]

TRITA-ICT-EX-2015:199