Revisiting the IETF Multipath Extensions on Transport Layer

D I S S E R T A T I O N

to obtain the academic grade
doctor rerum naturalium
(dr. rer. nat.)
in Computer Science

Submitted to the
Faculty of Economics
Institute for Computer Science and Business Information Systems
University of Duisburg-Essen

by
Martin Becke
born on 08.08.1977 in Ankum, Germany

President of the University of Duisburg-Essen:
Prof. Dr. Ulrich Radtke

Dean of the Faculty of Economics:
Prof. Dr. Volker Clausen

Reviewers:

1. Prof. Dr.-Ing. Erwin P. Rathgeb
2. Prof. Dr. Klaus Echtle

Submitted on: September 11, 2014
Date of Disputation: November 12, 2014

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständig ohne fremde Hilfe verfaßt und nur die angegebene Literatur und Hilfsmittel verwendet zu haben.
Weiter erkläre ich, dass diesem Promotionsverfahren keine Promotionsversuche in diesem Fach oder in einem anderen Fach vorausgegangen sind und dass die eingereichte Arbeit oder wesentliche Teile derselben in keinem anderen Verfahren zur Erlangung eines akademischen Grades vorgelegt worden sind.

Martin Becke
September 11, 2014

Abstract

Load sharing on the transport layer of the OSI reference model is an important topic in the IETF standardization. This approach is also supported by the industry to optimize the use of the resources in a network like the Internet. After many trials, two basic sets of mechanisms and functionalities on the transport layer have been proposed by the IETF to achieve load sharing. These basic sets extend the protocol mechanisms that were originally designed for the use in singlepath dominated networks and represent only a first step to introduce a real end-to-end multipath transfer on the Internet. These first basic sets must be investigated and improved for the next steps.

The Transmission Control Protocol (TCP) and the Stream Control Transmission Protocol (SCTP) provide the basis for the two IETF end-to-end multipath extensions. Both singlepath transport protocols have a different historical background but similar goals. These can be characterized by a reliable, connection-oriented and ordered data transport. However, initial experiments with the IETF multipath extensions in real networks show unexpected and in some cases clearly inadequate results. It is becoming rather apparent that the singlepath transport protocol specifications with their singlepath goals have a significant impact on the effectiveness of the load sharing mechanism and, furthermore, that the severity of the influence depends on the topology. The new mechanisms for multipath transfer include, in particular, an extended "path management" and "scheduling" task. The mechanisms addressing the path management organize the new, alternative paths and the scheduling mechanisms support their effective use. For both protocol extensions of TCP and SCTP, an interaction can be identified between the new load sharing mechanisms and the existing specifications for singlepath transfer. This thesis systematically identifies the impact factors of the singlepath specifications on the new load sharing mechanisms and demonstrates their effects. In addition to the focus on the optimal use, the fair distribution of resources across all connections must be taken into account in the IETF standardization process. This so-called "fairness" discussion is mandatory for a transport protocol in the IETF context and has a direct impact on the overall system performance. Furthermore, this thesis discusses the currently implemented load sharing extensions and analyzes their weaknesses. Moreover, in this work new design approaches are developed to decrease the impact.

Keywords:
Stream Control Transmission Protocol (SCTP), Multipath Transport, Multipath Transmission Control Protocol (MPTCP), Evaluation, Optimisation

Acknowledgements

This thesis is the result of my work as research associate in the Computer Networking Technology Group of the Institute for Experimental Mathematics at the University of Duisburg-Essen.

First and foremost, I would like to thank my primary advisor, Prof. Dr.-Ing Erwin P. Rathgeb, for his support and supervision of this thesis and the invaluable comments related to my research work. Furthermore, I also thank my secondary advisor Prof. Dr. Klaus Echtle for his thesis review.

I would like to express my special thanks to my former colleague Dr. Thomas Dreibholz for his great cooperation in the project, as well as Prof. Dr. Michael Tüxen from the Münster University of Applied Sciences for his insperation and support, especially in the context of our IETF standardization work. Furthermore, I would like to express my special thanks to my colleague Hakim Adhari for his great cooperation with respect to discussions on fairness and path management and debugging the testbed environments used for this work. I would like also to thank my former colleague Jobin Pulinthanath for his help with building up the testbed environment and negotiating the contract details with the ADSL Internet service provider. For his help with the acquisition of the testbed hardware, I would also like to thank our computer systems technician Nihad Cosic and my student assistant Sebastian Wallat. Clearly, I would furthermore like to express my special thanks to my colleagues Irfan Simsek, Adnan Aziz and Sebastian Werner for their input during the discussions about fairness and security. Also, I would like to thank Dr. Irene Rüngeler and Dr. Robin Seggelmann for the discussions and continuous support. In addition, I would like to thank the Deutsche Forschungsgemeinschaft (DFG) and the Bundesministerium für Bildung und Forschung (BMBF) for supporting parts of this project. Finally, I would like to thank my girlfriend Shima Shayanfar and my father Paul Becke for the years of encouragement and support.

Contents

Glossary

ACK	Acknowledgment
ADSL	Asymmetric Digital Subscriber Line
AIMD	Additive Increase Multiplicative Decrease
ARWND	Advertised Receiver Window
BDP	Bandwidth Delay Product
CC	Congestion Control
CCC	Coupled Congestion Control
CMT	Concurrent Multipath Transfer
CMT – SCTP	Concurrent Multipath Transfer extension for SCTP
ConSN	Confluent Sequence Numbers
CumAck	Cumulative Acknowledgement
CWND	Congestion Control Window
DCCP	Datagram Congestion Control Protocol
DFG	Deutsche Forschungsgemeinschaft
DFN	Deutsches Forschungsnetz
DiffServ	Differentiated Services
DSL	Digital Subscriber Line
DSS	Data Sequence Signal
DupACK	Duplicate Acknowledgment
ERiCA	Encapsulated Responsibility-Centric Architecture Model
FastRTX	Fast Retransmission
FIFO	First In First Out
FRA	Fair Resource Allocation
GUI	Graphical User Interface
ID	Internet Draft
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
INP	Internet Protocol
inp_faddr	Internet Protocol Foreign Address
inp_fport	Internet Protocol Foreign Port
inp_laddr	Internet Protocol Local Address
inp_lport	Internet Protocol Local Port
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISDN	Integrated Services Digital Network
LIA	Linked Increases Algorithm

MPTCP	Multipath TCP
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
N/PAT	Network/ Port Address Translation
NCA	Normalize, Cluster and Aggregate
NED	Network Description
NetPerfMeter	Network Performance Meter
NR – SACK	Non-Renegable Selective Acknowledgement
OLIA	Opportunistic Linked Increases Algorithm
OMNeT++	Objective Modular Network Testbed in C++
Opp – Rtx	Opportunistic Retransmission
OS	Operating System
PCB	Protocol Control Block
PDU	Protocol Data Unit
PF	Protocol Family
PPP	Point-to-Point Protocol
PPPoE	Point-to-Point Protocol over Ethernet
QoS	Quality of Service
RED	Random Early Drop
RFC	Request for Comments
RP	Resource Pooling
RP – MPv2	Resource Pooling Multipath Version 2
RSerPool	Reliable Server Pooling
RTO	Retransmission Timeout
RTT	Round Trip Time
RTTVAR	Round Trip Time Variance
RTX	Retransmission
SACK	Selective Acknowledgment
SCTP	Stream Control Transmission Protocol
SIGTRAN	Signaling Transport Working Group (IETF)
SimProcTC	Simulation Processing Tool-Chain (software package)
so	Socket
so_pcb	Socket Protocol Control Block
so_type	Socket Type
SQN	Sequence Number
sRTT	Smoothed Round Trip Time
SSN	Stream Sequence Number
ssthresh	Slow-Start Threshold
TCB	Transmission Control Block
TCP	Transmission Control Protocol
TLV	Type Length Value
TSN	Transmission Sequence Number
TSVWG	Transport Services Working Group (IETF)
UDP	User Datagram Protocol

Chapter 1

Introduction

The Internet is more than 30 years old. During this time period, the services in the Internet have changed a lot. Starting with simple file transfer, the use cases evolved to complex services, like for example video instant messaging (e.g. Skype or Google Hangout), cloud services (e.g. Dropbox or Amazon Software-as-a-Service) or social networks (e.g. Facebook or Twitter).

It might be surprising that although the requirements changed, the core architecture and the mechanisms to transport data through the Internet stayed quite similar since the beginning. In the end, all services deployed in the Internet are based on two Internet protocols (IPv4 [Pos81a] and IPv6 [DH98b]) and a handful of transport protocols. Even if the use cases, the connected hardware and the provided access media have changed and also their number has increased over time, the mechanisms deployed in the core Internet does not always reflect this. This can be observed for different use cases like for example when a mobile telephone has to decide whether it should connect via 3G or via WiFi – although both is possible. The original transport protocols just focus on one single "path" between two endpoints and, therefore, only on one single access medium or destination identifier. Thus, even if the mobile phone is connected via 3G and WiFi, the transport protocol is not able to use both paths. Therefore, the currently deployed mechanisms are not optimized to use all provided resources of today's Internet.

The main reason for this limitation is the risk that the basic service of the Internet may be compromised by this new kind of network usage. Even if a simple coupling of resources is addressed, it changes the current definition from a singlepath end-to-end connection to a multipath end-to-end connection. This change has a considerable impact on the deployed mechanisms. Thus, besides the potential benefits also – perhaps unknown – drawbacks exist, which have the potential to break the Internet and all services on top of it in the worst case. It is fair to say that even small changes can cause big issues.

Particularly critical is the "indirect" impact of changes as can be observed in the ongoing bufferbloat discussion [Get11b]. More buffer in routers enables the support of high speed links in a packet switched network. At first sight, this promises many advantages. However, more buffer can lead to increased delay, increased delay increases the update cycle time of the mechanism to calculate the send rate and this, furthermore, causes an inaccurate behavior of the send rate calculation. The mechanism to control the send rate does not consider the delay as queueing delay and that gives the basis for wrong assumptions and destructive behavior. The root cause for this inaccuracy is the fact that the original mechanism for send

rate calculation was not designed for these conditions with huge buffers. Thus, although at first sight the increase of buffer to improve the performance in the network seems to be a good idea, it can cause an erratic behavior at the endpoint that has again a negative impact on the network performance. Therefore, changes to basic Internet protocols are handled very conservatively, because the overall impact is not immediately clear.

Anyway, in this thesis, the use of more than one path at a time is discussed as a new feature with a real chance of wide deployment. Of course, the idea to deploy load sharing in the Internet is not new and several approaches exist as workarounds on top of the original architecture (e.g. shim layers) or as approaches for partial network segments, but it is the first time that real end-to-end solutions as part of the core technology have a real chance to become standardized for their deployment in the Internet.

Two load sharing extensions for two different transport protocols are in the standardization discussion of the Internet Engineering Task Force (IETF). Both extend the original architecture of the core transport protocol and support the usage of every connected interface for the same end-to-end transport connection. This thesis discusses the new tasks of these extensions of the Transmission Control Protocol (TCP) [Pos81b] and the Stream Control Transmission Protocol (SCTP) [Ste07]. Until now, it is still open whether the current goals of load sharing, primarily the improvement of throughput, can even be achieved. During the analysis, this thesis identifies the need to reconsider the current view on specific load sharing requirements, as for example the fair sharing of link capacities on flow level and proposes new approaches to optimize the current design.

Although the standardization process of the load sharing extensions has made some significant progress during the past years, there was no analysis done whether both extensions even have the potential to achieve the goals of the load sharing features under comparable conditions as they are discussed and proposed in the IETF standardization process.

1.1 Motivation

There is a strong need to revisit the approaches currently discussed in the IETF standardization. Although first basic sets to support load sharing on top of existing singlepath protocols have been provided, it is not known which usability these approaches have for TCP and SCTP. Furthermore, it is still open whether the current efforts even provide the basis to achieve the minimum goal of the standardization process [ABD+13, FRHB13] to achieve an improved performance. Until now, the load sharing extensions for the transport protocols are designed as very straightforward extensions of singlepath end-to-end protocols and it is not clear whether this focus on the singlepath protocols has drawbacks for the multipath service. Thus, the load sharing extensions come with legacy constraints and requirements whose influence is still unknown. No real optimization or adaptation has been done for existing singlepath mechanisms to provide the load sharing feature in an optimal way.

First real world experiments do not show satisfactory results [BAR+13] and provide motivation for a deeper analysis of this behavior. Even worse, some experiments show that not even a performance comparable to a singlepath flow on the "strongest" path can be achieved by both load sharing extensions, even if this is a minimal goal of the IETF standardization process [DBRT10]. Furthermore, the implementations do not always work in a predictable and expected way, particularly with regard to the competition with other end-to-end flows. All these issues will prevent a wide deployment of the load sharing extensions of TCP and

SCTP in the Internet and, therefore, require a systematic analysis before the next step of a wide rollout can proceed.

It becomes clear that the current designs of the load sharing extensions of TCP and SCTP have deficits. Potential reasons for these deficits can be identified in the design of the new tasks to support the load sharing feature on an end-to-end level. But it is not clear whether the poor performance can be addressed by the organization of the additional link resources, which will be denoted in the following as path management, or the new scheduler task, which can cause possible performance issues, especially if the mechanism adapts ineffectively to the network conditions.

However, besides these two new tasks there is another task with potential high impact on the performance. This task is the fair sharing of the link capacity and is an important requirement in the context of the IETF. A new approach is discussed and denoted as Resource Pooling idea [WHB08] to satisfy the requirements of the IETF. Multipath protocols implementing this idea try to perform no more aggressively than a singlepath transport protocol on the same link. But the impact of this implementation on a multipath transport protocol has not been investigated systematically. Furthermore, it is not even clear whether the IETF fairness requirement is achieved in every case.

First tests led to the assumptions that mechanisms based on this approach are not sufficiently well designed to provide the expected results. Thus, either important requirements are missed, the new functionalities are not well integrated, or limited by the basic singlepath protocol architecture. Therefore, the current design of the load sharing extensions must be analyzed. The results of this work will help to prepare the next steps in the standardization and development process. There is a strong need to restart the design process based on the results observed to improve the benefit of the load sharing feature or to even achieve the minimum IETF load sharing goals.

1.2 Goals of this thesis

The existing approaches of load sharing for TCP and SCTP show good results in simulation and in local domain testbeds [DBRT10, RPB+10], but first experiments in the Internet demonstrate serious weaknesses in the performance [BAR+13]. Until now, it is not clear whether an end-to-end load sharing solution is even able to work under the conditions of the Internet. There are known issues caused for example by middleboxes [ADB+11], but they do not explain that the performance of a connection with deployed load sharing does not even achieve the performance of a simple singlepath connection deployed via the strongest alternative path alone [DBRT10].

Therefore, the first main goal of this thesis is to analyze the current behavior and to identify the reasons for the weak performance under the challenging conditions [FODA14] of the Internet. Different test models[1] and platforms are necessary for the analysis of the two load sharing extensions to provide a complete picture. Of course, the Internet is the target platform, but it does not provide the optimal conditions to run controllable and repeatable performance experiments. Repeatable and controllable experiments should be done in real controllable testbeds or in a simulation environment. However, the simulation environment and the controllable local testbed are only as good as the parameterization that represents the Internet. Therefore, an additional goal is to understand whether the current test models for

[1] like the disjoint and the cross path setup test models as discussed in Subsection 5.1.2.

the load sharing extensions of TCP and SCTP fit to the requirements which are given by the Internet with its asymmetric, heterogeneous network characteristics. The research community has not considered the difference between path and network adaptation until now, it is a goal of this thesis to discuss this in more detail.

The second goal addresses the fair behavior in the Internet. Here, new congestion control mechanisms have been introduced based on the Resource Pooling idea [WHB08]. The implementation of the Resource Pooling idea comes with new IETF multipath fairness goals. These IETF multipath fairness goals describe minimum goals of the Resource Pooling idea, but allow different interpretations. Until now it is not clear what impact these IETF multipath fairness goals have on the usability of the implemented mechanisms. This thesis will demonstrate that these IETF multipath fairness goals detach from the well-known single link fairness view to a network fairness view and will argue that the fairness view is shifted to a new calculation basis without always providing benefits. Furthermore, it has not been analyzed whether the congestion control mechanisms implementing the Resource Pooling idea are even able to achieve the goal of a fair allocation of the resources in an asymmetric and heterogeneous topology like the Internet.

The third major goal of this thesis is to analyze the load sharing solutions and to propose optimizations to achieve at least the minimum performance goals. Improvements should be designed and implemented under the constraints of the IETF and without changing the characteristics of the core singlepath protocols.

1.3 Organization of this work

Chapter 2 provides a short introduction to the architecture of the Internet and to the transport service provided in this architecture. Chapter 3 introduces the load sharing terminology and the corresponding goals and gives an overview of existing load sharing approaches. Furthermore, it explains the motivation according to Voice and Kelly [KV05] to apply load sharing on the transport layer and discusses the resulting requirements for transport protocols in general. The major focus in this chapter is on the design of the load sharing extensions and the impact of resource coupling on the transport protocol mechanisms. Here, in particular the buffer management, congestion control, receive and send queue behavior of SCTP and TCP are in the focus. The simulation models developed for these approaches are described in Chapter 4, as well as the real world testbed and further toolchain components. Chapter 5 analyzes the two different load sharing alternatives and demonstrates the impact of different design decisions made for the singlepath protocols. Furthermore, this chapter is concerned with the deployment of the load sharing protocols in the Internet and upcoming routing questions. Chapter 6 discusses the impact of the currently deployed congestion control mechanisms and the fairness aspect. In the following Chapter 7, the achievement of the load sharing goals is discussed with focus on the ability of the multipath scheduler to adapt to different network conditions. Chapter 8 relates the results of the technical discussion to the IETF discussion and goes beyond the OSI reference model to give input for a Future Internet discussion, too. The conclusion and an outlook on future work are given in Chapter 9.

Chapter 2

Basics

The Internet, as it is known today, is based on a number of basic ideas about networking. The key concept of exchanging data in a network is described by [Koz05] as:

> A network is a set of hardware devices connected together, either physically or logically. This allows them to exchange information.

However, a network like the Internet has an uncounted number of hardware components, which are connected together to provide a multitude of services. These services are used by billions of users day by day.

A common approach in information technology is to separate the responsibilities in a layered structure to understand complex systems. Different models have been designed to achieve this goal, the most well-known model to describe the functionalities is the ISO/OSI reference model, as also discussed in [Tan96] and illustrated in Figure 2.1.

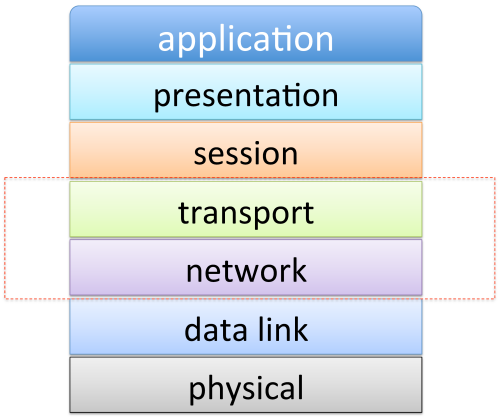


Figure 2.1: OSI reference model

This model structures networks like the Internet by using a layered model of components called protocols. Each layer is assigned to specific tasks. Of course, this model covers much more than the Internet protocols, but only these protocols are in the focus of this thesis. The layers from bottom up are defined as follows:

- The physical layer deals with providing the mechanical and physical specifications to ensure the transmission across a physical medium. It is also concerned with transforming digital information to signals which can be transferred via the chosen medium, like for example a copper cable.

- The major task of the (data) link layer is to make the physical layer appear error-free to the upper layer (network layer). The link layer protocol data unit (PDU) is denoted as frame. In this context the Institute of Electrical and Electronics Engineers (IEEE) provides the most important sets of standards.

- In the Internet of today, the network layer is mainly responsible for the addressing and forwarding of data via different networks. This layer is partly in the focus of this thesis, especially the task of forwarding information. The protocol data unit is denoted as packet.

- The transport layer provides different services in the Internet, which extend the simple forwarding functionality of the network layer by additional end-to-end services, like a connection-oriented delivery of in-order user data. The load sharing transport via multiple end-to-end "paths" through a network is in this thesis provided by this layer, although the load sharing service can be provided by alternative layers as discussed in more detail in Subsection 3.1.2. The transport layer PDU is called segment.

- The session layer supports the establishment of a session between end-user applications.

- The presentation layer is also sometimes called the syntax layer and describes the functionality related to formatting of data. Also encryption and compression is assigned to this layer.

- The application layer represents protocols which are concerned with supporting a specific service, e.g. the File Transfer Protocol [PR85].

For a complete discussion of load sharing extensions in the Internet nearly the complete OSI reference model is necessary, even if this thesis focuses clearly on the network and transport layers. Anyway, if researchers or protocol developers start to implement protocols for the Internet, they often denote it as *protocol stack* [BFH03].

Before a discussion of the load sharing approaches on the transport layer is done in Chapter 3, this chapter gives some basic background information about the transport protocols on the transport layer.

2.1 Transport protocols in the context of the IETF

Different central organization units help to specify common standards for the Internet. The Internet Engineering Task Force (IETF)[1] is one of the most important organization units of the Internet and provides specifications for different protocols. This applies particularly also to the protocols on the transport layer. Protocols defined by the IETF have a huge impact on the Internet because many vendors provide solutions which support the standards defined by the IETF or at least derive their functionality very closely from the IETF specification.

[1] IETF http://www.ietf.org/

As an example, the IETF adopts mainly two variants of protocols on the network layer. These variants are the Internet protocol in version 4 (IPv4) [Pos80a] and version 6 (IPv6) [DH98b]. These protocols describe the way of addressing and forwarding packets in the Internet and are without any alternative. Thus, if someone wants to be part of the Internet he has to use IP.

The most well-known and widely deployed transport protocol standardized by the IETF is denoted very straightforward as Transport Control Protocol (TCP) [Pos81b]. Together with the User Datagram Protocol (UDP) [Pos80b], TCP is the most used transport protocol and was developed in the early years of the Internet. However, there also exist newer alternative approaches of transport protocols in the IETF, like the Stream Control Transmission Protocol (SCTP) [Ste07] or the Datagram Congestion Control Protocol (DCCP) [KHF06]. Even if they are not as widely deployed as TCP or UDP, they give a good idea of what a transport protocol in the Internet of today can provide. The design process of these protocols never really stopped after the first IETF document was adopted. An adopted standard is also denoted as request for comments (RFC) and can be set as obsoleted or can be extended by other ideas. Therefore, the development of TCP, UDP, SCTP and DCCP was always a continuous process in the IETF. Many examples for this can be given, like the replacement of the RFC2960 [SXM+00], with the RFC4960 [Ste07] to describe the core SCTP. But also new extensions can be mentioned, e.g. the introduction of congestion controls to protect the network and to achieve a fair behavior with RFC2914 [Flo00], RFC2581 [APS99] or RFC5681 [APB09].

Although the standardization process is not in the focus of this thesis, an important specialty of the IETF should be highlighted in this context. Before an Internet standard can be accepted, a reference implementation must exist. RFC2026 [Bra96] is quite clear in this point:

> A candidate specification must be implemented and tested for correct operation and interoperability by multiple independent parties and utilized in increasingly demanding environments, before it can be adopted as an Internet Standard.

The main work of this thesis addresses two different IETF protocol extensions which are currently in the IETF discussion. Therefore, there exist two reference implementations, too. The basic idea of both approaches is to deploy a load sharing functionality on top of existing transport protocols. The currently discussed approaches are the extensions for TCP, called Multipath TCP (MPTCP) and for SCTP, called Concurrent Multipath Transfer extension for SCTP (CMT-SCTP). The most important reference implementation for MPTCP can be installed as a patch for the current Linux kernel[2]. For CMT-SCTP, it is available in the current FreeBSD kernel[3] without any further efforts. It should be mentioned that at the beginning of this thesis, no common platform existed to compare both approaches in the same environment and as far as the author knows there still exists no other platform besides the solution developed as part of this thesis project[4].

[2]MPTCP patch https://github.com/multipath-tcp/mptcp
[3]FreeBSD http://www.freebsd.org/
[4]During the time alternative approaches come up, like the MPTCP implementation for FreeBSD http://caia.swin.edu.au/urp/newtcp/mptcp/tools.html, but they are in an initial state (or not open).

2.2 Introduction to TCP and SCTP

As mentioned before, this thesis focuses on the load sharing extensions of TCP [Pos81b] and SCTP [Ste07]. Because both extensions are based on the functionalities of the corresponding singlepath protocols, some details of a singlepath transport protocol service have to be discussed in the following. For a discussion of TCP and SCTP it must be mentioned first that both protocols were developed at different times and for different use cases.

As [Koz05] describes, the first version of TCP was written in 1973 as a core protocol of the so called ARPAnet, with the main goal to provide basic mechanisms for connection establishment, management and reliable data transport between two software processes. However, in the following years, the Internet replaced the original ARPAnet and new ideas came up, e.g., to bring the telephony service to the Internet. In 1988, the Signaling Transport Working Group (SIGTRAN[5]) was formed to discuss alternatives to provide IP-based networks for telephony. One major quality requirement was the support of a reliable data transfer with the possibility to support network or path redundancy. Thus, a major SIGTRAN use case was to provide the support of more than one network for system stability. This requirement led to the support of multi-homing at the endpoints. An outcome of the SIGTRAN working group was SCTP. Even if TCP and SCTP share a lot of services (as different surveys show, like e.g. [Dre12a, Seg12]), they are different in the use case they were designed for. The following subsections discuss basic mechanisms of both protocols.

2.2.1 Services of transport protocols in the Internet

The major task of a transport protocol is to provide the transfer of user data from one endpoint of the network to another by using the service of the network layer. This is also called end-to-end connection. Thus, while in the Internet the IP address of the Internet protocol (IP) identifies the endpoint, the service itself is identified with a port number on the transport layer.

So, every protocol instance has at least a combination of source/destination address and the related source/destination port. This information four tuple in combination with the protocol is used describe a connection as a so-called unique 5-tuple. This 5-tuple of information is also used by the de-facto standard interface for the network in most operating systems. This interface API is known as Berkeley API [Koz05] and is discussed in more detail in Subsection 2.3.3.

Sending data with TCP or SCTP requires signaling of control information. In networks like the Internet this information is added to each user data packet. This process is also called encapsulation and is located on the sender side. The analog decapsulation process is located on the receiver side. The control information is encoded for TCP and SCTP in a different way and includes the signaling for the services provided by both transport protocols. TCP extends the user data by the header information illustrated in Figure 2.2. This information is 32 bit aligned and defined by [Pos81b]:

- The source port is part of the 5-tuple to address the endpoint service on the sender side.
- The destination port is part of the information needed to address the endpoint service at the receiver side.

[5]SIGTRAN charter http://datatracker.ietf.org/wg/sigtran/charter/

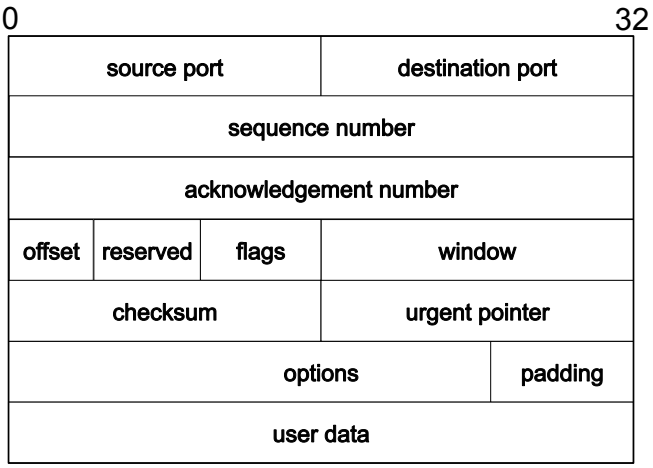


Figure 2.2: TCP header

- The sequence number is assigned in-order to the user data to support a re-ordering process and reliable transfer. This re-ordering process will become also very important for the load sharing extension.
- The acknowledgment number is used to support the reliable transfer and is reported by the receiver.
- The data offset field is used to expand the TCP header. The headers can have different sizes; in minimum 20 bytes and in maximum 60 bytes, if all option space is used.
- The reserved block is not yet assigned to an official function. In some implementations a misuse of these elements can be observed for individual features.
- The flags are used to signal different functions. If the flags are important for the thesis they will be discussed in detail in the relevant context.
- The window is part of the flow control. The flow control is a mechanism used to protect the receiver and will be discussed in more detail in Subsection 3.2.3.3.
- The checksum is used for error detection.
- The urgent pointer can be used to prioritize specific data.
- The variable option field is used to extend TCP. The possibilities to signal new services are limited (as will be discussed in Subsection 3.2.3). This will also become very important for the TCP load sharing extension that needs signaling of control information, too.

The encapsulation process for SCTP is technically the same. The only difference is that SCTP encapsulates messages instead of a byte stream. In comparison to the rather fixed structure of TCP, SCTP allows more flexibility due to an extendable Type-Length-Value (TLV) design and the use of a fixed common SCTP header structure. The SCTP common header [Ste07], also illustrated in Figure 2.3, includes only minimum information as described in the following:

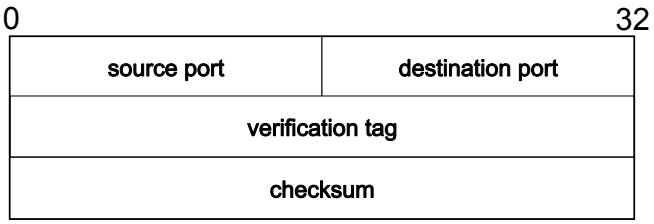


Figure 2.3: SCTP Common Header

- As for TCP, the source port is used to address the connection on the sender side and is used to identify the SCTP connection, which is denoted as association in SCTP terminology.
- Also the destination port is used to address the receiver side as known from TCP.
- The receiver can use the verification tag to validate the sender.
- The checksum field has the same functionality for SCTP as for TCP, even if another algorithm is used to check the data integrity.

It is obvious that this common header is not enough to provide a service comparable to TCP. Additional units of information are needed to support this, which are denoted in SCTP as "chunks" [Ste07]. SCTP distinguishes two kinds of chunks.

- DATA chunks are used to transport user messages. DATA chunk header information is used to extend the user data to support a reliable, in-order transfer as discussed in more detail in Subsection 2.2.3.
- Control chunks are used to signal a specific functionality. Besides using the control chunks for connection management (see Subsection 2.2.2), they are also used to provide a heartbeat mechanism. The heartbeat mechanism monitors the reachability of an endpoint address by sending a heartbeat request. The receiver has to answer this request with a heartbeat acknowledgment.

The benefit of the architectural concept of SCTP is that new features can easily be integrated without changing the core of SCTP. A good example for this is the RFC5061 [SXT+07] that describes a solution to deploy a dynamic address reconfiguration for SCTP [Ste07], which is not part of default SCTP. SCTP is able to react to dynamic address changes with this extension by adding and removing IP addresses. To add such a functionality, new protocol mechanisms have to be defined and new types of Control chunks have to be created. However, this thesis just focuses on the core SCTP, as specified in RFC4960, and the extensions to provide a load sharing service, which are discussed in the following.

One segment – including the header structure – has to fit into an IP packet which can be forwarded through the network. The maximum size of an IP packet is specified to be 64 KiB, but this is not used as Maximum Transmission Unit (MTU) in the network topology of today's Internet. The default packet size is currently based on Ethernet which provides 1500 byte sized frames, or if using PPPoE (Point-to-Point Protocol over Ethernet [MLE+99]) over a Digital Subscriber Line (DSL) access link, it is 1492 byte. However, there exist multiple variants and so in the end the Maximum Segment Size (MSS) for TCP, i.e. the maximum

length of the user data field, is based on the difference of the MTU and the header overhead of IP. Also, SCTP uses the MTU to calculate the maximum message size for the transfer of the user data. But SCTP defines user data in a different way compared to TCP. TCP works on user data as an unstructured byte stream. Thus, the application itself is responsible to identify the messages in the segments of the byte stream. In contrast, SCTP encapsulates user messages in a DATA chunk and can bundle several small chunks in an IP packet to fill the MTU. If the MTU size – reduced by the header size – does not provide enough space to encapsulate the complete message, a fragmentation is performed by SCTP.

2.2.2 Connection management

TCP and SCTP are both connection-oriented protocols. Thus, data can be allocated to a specific connection. The receiver and the sender have to provide resources to support this kind of stateful connection. The connection life cycle for both protocols starts with a handshake and ends with an abort or controlled tear-down. The signaling used by both protocols differs and also the timing of the resource allocation. As RFC4960 [Ste07] points out, the SCTP connection is called "association" and is a broader concept than the TCP connection. A TCP connection is defined by the unique 5-tuple, whereas an SCTP association is able to span a data transfer over all known destination addresses on the same port.

Because this is also important for the load sharing extension and the corresponding path management (see Chapter 5), the handshake and tear-down process will be discussed in the following.

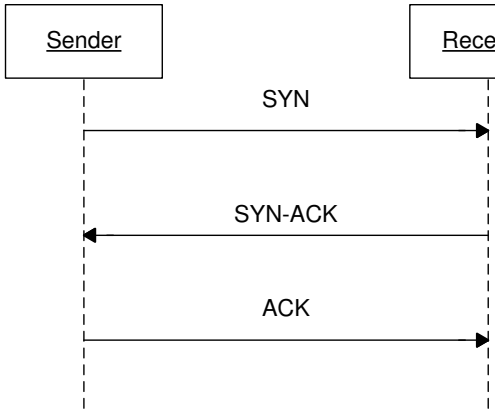


Figure 2.4: TCP handshake

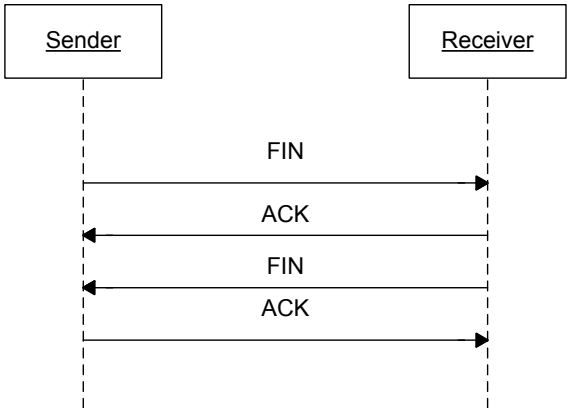


Figure 2.5: TCP tear-down

TCP uses a so-called three way handshake to establish a connection (see Figure 2.4). A TCP connection is defined as a combination of source and destination address and the corresponding ports. Flags of the TCP header are used to signal the handshake. If the port on the server side is not closed, i.e. the server is waiting to accept new connections, a SYN flag set in the first message is interpreted as a request to establish a TCP connection. If the receiver accepts, it allocates resources for the connection and sends an acknowledgment with the ACK flag set combined with a SYN to establish a bidirectional connection back to the sender. In this state, the connection is also described as half-open. After the sender accepts the incoming SYN-ACK, it answers with an additional acknowledgment and once

this acknowledgment with a set ACK flag arrives at the receiver, the connection is fully established. A segment with a set FIN flag signals the termination of a TCP connection, as Figure 2.5 illustrates. The opposite side should acknowledge this FIN request. The receiver should start the same message exchange, too.

There exists no unique connection identifier for TCP to identify a specific connection. Thus, for every incoming data segment a table lookup has to be performed over all TCP connections described by the corresponding 4-tuples. Only if there is a match during this lookup for the source/destination address and ports, the data can be assigned correctly to a connection on the endpoint. This will become important, if later a connection with more than one IP address has to be identified.

However, the three way handshake for TCP comes with some issues, so it is possible for an attacker to mislead the receiver to allocate unneeded resources by flooding it with SYN messages [Edd07]. Furthermore, the three way handshake provides an easy target for man in the middle attacks [Ste07]. The standardization process of SCTP addressed these

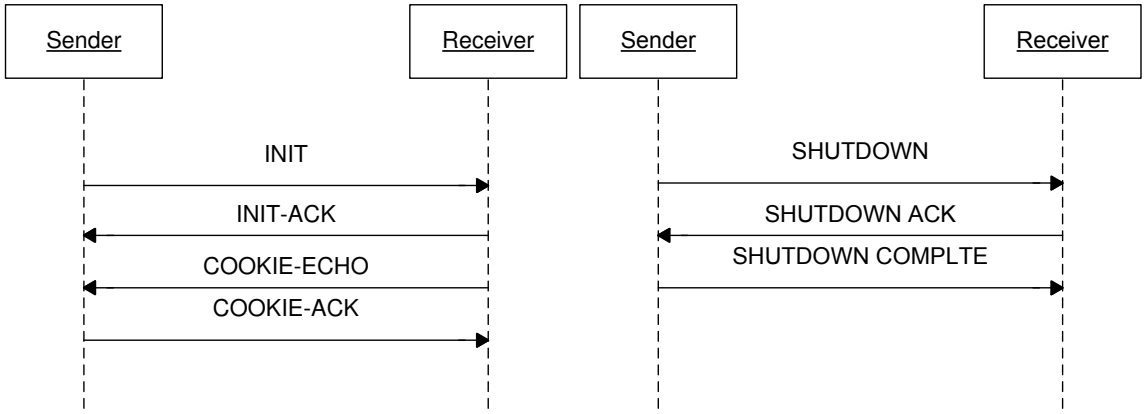


Figure 2.6: SCTP handshake

Figure 2.7: SCTP tear-down

known issues by an extended handshake, also called four-way handshake. As illustrated in Figure 2.6, SCTP uses an INIT control chunk to request a connection establishment at the receiver. This control chunk is encapsulated in a common SCTP header. Unlike TCP, the receiver does not allocate any resources at this time, it just acknowledges this INIT with an INIT-ACK control chunk. This INIT-ACK includes an encrypted state cookie generated by the receiver, to identify the connection request in case of an incoming COOKIE-ECHO control chunk including a copy of the state cookie. A similar mechanism is also known from a TCP extension, which is called SYN-COOKIE [Edd07]. This procedure forces the sender also to allocate resources to establish a connection, which avoids an attack similar to SYN flooding and prevents man-in-the-middle attacks.

During the handshake, further information is exchanged, like the additional IP addresses to support a feature that is called multi-homing. Multi-homing describes the fact that an endpoint may be reachable via more than one address. Thus, an SCTP association is not limited to a 4-tuple to identify a connection, but is rather defined by a list of IP addresses and one port number.

However, even if there exists more than one alternative path to exchange user data between two endpoints, SCTP uses only one address combination for data transfer at a time. This

combination is denoted in RFC4960 [Ste07] as primary path. The RFC4960 defines the primary path as follows:

> The primary path is the destination and source address that will be put into a packet outbound to the peer endpoint by default.

To be clear, the definition of the primary path defines the source and destination address which is used in the IP header to forward the user data through the network. All other alternative paths are observed by the heartbeat mechanism to be able to use them as fallback in case of an error on the primary path. The basic idea of multi-homing in SCTP is to support better network level fault tolerance, not to provide load sharing to increase throughput. This addresses the SIGTRAN use case.

Another difference to TCP is that SCTP supports the transport of more than one logical data flow. For this reason, SCTP introduces the stream concept, which is defined in RFC4960 like follows:

> The term "stream" is used in SCTP to refer to a sequence of user messages that are to be delivered to the upper-layer protocol in-order with respect to other messages within the same stream.

This has impact on the sequence number definition. A stream sequence number (SSN) is introduced by SCTP to organize the ordered transmission on stream level. But to support the reliable sending of the complete message flow and to detect duplicate acknowledgments, another sequence number space, the transmission sequence number (TSN), is introduced. The TSN sequence number space is also denoted as out-band sequence number and describes the overall sequence number space of the complete flow used by all mechanisms to support a reliable transfer.

The tear-down of the complete association is organized by three messages as illustrated in Figure 2.7.

2.2.3 Reliable and ordered transfer

The basis of the Internet is the Internet protocol (IP in Version 4 and 6). One major characteristic of both IP variants is that they provide only unreliable transfer. Thus, if packets are sent to the network, it is not sure if they will reach the receiver at all and in the right order. Even worse, neither the sender nor the receiver will be notified by the network when a loss occurs.

A TCP connection and an SCTP association provide a reliable, in-order transfer of user data. TCP and SCTP use similar mechanisms to support this service but work on different logical units. TCP exchanges a byte stream between sender and receiver, where an unique sequence number identifies each byte of the stream. In currently used TCP implementations, the TCP timestamp option is used to ensure uniqueness and is added to every segment [JBB92]. The sequence number set in the TCP header is assigned to the first byte of the encapsulated data. The initial sequence number is negotiated during the handshake within the SYN flagged segment.

The acknowledgment (ACK) confirms the maximum sequence number that arrived in-order at the receiver by signaling the next sequence number the receiver expects. Because

the ACKs are used cumulatively here, the signaled sequence number is also often called cumulative sequence number. In case of out-of-order segments, the byte stream can be re-ordered by using the sequence numbers. SCTP also uses cumulative acknowledgments, but with them only the overall flow TSN space is addressed (based on user messages instead of bytes). Re-ordering is performed on stream level is only possible by using the SSN.

A mechanism to ensure a reliable data transfer is the use of timer-based retransmission. Simplified, a timer-based retransmission waits for a pre-defined or calculated time, during which an acknowledgment for a specific byte in case of TCP, or DATA chunk in case of SCTP is expected. If during this time no acknowledgment occurs, a retransmission of the missing data will be triggered. This kind of retransmission represents a kind of last resort, which can be implemented independently on the sender side and can be repeated as often as necessary to fill the gaps.

The calculation of the timer-based retransmission for SCTP and TCP is based on the Round Trip Time (RTT) of a packet exchanged between sender and receiver. Thus, the retransmission timeout (RTO) is computed by using the RTT. Because the RTT can be very unstable, a smoothed variant is used, the so-called smoothed Round Trip Time (sRTT). A detailed calculation is given in the currently used version of RFC6298 [PACS11]. On startup, an initial measurement R_0 of the RTT_0 occurs. The calculation of R_0 requires a time base for the calculation which is given by the clock granularity (G):

$$\begin{aligned} \mathrm{sRTT}_0 &= R_0, && (2.1)\\ \mathrm{RTTVAR}_0 &= \frac{R_0}{2}, && (2.2)\\ \mathrm{RTO}_0 &= \mathrm{sRTT}_0 + \max\{G, 4 * \mathrm{RTTVAR}_0\} && (2.3) \end{aligned}$$

Each following measurement of the RTT R_i results in an update of the variables:

$$\begin{aligned} \mathrm{RTTVAR}_i &= (1-\beta) * \mathrm{RTTVAR}_{i-1} + \beta * |\mathrm{SRTT}_{i-1} - R_i|, && (2.4)\\ \mathrm{SRTT}_i &= (1-\alpha) * \mathrm{SRTT}_{i-1} + \alpha * R_i, && (2.5)\\ \mathrm{RTO}_i &= \mathrm{SRTT}_i + \max\{G, 4 * \mathrm{RTTVAR}_i\}. && (2.6) \end{aligned}$$

α and β are smoothing factors and given by the current RFC6298 [PACS11] with $\alpha = \frac{1}{8}$ and $\beta = \frac{1}{4}$.

Anyway, it should be avoided that a timer-based retransmission occurs, because it interrupts the sending process. Missing packets should be detected earlier. Fast retransmission was introduced to achieve this. The idea behind this fast retransmission mechanism is to detect loss early by using certain loss indicators and to retransmit the lost segment quickly to repair the damage. The most common loss indicator for TCP and SCTP is a duplicate acknowledgment (DupACK). This DupACK is generated by a receiver if an out-of-order segment arrives, which produces a gap in the received sequence number space, i.e., one or more segments with lower sequence numbering than the received one are still missing. In case of a missing segment, the receiver sends a duplicate of the last regular (cumulative) ACK. Since reordering of segments may occur within a network, a single out-of-order segment might not always indicate loss. Therefore the sender waits until it receives a certain number of DupACKs for the same sequence number, defined by a threshold, before it retransmits any seemingly lost segment.

The selective acknowledgment (SACK) was introduced [MMFR96] to support an additional mean for loss indication by using so-called “gap” blocks. A gap block defines an

in-order sequence number space of already received data. Thus, also messages not delivered in-order beyond the cumulative sequence number are part of this gap reporting. Duplicate acknowledged sequence numbers signaled by these gap blocks are interpreted as indication for loss and will trigger a fast retransmission. SACKs in TCP use flags and a part of the option space to signal information. SCTP uses the same basic idea but here a specific control chunk signals a SACK. In any case, a timer-based retransmission will occur if the fast retransmission also gets lost.

2.2.4 Congestion and flow control

Congestion and flow control are in general complex mechanisms of both transport protocols. Both are in the main focus of this thesis. In both protocols both mechanisms are based on the sliding window mechanism and try to keep the overall system in equilibrium. Sliding window describes a mechanism where the send rate is limited by a dynamic calculation. The result of this calculation is denoted as window. In general, a window defines the maximum amount of data that is allowed to be sent unacknowledged. Unacknowledged data that was already sent is also often denoted as "in flight". Thus, the data which is allowed to be sent during one send cycle can be calculated by the given window size reduced by the amount of data that is still in flight. The send window for TCP and SCTP is in the end the minimum of two different windows provided by two different mechanisms.

- (Receiver-side) flow control is a mechanism to protect the receiver. It can eventually happen that the receiver is not able to process all the data sent by the sender. Thus, the source should limit the amount of new data. The amount of data should be equal to the amount of free buffer that is provided by the receiver. Thus, at startup, the advertised receiver window (ARWND) is initialized equal to the receiver buffer size configured at the receiver side. If the ARWND is decreased down to zero, this state is also denoted as closed window.

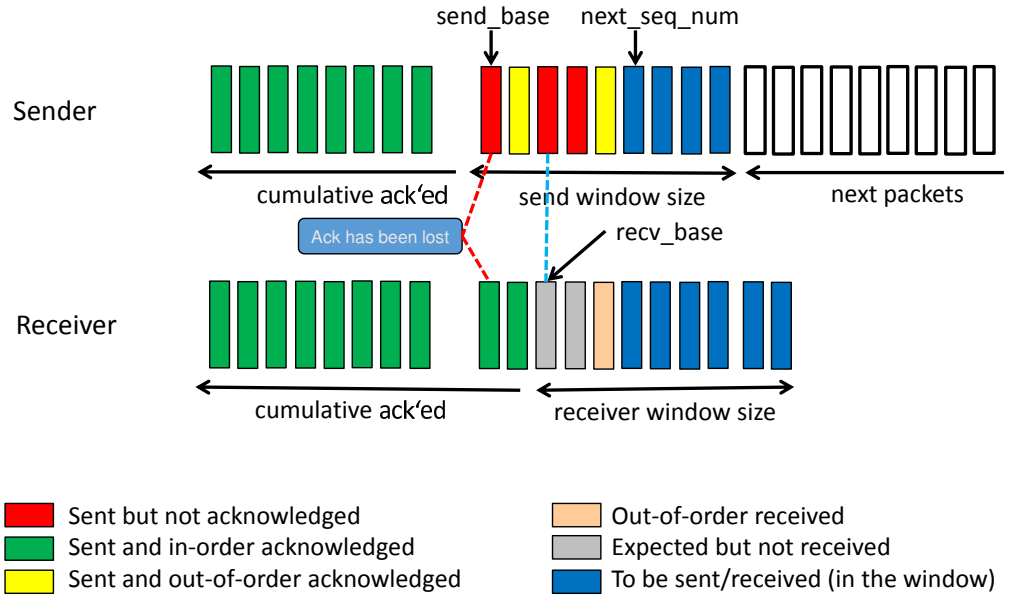


Figure 2.8: Sliding window example

- The main goal of the congestion control is to protect the network. Although congestion control was not part of the first core standard of TCP in RFC793 [Pos81b], the experience in the first years had shown that the Internet would collapse without [Wel05]. So, the congestion control became a strong requirement for new transport protocols like SCTP. Besides the network protection also the task of a fair sharing of the network resources was assigned to this mechanism. For the currently used protocols a window is calculated that defines the maximum amount of outstanding data in bytes and is called congestion window (CWND). Different variants exist here, the most used New Reno algorithm will be discussed in more detail in Subsection 2.2.4.1.

The minimum of the ARWND and the CWND gives the maximum send window. The Figure 2.8, which is based on an example given in [Wel05], illustrates the most important identifiers of this mechanism. The queue on the sender side and the queue on the receiver side are illustrated in this figure. The queues are filled with user data – represented by rectangles – which are categorized by their assigned status. Because their status is important for the buffer blocking issues (see Subsection 7.1.3), it is discussed now in more detail:

- Sent but not acknowledged user data are represented by red rectangles. The lowest sequence number at the sender is stored as *send_base*. This *send_base* is in many implementations and documentations also called *snd_una* (send-unacknowledged) and, furthermore, often used to calculate the data in flight.

- Sent and in-orderly acknowledged user data completed the send cycle already and is illustrated by the green rectangles. The situation can exist where the sender has less sent and in-orderly acknowledged data than the receiver, because the acknowledgment is on the way or got lost. However, all in-orderly acknowledged data can be removed from the buffers and queues. This kind of data does not block any resources in a normal send cycle on a singlepath.

- Sent and out-of-orderly acknowledged data is reported by the SACK mechanism and can be identified in the Figure 2.8 by the yellow rectangles. Even if they are reported as delivered, they cannot be removed from the buffer by default, because their state can be redefined at any time.

- Received out-of-order data causes a decreased ARWND, because this data must be kept in the receiver queue until the gaps can be closed. The beige colored rectangle represents this segment in the figure.

- Expected but not received segments are illustrated by the grey rectangles. The sender should send these segments next. Out-of-order segments or loss mostly causes this kind of gaps on the receiver side. These segments build the gaps in the gap report of a SACK. The first segement *recv_base* is often also called rcv_nxt in implementations or their documentation.

- At last there are the packets which are not yet sent or received in-order but fit into the window size of the sender, which is based on the CWND or ARWND. The next in–order sequence number (next_ sequence_num) can only be assigned if there is space in the send window. This sequence number is also often called *snd_nxt* and the rectangles for this group of segments is colored blue.

Several approaches for the congestion control exist. [Wel05] gives an overall overview of different congestion controls, for example Cubic, TCP Reno, TCP Vegas, TCP Westwood+ and discusses their pros and cons. However, currently, the two most common congestion control versions for TCP are New Reno and Cubic. SCTP [Ste07] also applies New Reno as congestion control that is why this thesis focuses in the following fairness discussion on this approach defined in [APB09].

2.2.4.1 New Reno

The New Reno congestion control is a loss-based congestion control and is based on the so-called additive increase/multiplicative decrease (AIMD) [Wel05] approach applied by TCP [APB09] and SCTP [Ste07]. It builds the base congestion control for the following fairness discussion. In this approach, the congestion window (CWND) of a path c_P will be changed in case of a congestion indication by using a multiplicative decrease of c_P and in case of positive acknowledgments by using an additive increase of c_P. As mentioned, c_P limits the number of outstanding bytes on path P. The additive increase of c_P can be split in two different phases:

- **Slow start** is used once probing for the allowed send rate starts. It results in an exponential growth of c_P. This is achieved by an increase of one message unit for every new cumulative acknowledgment received.
- **Congestion avoidance** represents the second phase and leads to a linear growth of c_P when c_P exceeds the slow start threshold s_P. The partially acknowledged bytes p_P are used to ensure that the complete c_P is acknowledged, before the window is increased again. α represents the acknowledged bytes.

In detail, the algorithm can be described for TCP as follows:

$$c_P = c_P + \begin{cases} \min\{\alpha, \mathrm{MSS}_P\} & (c_P \leq s_P) \\ \mathrm{MSS}_P & (c_P > s_P \wedge p_P \geq c_P) \end{cases}.$$

MSS_P denotes the Maximum Segment Size (MSS) on path P. On a retransmission (RTX) on path P, s_P and c_P are adapted as follows:

$$\begin{aligned} s_P &= \max\{c_P - \frac{1}{2} * c_P, 4 * \mathrm{MSS}_P\}, \\ c_P &= \begin{cases} s_P & \text{(Fast RTX)} \\ \mathrm{MSS}_P & \text{(Timer-Based RTX)} \end{cases}. \end{aligned}$$

2.3 Special network components and relevant aspects

IETF protocols have to adapt to real network conditions. That means, they have to address existing constraints given by hardware or interfaces. Over the time a lot of elements were introduced in the Internet which deal with optimization or limited resources.

2.3.1 Issues with middleboxes

The term middlebox was introduced to address functions in the network which are beyond the regular standard functions of an IP router [CB02]. In today's Internet, middleboxes have a

deep impact on transport protocols like TCP. Some middleboxes are used to integrate security infrastructure, like firewalls, but most of them are used to provide a specific network service. A very common middlebox function is the network and port address translation (N/PAT) [EF94]. This service is used to manipulate the IP addresses and the port number of the transport protocol to map for example one IP address space to another. An often observed use case is the masquerading of an entire private IP network behind a single public address. Other examples are the support of Differentiated Services (DiffServ) [Gro02], which supports Quality of Service (QoS) in the network or different kinds of packet filters [ZMD+09]. A packet filter can possibly block packets or replace parts of the content included. [DHB+13] did some research on this topic and they figured out that only the IP version number and the protocol number for an encapsulated TCP segment stays untouched in the worst case. This has a huge impact on the degrees of freedom of a protocol design. If options, for example, are manipulated, replaced or removed it has a direct impact on the usability of extensions.

2.3.2 Queueing discipline

In a packet switched network like the Internet, resources are shared among different connections. Thus, a logical arbiter is needed to control the access to these shared resources. This logical arbiter has to work on the physical buffers represented by the queues at the endpoints and connecting routers. For these queues, different queue management strategies can be applied. Different queueing disciplines were introduced to address different network conditions even if in the Internet the First In, First Out (FIFO) queueing discipline is most widely used in the routers or endpoints. Nevertheless, studies show that for example the usage of queues supporting Random Early Drop (RED) can increase the overall network performance significantly [BCC+98].

The principle of a FIFO queue (queue with applied FIFO queueing discipline) is simple. The packets are enqueued in the order of their arrival time. The queue will be freed in the same order as the queue is filled. If the input rate exceeds the output rate, the FIFO queue will be filled with packets. Because physical memory which provides queues with buffer, is not unlimited, a queue size gives a maximum limit of bufferable packets. All packets that arrive once the queue limit is reached will be dropped and lead to packet loss. The RED queueing discipline [FJ93] extends the idea of the FIFO queues. Queues with applied RED queueing discipline have a linearly increasing drop probability depending on two threshold parameters and drop packets before the buffer is full. For the simulation experiments with RED queues in this thesis, the parameter recommendations of [Flo97] have been used.

2.3.3 Socket concept

In currently used operating systems like Linux, FreeBSD or Microsoft Windows, the transport layer of the ISO/OSI reference model also has a specific task. It represents the border between an application in user space and the tasks assigned to the kernel. There is only one implementation for all protocols which is assigned to the kernel, thus there is one protocol stack. The kernel of the operating system manages all outgoing and incoming TCP or SCTP connections in one data structure and has to identify them correctly.

Most OSs provide an interface to this functionality which is based on the Berkeley API [SFR03]. This de-facto standard evolved over time into the Portable Operating System In-

terface (POSIX) socket API, which provides a common interface to standardized structures of a transport protocol. Core element of this API is the socket structure, which describes a unique connection entity in an inter-process communication[6]. A socket request for a TCP or an SCTP connection requires the choice of the protocol family (PF), which can be mainly split in the choice of PF_INET for IPv4 and PF_INET6 for IPv6. Furthermore, the socket type (so_type) and the protocol number of the protocol are required.

A tuple of communicating local and remote sockets is often denoted as socket pair. The information of a socket is held in a so-called socket protocol control block (so_pcb). The protocol control block (PCB) represents the root of the data structures to identify a transport protocol connection by its 5-tuple. The PCB is the basis of the discussion in Section 5.1

[6]Even if a network connects this inter-process communication.

Chapter 3

Basic function set of load sharing for TCP and SCTP

This chapter gives a general introduction to the benefits of load sharing, in particular on the transport layer. There exist many load sharing approaches for the different layers of the OSI reference model, which all address more or less the same issues, like for example the out-of-order problem complex. This chapter discusses the reasons to place the coupling of resources on the transport layer. Furthermore, this chapter introduces the transport protocol extensions discussed by the IETF to create a concurrent multipath transfer.

3.1 Concurrent multipath transfer and load sharing

The structure of the Internet, as a meshed network, provides most of time more than one path to reach the destination. So, it should not surprise that the usage of more than one path has a long history in the Internet protocol development and in particular in the development on the transport layer. Researchers and developers all around the world worked on this challenge by using different umbrella terms as for example load sharing [ELZ86], bandwidth aggregation [MK01a], concurrent multipath transfer [IAS06], inverse multiplexing [Dun94] or other terms like coupled resources or utilization of multi-homing [Ste07]. Anyway, the main idea is the same in every case, i.e. using more than one resource to increase the quality of service. The approaches only differ in the definition of the resource and the defined goals that should be achieved.

3.1.1 Goals

In the following, the benefits of load sharing are discussed for the endpoints only. This requires that the endpoint is connected to the network in a multi-homed manner. Therefore, the endpoint is able to use more than one IP address to access the same destination. The following goals with respect to load sharing on a multi-homed system can be identified [NUO+07]:

- **Redundancy**
  The connectivity can be guaranteed, if at least one path through the network can be established by using more than one access to the network.

- **Load sharing**
  The basic idea is to pool all resources and use what is needed to increase the throughput. Traffic can be distributed among different paths through the networks to improve the overall service. A scheduling process is required to support load sharing. This scheduling process has mostly the goal to increase the throughput, although other policies could be applied like to decrease costs or delay. Even non-technical, e.g., political reasons can have an impact on the scheduler. This work just focuses on the throughput goal.

- **Load balancing**
  Balancing load is particularly interesting whenever parts of the network are congested. By shifting load from one path to another, the overall system can be relieved without losing performance for a specific connection. This helps to improve the resource utilization.

- **Bi-casting**
  Here, the data is be duplicated and sent via different paths redundantly. This has a benefit e.g. for a delay sensitive communication, like a IP video conference, because errors or loss can be compensated with the redundantly sent data. Furthermore, if all paths are able to transport the data, it improves the handover performance in a mobility scenario.

It is obvious that not all goals can be addressed by one multipath approach and that besides these major goals also the context of the deployment matters. The same network technology in different setups leads to different degrees of freedom in the usability. As an example, compared to a data center, a smart phone user connected via a 3G provider has less or no control of the access medium, the network behavior, kind of traffic, topology, hardware or their preferences. But all these characteristics have a huge impact on the performance of a load sharing approach.

In the related literature it can be distinguished between local domain and Internet solutions. Where the architects of a local domain solution have the possibility to adapt the network to the load sharing requirements, the architects of the Internet solutions have the goal to create an approach with the ability to adapt to the conditions of the Internet. The architects of the standards in the IETF focus on the condition adaptation, as it can be observed in the discussions on the mailing lists of the MPTCP working group or the TSVWG working group.

3.1.2 Alternative approaches in the context of the IETF

The IETF provides different approaches to couple resources and to improve the quality of the service almost on every layer below the transport layer of the OSI reference model. The resource definition itself is mostly based on the definition of the layer. As example, on the data link layer the multilink Point-to-Point Protocol (multilink PPP) can be mentioned, which was standardized by the IETF in [SLM+96]. The main goal of this draft is to couple different PPP [Sim94] connections. Unfortunately, the link layer approaches are only usable in dedicated segments of the Internet, e.g. on an Integrated Services Digital Network (ISDN) access link [SLM+96]. These approaches are not suitable for a wide deployment in the Internet.

The network layer keeps the Internet together by providing the logical addressing in IPv4 and IPv6 and by providing the forwarding service that enables the transport through the

network. A path through the network is defined on the network layer as a hop-by-hop decision performed by the routers for every packet. In principle, every packet can be routed through the network on a packet specific path. Thus, the paths for different packets are depending on routing decisions and that can be used for a kind of load sharing in the network. The routing decisions are results of policies based on the routing protocol. A routing approach that considers more than one path is denoted as multipath-routing [TH04]. This is a useful way of deploying load sharing, because routers can react quickly to changes. However, it is nearly impossible to replace the complete infrastructure of the Internet with routers providing this specific feature, although this replacement can be a real alternative for local domains.

Adding a new shim layer in the layered network model is also an option and provides an alternative view on coupling resources on the network layer. This shim layer is located between the network layer and the transport layer. Goal of this shim layer is to support the split of identifier and locator. At first sight, this approach is more a question of mobility, but mobility is an extreme scenario for the transport of data via more than one path that needs to consider dynamic address changes, too. Approaches of protocols which can be used in this context are the Host Identity Protocol (HIP) [MN06] for IPv4 and the site multi-homing by IPv6 intermediation (Shim6) [NB09]. These approaches also provide a solution to support the end-to-end transport via multiple paths, but show negative effects in combination with TCP or SCTP as discussed in the following subsection.

Besides the IETF and IEEE standardization, the research community developed many alternative ideas and prototypes. However, to shift this functionality to the transport layer has significant advantages, which also are discussed in the next section.

3.2 Multipath transfer for TCP and SCTP

The basic mechanisms of TCP and SCTP were introduced in Chapter 2. It is neither provided by SCTP nor by TCP to use more than one path at same time for data transfer. TCP itself provides not even a mechanism to use more than the 4-tuple of source/destination addresses and ports to establish an end-to-end connection. In general there are two alternatives to realize a connection over multiple paths on the transport layer:

- **Multi-homing unaware**
  The transport protocol itself is not aware of the multi-homing feature. This can be realized today by running the service over HIP [MN06] or Shim6 [NB09]. The solutions are easy to deploy, because no changes are required in the transport protocol itself, as from its view nothing in the connection setup changes. But this straightforward deployment causes a lot of problems for the transport protocol, because many mechanisms assume the use of one path, like the RTT measurement, the re-ordering process or the congestion control. Furthermore, the unaware load sharing approach has an indirect impact on other mechanisms like timer-based retransmissions, because for example the RTO calculation is based on the RTT measurement (see Subsection 2.2.3). This is a possible solution, but it is obviously not a good one.

- **Multi-homing aware**
  Over the years many approaches came up, in particularly as extensions for TCP. In 1995, Christian Huitema proposed the multi-homing feature for TCP in [Hui95]. With R-MTP [MK01b], pTCP [HS02] or mTCP [ZLK+04] other examples can be added to the

list. However, none of these approaches had a chance for standardization. SCTP [Ste07] was at the starting point of this thesis the only protocol on the transport layer which was standardized by IETF with real multi-homing support. However, this approach had the goal to increase reliability and, therefore, no load sharing was supported by SCTP.

Even if no load sharing approach on the transport layer finally succeeded in the IETF standardization process, there are good reasons to spend further efforts to provide load sharing on the transport layer. Voice and Kelly [KV05] first suggested the placement of mechanisms to support load sharing in the congestion control of a transport protocol. The reason for this is trivial. In the current Internet structure only the endpoints have an overview of the connected networks. Internet service providers (ISP) mostly do not share information across provider borders and are not aware of this information. The authors of [WHB08] developed a quite good model to illustrate the benefits, which is represented in Figure 3.1. The figure describes two scenarios, where endpoint A has a default singlepath connection, e.g., a regular TCP connection. The endpoints B and C are multi-homed. Therefore, if the congestion control supports load balancing, B and C can react to congestion across provider borders. This is illustrated on the right side of the Figure 3.1 by the thicker lines. A provider cannot achieve this load balancing, because a provider has no knowledge about the other networks.

But if such good reasons speak for the usage at the transport layer, there must be other reasons which have prevented standardization in the IETF so far. The main reason for this is the strict requirement to provide TCP-friendliness. The issue with that is discussed after introducing some terms used in the context of multipath transfer.

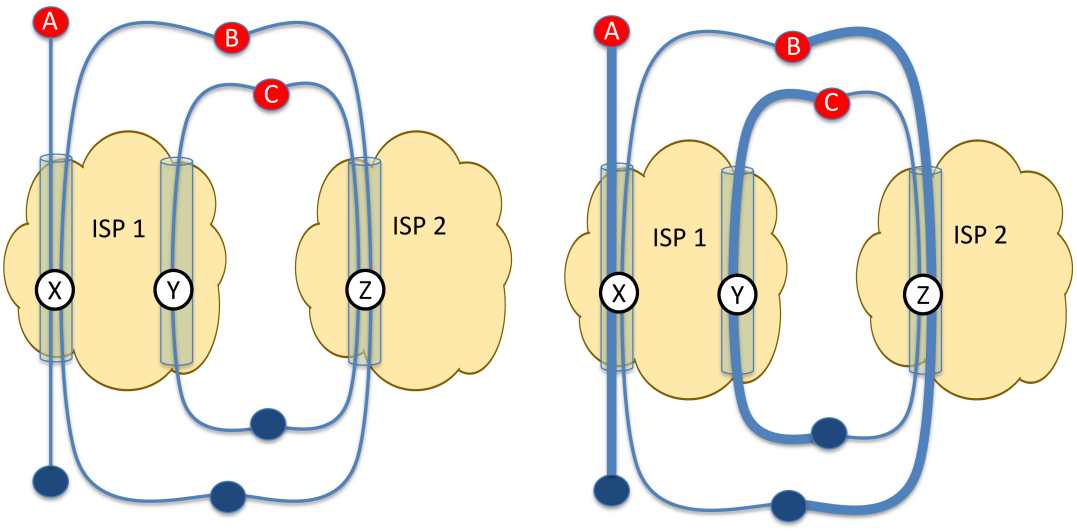


Figure 3.1: Load sharing example on the transport layer (is based on [WHB08])

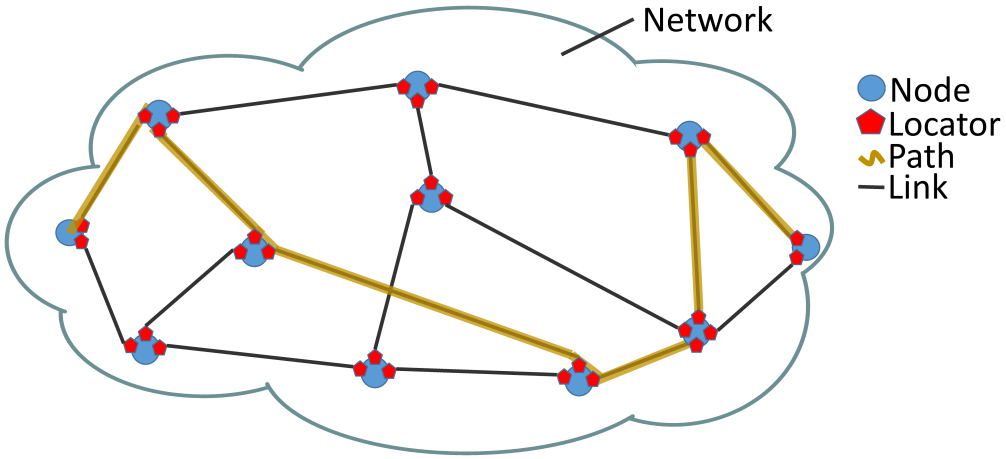


Figure 3.2: Network topology example

3.2.1 Definition of multi-homing, multipath and flow

An endpoint is multi-homed for a transport protocol if the system provides more than one IP address to access the network. A multipath flow can be defined by a flow addressable via more than one IP address. The terms flow, network, path and subflow are not defined consistently in the IETF. Definition, scope and functionality can differ with the point of view and the context in which they are used.

As preparatory work of this thesis, some basic definitions were formally introduced by [BDAR12b]. Furthermore, [Dre12a] provides a detailed discussion of the terms used in this context. However, this thesis only uses few entities of this overall definition, therefore only entities relevant for the discussion are illustrated in the Figure 3.2.

Important for this thesis is the "path" definition. A path on the network layer is a different entity than the path addressed by an end-to-end connection on the transport layer. As formally introduced in [Dre12a], the Figure 3.2 uses a finite locator set, node and link set to describe a network. Every node (see blue circles in Figure 3.2) provides one or more unique locators (see red hexagons in Figure 3.2) which are for example unique IP addresses in the Internet. These locators can be allocated to connecting links (see the black lines connecting the red hexagons in Figure 3.2). Theses links provide specific characteristics like a capacity, error rate or delay. A sequence of locators identifying a sequence of nodes and links represents a path through the network on the network layer (see the yellow line in Figure 3.2). The endpoints of these network layer paths are the locators of the end-to-end connection endpoints. The locators on the connection endpoints define the end-to-end path on the transport layer. The bandwidth of the end-to-end path is based on the capacity of the weakest link in the link chain. Delay and error rate are cumulative. Different paths on the network layer can be used for the end-to-end path as long as the edge locators at the endpoints remain the same. An end-to-end connection uses an end-to-end path to transfer the user data within a flow.

The flow of a singlepath connection, like e.g. a TCP connection, only consists of one subflow. However, for a multipath transport, e.g. based on CMT-SCTP or MPTCP, one flow may consist of multiple subflows. A subflow is always assigned to one specific end-to-end

path. Furthermore, it is not mandatory that locators are in a 1:1 relationship to an interface. The combinations here are diverse and can be characterized as follows

- **Single link, multiple IP addresses**
  In IPv4 this concept is called IP aliasing. IP aliasing is used to provide different virtual sites for, e.g., webservers. In IPv6 this relation is more the default than the exception, like for example to support a global and a link-local unicast address.
- **Multiple interfaces, single IP address per interface**
  This is the most straightforward interpretation of a multi-homing endpoint. Thus, an endpoint has more than one network interface and every interface has its own IP address.
- **Multiple links, single IP address**
  This concept is used less often and is mostly known from the routing context, where routing protocols can address different interfaces to forward to the same IP address.
- **Multiple links, multiple IP addresses**
  Load balancers on the link level use this concept to ensure that they are able to use all possible path combinations, with the goal to utilize the network in an optimal way.

Each of these combinations can occur in the Internet.

3.2.2 TCP-friendliness

As introduced in Chapter 2, TCP and SCTP use a congestion control mechanism to protect the network and to achieve a fair behavior. Two classes of approaches have been proposed to achieve a fair distribution of resources in a network: centralized and decentralized. Centralized approaches use a global management instance for making decisions on resource distribution. These decisions may be based on fairness definitions such as *max-min fairness* [Hah91], *proportional fairness* [MMD91] or *weighted proportional fairness* [Kel97]. In this case, connection endpoints and flow characteristics (e.g. bandwidth, delay, etc.) are supposed to be fixed. In reality, however, a communication in the Internet may be a highly dynamic process, where the actors as well as the network characteristics are continuously changing. Therefore, decentralized approaches – where the decisions about resource allocation are made by the connection endpoints themselves – have been introduced. With the widespread deployment of TCP, the discussion about fairness – especially *flow rate fairness* [Bri07] which is denoted in the followings chapters as "flow fairness" – has moved towards a discussion about *TCP-friendliness* [BCC+98]. The TCP-friendliness term was introduced by [Wel05] and is based on the TCP-compatible definition of RFC2309 [BCC+98], which can be cited as follows:

> A TCP-compatible flow is responsive to congestion notification, and in steady-state it uses no more bandwidth than a conformant TCP running under comparable conditions.

It should be highlighted that this definition is based on flow level. In this historical context the flow is based on the definition of a TCP connection, which is based on the 4-tuple introduced in Chapter 2. Therefore, today all multipath transport protocols have to be designed to achieve the goal to allocate no more bandwidth than a TCP flow under comparable conditions. "Comparable conditions" imply that all relevant parameters are identical. However, the list of parameters given in the standards is very vague ("drop rate, RTT, MTU, etc.") [Flo00, BCC+98].

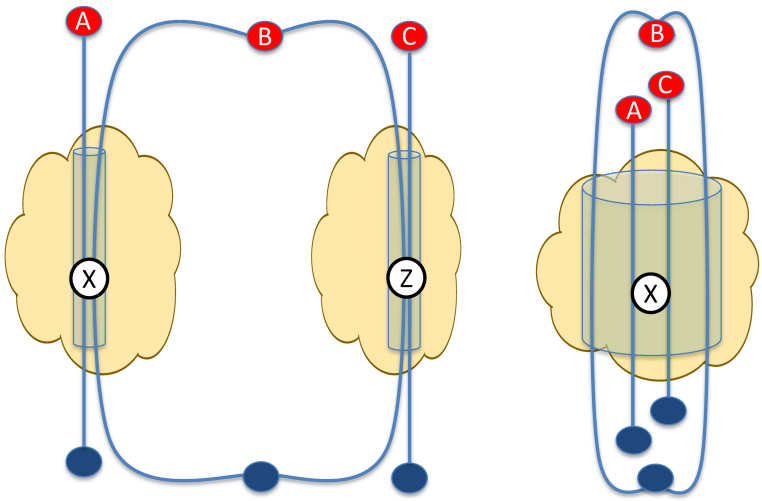


Figure 3.3: Bottleneck scenario

3.2.2.1 Shared bottleneck scenario

The congestion control has a huge impact on the rate control of the sender. If a straightforward implementation of New Reno on each subflow is used to deploy MPTCP in the network, the TCP-friendliness cannot be achieved in every case. It is depending on the topology of the network. Figure 3.3 illustrates two possible scenarios. In the first scenario the multipath connection uses two independent physical paths (X and Z) through the network. The flows of A, B and C have to share the resources of these paths, which is represented by the link capacity of X and Z. On each bottleneck link a fair allocation of the capacity can be expected if a congestion control like New Reno is deployed on each subflow. However, the second scenario describes the case where the capacity of only one bottleneck link X is shared among different connections. Here, the two independent singlepath connections (A and C) have to share the capacity with a multipath connection B. The TCP-fairness requires a sharing of the capacity on flow level in three equivalent shares, because every connection represents one flow. However, by using the uncoupled New Reno congestion control, the TCP-friendliness on flow level will not be achieved. The multipath connection will allocate twice the capacity on the shared bottleneck as allowed. The capacity allocation results in $\frac{1}{2}$ for the multipath connection and $\frac{1}{4}$ for every singlepath connection. In the IETF, this behavior is defined as TCP-unfriendly and, therefore, unfair and this scenario is denoted as the *shared bottleneck scenario*. Thus, an end-to-end transport protocol in the Internet has to be aware about the topology used. [Dre12a] provides three alternatives to work against this issue:

- **Bottleneck detection**
  This is a new mechanism with the goal to detect shared resources of a multipath flow. Mechanisms exist like [YWY08b], [YWY08a] [Wel13] or [RKT02]. But even if these

mechanisms work in theory, they are based on complex data aggregation, which needs at least a stable database of measured data, like for example RTT or loss rate.

- **A-priori knowledge**
  This approach is useful in scenarios where the user is aware of the topology or the network provider aware of the needs of the user. That is true for controlled networks but usually not for the Internet.

- **A topology adapting congestion control**
  The congestion control adapts on the topology of the network. As long as the congestion control is not more aggressive than for a singlepath TCP flow, the TCP-friendliness goal is achieved.

For the current thesis, the progress with respect to the topology adapting congestion control plays a specific role. This new approach has re-launched the standardization discussion for multipath TCP and SCTP and is denoted as Resource Pooling.

3.2.2.2 Resource Pooling principle

The idea of Resource Pooling (RP) was presented by [WHB08] and is initially quite simple. RP couples the congestion control of all paths with the goal of:

> making a collection of resources behave like a single pooled resource.

Thus, an approach implementing the Resource Pooling idea couples the per path congestion control mechanisms in order to shift traffic from more congested to less congested paths. Releasing resources on a congested path decreases the loss rate and improves the stability of the whole network. However, this Resource Pooling idea has a severe impact on the congestion control, because it extends the congestion control by adding the additional task to adapt to the topology. So, the goals for a coupled congestion control based on the Resource Pooling idea can be identified as:

- **Protect the network**
  It is the goal of the multipath transport protocol to keep the system in equilibrium like the singlepath protocol. Any network can only transport a limited amount of traffic in a certain period of time. Thus, if senders inject more traffic into the network than given by its nominal capacity, the network reacts with a congestive collapse. The send rates have to be adapted to the network capacity but the capacity is not known to the endpoint. Therefore, algorithms are required to detect and avoid congestion. Solutions for this are for example discussed in [Ram12].

- **Achieve fair behavior**
  The Resource Pooling idea introduces a new perspective on the multipath fairness goals. These multipath fairness goals are described by [RWH09] as:

> 1. *Improve throughput*: a multipath flow should perform at least as well as a singlepath flow on the best path.

2. *Do not harm*: a multipath flow should not take more capacity on any one of its paths than a singlepath flow using only that path.
3. *Balance congestion*: a multipath flow should move as much traffic as possible off its most congested paths.

- **Adapt to topology**
  This is an additional task of a multipath congestion control and addresses the need of topology detection to achieve a TCP-compatible behavior. In the end, Resource Pooling is just a workaround. It does not solve the TCP-friendliness discussion, it bypasses this issue by balancing congestion. The discussion of the impact of this suboptimal approach is a major goal of this thesis (see Chapter 6).

Even if this approach to solve the TCP-friendliness issue is accepted by the IETF, it is not clear which impact these coupled congestion controls have on the transport protocol behavior. Therefore, Chapter 4 of this thesis focuses on the impact of their deployment.

In addition, it is important to note that with the new congestion control approach new fairness goals were introduced which gave the standardization process for MPTCP and CMT-SCTP a new basis. The impact of these multipath fairness goals has not been discussed sufficiently so far and is therefore also part of this thesis.

3.2.3 Impact on the transport protocols

One major requirement for an IETF load sharing approach is to provide the services known from the singlepath protocols in the same way. The basic idea is to be completely transparent to the application and the network. For CMT-SCTP this raises less issues, because SCTP was designed to support more than one path, but MPTCP was not. That is why three different compatibility constraints were defined for multipath TCP (MPTCP) [FRHB13]:

- *External Constraints*
  The protocol must function through the vast majority of existing middleboxes such as NATs, firewalls, and proxies, and as such must resemble existing TCP as far as possible on the wire. Furthermore, the protocol must not assume the segments it sends on the wire arrive unmodified at the destination: they may be split or coalesced; TCP options may be removed or duplicated.
- *Application Constraints*
  The protocol must be usable with no change to existing applications that use the common TCP API (although it is reasonable that not all features would be available to such legacy applications). Furthermore, the protocol must provide the same service model as regular TCP to the application.
- *Fallback*:
  The protocol should be able to fall back to standard TCP with no interference from the user, to be able to communicate with legacy hosts.

Clearly, no change in the API should occur and each subflow of an MPTCP connection should look like a singlepath TCP flow, if it is sent over a link. Thus, an MPTCP subflow has to open an additional connection with an additional three way handshake and for terminating

a FIN must be used at subflow level. It is important to retain the well-known singlepath signaling on path level, because it is important for middleboxes. The risk that a connection will be blocked, for example by a firewall, is decreased if it looks like a default singlepath connection. If no multipath connection is possible or the multipath connection will be blocked, MPTCP falls back to a singlepath connection. The impact of the constraints on the extension design for MPTCP and CMT-SCTP will be discussed in the following.

3.2.3.1 Impact on connection management

TCP provides – in contrast to SCTP – no mechanism to exchange additional address information in an initial handshake. Thus, there is a need to signal to the endpoints that a connection via more than one address is possible. For MPTCP this signaling is described in [FRHB13] and introduces additional options for TCP:

- **MP_CAPABLE**
  This option signals to the endpoint that an MPTCP connection is requested. If this option is filtered or replaced by middleboxes, the MPTCP handshake will be proceed like a singlepath TCP handshake. Key material will be exchanged during the initial MPTCP handshake which is used to create a unique connection token (see RFC6824 Subsection 2.1). This unique connection token is used to create a start sequence number. This start sequence number is used for the overall flow reordering process (see RFC6824 Subsection 3.3.1).

- **MP_JOIN**
  The MP_JOIN option is used for every additional handshake of the multipath connection if the initial handshake was successful. In principle the additional handshake looks like a default singlepath TCP handshake. The only difference is that the handshake segment includes the MP_JOIN option. This MP_JOIN option includes, furthermore, the connection token of the multipath connection and a SHA-1 HMAC. The SHA-1 HMAC is used for endpoint authentication. Because the correct HMAC has to be confirmed, an additional ACK is required. This additional ACK is often used to signal an additional address if explicit address notification is used.

MPTCP provides two ways to add further addresses to the connection.

- **Implicit address addition**
  This is a straightforward extension of the multipath connection by adding every new source or destination IP address available. If an endpoint is aware of a new IP address or a new port, the connection management creates all possible 4-tuples with the local IP addresses and ports and initiates an MP_JOIN handshake for every unknown combination. This approach works fine as long as no middleboxes like Network Address Translation (NAT) come in and both endpoints are allowed to create new connections. That both endpoints are allowed to create new connections is not always the case, e.g., if one endpoint is behind a firewall. Thus, even if this multipath connection setup alternative is a functional approach, the implicit addition of addresses will not be the default behavior in the Internet.

- **Explicit address addition**
  This approach is illustrated in Figure 3.4. The process itself is split in two phases represented in the Figure 3.4 by the red and the green arrow. In the first phase an endpoint

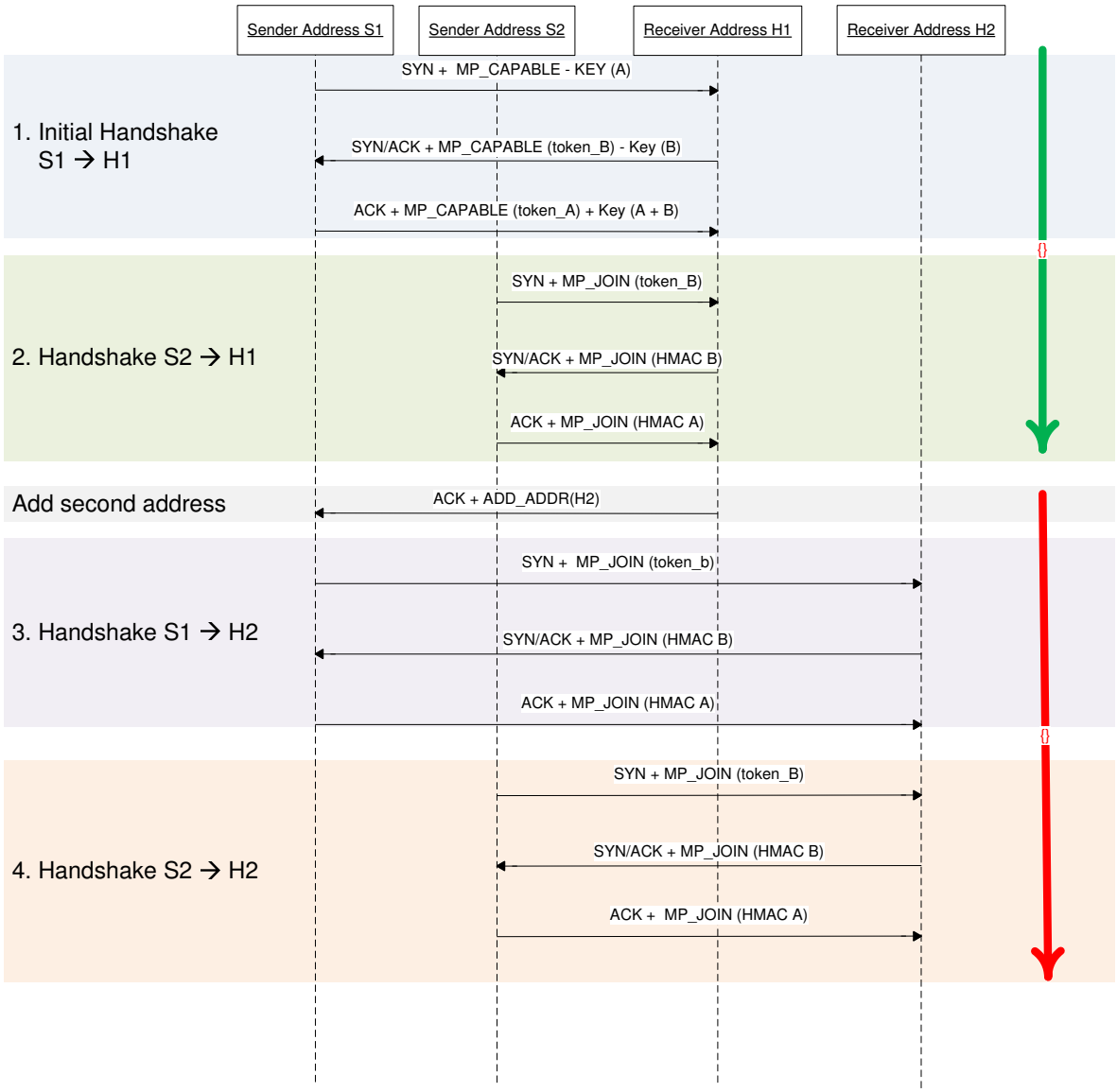


Figure 3.4: MPTCP handshake

(the sender) tries to perform a handshake for every 4-tuple the endpoint becomes aware of. Like in the implicit approach the sender uses for the first handshakes, the destination address that is set by the application socket call (see basics in Section 2). In the scenario of Figure 3.4 the application uses the destination address $H1$. For a dual homed endpoint with the IP addresses $S1$ and $S2$ this leads to a handshake for S1↔H1 and S2↔H1 (see handshake 1 and 2 in Figure 3.4). It is not defined in which order these first handshakes should be performed, therefore, it is also possible that these handshakes can be reordered (this potential issue is also discussed in Chapter 5). After the first phase, the passive endpoint has to provide new addresses by using the "ADD_ADDR" option. This is the first message in the phase two. Thus, for every address the other endpoint is not aware of, the endpoint needs the additional ADD_ADDR option. This

information is sent most of the time during an additional ACK directly after the first initial handshake, but the RFC is not clear here so it is not a "must". The sender is able to build the missing 4-tuples for the address combinations S1↔H2 and S2↔H2 by getting an additional address. These handshakes (handshake 3 and 4 in Figure 3.4) in addition to the first two handshakes lead to a fully meshed connection setup between the endpoints.

Each established subflow also has to be terminated at the end of the session. In singlepath TCP, both sides send a FIN and the connection will be closed. However, this is not adequate for an MPTCP connection. Even if a FIN is sent on subflow level, it should potentially not tear down the whole multipath connection. That is why the Data Fin option on flow level was introduced. This option can signal a complete multipath connection shutdown with the same semantic as the default TCP FIN.

The CMT-SCTP handshake is equal to the default SCTP handshake, see Subsection 2.2.2.

Summarized, the handshake and the addresses used by the multipath extensions of TCP and SCTP differ a lot. Chapter 5 discusses the impact of these differences and compares both approaches in a real world Internet setup. Furthermore, the IETF has not discussed the different path management strategies until now. It was not discussed whether this has an impact on the goals of the multipath transfer. However, to design an effective scheduling, a reordering process must be provided for both approaches. This requirement is discussed in the following section.

3.2.3.2 Impact on reliable and ordered transfer

SCTP provides in-order, reliable end-to-end transfer over more than one path, even if it is not provided concurrently [Ste07]. The authors of [IAS06] have investigated the possibility of concurrent multipath transfer and denoted their approach as CMT-SCTP. This CMT-SCTP approach was extended by different mechanisms to ensure a reliable and ordered transfer via more than one path. The performance of the protocol would suffer a lot without these mechanisms. As also [Dre12a] pointed out, at least three optimizations are needed to provide a basic transport service by deploying CMT-SCTP.

- **Split fast retransmission**
  This mechanism forces the SACK mechanism – compare Subsection 2.2.3 – to work on a specific path. This avoids re-ordering issues in the gap handling. Thus, gap blocks only increase duplicate ACK counters if the gap block is smaller than the highest successful acknowledged chunk/message on the path.

- **Congestion window update**
  A congestion control window update should be done independently on each path. That means, a path should maintain a virtual cumulative sequence number per path. This virtual cumulative sequence number can be increased by a virtual acknowledgement, a so-called PseudoCumAck. This PseudoCumAck is caused if gaps of one path are closed by a new cumulative acknowledgment on another path. Therefore, a new valid PseudoCumAck leads to a congestion window growth.

- **Delayed acknowledgment**
  Out of order data is a common problem in a multipath connection [Jun05]. This has

impact on existing mechanisms like, e.g., the delayed SACK mechanism in SCTP. In singlepath SCTP, delayed SACKs are used to decrease overhead traffic. In detail, the acknowledgments will be sent delayed until an additional segment arrives or a timer exceeds. However, an acknowledgment will be sent immediately if an out-of-order transmission is detected. This helps to detect loss earlier and to trigger a fast retransmission more early. In CMT-SCTP this delayed ACK mechanism causes unnecessary overhead. The main idea is to delay all SACKs for CMT-SCTP.

These mechanisms are also part of the individual IETF draft on load sharing for SCTP [ABD+13]. However, CMT-SCTP, as proposed in [IAS06], was not designed for an asymmetric, heterogeneous topology as represented by the Internet. The work of [Dre12a] and this thesis address these issues. In Chapter 5 the deployment in a real world Internet setup will be discussed, in Chapter 6 the required changes in the congestion control will be analyzed and in Chapter 7 an improvement for the scheduling task will be introduced to achieve the IETF load sharing goals [Dre12a].

In contrast to the adaption of SCTP mechanisms in case of CMT-SCTP, the design of the MPTCP extension has to address the complete re-ordering task, with respect to the constraint that an MPTCP subflow must look like a singlepath TCP connection on the wire. The MPTCP design has to address this backward compatibility. Particularly important is that the sequence numbers have to look "normal" on path level. Thus, besides the path assigned sequence number space another overall sequence number (SQN) is required to organize the overall flow. The signaling of the additional SQN is done by additional options. These options organize the so-called data sequence signal (DSS). Every byte gets its own DSS, which is uncoupled from the singlepath sequence number – which will become important later in the scheduling discussion of Chapter 7. This overall sequence number cannot be used for the re-ordering process at path level, but it is necessary for the retransmission process at flow level.

This has a direct impact on the behavior compared to SCTP. It can be a possible benefit to try a retransmission of the same data on an alternative path to free buffer space as will be discussed in Chapter 7. SCTP segments can be retransmitted via an alternative path. Therefore a retransmission in CMT-SCTP via an alternative path causes no additional overhead, whereas a retransmission via an alternative path in MPTCP does. The reason is the per path sequence number of TCP. This sequence number space must be transmitted ordered, even if it is not necessary for the overall sequence number space. Thus, for MPTCP a retransmission on an alternative path always uses a copied segment and, therefore, requires additional resources.

3.2.3.3 Impact on flow control

The receiver side flow control is a good example for a mechanism that should be deployed as a coupled mechanism. If it is not deployed as a coupled mechanism, the flow control can cause a potential deadlock.

Figure 3.5 demonstrates a deadlock for a de-coupled flow control caused by a path failure during a reliable and ordered transfer. In this example, a sender schedules data over different destination addresses, where the first segment (SQN 1) gets lost. The receiver de-couples the flow control and assigns a receiver buffer of three segments per subflow. Until the sender notices the loss, the sender has transmitted as much as the advertised receiver window of the

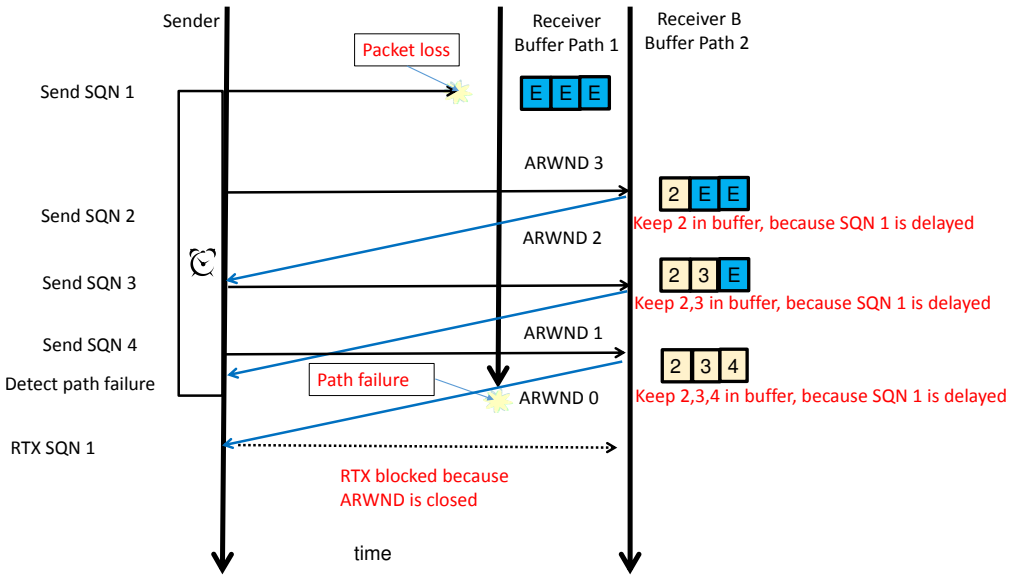


Figure 3.5: Illustration of a deadlock based on a decoupled advertised receiver window

second path allows. Thus, the receiver queue on path 2 is filled with 3 segments and the ARWND is decreased to zero. After the sender detects a failure on path 1 the packet with SQN 1 should be retransmitted on the path 2, but here is no free space anymore, because the receiver must hold the data in its buffer as long as the data cannot be delivered in order to the user. To also use the buffer of path 1 is not possible, because the receiver cannot detect the link error.

There might exist potential solutions to resolve this deadlock. As an example, the receiver could reopen the window of path 2 by the free space of path 1. Anyway, even if the receiver drops all segments from the receiver queue, the receiver cannot be sure that the segment with SQN 1 is under the first 3 packets, which arrive thereafter. The ultimate solution would be to reset the window triggered by the sender, but also in this case middleboxes like proxies can transmit buffered data instead of the missing packet with the SQN 1.

So this situation should be avoided by setting up a coupled ARWND to coordinate the maximum sequence number, which is sent via all available paths. This requirement is valid for both the extensions of TCP and SCTP. While this coupling of the ARWND avoids the deadlock it creates potential performance issues for the scheduling task (see Subsection 7.1.3.2).

3.2.3.4 Implementation dependent options

A protocol standardized by the IETF leaves the implementation details to the protocol developers. Thus, it is not mandatory that every protocol mechanism is specified. Therefore, for every protocol specified in the IETF different implementation variants will exist.

For the load sharing extensions of SCTP and TCP, a scheduler is needed to distribute the user data over the paths, but there exists neither a standardized specification for CMT-SCTP nor for MPTCP. Therefore, already today different variants can be observed in the different implementations. For example, the Linux reference implementation for MPTCP uses an RTT weighted scheduler to distribute the data among the paths whereas the CMT-SCTP reference implementation on FreeBSD uses a Round Robin scheduler. In detail, whereas the FreeBSD

implementation cycles sequentially over all available paths to fill each path with data up to the limit of the congestion control window, the MPTCP Linux implementation always choses the path with the best RTT and sends as much data as the congestion control window allows to send. If the "best" path provides no more free resources, the path with the next best RTT will be chosen. Thus, while every path will be used in the FreeBSD implementation, it is possible in the Linux implementation that only a subgroup of paths is used that allocates and blocks all coupled resources.

3.3 Conclusion

The proposed multipath extensions for TCP and SCTP are quite straightforward. Besides security [Bag11] and middlebox problems, the approach of MPTCP is discussed as fully functional in the IETF. The standardization process for CMT-SCTP is expected to start soon. However, the detailed implementations of these approaches and their deployment have not been systematically investigated until now. The behavior in the Internet is not well analyzed. It is even an open questionif a successful deployment is possible at all is an open question. Furthermore, the impact of the new multipath mechanisms on the transport protocol service is not clear yet.

One important challenge is to analyze if the re-ordering issue can be solved in the scenarios given by the Internet and which impact a limited resource has, like e.g. send or receive buffer. Furthermore, until now it is not clear which effect the new fairness goals [RWH09] have on the ability to achieve the goals of the load sharing extensions and if the expected throughput can be achieved.

In addition, besides the multipath extensions itself the multipath fairness goals must be discussed, because they are not that precise, so that it is possible to interpret them in different ways. The impact of a different interpretation has not been discussed sufficiently so far.

Chapter 4

Testbeds and tools

The development of a protocol starts with an idea to change or improve the way to transfer data. This idea has to be described, evaluated and tested. That is even more important for the standardization process in the IETF, where every standardization draft has to provide a reference implementation that demonstrates the idea in running code. With Multipath TCP (MPTCP) [FRHB13] and Concurrent Multipath Transfer for SCTP (CMT-SCTP) [ABD+13], two different protocol extensions are evaluated in this thesis. CMT-SCTP is discussed in the IETF in the Transport Area Working Group (tsvwg[1]) and MPTCP in the Multipath TCP Working Group (mptcp[2]) of the Internet Engineering Task Force (IETF). This thesis addresses the new tasks of a load sharing extension with the management of paths (see Chapter 5), the fair sharing of the resources (see Chapter 6) and their effective usage (see Chapter 7).

As author of this thesis and as co-author of one IETF load sharing approach [ABD+13], it is important to stay close on the current development state and to work in cooperation with other researchers to provide toolchains that allow a systematic analysis of the new ideas. Efforts that are claimed as *preparatory work* in this thesis are the result of cooperation in implementation, analysis and evaluation. But even if this preparatory work is not the main contribution of this thesis, it should be mentioned, because it builds the basis.

Two possible alternatives are in general available to test new ideas, namely simulation and real implementation. Simulations are suitable to perform a wide range of platform independent experiments with low costs compared to a real testbed setup. However, implementations on real operating systems are necessary to verify the simulation setup, determine relevant parameters and test for real world side effects. Thus, for this thesis different tools and implementations were required:

- A simulation model of the load sharing extensions for TCP and SCTP
- A real world testbed with measurement instances for validation of the simulation model
- Monitoring tools to allow a comparison of the new load sharing extensions

It is important to know that for both load sharing extensions a reference implementation on different operating systems exists. For MPTCP it is Linux and for CMT-SCTP it is FreeBSD. The main cause for these different implementations is the free platform choice of the main authors of both extensions. This free choice is also the reason why parts of the

[1]tsvwg: http://datatracker.ietf.org/wg/tsvwg/charter/
[2]mptcp: wg http://datatracker.ietf.org/wg/mptcp/charter/

implementation differ a lot and there is a need for a common platform in order to compare the results.

However, upon starting this thesis neither a common platform for real systems nor simulations existed for the comparison of load sharing extensions. So, one goal of this thesis was to provide such a common platform for experiments avoiding platform dependent behavior. The following sections describe in some detail the different implementations used to create valid testbeds for protocol evaluation and analysis.

4.1 Simulation model

One major constraint for the simulation environment was the support of SCTP and TCP. There are not that many network simulation tools which provide this support as [Seg12] pointed out. Furthermore, the reasons to choose an open source solution to avoid expensive licensing and to support easier debugging are also true for this thesis as they are for [Seg12].

But the most important reason to choose OMNeT++ [Var12] with its INET framework [Var11] for this thesis was the detailed knowledge of the existing SCTP model and the long term experience with the tool. The SCTP model is a result of different successful research projects with the goal to improve the overall performance of SCTP. Furthermore, the basic SCTP model was validated with the external interface [TRR08] under real network conditions and can be used as a basis to evaluate all other mechanisms and protocols.

4.1.1 OMNeT++

OMNeT++ in general is a discrete event simulation tool [Var05]. A strong benefit of OMNeT++ is its modular design. Thus, an already existing module can be reused or extended easily. The modules can be described by a specific network description (NED) semantic, and stored in a corresponding "NED file". Mechanisms and functions assigned to this module are developed using the programming language C++. Each module can be connected to other modules by "gates". The connecting medium is called connection. Simple modules can be combined to new compound modules in a hierarchic way by applying this principle.

The dynamic event is represented in the OMNeT++ simulation by a message, which can be scheduled to the module itself by "self-messages" or via gates to other connected modules. This way complex networks can be created. A NED file provides the base for an experiment. The experiment parameters can be manipulated in an "ini" file. Here, information about start time, duration or specific module parameters is set.

The output of a simulation can be split into two different result files. The vector files provide a series of parameterized events and their status over time. The scalar files include the aggregation of specific information. The version of OMNeT++ used in this thesis is 4.4.1 and can be downloaded from the project page[3]. But OMNeT++ provides only the core system. The INET framework is required to use a complete existing IP stack, too.

The development support for the OMNeT++ simulation is provided by an integrated development environment (IDE). This IDE is based on the Eclipse platform[4] but provides an individual bundle to support the work on NED files or the analysis of the scalar and vector files. Even if the analysis tools are not suitable to process the results in an adequate manner

[3]OMNeT++ http://www.omnetpp.org
[4]Eclipse http://www.eclipse.org

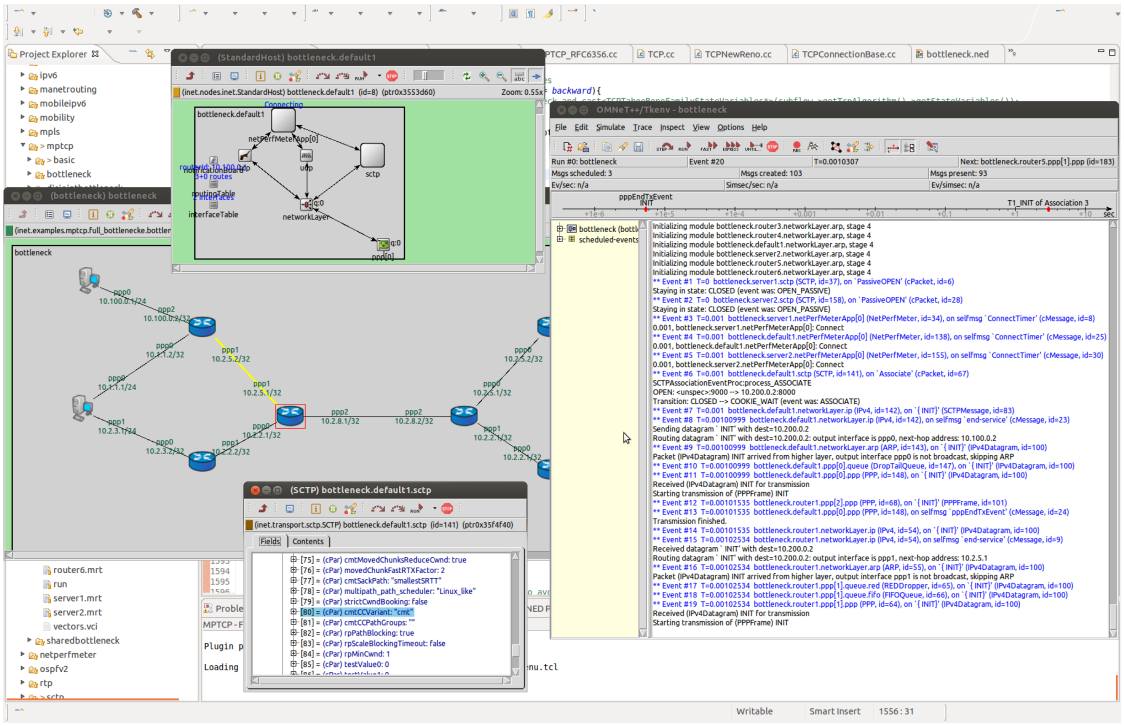


Figure 4.1: OMNeT++ with its INET framework running an MPTCP simulation

for this thesis, the plugins are important for the first initial development steps. In Figure 4.1, the IDE is visible in the background of the OMNeT++/INET graphical user interface (GUI) that is helpful for debugging the experiment setups. Anyway, OMNeT++ should be used without the GUI in complex simulation setups, because it increases the time required for a simulation run dramatically. For this use case the OMNeT++ environment provides the usage from the command line. This command line is also helpful if a distributed simulation pool is used. A distributed simulation was used in this thesis to provide the high number of simulation runs that are required for an evaluation of the fairness and the scheduling mechanisms.

4.1.2 INET framework

OMNeT++ itself provides no model for network communication. There exist different models to extend OMNeT++ with this missing functionality, even if most models are very specialized, like for example MixiM [KSW+08] for wireless communication. The INET framework [Var11] provides a more common model with all layers of the OSI reference model. The existing model provides protocols like Ethernet or Point-to-Point (PPP) protocols on the link layer, IPv4 and IPv6 on the network layer and SCTP, UDP and TCP on transport layer. Furthermore, it supports the development of tools on the application layer that are used in this thesis as interface for the traffic generator.

Figure 4.1 shows an experiment using the INET framework in the OMNeT++ GUI. The topology figure of the screenshot in Figure 4.1 shows a simple experiment and gives an idea of the graphical interface. Furthermore, different module windows are shown which are used

for parameter configuration or debugging.

The original INET framework comes without any load sharing support. The main work to extend SCTP with the CMT-SCTP features was done in [Dre12a] and was extended by additional functionality in this thesis. In Mai 2014, first code segments were committed to the main developer tree of the INET framework and will be provided in the next official INET framework version. The MPTCP model as extension was created exclusively for this thesis on top of the existing TCP model.

4.1.2.1 Enhancement of the INET TCP model

Similar to the SCTP model, a TCP model exists in the INET framework. However, the model of TCP was rudimentary and error-prone. In a first step, the old TCP model was extended by a basic SACK mechanism [RDB+10]. These changes are available as new default TCP model in the original INET codebase. However, a series of basic experiments for this thesis showed that these changes are not sufficient, and even worse, parts of the TCP module did not show the expected results. The most important extensions to the core TCP module are discussed in the following.

Singlepath congestion control
One major challenge was the detection of the issues with the singlepath congestion control that showed an erratic behavior in an experiment setup with a competing singlepath SCTP flow. TCP in competition to itself or alone on the link showed an acceptable behavior. But here only the good case was considered without any problems on the link. Furthermore, because both TCP flows reacted on network events – like RTT and error rate – in the same wrong way. This led at first sight to the same "fair" behavior. However, the issues with the existing implementation in the INET framework get visible when an alternative SCTP flow comes in that implements the congestion control in an alternative way[5]. Figure 4.2 demonstrates the TCP flow behavior with the OMNeT++ IDE plot plugin. The TCP flow is repressed over time by the SCTP flow and prevents the expected fair capacity sharing of the link. Different issues in the retransmission, fast retransmission and window calculation have been identified and fixed to get the result as expected and used in the next chapters.

Scaling option
There was also an issue with the scaling option of TCP [JBB92] in the INET version that was used as basis for this thesis. Even if the scaling factor was exchanged during the TCP handshake, the factor was not used to calculate the advertised receiver window (ARWND) as expected. Furthermore, there existed a calculation issue in the retransmission timer setup that was not in line with the RFC6298 [PACS11].

Send buffer
In the current INET framework, TCP can only be parameterized to send a fixed number of bytes. Thus, to ensure a saturated sender over time was not possible. Here a mechanism was needed to ensure that always enough data is stored in the send queue. In the context of the work of [Seg12], the author implemented a first initial version to support this kind of sending, even if it was not usable in complex setups, because the send queue size was not configurable

[5] And this implementation was also tested against a real world implementation.

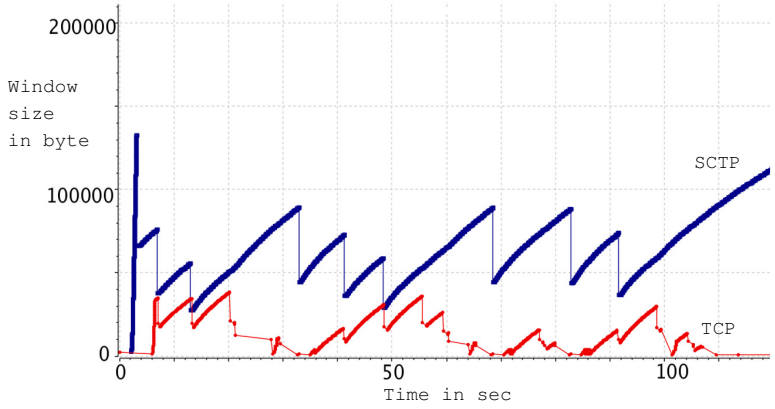


Figure 4.2: Old congestion control window (created with OMNeT++ plot tool)

in this approach. Thus, the mechanism was designed in a way that it did not consider any limitations of the send buffer. A complete redesign of this functionality was necessary to support the same send behavior as provided by the SCTP module [Dre12a].

4.1.2.2 MPTCP

The main extension for the TCP model was the development of the MPTCP module in a way that TCP was affected only to the minimum possible extent. It was a main requirement for the implementation to allow a strong separation between the core TCP module and the MPTCP extension. The created code is separated in eight main C++ classes and is comparable in functionality with the kernel implementation on Linux done as part of the thesis in [Bar11][6]. Even if the MPTCP module was designed independently from the TCP model, the parameters have to be set by the TCP module, because it is a TCP connection, even if the MPTCP extension is used. Thus, the TCP model provides new parameters in the INET framework to control the MPTCP extension. Like for SCTP [Dre12a], a parameter *cmtCCVariant* exists in the NED file that enables the MPTCP extension with a specific congestion control. Table 4.1 gives an overview of the available congestion control algorithms. Furthermore, there exists a parameter that is called *multipath_Scheduler* and controls the factory method pattern [GHJV95] for the scheduler instance in the MPTCP implementation. The factory method pattern supports an exchangeable codebase that can be changed dynamically for the needs of an experiment. Currently implemented are Round Robin and Weighted Queuing (for more details see also Subsection 3.2.3.4). Furthermore, in the MPTCP extension of the INET framework elements were implemented that are not part of the IETF specification. Here, in particular the mechanisms to avoid buffer blocking are addressed. These mechanisms are explained and discussed in more detail in Subsection 7.1.3.1 and in [RPB+12]. The new OMNeT++/INET parameters *multipath_opportunistic_retransmission* and *multipath_penalizing* were introduced to enable the implemented mechanisms. Furthermore, the ConSN approach is configurable with *multipath_consn* (see Subsection 7.5.2).

Table 4.2 shows a survey that is based on the work of [Ear14] to compare the result of these

[6] [Bar11] also created a comparable codebase for the Linux kernel, but described the required functionality in C under the constraints of the kernel development.

Value	Description
off	Singlepath congestion control is based on the parameter *tcpAlgorithmClass*
like-mptcp	Coupled congestion control LIA (explained in Subsection 6.2.2)
like-olia	Coupled congestion control OLIA (explained in Subsection 6.2.3)
cmtrpv2	Coupled congestion control RP-MPv2 (explained in Subsection 6.2.1)
cmt	New Reno (see Subsection 2.2.4.1) congestion control for every subflow

Table 4.1: Parameter values for *cmtCCVariant*

Institution	UCLouvain	Swinburne	Anonymous	Citrix	Thesis
Platform	Linux	FreeBSD-10	Commercial	NetScaler	OMNeT
MP_CAPABLE	Yes	Yes	Yes	Yes	Yes
MP_JOIN	Yes	Yes	Yes	Yes	Yes
#subflows	32	8	no limit	6	2^{32}
DSS	Yes	Yes	Yes	Yes	Yes
DATA ACK	4 bytes	4 or 8 byte	4 or 8 byte	4 or 8 byte	4 bytes
Data seq num	4 bytes	4 or 8 byte	4 or 8 byte	4 or 8 byte	4 bytes
DATA_FIN	Yes	Yes	Yes	Yes	Yes
ADD_ADDR	Yes	No	No (never)	No (never?)	Yes
REMOVE_ADDR	Yes	No	Partly	Yes	No
Sharing	shared, RTT	shared	active/back	active/back	shared, RR, RTT
Handover	Yes	No	Yes	Yes	No
Coupled CC	Yes	No	No	No	Yes
Other CCC	Yes, OLIA	No	No	No	Yes, OLIA, MP-RP-v2

Table 4.2: Implementation overview in comparison to other implementations

implementation efforts with the other existing MPTCP implementations. Most important in this survey is the open and usable reference implementation of UC Louvain on Linux [Ins13], which is also discussed in more detail in [Bar11] and provides the Linux reference implementation of the IETF. It should also be mentioned that there exist official implementations for productive systems like the NetScaler of Citrix [Man10], even if the access to the implementation is restricted.

4.1.2.3 Enhancement of the INET SCTP model

This thesis uses the existing SCTP implementation in the INET framework to limit the implementation work. The SCTP module was introduced by [RTR08] and is up to now the default SCTP module in the INET framework. However, in cooperation with the author of [Dre12a], this module was completely redesigned and extended with CMT functionality and mechanisms to support NR-SACK, Chunk Rescheduling as well as Buffer Splitting. With respect to the SCTP module, this thesis just focuses on the features which are important for

the following chapters and differ from [Dre12a].

The first step was to extend the existing simulation module with the coupled congestion control OLIA (see Subsection 6.2.3). The SCTP model initially provided the same parameters for *cmtCCVariant* as TCP (see Table 4.1), but only with the support of LIA and RP-MPv2. In the second step, the scheduler system was redesigned with the goal to support a parameterization similar to the MPTCP implementation. Thus, there exists a comparable *multipath_Scheduler* parameter for SCTP.

4.1.2.4 Enhancement of other models

The NetPerfMeter simulation model was created in [Dre12a] and was used in the preparatory work for evaluation and testing [BRW+13, DBRT10, DBPR10a, ZDB+10, DABR12b]. A first detailed description was done in preparatory work of [DBPR10b]. As an independent module, NetPerfMeter can be integrated in the INET stack and can act as an active sender or as passive receiver, depending on the parameterization. NetPerfMeter is used in this thesis as a traffic generator to simulate a saturated sender. A more detailed description of the parameters and design is given by [Dre12a]. For this thesis, the NetPerfMeter module was extended to support the send process of an MPTCP connection, in particular to create new messages in time before the send queue runs empty. The MultihomedFlatNetworkConfigurator is used to configure the endpoints. This configuration tool was also published as preparatory work of this thesis and is also described in more detail in [Dre12a]. For this thesis some adaptation work was required to keep it operational on the codebase used for the MPTCP implementation. This was necessary because the development process on the main branch of the INET framework proceeded during the thesis work.

4.1.3 SimProcTC

An important tool for this thesis is the simulation processing toolchain (SimProcTC). This toolchain was initially created by Thomas Dreibholz [Dre12b]. A brief introduction is also given by [DZR09] and a more detailed one in [Dre12a]. As [Dre12a] describes, this tool chain provides a framework to organize the modules CMT-SCTP, NetPerfMeter and the MultihomedFlatNetworkConfigurator for a quantitative performance analysis. A task during this thesis was to extend this toolchain with the required functionality for MPTCP.

4.1.3.1 Basic scenario setup

As [Dre12a] points out, a simulation may consist of thousands of individual runs. The SCTP-based RSerPool [LOTD08] framework [Dre07] is used as part of the SimProcTC to organize the distribution of such a high number of runs using a pool of distributed servers. Thus, it is possible to address one service on many different endpoints at the same time. The pool used in this thesis provide 44 parallel sessions. That means that in total up to 44 simulation runs were computed at the same time.

The toolchain uses GNU R [R D12] to plot the figures. A point in the graphic represents the average value of a minimum of 50 simulation runs for every parameter combination. Furthermore, the confidence interval is plotted around this point. The confidence interval is set to 95% for all experiments. Furthermore, each of these minimum of 50 runs has a minimum duration of 120 s (without transient phase) or 400 s (including the transient phase).

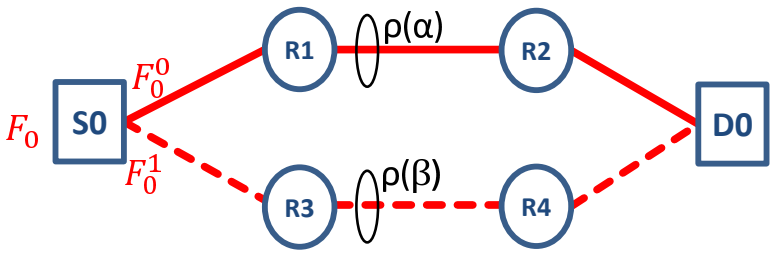


Figure 4.3: Test scenario under perfect conditions

4.1.3.2 Baseline experiment

A simple multipath connection setup was created to demonstrate the results of the toolchain and, furthermore, to demonstrate the expected behavior of the implemented extensions of MPTCP and CMT-SCTP in INET in a simple test case. Figure 4.3 shows two endpoints $S0$ and $D0$. Both endpoints are connected via two access links. $S0$ sends a flow $F0$ by using the links α and β. The capacity $\rho(\alpha)$ is constant with 800 Kbit/s, the link capacity $\rho(\beta)$ varies from 200 Kbit/s to 9 Mbit/s. Thus, $\rho(\beta)$ is set to more than 10 times the capacity $\rho(\alpha)$ in the extreme. Three experiments were configured for this setup by using MPTCP in a first, CMT-SCTP in a second and singlepath TCP via β in the third experiment. Every connection used the path β for the initial handshake.

The results of the simulation are shown in Figure 4.4. All three curves are showing the expected results. Curve #3 illustrates the performance of the singlepath flow depending on the capacity $\rho(\beta)$. Curve #1 and #2 show the result curve as a sum of the capacity $\rho(\alpha)$ and $\rho(\beta)$[7].

4.2 Real world testbed setup

An evaluation with the simulation environment is not sufficient, as for example the preparatory work in [ADB+11] shows. Especially issues caused by the CPU load are not sufficiently addressed in the current simulation model. Thus, even if the protocol mechanisms show perfect results in the simulation setup, the mechanisms also have to be evaluated in a real world setup. Furthermore, it should be kept in mind that the simulation only represents the network as good as the parameterization used to set the experiment. Whether this network represents the complete picture is not ensured as this thesis will demonstrate, too.

The reference implementation of CMT-SCTP was extended in this thesis for example by the coupled congestion controls (see Section 6.2) to investigate the mechanisms also in real world setups. Anyway, for the measurement itself, endpoints were configured to run the reference implementation of MPTCP on Linux and CMT-SCTP on FreeBSD. Thus, all hosts had a dual boot installation for both Linux (Ubuntu) and FreeBSD. For the endpoints, state-of-the-art computers were used, in this case Dell Optiplex 760 with 4 GiB RAM and Intel Core™ 2 Duo CPU E8600 3.33 GHz. Measurements in a real world setup are separated in test setups in the Internet and a local testbed setup. The local testbed setup is mandatory to create controllable and repeatable experiments, which are not possible in the Internet. The link

[7]The overall performance is of course reduced by the protocol overhead. Thus, only the goodput is measured.

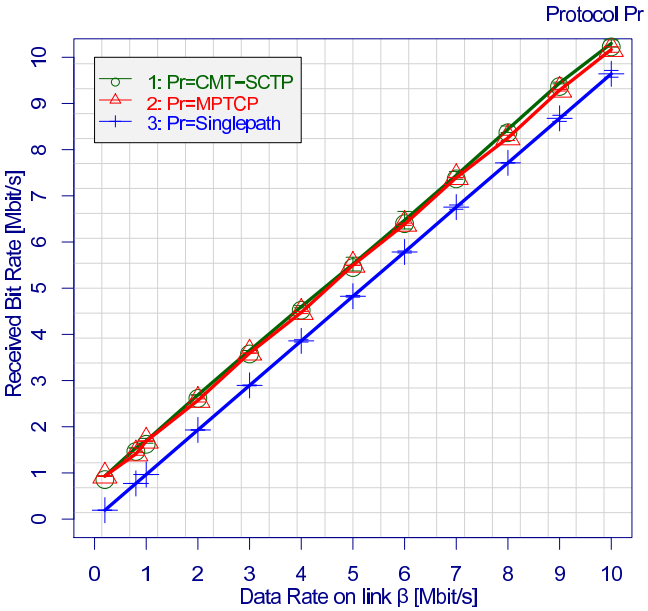


Figure 4.4: Testbed experiment with 0 ms delay

characteristics of the local testbed setup were manipulated by using Dummynet [BDRF11]. The NetPerfMeter toolchain for Linux and FreeBSD was installed on the endpoints. The NetPerfMeter toolchain for Linux and FreeBSD was introduced as a tool for multipath protocols as preparatory work in [DBAR11a] and is comparable to the NetPerfMeter module of the OMNeT++/ INET framework. It also produces vector and scalar files as outcome, which can be plotted with the existing toolchain of the simulation environment.

Chapter 5

Revisiting path management

This thesis is based on an initial situation, where two IETF load sharing extensions — MPTCP [FRHB13] and CMT-SCTP [ABD+13] — existed for the transport layer. Both approaches are alike, with similar challenges (see Subsection 3.2.3) and goals (see Subsection 3.1.1) and for both the path management task is essential. Maximum network resources can only be used if the path management works as expected. The initial measurement tests with the currently deployed path management strategies show impressive results. For example, the experiment described in [Chr13] demonstrates that it is possible to transfer the data volume of more than one DVD per second between two endpoints. So far, for the load sharing connection in this experiment setup is the fastest TCP connection ever established. Furthermore, the research publications [DBRT10] and [RPB+12] show nearly perfect results in their simulation models or emulated network setups. Although the authors identified possible problems in asymmetric, heterogeneous link setups, they also provided potential solutions to decrease the effects. So far, the research community identified no challenges for the path management with impact on the performance. This chapter will change this view.

Until now, the path management was not identified as performance limiting factor and the default test model used in the experiment setups can be identified as reason for this. The test model consists of 1 to n endpoint pairs, where one endpoint acts as client and the other one as server. Furthermore, every endpoint pair is connected by 2 to m links. It is important to note that the test model used always ensures that the transport protocol extensions are able to use at least one path per endpoint address pair.

Many publications are based on this simple multipath model. One example is the research paper of [RBP+11]. In this work the authors show a possible improvement in a datacenter by using the load sharing extension of TCP. In the context of fairness the publications [BDAR12b, DBPR10a, AWDR14, ADB+11, VBOMT13a, DABR12b] can be mentioned and, furthermore, in the context of scheduling all research papers, e.g. [SGTG+12, RPB+12, DSTR10], apply this model.

All results lead to the assumption that the extensions for load sharing of TCP and SCTP provide the same results under the same conditions in a nearly perfect way. These results can lead to a statement that it does not matter for an application which protocol – TCP or SCTP – will be used for load sharing. But, these protocols are not identical and provide a completely different architecture and philosophy. So far it was not discussed if this difference has an impact on the ability to achieve the load sharing goals. This chapter, therefore,

addresses this question which is important for the application.

Furthermore, first real world measurements show that the use of more than one link is not always a straightforward task and needs a more detailed discussion. The efforts spent in this thesis led in the end to the first intercontinental multi-homing testbed which supports both IETF extensions on the transport layer in a comparable setup in the Internet. Here a lot of initial work and topology analysis were necessary to achieve comparable results and to demonstrate the impact of the design on real Internet topologies.

The ability to support load sharing on the transport layer in every reasonable connection setup was not discussed so far, because the focus was on the design of the load sharing extensions themselves. This thesis focuses on further constraints which even have a direct impact on the ability to support the load sharing feature. The constraints identified in this thesis are:

- **Layered protocol architecture**
  As discussed in Chapter 2, the currently used network model is a layered model. In this model the transport protocol is not aware of any topology information. It only has one interface to the network layer to exchange data. There exists no interface to exchange link information. The network layer also does not know about the existence of links or their status. The network layer protocol is only aware of the configured or assigned addresses. Therefore, for a network layer protocol, e.g. IPv4, the local endpoint is multi-homed and provides potentially the chance for load sharing if more than one IP address – with a scope beyond endpoint borders – exists. But so far nothing is defined about the mapping of the addresses to the available links and, furthermore, nothing is defined about the way a transport protocol has to use this information to establish an end-to-end connection.

- **Backward compatibility**
  Subsection 3.2.3 explains that the load sharing extensions MPTCP and CMT-SCTP are based on existing singlepath protocols. Thus, the extensions provide a design where the application is not aware of the load sharing feature. All interfaces and descriptions of an end-to-end connection remain the same from application view, although the protocols support the multipath extensions. Therefore, the application must be able to address a multipath and a singlepath connection in the same way. As discussed in Subsection 3.2.3, TCP uses a so called 4-tuple to identify a connection by setting the source/destination IP and the source/destination port. SCTP provides a list of alternative address pairs for a specific port to address an association.

  The design of the extensions requires a network transparent behavior, too. The middleboxes (see Subsection 2.3.1) or any other observer should be unable to identify a different behavior as for a TCP or SCTP connection.

This chapter discusses the path management as a task with:

- Complex interaction of different protocol layers.
- Challenging mix of specifications, which are optimized for a singlepath network setup.
- The need to develop new mechanisms for multipath support.
- A possible impact on every network component.

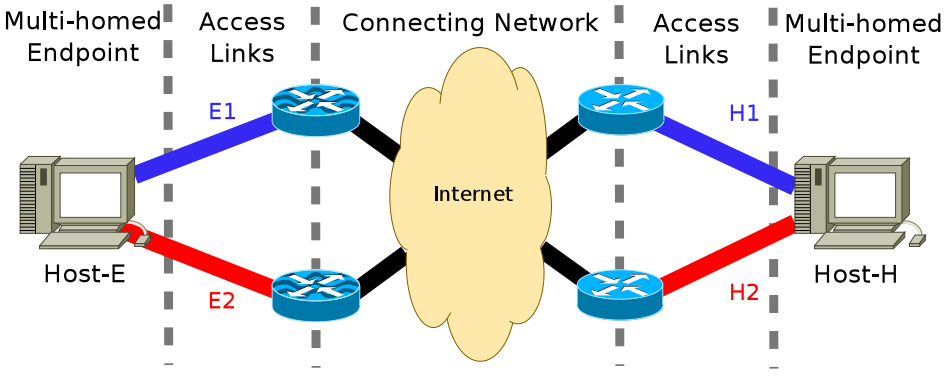


Figure 5.1: General setup of a multipath scenario

The behavior of the mechanisms in a real network is discussed by using the components illustrated in Figure 5.1. In this figure, a simplified multipath topology is illustrated consisting of the main roles in a multipath connection setup. The endpoints at the network borders represent the source and the sink of the connection. The next points of interest are the access links, which are possibly under the control of the sink or the source, even if this is not the case in most scenarios. The last major component is the connecting network, normally the Internet for Internet protocols. In the worst case the customer has no control over these resources and the worst case is the default case.

This and the following chapter focuses only on scenarios with a static topology. Neither standard SCTP [Ste07] nor CMT-SCTP [ABD+13] provides any mechanism to add or remove addresses – as described for example in RFC5061 [SXT+07][1]. The dynamic path management aspects are out of scope of the following discussion.

The sections are structured as follows. First, the constraints for the load sharing extensions of TCP and SCTP are discussed to understand the resulting requirements and the degrees of freedom for the design. There is the need to understand the impact of the layered protocol architecture and backward compatibility. In the second step, the impact of the network layer on the path management is discussed. Altogether, this discussion provides insights which protocol manages load sharing in the Internet in a better way.

5.1 Path management

Compared to the multipath setup, it is easy to manage a functional connection "path" in a singlepath environment. Here, only one "path" to the destination exists. For the transport protocol it does not matter if there are routers in the network or how these routers forward the data. There is always a strong relationship between the interface, the IP address and the "path". This dependency allows the utilization of the resource by the application for both SCTP and TCP immediately after the initial handshake is done (on the basis of one destination address). For concurrent multipath transfer this task is much more complex.

The unambiguous assignment of the address, path and link is dissolved. There exists no common rule how to map a usable "path" of a transport protocol – whatever the definition

[1]The RFC5061 exists as an extension for SCTP on base of RFC4960 with the goal to support the add/remove of addresses, but RFC5061 is not an explicit part of the CMT-SCTP approach.

is – to a physical link. For an observer outside the system this issue might be a surprising challenge because it seems obvious to use a path per physical link. But it is not obvious for the transport protocol in the Internet. By using the default Internet protocol (IP) the network layer only has an abstract view on the network by providing just an addressing and forwarding functionality. Thus, if more than one network layer address is provided, it is completely under the control of the network layer to map these addresses to the physical links. Therefore, this mapping is out of control of the transport layer. The impact on the design of a multipath protocol and a successful deployment is underestimated.

5.1.1 Constraints

The load sharing extensions of TCP and SCTP were not designed from the scratch. Therefore, the new mechanisms for connection setup, connection usage and capacity sharing have to adapt to the existing conditions and requirements. The constraints are based on the specifications for singlepath TCP and SCTP and their multipath extensions. It should be kept in mind that the requirements for the singlepath specifications are the IETF consensus. Therefore, modifications to these existing specifications are not possible, even if it would make sense considering the new conditions of the multipath goals. However, the existing specifications are extendable to achieve the new goals.

5.1.1.1 General

For MPTCP the interaction of the relevant specifications is more complex than for CMT-SCTP. TCP is defined in RFC793 [Pos81b] (see Section 2.1). The general load sharing design approach for MPTCP is defined in RFC6182 [FRH+11] and addresses the architecture as an extension of RFC793 with the application and network compatibility discussed in more detail in RFC6824 [FRHB13] (see Subsection 3.2.3). RFC6824 also includes the general protocol design and the additional signaling for connection setup and the new sequence number space (see Subsection 3.2.3.1 and 3.2.3.2). Furthermore, RFC6824 addresses the issues with middleboxes – for example firewalls [Fre00] – and the need for a new congestion control to achieve fairness at the bottleneck (see Subsection 3.2.2.1). The RFC6356 [RHW11] addresses the congestion control issue and provides a very first solution (see Subsection 6.2.2). The RFC6824 and RFC6356 are dependent on other RFCs, e.g., the RFC5681 [APB09] for the singlepath congestion control and also on RFC2018 [MMFR96] for selective acknowledgment (SACK). Furthermore, RFC5681 is an update of RFC2581 [APS99] and RFC2001 [Ste97]. The RFC2001 is based on RFC1122 [Bra89] which itself is root for some more dependencies e.g. the retransmission timeout calculation in RFC6298 [PACS11]. Also, another branch of RFCs is based on the backward compatibility defined in RFC6182 and RFC6824. A general overview of the TCP load sharing extension is provided in RFC6897 [SF13]. The RFC6897 also describes clearly that the multipath design has to manage the data transport via different subflows automatically, i.e., without any information from the application or other upper layers.

As compared to the load sharing extension for MPTCP these dependencies are less complex for CMT-SCTP. The RFC4960 [Ste07], which describes the core transport protocol, is the base for most of the other specifications.

The RFC4960 is an update of the RFC2960 [SXM+00] that provides the multi-homing feature (see Subsection 2.2.2) but no load sharing function. The load sharing is addressed

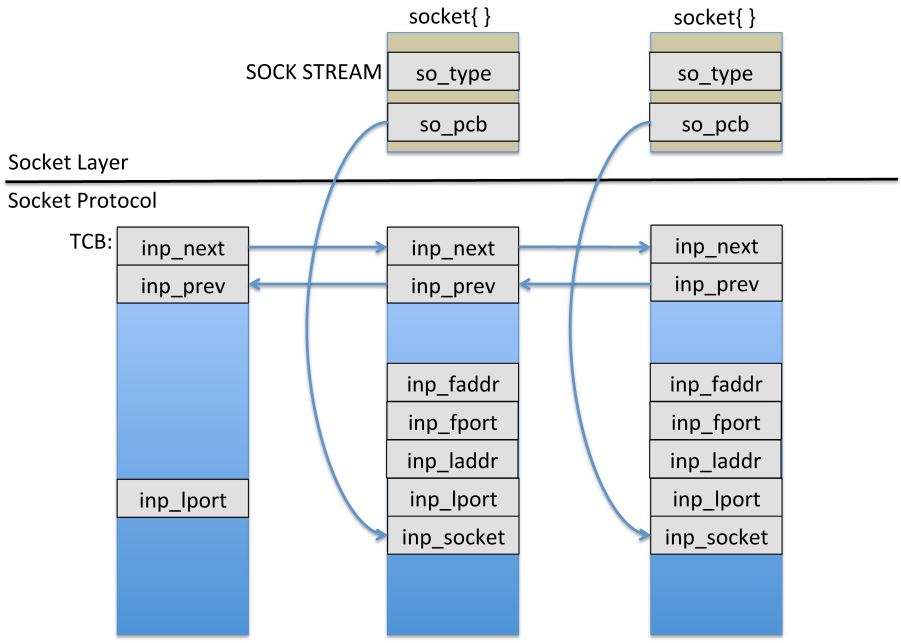


Figure 5.2: Simplified illustration of a TCP TCB list with relation to a socket interface

in an IETF Internet Draft called "Load Sharing for the Stream Control Transmission Protocol" [ABD+13].

It should not surprise that the existing protocol specifications establish a frame for the load sharing extension where the degrees of freedom for an implementation are reduced. This legacy support is a limitation for MPTCP and CMT-SCTP. It becomes for example visible in the identification of an MPTCP and CMT-SCTP connection and the paths used for the multipath transfer. As discussed in Subsection 2.2.1, at least an information 5-tuple (source/destination address, source/destination port and protocol identification) is required to address a connection on the transport layer. A query with this information 5-tuple always provides a unique identifier in a protocol control block (PCB).

Furthermore, as [Koz05] explains, each connection has to maintain data about its status and unique identifier separately. Therefore, a specific description structure called transmission control block (TCB) has evolved in different RFCs over time as element of the PCB to describe a TCP connection. Figure 5.2 gives a simplified illustration of the TCP TCB in the context of the TCP socket connection, which is also used by the Berkeley Socket API [WS95], the de-facto standard interface to the transport protocol on most operating systems (see Subsection 2.3.3).

The connection description was introduced in rudimentary form in the first TCP RFC [Pos81b]. However, it was illustrated first in detail in [WS95] as a doubly linked circular list. The list has always a fix start point which allows a direct access to the root element of the list, e.g., as root element for a connection lookup. The TCP TCB is used in a singlepath environment to allocate singlepath connections to the endpoints with minimum information. This information structure includes source address (*inp_laddr*), destination address (*inp_faddr*), source port number (*inp_lport*), destination port number (*inp_fport*) and also other informa-

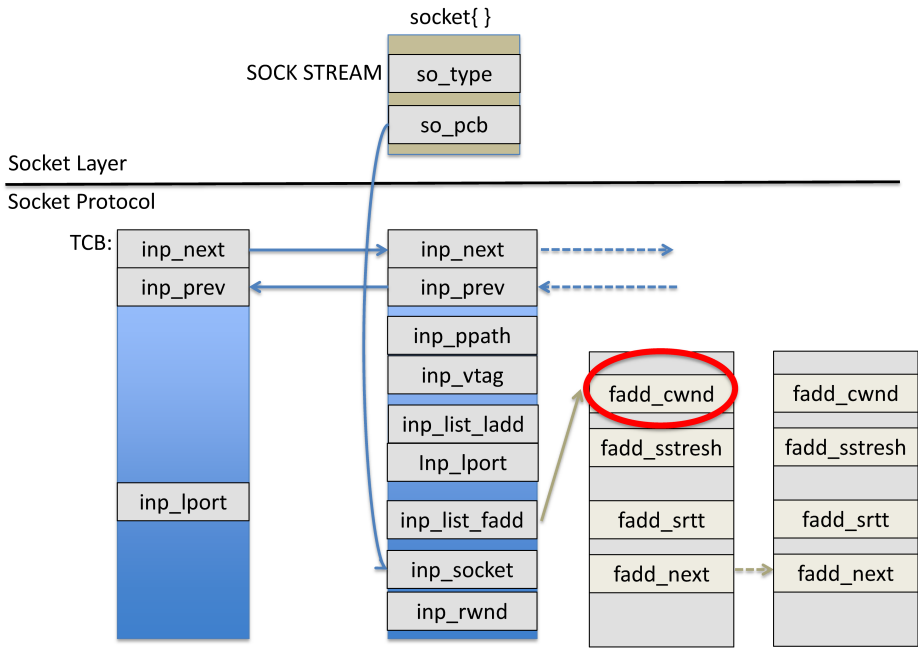


Figure 5.3: Simplified illustration of an SCTP TCB list with relation to a socket interface

tion assigned to a connection like the status of the state machine [Pos81b] or the necessary information for the flow control as described in RFC6582 [HFGN12] for TCP. The variable names are encoded mostly by *inp* for Internet protocol, *l* for local and *f* for foreign.

It is important to understand that the TCP TCB is a result of discussions done over the years considering different constraints and goals. The constraint of "backward compatibility" results in the fact that this structure cannot be changed, only extended. Clearly, a change of this structure would not lead to an extension of the existing protocol, rather to a specification of a completely new protocol and is, therefore, not possible.

Figure 5.2 illustrates that for each socket (so) – as handle for the interface in the application – a socket protocol control block (*so_pcb*) exists that points to the assigned transport protocol. This connection entry itself is part of the list that describes the TCB and, therefore, the complete TCP connection. This relationship is characterized in a generic part assigned to the general socket layer structure and a protocol specific interpretation by the TCB for TCP.

In SCTP [Ste07] the TCB also exists as a list of SCTP associations, but the structure is more complex. However, the basic idea is similar to TCP. The RFC4960 [Ste07] defines a recommended structure of parameters, which is useable as basis and gives the developers more detailed information about the standard (see Figure 5.3). SCTP provides, in contrast to TCP, a structure to identify more than one subflow. Thus, the existing structures of SCTP can be used. This leads to the fact that parts of the path information of SCTP – like the congestion window (CWND) – are given for each destination address (see entry *faddr_cwnd* Figure 5.3) as a unique value.

In contrast to SCTP, the TCB for the default TCP provides no room to identify more than one subflow. Thus, the existing TCB structure has to be extended. Every new entry must represent a complete TCP connection, because every subflow has to interact on the path

like a "normal" TCP connection (see Subsection 3.2.3). Therefore a subflow entry is similar to a default TCB TCP connection structure[2].

This historical relationship of status variables to addresses has an impact on the behavior of SCTP and TCP with the additional load sharing extension, as explained in the following sections.

5.1.1.2 MPTCP

Until now, the general discussion about existing constrains leads to a subflow description as a copy of a default TCP connection. The relation between the MPTCP connection and a subflow is defined in RFC6182 Subsection 1.2 [FRH+11].

> Subflow: A flow of TCP segments operating over an individual path, which forms part of a larger Multipath TCP connection.

So it has to be understood that a subflow – organized as a default TCP flow – is defined as all segments using a specific path between two endpoints identified by a specific source/destination IP address pair. Therefore, a path is only a logical entity defined on the transport layer and does not imply the usage of a specific access link as this mapping is done by the routing layer (see Subsection 1.2 of RFC6182 [FRH+11]).

> Path: A sequence of links between a sender and a receiver, defined in this context by a source and destination address pair.

Every individual source/destination address combination in MPTCP identifies a new usable path, because every combination is required to ensure the support of the goal defined in [FRH+11]:

> Multipath TCP is primarily concerned with utilizing multiple paths end-to-end, where one or both of the end hosts are multi-homed.

With this definition a complex requirement is addressed. The TCB describes a multipath connection created by an application with the normal information 5-tuple. Furthermore, the TCB includes the list of the subflows. In this subflow list every entry provides the same information and functionality as a default singlepath TCP entry. An entry to this subflow list has to be added if the source/destination address pair does not exist so far, even if one address of this new pair is already a part of an existing subflow. The support of multiple paths to one destination address – as requested in the scenario with one multi-homed host – requires a handshake to every destination address or from every source address. Clearly, this definition is the reason for the full cross table of all address combinations. The cross table is a result of the singlepath TCP constraints and the multipath goals and these constraints create a pool of multiple potential paths without any knowledge, whether an address combination is valid or not. If a new address pair is identified or signaled – for example by an ADD_ADDR option (see Subsection 3.2.3.1) – the resources for this path have to be reserved and the additional "Join" Handshake (see Subsection 3.2.3.1) has to be performed.

[2]Subflow information includes information like sequence number, window size, etc..

5.1.1.3 CMT-SCTP

The multi-homing feature is already available in standard SCTP defined in RFC4960 [Ste07], and later on provided as basis for CMT-SCTP. Thus, the RFC4960 and the CMT Internet draft give exact information about how SCTP identifies a specific association and how SCTP addresses usable paths between two endpoints. For the load sharing extension this path identification and definition of an association is important and also every detail in this context, because it describes the usable resources. The definitions are presented in the following to give more detailed information for the discussion. First, the RFC4960 provides the path definition:

> Path: The route taken by the SCTP packets sent by one SCTP endpoint to a specific destination transport address of its peer SCTP endpoint.

In this definition the sender is not mentioned. This strict focus on the destination addresses is visible throughout the entire RFC and is also the reason for the mapping in the SCTP TCB (see Figure 5.3). It is, therefore, not surprising that there exists a multi-homing definition with the same intention.

> The endpoint is considered as multi-homed if there is more than one transport address that can be used as a destination address to reach the endpoint.

The authors of the RFC4960 have pointed out their intention in the Subsection 13.1. Here, they recommend the TCB parameters for a default SCTP implementation. In this proposal they define one congestion window ($CWND$) for every destination address. Applying load sharing based on SCTP, as initially described in the CMT-SCTP approach [ABD+13, IAS06], is a straightforward extension and, therefore, CMT-SCTP uses the existing SCTP destination based path management strategy, too. Thus, the association establishment – including the setup of more than one path – and the path definition are given by the original SCTP.

5.1.1.4 Conclusion

The core protocols of MPTCP and CMT-SCTP have a different history and design philosophy which leads to different definitions of "path". CMT-SCTP provides one path per destination address, whereas MPTCP manages a path per source and destination address pair. This path definition has a direct impact on how subflows are distributed over the network. So far no mapping rules exist to assign an address pair to a specific physical link. In the layered model this is a task of the network layer. The next step is to analyze the ability of the network layer to support the needs of this mapping and the impact on the deployment of the load sharing extensions.

5.1.2 Network layer

First of all it should be clear that routing is not a task of the transport layer. The transport protocol is an end-to-end protocol and is not aware of any infrastructure information except the information which is exchanged, estimated or measured during a connection by the transport protocol itself (see Chapter 2 or compare with the general fairness discussion in

Subsection 3.2.2). However, the load sharing extensions of TCP and SCTP are unable to use the resources without any further routing adjustments. The reason is simple: the data has to reach the destination via different paths on different physical links to exploit the idea behind load sharing. The transport protocol depends on the network layer information and identifies a "path" by the IP addresses. However, the IP addresses do not ensure that the paths are mapped correctly to the logical and/or physical infrastructure. Most operating systems are not aware of the need to support multiple interfaces at the same time [BS11].

The default operation for protocols on the network layer is based on singlepath behavior in the Internet. For IPv6 there also exists a standardized way to support multipath, if the infrastructure is supporting this [CA11], but this support is not the default case. However, the routing process is based, from a historical point of view, on the assumption that only the destination address is required to forward a packet through the network [Pos81a].

Clearly, in many operating systems the routing focuses only on the destination address. The source address is only used to support the bi-directional connection. Thus, every mapping decision from path to link is made on the basis of the destination address. This is also denoted as destination-based routing. The rules for the destination-based routing are placed in a specific routing table. This routing table provides an assigned gateway per endpoint or network address. This gateway information leads to a physical interface and a usable link. Thus, a routing table consists of information about at least the reachable network, the metric and the next hop. If there exists no information for a specific network or endpoint, a specific kind of routing information is considered denoted as default route. A default route describes the rule for the routing decision if no other rule exists. In IPv4 the default route entry starts with the unspecified network address 0.0.0.0. The destination-based routing has a direct impact on the multipath extensions of TCP and SCTP.

5.1.2.1 Routing and MPTCP

A source/destination address pair defines a path in MPTCP and the paths are organized as a list in the TCB of TCP. A handshake is required for every subflow to exchange data via the new subflow. The information used to create the handshake message flow depends on the implementation and the routing setup in the operating system (see Subsection 3.2.3.1). Thus, the initial handshake depends on the implementation and configuration of the endpoint. The handshake variants can differ in the IP addresses or interfaces used.

The default behavior of most operating systems is not supportive for the MPTCP extension. In this case, without any further work, the goal of better network utilization cannot be achieved by deploying the load sharing extension. Of course, theoretically MPTCP is also able to establish more than one subflow in default system setups, but it is necessary to understand that these subflows are not mapped correctly to the physical paths in every case, especially if only the default route is used.

If only the default route exists, the routing always uses the same gateway and the same physical access link. Therefore, no real load sharing will be achieved, because the subflows are not mapped to different physical links. A routing table setup to differentiate multiple physical access links is required. This is possible by a manual configuration of appropriate routing rules on the basis of each destination address. But a manual manipulation is not really practical, because it creates the requirement for a manual routing entry for every connection partner.

The IETF Internet draft from the main authors of the MPTCP extension [BPB11b] claims

the definition of a more general approach. From their point of view a switch to source-based routing provides a universal implementation approach for MPTCP. The idea of source-based routing is similar to destination-based routing with the important difference that there exist additional routing tables per source address. Many OSs support this routing feature – even if it is not always activated by default or must be compiled into the kernel.

The Internet draft [BPB11b] has proposed an example routing table in its Subsection 4.1.2, which should be used as universal approach for MPTCP routing. First approaches of automatic multipath configuration tools already exist [Kri13, Ond13]. These tools create an extra routing table with a single default route entry per source address. The basic idea behind this is to map traffic of a specific source address to a specific interface to support the mapping of paths to links. But this approach has a weak point in design that has not been discussed so far. It is denoted in the following as the MPTCP routing issue. This routing issue demonstrates that the source routing does not cover all use cases. This thesis will discuss that these drawbacks prevent a connection setup in the worst case.

The MPTCP routing issue

The Figure 5.4 illustrates a generic multi-homed endpoint connected to the network via two access links. The endpoint is not aware of the network structure, which is the default case in the Internet. A default application is used in this setup, which is not aware of the MPTCP feature. This setup establishes the client side of the base model discussed in the first section of this chapter. It is addressed by most publications in the context of the SCTP and TCP load sharing discussion, e.g., [RBP+11, PEK11, BPB11a, DBRT10, BDAR12b, DBAR11b]. This setup gives no further information about the topology used, except that a connection via the Internet will be established. Non-Internet based setups have not been addressed so far. However, the scenario described can lead to two different setups which are illustrated in Figure 5.5a and Figure 5.5b. Both setups are characterized as follows:

- **Cross path setup**
  This is the default Internet setup without any individual configurations. In a meshed

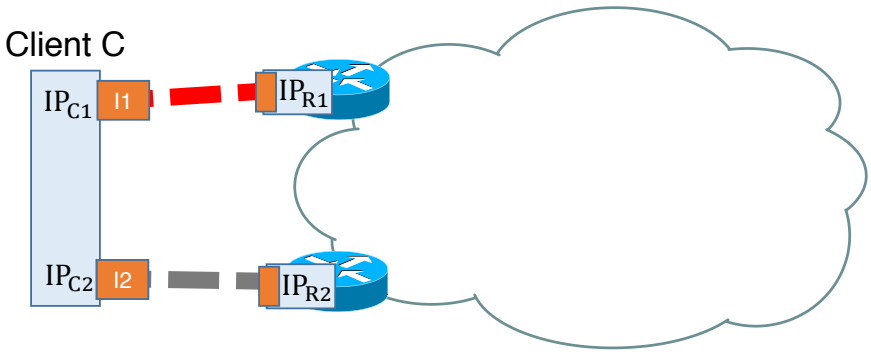
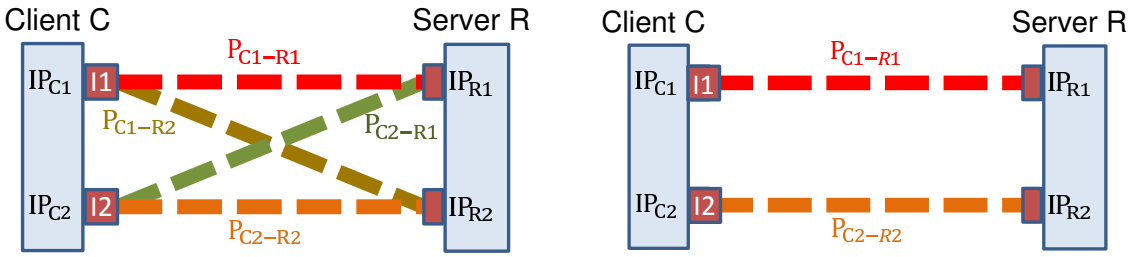


Figure 5.4: Simple multipath routing setup



(a) Cross path setup

(b) Disjoint/linear path setup

Figure 5.5: Path setup

network always a path from any source to any destination exists. Subfigure 5.5a illustrates every possible path which is useable for the initial 3-way handshake in a simple meshed network model. In this scenario the proposed MPTCP routing approach works perfectly, because in the end every possible address combination will be used, independently of the destination address of the initial handshake.

- **Disjoint/linear path setup**
  Using disjoint paths is typically an intra-domain network setup. Thus, if reliability is the goal, it is the preferred network setup for example in a data center [RBP+11]. Furthermore, it is a default setup to provide a high performing setup as described in [Chr13]. However, apart from the intra-domain setup, this disjoint path setup is also a valid Internet scenario, if paths are blocked by middleboxes – e.g. NAT – or overlay networks are used – e.g. virtual private networks. This setup is not covered by the currently discussed routing approaches.

  The network access setup illustrated in Figure 5.4 is used as basis for a simple example. This example uses furthermore a routing table as introduced in [BPB11b] and shown in Table 5.1. This routing table provides a "Default Table" which is used if no source address is provided. This "Default Table" provides a default route which is used, if no other rule fits. Furthermore, it provides three additional routing tables "Table #1", "Table #2" and "Policy Table". "Table #1" is used for IP packets with an assigned source address C1 and "Table #2" is used for IP packets with source address C2. Both tables provide a specific default route per source address. The "Policy Table" provides the corresponding linking from the "Default Table" to the additional tables assigned to a specific source address.

  An application tries to establish a connection to the server just by using a given destination address. The local address is per default set to 0.0.0.0 in case of IPv4 or :: in case of IPv6. This is interpreted by most operating systems as wildcard and named *INADDR_ANY*. In case the *INADDR_ANY* is used, the operating system chooses the source address. Therefore, the setup of the TCB and the chosen path for the connection setup depends on the routing tables and the protocol implementation. The further details will be explained by using the behavior of the current MPTCP reference implementation for Linux. Two different scenarios can be discussed for the disjoint path setup:

  In the first scenario everything works fine. If the application tries to connect to the

Default Table
Dst: 0.0.0.0/0 Via: Gateway-R1 Dev: I1
Dst: Gateway1-Subnet Dev: I1 Src: C1 Scope: Link
Dst: Gateway2-Subnet Dev: I2 Src: C2 Scope: Link
Table 1 (for C1)
Dst: 0.0.0.0/0 Via: Gateway-R1 Dev: I1
Dst: Gateway1-Subnet Dev: I1 Src: C1 Scope: Link
Table 2 (for C2)
Dst: 0.0.0.0/0 Via: Gateway-R2 Dev: I2
Dst: Gateway2-Subnet Dev: I2 Src: C2 Scope: Link
Policy Table
if src == C1, Table 1
if src == C2, Table 2

Table 5.1: Routing and policy table based on the proposal of [BPB11b]

server with the address $R1$, Linux uses destination-based routing and lookup for this address in the routing table. Initially, no source address lookup is possible, because the source address cannot be identified by the $INADDR\_ANY$ configuration. Therefore, the system searches in the main routing table, as this is default. Thus, in this example it will use the Gateway-R1. Via the Gateway-R1 $R1$ is addressed and this results in a successful connection setup. It is important to understand that even if source-based routing is configured, the first lookup is done in the main routing table where only one default route exists for every destination address.

In the second scenario, the application uses the destination address $R2$ instead of $R1$. In this case the OS routes the handshake via an interface that does not provide a route to the destination address[3]. In this case the initial SYN will never reach the destination and no connection will be established. This is a significant drawback if all possible address pairs are built without the knowledge whether a real connectivity exists. The same negative result will occur for the destination address R1 if the default route will be changed to $R2$.

The discussion shows that the proposed MPTCP routing approach is not valid in every scenario. That is the reason why this thesis proposes three different alternative approaches to work around this issue. The mechanisms and pros and cons are discussed in the following:

- The first option – an alternative to the manual configuration of the routing table – is to start the connection on every possible source/destination pair and work in a first come first serve manner on the first fully established 3-way handshake. However, this results in some resource management drawbacks. The system has to manage $n - 1$ half-open TCP connections for every successful initial handshake – where n is the number of all source/destination address pairs – which then have to be aborted in the worst case. To reuse the started handshake is not possible because this requires the key material (see Subsection 3.2.3).

[3]The source IP address is given by the default routing table for the paths ($INADDR\_ANY$,$R2$).

- MPTCP can introduce a kind of heartbeat mechanism that has to take place before the initial handshake starts. MPTCP could check with this heartbeat mechanism whether the destination address is reachable. This approach requires a completely new message exchange similar to the heartbeat mechanism of SCTP, where upon request the receiver has to answer. However, for MPTCP this must be allowed before the connection even exists. Although this is a feasible approach, it needs an implementation on client and server side. Currently deployed MPTCP implementations will not understand this request and will not give any answer, which would prevent a connection setup. As a consequence, this introduces a completely new MPTCP version.

- As alternative, a multipath SYN retransmission strategy is proposed by this thesis, which organizes the retransmission of the initial SYN on alternative paths. Today, there exists a policy to retransmit the SYN if it is lost in a singlepath setup. Until now, this mechanism was not adapted to the multipath extension of MPTCP. In MPTCP the SYN retransmission just focuses on the singlepath connection setup, but this retransmission should be done via all available paths. The timeouts should base on existing suggestions for IETF singlepath protocols, for example, on the old RFC2988 [PA00] – which suggests a retransmission timeout of 3 seconds – or the RFC6298 [PACS11] – which leads to 1 second timeout. This approach can be optimized by considering address-scoping and using a history of known connections. Thus, if the MPTCP implementation is aware of a functional address combination from history, this combination should be used first.

In summary, the proposed routing approach for MPTCP is not sufficient to enable full topology support. An equivalent to a singlepath SYN retransmission is proposed to solve this issue. Even if this timeout approach needs in the worst case the sum of all path RTOs to notify the application about an error, it provides a generic solution without requiring a manual editing of the routing table and also provides full application backward compatibility.

5.1.2.2 Routing and CMT-SCTP

The design goals of SCTP must be kept in mind to understand the routing issues of SCTP. The primary goal of RFC4960 was to support fault tolerance (see Chapter 2). Here, the SIGTRAN use case (see Section 2.2) has been considered, which means that the SCTP protocol is used by default for networks which support a failure-tolerant behavior by design. Thus, if there is no support by the network, there will be no support by SCTP. The architects of SCTP had primarily the disjoint path setup in mind – illustrated in Figure 5.5b – not a meshed and uncontrolled Internet scenario. Without this background, application developers would expect a similar behavior like for MPTCP, but the designs of both approaches are based on completely different assumptions.
This initial intention has an impact on the routing, because SCTP does not organize paths depending on source addresses and there exists no information in most SCTP stacks about these addresses. In FreeBSD an extra feature was developed, which is called the source address selection. The source address selection mechanism fills in the source address depending on the outgoing gateway. There are two reasons to do it that way. On one hand, there is no need to manage the relationship of source addresses to destination addresses in the SCTP implementation because SCTP has not to be aware of the addresses used locally. This is a benefit if the transport protocol must be separated from the network layer as, e.g., done for the RTCWeb stack [BRW+13].

On the other hand, it has a positive impact on the scalability. In the high speed testbed setup of [Chr13], MPTCP was used to send 50 Gbit/s using one MPTCP flow. The default MPTCP connection setup – using a cross table of all possible address pairs – leads to an quadratic increase of the potential subflows assigned to the path. But investigating the experiment setup shows that the researchers also noted that building the full cross table was turned off and only one connection per destination address was established[4]. Therefore, this specific MPTCP connection setup leads to the disjoint path setup of Figure 5.5b[5] that is used by CMT-SCTP by default. Scalability can be an important argument. The calculation of the memory required for the buffer management to support reliable data transfer in this high speed setup is a good example.

It has a high impact on the buffer resource needed by the sender if the minimum buffer has to be calculated for the usage of 36 10 Gbit/s paths – even if only six subflows are really useful in this setup. The system has to preset every source/destination address combination with the resources required for a normal TCP connection, because during the subflow and path building process it is not clear which subflow will be usable for the connection. The base of this resource booking is the calculation of the bandwidth delay product (BDP). The BDP equals the amount of buffer the sender has to provide to support an ordered, saturated transmission. Retransmission timeouts also must be considered if reliable transmission is expected,

That is, the minimum buffer size $B_{\min}$ – for the sender as well as for the receiver buffer – in a setup with paths $P = \{P_1, \ldots, P_n\}$, BW_i the bandwidth and RTT_i the RTT of path P_i is:

$$B_{\min} = 2 * \max_{1 \le i \le n} (\mathrm{RTT}_i) * \sum_{i=1}^{n} \mathrm{BW}_i.$$

Furthermore, the timer-based retransmissions (see Chapter 2 for more details) must be considered in case of high congestion (even if they should be rare). To cover a timer-based retransmission, the minimum buffer size $B_{\min}$ for the sender and receiver buffers is:

$$B_{\min} = (3 * \max_{1 \le i \le n} (\mathrm{RTT}_i) + \max_{1 \le i \le n} (\mathrm{RTO}_i)) * \sum_{i=1}^{n} \mathrm{BW}_i.$$

That is, in the worst case it takes 3 times the RTT (first transmission, fast retransmission, timer-based retransmission) plus the highest path retransmission timeout (RTO). Since the default *minimum* RTO is 1s (as described for example in [Ste07] and [PACS11]), the timer-based retransmission coverage by the buffer space is usually too expensive. Adapted to the 50 Gbit/s experiment the buffer has to be calculated with n=6, BW_i=10000 Mbit/s and a basic setup with RTO=1s and RTT=110ms. This results in a memory need of 7 GiB for the CMT-SCTP and 45 GiB for the MPTCP end-to-end connection. Of course, no optimization is considered here and this represents an extreme/worst case scenario. But nevertheless, it demonstrates the issue with a path definition using all source-destination address pairs. Furthermore, besides memory usage also the impact on CPU load and the organization overhead should be considered.

[4]Not wanted subflows can be blocked for example by firewalls.
[5]Of course with six paths instead of two.

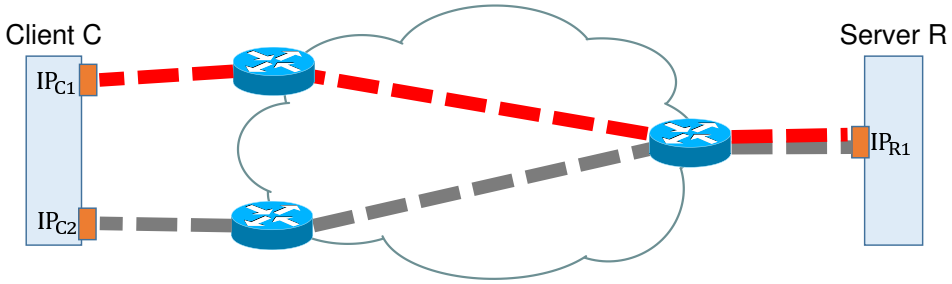


Figure 5.6: Simple multipath scenario

5.1.2.3 Conclusion

Even if the routing of packets is not a task of the transport layer, it has a severe impact on the ability to support the load sharing extensions. The impact of the network layer leads to different results, where only in the best case a load sharing feature for both transport protocols is supported. But it is also true that the mapping of the paths to the links is not straightforward. Efforts have to be spent here in every case for CMT-SCTP. SCTP requires by definition a managed network and the correct mapping. For MPTCP, a universal approach for a meshed network is provided[6] even if it lacks to cover every setup scenario as discussed for the MPTCP routing issue. MPTCP supports – with the proposed improvements – a more practical and easier way to setup conditions required for a successful load sharing deployment, particularly with diverse connection partners in a meshed network like the Internet.

5.2 Path management scenarios

The comparison of the two path management approaches of MPTCP and CMT-SCTP requires comparable scenarios with a real world impact. The first approach for a possible test model is a design that is as simple as possible, but also close to an expected use case. The simplest multi-homing setup requires the existence of more than one IP address at at least one endpoint. This scenario also is addressed in RFC6182 [FRH+11].

5.2.1 Basic scenario 1: One multi-homed host

The most important use case in the Internet is still the *application server model* [KR02]. That means, a provider outside the control of the customer provides a service and the customer uses this service by sending packets to a specific address. This address will be translated during the path setup to a specific IP address and port number pair. This information defines the connection in case of TCP and the well-known 5-tuple.

Just extending this scenario with an additional address leads to the simplest multipath scenario, which is also illustrated in Figure 5.6, by using only one multi-homed endpoint with one additional address. In the multipath scenario of Figure 5.6, the server provides only one interface with only one IP address and the client is connected simultaneously by two, e.g. via a WiFi and a 3G network. Today this setup is a very common use case, e.g., for mobile devices like smart phones or tablets.

[6] with existing software tools.

MPTCP
From the standardization view this setup represents a valid scenario for MPTCP. If the routing works as expected, the first initial 3-way handshake includes the new TCP MP_CAPABLE option and this option signals an incoming MPTCP connection to the server. After the first handshake is performed, the client starts an additional handshake to establish the alternative subflow. As a result, a multipath connection is established through the network using both possible address pairs. Summarized, in case of MPTCP this simple scenario is useable for load sharing even if additional routing configuration is necessary.

CMT-SCTP
As explained in Chapter 2, SCTP uses a 4-way handshake to establish a multipath connection. During this process the client C sends an INIT message including addresses $C1$ and $C2$. The server R answers by sending the INIT-ACK with address $R1$.

It is important to understand that two different mechanisms are working to support multiple paths, one from the singlepath specification and the other from the multipath specification. The RFC4960 was designed to increase reliability, but to support the transfer only via one path.

The load sharing extension uses all existing paths simultaneously. A path in CMT-SCTP is defined only by the destination address. Therefore, if only one destination address exists, only one path is useable by the load sharing extension. Thus, no load sharing is supported in this setup. But this restriction does not mean that the alternative path is not usable. In a failure scenario, SCTP is able to use the existing resources – based on the definition in RFC4960 Section 6.4.1 [Ste07]:

> When retransmitting data that timed out, if the endpoint is multi-homed, it should consider each source-destination address pair in its retransmission selection policy.

Thus, SCTP is in case of a timer-based retransmission able to consider also every other source/destination pair for further transmission. Summarized, in this scenario, the CMT-SCTP extension provides increased redundancy but does not to achieve the goal of increased throughput. This is an additional drawback in comparison to MPTCP.

5.2.2 Basic scenario 2: Two multi-homed hosts

The second scenario is defined by a multi-homing support with two addresses on both endpoints. As discussed so far, this setup leads to two possible scenarios, namely the disjoint path setup (see Figure 5.5b) and the cross path setup (see Figure 5.5a). The disjoint path scenario represents the simple basic test model used so far. It provides routing challenges, but it is a suitable scenario for a performance comparison for both multipath extensions. In contrast, the cross path setup changes the basis for this comparison which was not considered so far. A local testbed (see Section 4.2) with the topology illustrated in Figure 5.7 is used to demonstrate the impact of the link configuration. A typical German DSL setup with specific bandwidth restrictions on every access link was used for the link configuration. The access links $C1$ and $R1$ have been limited to 800 Kbit/s and link $R2$ has been set to 3 Mbit/s. The bandwidth of $C2$ has been varied between 200 Kbit/s and 3 Mbit/s. Delay and error rate were assumed as homogenous and low. The sender was saturated. The setup shows no bottlenecks

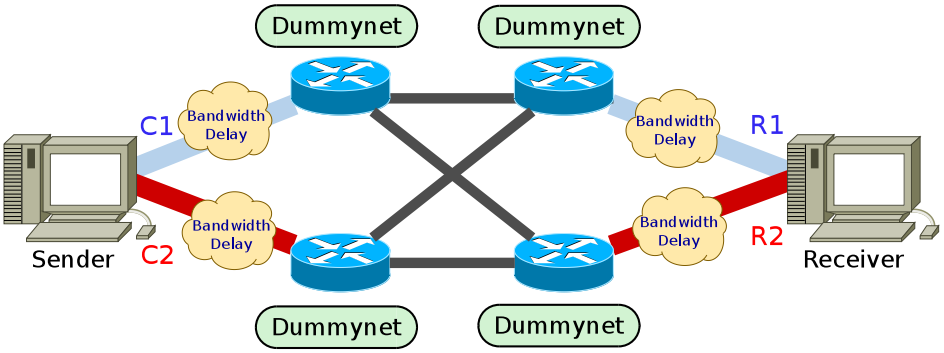


Figure 5.7: Experiment setup to illustrate the impact of source-based selection

in the network. This experiment setup demonstrates a different behavior for MPTCP and CMT-SCTP.

MPTCP
The load is shared by MPTCP among all available resources completely using the existing infrastructure. For the overall system performance it does not matter which path is chosen for the initial handshake (P_{C1-R1}, P_{C1-R2}, P_{C2-R1} and P_{C2-R2}). In every case, every path will be established and used. So, the optimal throughput is achieved, independently of the initial path.

CMT-SCTP
For CMT-SCTP, the achieved throughput depends on the path chosen for the initial handshake, if there exists an asymmetric configuration of the access links. The results are shown in Figure 5.8. If the initial path is set up via the low speed link at the sender ($C1$) and the high speed link at the receiver ($R2$), the throughput does not benefit from the faster link combination (P_{C2-R2}). Therefore, the initial choice of the destination address is crucial for the possible throughput improvement of CMT-SCTP. The ideal load sharing result is only achieved if the right routing decision is configured in the routing table to support the strongest path. A real world issue can be identified, because the configuration of the destination is typically not known. Thus, even if a user or the application has access to the routing table this is not fruitful and does not ensure a correct mapping for the topology. An obvious remedy would be to adopt the mesh-type path management of MPTCP. However, the decision for the (CMT-)SCTP strategy is based on the IETF working group consensus. The goal was to keep SCTP and the multi-homing feature scalable. Therefore, the CMT-SCTP feature has to be designed without any change on the TCB of SCTP.

5.2.3 Specific scenarios

The ability to use every possible path combination also has an impact on other scenarios, which is discussed in the following subsections.

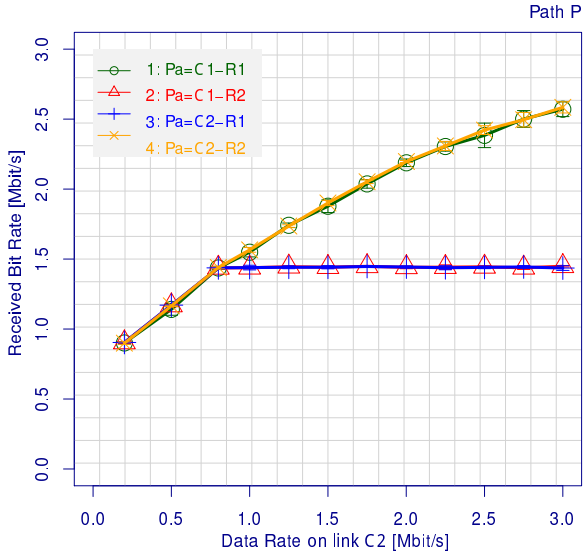


Figure 5.8: Impact of initial handshake

5.2.3.1 Asymmetric load sharing scenarios

Before the path management in a real world scenario is investigated, the impact of an asymmetrical multi-homing setup with more than two IP addresses has to be discussed. The following discussion is based on the simple example illustrated in Figure 5.9. Two endpoints are connected via different networks. The client C has two IP addresses and the server S has three.

As discussed, MPTCP will build all available path combinations. Therefore, MPTCP uses the same number of subflows for both directions. In the scenario given, three subflows are used in every direction. However, CMT-SCTP only uses the destination addresses as pointed out, resulting in an asymmetric number of paths used for each direction. Even if the client C knows three addresses and uses three different paths, the server R answers just by using two paths. However, as long as the network is homogenous it makes no difference, because the congestion control mechanism acts on the bottlenecks in a fair manner among all paths.

The difference will be observed when the bottlenecks are located in the network and can be bypassed via alternative links. Here MPTCP has a benefit, because it is able to use every path combination and, thus, more network resources.

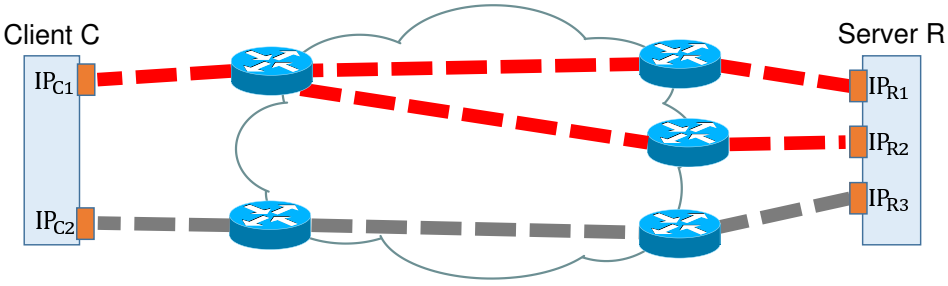


Figure 5.9: Asymmetric multipath scenario example

5.2.3.2 Poor man's multi-homing

This scenario demonstrates that the choice of the technology has an impact on the ability to support load sharing on the transport layer. As discussed in Subsection 2.3.3, with IPv4 and IPv6 there exist two major network protocols. IPv6 introduces a new auto configuration feature by using link-local addresses. Since IPv6 link-local addresses can only be used in a link scope, the transport protocol cannot differentiate between the networks. Thus, an endpoint cannot identify to which network an address is assigned.

Strictly speaking, the "Poor man's multi-homing" scenario reflects a topology where two endpoints are connected directly without any routers or middleboxes in a disjoint path setup. Figure 5.10 illustrates an example and for the following discussion it is assumed, that the initial handshake is performed on the upper path ($IPv6_{CLL1}$,$IPv6_{RLL1}$).

As discussed for CMT-SCTP in Subsection 2.2.2, the IP addresses are exchanged during the initial handshake. The specific aspect about this example is that only IPv6 link-local addresses can be exchanged. These exchanged IPv6 link-local addresses are only usable in the link context and cannot be routed on network layer. Therefore, there exists no routing information for the link-local IPv6 addresses on the sender side, thus there exists no information how to route $IPv6_{RLL2}$ or $IPv6_{RLL3}$. The transport protocol is unable to identify this important information. But a correct assignment of the paths to an outgoing interface is a strict requirement to support the load sharing idea as discussed in Subsection 5.1.2. In the example the information about the local interface $L2$ is needed for $IPv6_{RLL2}$ and $L3$ is needed for $IPv6_{RLL3}$. Thus, instead of the routing information at least the link scope – here the interface – is necessary to provide a valid IPv6 address pair for an end-to-end path (($IPv6_{CLL2}$,$IPv6_{RLL2}$),($IPv6_{CLL3}$,$IPv6_{RLL3}$)). CMT-SCTP uses only the destination to identify a specific path and, furthermore, CMT-SCTP does not manage the source address which would be needed to distinguish between the interfaces. In CMT-SCTP, the source address selection identifies the outgoing interface by using the routing information and here the issue becomes visible, because no information exists for a link-local address. Therefore, CMT-SCTP is not aware of the correct source address and a clear identification of the outgoing interface is not possible in setups like illustrated in Figure 5.10. As a consequence of this ambiguity, it was consensus to filter and ignore all link-local IPv6 addresses for SCTP except the source and destination link-local addresses in the IP packet used by the initial handshake. Therefore, CMT-SCTP will only be able to use a single path in this scenario. Again, MPTCP

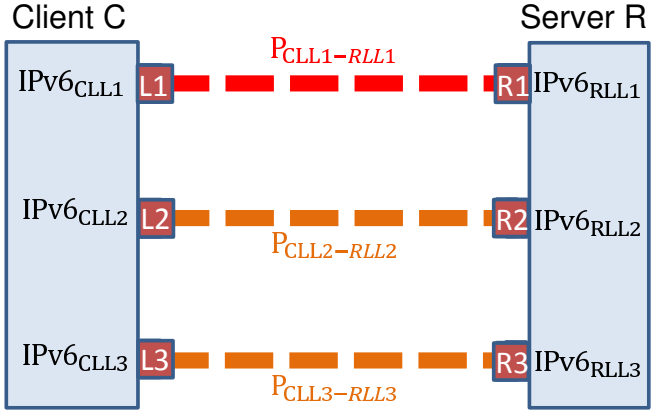


Figure 5.10: Poor man's multi-homing example

has a benefit in this scenario, because MPTCP manages the paths by source and destination address pairs. Therefore, every possible combination can be tried by performing the additional MPTCP connection handshakes. Of course, not every MPTCP handshake started will result in an established connection, but it covers every possibility. Therefore, MPTCP is able to use all valid IPv6 address combinations, i.e. it will use three paths.

5.2.4 Conclusion

Summarized, the success of the path management depends strongly on the routing configuration. Furthermore, the used topology has a high impact on the performance of CMT-SCTP. For a correct comparison of MPTCP and CMT-SCTP it is important to understand that both load sharing extensions are based on different assumptions and CMT design goals. CMT-SCTP argues with a scalability benefit and assumes a network configured for its needs. MPTCP is more flexible and shows the expected throughput independent from the initial handshake.

So far, this path management discussion makes one thing clear: the path management has an impact on the performance of the load sharing extensions. Furthermore, the ability of the transport protocols to adapt to the topologies must be considered. This leads to two different views on the performance criteria of a load sharing transport protocol.

- **Ability of network adaption**
  The discussion shows that it is not enough to focus on the simple test model with two disjoint paths. A network like the Internet provides different topologies. Here the evaluation practice of multipath protocol mechanisms has to be reconsidered. The path management has an impact on the overall system performance, and this impact has to be taken into account.

- **Ability of path adaptation**
  The goal is to investigate the behavior of protocol mechanisms compared to each other based on the same resources defined by the same number of paths. Thus, the testbed environment must be identical for both protocols, e.g., with respect to CPU power and buffer size, but also with respect to the number of useable paths. Only if comparable conditions can be achieved a fair comparison of the mechanisms of MPTCP and CMT-SCTP can be achieved.

The next step is to discuss the impact of the path management for a real world scenario by investigating the behavior in the Internet to demonstrate the practical impact.

5.3 Behavior in a real Internet setup

What should be kept in mind from the discussion above is that path management is only as good as the ability of the path management strategy to adapt to the network conditions. Until now only the impact of the path definition, the routing and the mapping to the access links were discussed. Therefore, the next step is to analyze the ability to adapt to the conditions of the connecting network. In case of the Internet, this aspect cannot be defined easily. The Internet is not a homogenous network, it is a complex structure with a wide range of possible network characteristics. Furthermore, it should be mentioned that it is not an easy task to create a controllable and repeatable experiment in this context. Anyway, in the end it is

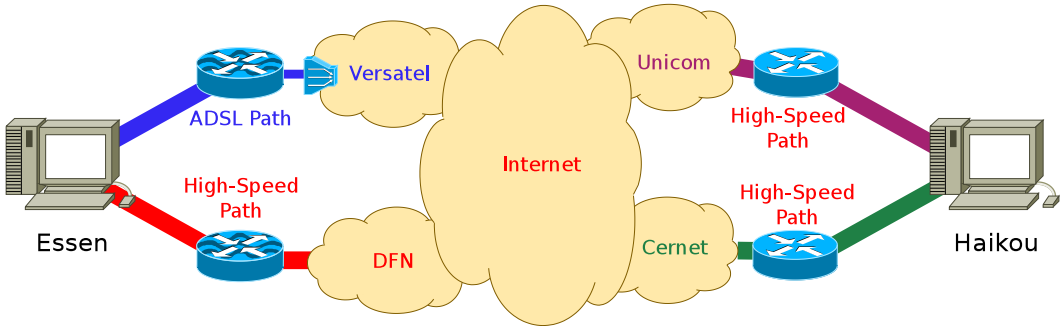


Figure 5.11: Internet scenario

the main target platform and must be investigated. Simulations and testbed setups under laboratory conditions like in [DBRT10, BDAR12b, ADB+11, DABR12b, DBAR11b, BDB+13, DBPR10a] are always a good starting point, but it is the behavior under real conditions that completes the picture. For this picture in particular the extreme scenarios are interesting for an analysis.

This thesis provides a test case that reflects the common use case to increase the performance over a long distance by using a topology that spans the globe. This test case confronts the load sharing extensions with extremely challenging conditions. The evaluation is divided into two steps. The first step is to investigate which path management strategies are deployable and the second step is to demonstrate their impact on the performance in a real world setup. That is why a network access model was chosen as in the initial experiment model illustrated in Figure 5.1.

5.3.1 Global multipath testbed

It was not possible to use existing research networks like Planet Lab [CCR+03] or G-Lab [KSc+11], because this kind of platforms allows no manipulation of the network stacks below the application layer and does not provide multi-homing at the endpoints. However, it is necessary to deploy the patches of the currently available reference implementations of both IETF load sharing extensions to get a complete picture of their behavior in the network. Furthermore, finding an environment that allows manipulation of the initial routing through the network was also the goal which was provided by two endpoints in Essen/Germany and in Haikou/China. The setup itself was considered challenging. The biggest issue was that the topology and the characteristics of the links were not predictable. The next sections describe the work done to define the baseline for the experiment.

5.3.1.1 The endpoint and access link setup

Figure 5.11 illustrates the testbed with two multi-homed hosts located in Essen/Germany and in Haikou/China, respectively. High speed links as well as common customer links were used. That is, the endpoint in Essen was connected via a high-speed fiber optic connection to the Deutsches Forschungsnetzwerk (DFN)[7]. The DFN connects most of the universities in

[7]DFN http://www.dfn.de

Germany. The second path connected to the Essen site was an ADSL connection. This was – compared to the DFN link – a low-performance link with a bottleneck of 800 Kbit/s upstream. The downstream was usable up to 16000 Kbit/s. On the other side, the endpoint in China was connected via two high-speed fiber optic connections. One connection was connected to the China Education and Research Network, also called Cernet[8]. The alternative access link on the Chinese side was connected to the China United Telecommunications Corporation Hainan Province Network, or shortly Unicom[9]. So, in the following discussion, four different paths were possible and are denoted as Path #1: DFN-Unicom, Path #2: DFN-Cernet, Path #3: Versatel-Unicom and Path #4 Versatel-Cernet. At the endpoints the same hardware as discussed in Section 4.2 and Subsection 5.1.2.1 was used.

5.3.1.2 Analysis of the Internet environment

The main focus of this section is to give an inside view of the path setup and the topology of the intercontinental Internet testbed. The tools Traceroute [Mal93], Ping [DH98a] and NetPerfMeter [DABR12a, DBAR11a] were used to get a deeper understanding of the paths resulting from the Internet topology. The database for the following discussion was built by using the information gathered during a long term observation of all paths. The important information about delay, throughput and path building has been traced between 2012-12-07 and 2013-01-01. The interval between the traces was three minutes. This resulted in a total of around 51,000 usable records of bandwidth, delay and path information.

Topology analysis

Most of the times a user is not interested in the hops which are used to reach the destination and it is only a matter of throughput and, more and more, also delay [BRW+13]. However, the first look at the data gave a very confusing picture. There existed more than 100 routers. These routers built multiple hop-to-hop combinations, which did not look very stable at first sight.

A big challenge was to identify measurement failures. Measurement failures happen for example if the path is changing during a Traceroute call interval. In the first analysis of [BAR+13] this impact was not clear. However, after introducing the new failure detection the average hop counts could be recalculated as shown in Table 5.2. For DFN-Unicom the detected hop average was 19, for DFN-Cernet 16, for Versatel-Unicom 19 and for Versatel-Cernet 23 hops. A high variation existed, for example, for the path DFN-Cernet with a difference of 20 hops between the minimum and the maximum hop count (see Table 5.2).

Creating a topology picture which is based on this data would be quite confusing. Therefore, the IP addresses of the routers were just the basis of a second analysis step. The Whois interface of the RIPE NCC [RIP13] has been used. This database was questioned for every known router IP address. Aggregating records of the Whois requests results in a quite short list of 21 different networks (see Table 5.3), and further simplified just 11 different network providers. Furthermore, if the focus is on the paths which were used more than 99% of the connection time, just four routes exist, and none of them uses more than four network providers (see Table 5.4). All identified paths are plotted in Figure 5.12. In the end it is very surprising that this number of IPs represents a simple setup of four different routes, with partially non-disjoint networks. An IP localization service [Max13] was used to locate

[8]Cernet: http://www.edu.cn/english
[9]Unicom: http://eng.chinaunicom.com/

the geographical position of the routers and, therefore, to give a more detailed picture of the networks. Figure 5.13 illustrates the geographical setup. The illustration shows that depending on the Chinese ISP targeted in the destination address, the traffic was routed from Germany directly to the east or to the west via a transatlantic and a transpacific line. The path DFN-Cernet can be confirmed by relevant literature: The DFN is directly connected to the Gigabit European Academic Network backbone (GÉANT) [GÉA13], which is linked to the Chinese research network[10]. For the other routes the results had to rely on the IP localization services. All paths either originating or terminating at the same interface obviously share the first or last hop, respectively. If these are low-capacity links (e.g. the DSL link in Essen), they should most likely constitute the shared bottleneck for these paths.

	DFN-Cernet	DFN-Unicom	Versatel-Cernet	Versatel-Unicom
Complete Path Records (Total)	14290	14290	11379	11380
Different Routers	76	90	41	84
Min Hop Count	7	14	11	7
Max Hop Count	27	30	30	24
Average Hop Count	16	19	23	19

Table 5.2: Routing statistics for each path

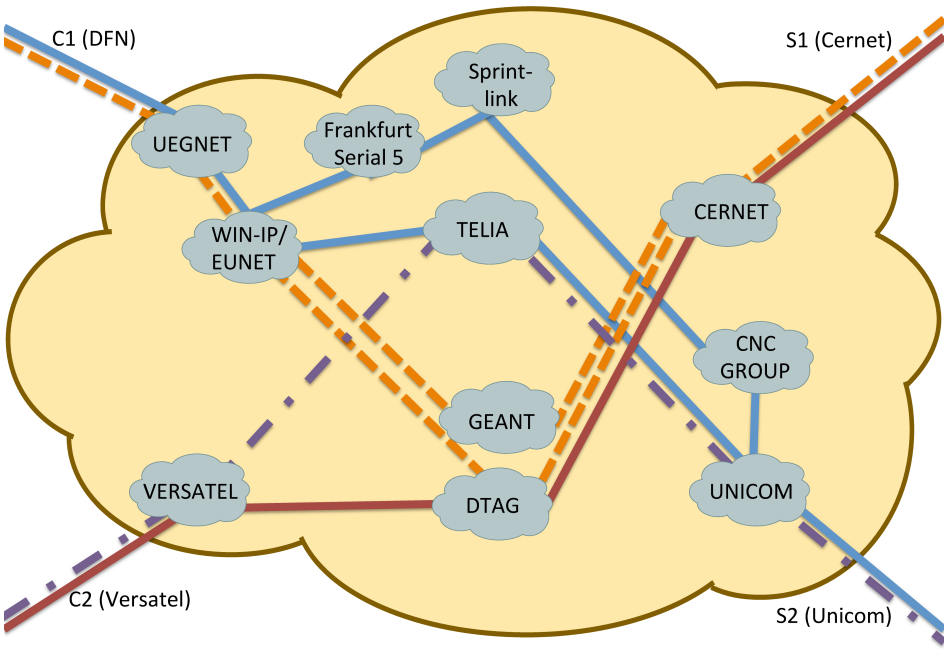


Figure 5.12: Providers used through the Internet

[10]The link "ORIENTplus link6" connects London and Beijing directly via Siberia (http://www.orientplus.eu)

Network Name	Network Provide
TELIANET	**Telia** Sonera International Carrier
CNCGROUP	CNC Group
China Unicom-BACKBONE	Backbone of China **Unicom**
BJCOMP-CN	**Cernet** Super Computer Center
BJREGIONB-CN	**Cernet** Super Computer Center
CER2113-CN	**Cernet** Super Computer Center
GZCOMP-CN	**Cernet** Super Computer Center
DTAG-BB16	Deutsche Telekom AG (**DTAG**)
DTAG-INT1	Deutsche Telekom AG (**DTAG**)
DTAG-TRANSIT5	Deutsche Telekom AG (**DTAG**)
DTAG-TRANSIT12	Deutsche Telekom AG (**DTAG**)
EUNET-BACKBONE	EUnet IP Backbone Network
FRANKFURT-SERIAL5	DE Customer Links
HAINU-CN	**Cernet** Super Computer Center
Sprintlink	Sprintlink
TELIA-TIC-NET-1	**Telia** International Carrier
TELIANET	**Telia** Sonera International Carrier
UEGNET	Universitaet Duisburg-Essen
UNICOM-HI	China **Unicom** Hainan province network
VT-TRANSFER-82-140-22	VT-Network **Versatel**
WIN-IP	IP Networking on **DFN**'s Wissenschaftsnetz

Table 5.3: Identified network providers used by the intercontinental testbed setup

Endpoints	Route used (Probability >99%)
DFN-Cernet	UEGNET - DFN (WIN-IP) - GEANT - Cernet
DFN-Unicom	UEGNET - DFN - TELIA - UNICOM
Versatel-Cernet	VERSATEL - TELIA - UNICOM
Versatel-Unicom	VERSATEL - DTAG - Cernet

Table 5.4: Route per endpoint combination

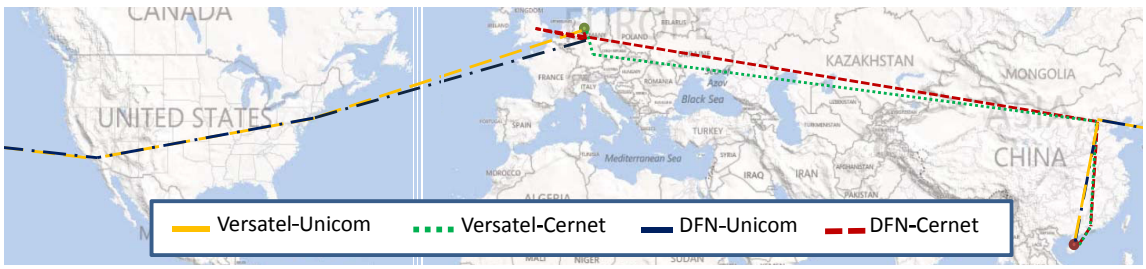


Figure 5.13: Paths between Essen/Germany and Haikou/China

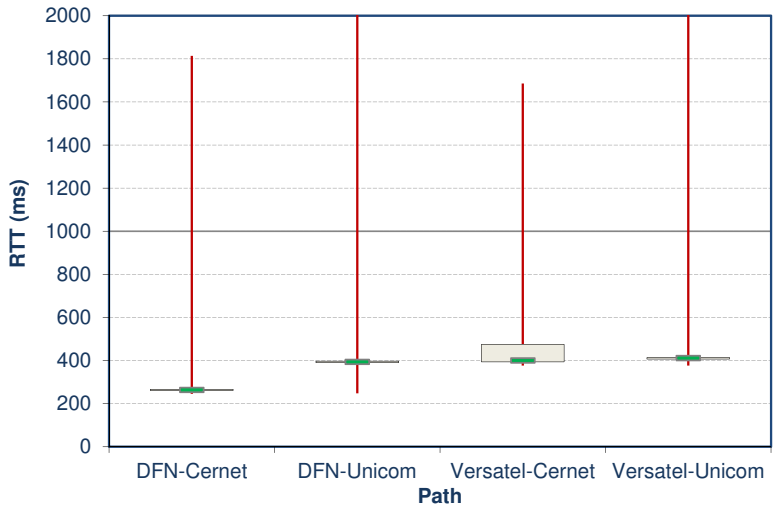


Figure 5.14: RTT statistics for each path

	DFN-Cernet	DFN-Unicom	Versatel-Cernet	Versatel-Unicom
Mean Throughput (Kbit/s)	208	255	496	772
Min Throughput (Kbit/s)	94	101	102	463
Max Throughput (Kbit/s)	295	416	800	800

Table 5.5: Throughput statistics for each path

Delay measurement

Delays and round trip times (RTT) were estimated for the four possible address combinations by sending ICMP packets. A statistical evaluation of the results is shown as a box plot in Figure 5.14. For this plot box all RTT values observed have been sorted. The vertical red lines denote the complete range of observations with the endpoints indicating minimum and maximum. The green colored dash denotes the median dividing this sorted list in the middle. The black box (best visible for Versatel-Cernet) holds 50% of the observed values with the upper and lower 25% (upper and lower quantile, UQ, LQ) is outside. It can be observed that the DFN-Cernet connection has the lowest RTT median of 263 ms, the three other paths are around 400 ms. The small difference between the LQ and UQ for DFN-Cernet, DFN-Unicom and Versatel-Unicom indicates that the RTT values are quite stable for these paths. This is different for Versatel-Cernet where half of the measured values are between LQ=394 ms and UQ=475 ms. Some delay values are very high, but these values were rarely observed and can be considered as outliers. The highest RTT measured on DFN-Unicom, e.g. was 7282 ms. However, only 3 of all values on this path were over 1000 ms. Therefore, the maximum value – shown at 2000 ms – was cropped to enlarge the relevant areas. In conclusion, 200 ms is a reasonable estimate for the end-to-end delay (half of the RTT) and this delay is fairly stable. This is surprising on the one hand, if it is compared with the huge variation of the hop count, but less surprising if keeping in mind that the networks used in the end are always very similar, even though the routers are changing.

Bandwidth measurement
Bandwidth estimations by using the test application NetPerfMeter (introduced in Chapter 4) [DBAR11a] have been performed. NetPerfMeter generated a saturated TCP flow between pairs of addresses and measured the achievable throughput. TCP was chosen because the main goal was to investigate the possible maximum bandwidth that can be utilized by a flow using a default congestion control. The results are summarized in Table 5.5. The first conclusion is that the average achievable throughput is well below 1 Mbit/s for all paths. While the Versatel DSL link can be loaded fully in some cases (800 Kbit/s), the high speed DFN connection performs significantly worse with respect to mean, minimum and maximum. That was really surprising, because each side was connected with at least one glass fiber optic connection. It was not possible to identify the reasons for this unexpected behavior, but the tendency was stable. The best reasons that can be guessed are that there was a constantly high usage of the Internet backbone, a physical bottleneck that impacts every path or a kind of traffic shaping occurs for each flow. The first two reasons are less plausible than the last, because if there was a persistent bottleneck caused by traffic or physical reasons, the flow performance should be decreased more if an additional flow was established, but this behavior could not be observed. But even for the traffic shaping no unambiguous proof could be identified in the network.

5.3.2 Valuation of the Internet throughput

Until now, the picture of a valid experiment in an international testbed is not complete. A baseline analysis of the protocol implementations is still missing. Measurements done as work for this thesis demonstrated a significant difference between the theoretical throughput and the throughput achieved with real world endpoints. As for example the work of [ADB+11] demonstrates, the CPU load has a high impact on the performance of the load sharing protocols. So, a further step was to analyze the ability of the testbed and the protocol implementations to achieve the required performance.

The basis of this analysis were the protocols as they were defined in the initial IETF load sharing specification, without any extensions or optimizations that have been developed on top of this initial idea. Based on the parameter setup of Chapter 4 the following experiment parameters have been used:

- For MPTCP, the MPTCP congestion control mechanism LIA as defined in [RWH09] has been activated.

- CMT-SCTP used LIA as MPTCP, which was adapted on the basis of [RWH09] to get an unbiased comparison. The other protocol mechanisms, like e.g. Delayed SACK or NR-SACK, were deployed as defined in [ABD+13, IAS06].

- The sender has been saturated (i.e. it has tried to transmit as much data as possible). All messages have been transferred using an ordered, reliable service as provided by the TCP default service.

- The measurement runtime has been 300 s, preceded by a transient phase of 20 s. Each run has been repeated at least 10 times in order to ensure a reasonable statistical accuracy.

- The result plots show the average values and their corresponding 95% confidence intervals.

For this the local testbed was reused and configured for the evaluation (see Figure 5.7 on page 63). Based on the intercontinental Internet setup, DUMMYNET has been used on path (C1↔R1) to limit the bandwidth to 800 Kbit/s which corresponds to the maximum upload rate available on the DSL link in Essen. The bandwidth on lower path (C2↔R2) has been varied between 200 Kbit/s and 10 Mbit/s. The parameter range was extended to a scenario closer to the parameters of the realistic wide area connections. A delay difference of 200 ms was configured on the access links, the delays represents the strongest and weakest path of the Internet setup[11]. The results of this experiment are shown in Figure 5.15 and with focus on lower bandwidth in Figure 5.16. For a low bandwidth setting on path B the expected behavior for both multipath extensions can be observed. However, this is not the case when the paths become more dissimilar[12]. In fact, the achievable throughput starts to saturate quickly and drops below the singlepath throughput. This can be attributed to the high CPU load. Peaks of up to 100% have been observed during the experiment. The reason for this high CPU load is that the protocols have to maintain lists of missing data (sequence number gaps) as both use selective acknowledgements (SACK) and lists for re-ordering. These lists have to be searched and updated for every received packet. The queue management of the router queues mainly causes these gaps. For example for RED queues (see Subsection 2.3.2), routers discard packets systematically and the increasing asymmetry between both paths amplifies the problem resulting in a tremendous growth of the lists and the CPU resources required to manage them. This effect is more dramatic for CMT-SCTP as this protocol has to maintain a second list for Non-Revocable SACKs (NR-SACK [NEY+08]) in order to avoid buffer blocking effects [DBRT10]. However, this is not really relevant as argumentation for an Internet protocol as both protocol implementations clearly fail to perform as expected for highly asymmetrical links under real world limitations. It should be mentioned that this CPU limitation could not be observed in typical event based simulation experiments. In this case, the high computation requirements only cause a longer simulation time without influencing the results.

Furthermore, it has to be mentioned that increasing CPU capacities would not be enough to solve the problem since the costs of maintaining the lists increase in a disproportional way with the dissimilarity of the links. Therefore, an optimized list management or an alternative "scheduling" mechanism is required.

However, even though the protocols in the initial definition do not perform as expected, they are at least able to saturate the observed bandwidth range of the intercontinental testbed setup. With these insights, it is now possible to design and conduct experiments in the global testbed to verify whether any of the limitations causes significant issues in a real world intercontinental Internet connection scenario.

5.3.3 Analysis of the protocol behavior in the Internet

After confirming that the Internet setup provides an environment where the multipath protocols can – in principle – operate as expected and the experiments are feasible, the discussion

[11] Values observed in the Internet measurements between Essen/Germany and Haikou/China (see Section 5.3.1.2).

[12] bandwidth on the path (C2↔R2) is higher than 2.5 Mbit/s, while the bandwidth on the path (C1↔R1) is 800 Kbit/s.

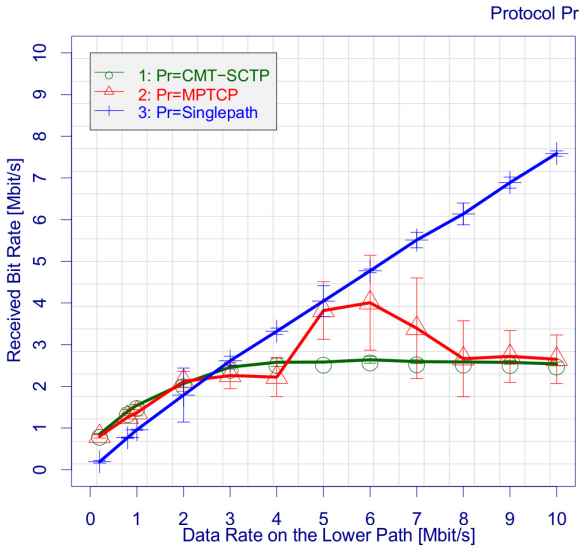


Figure 5.15: Testbed experiment with 200 ms delay difference

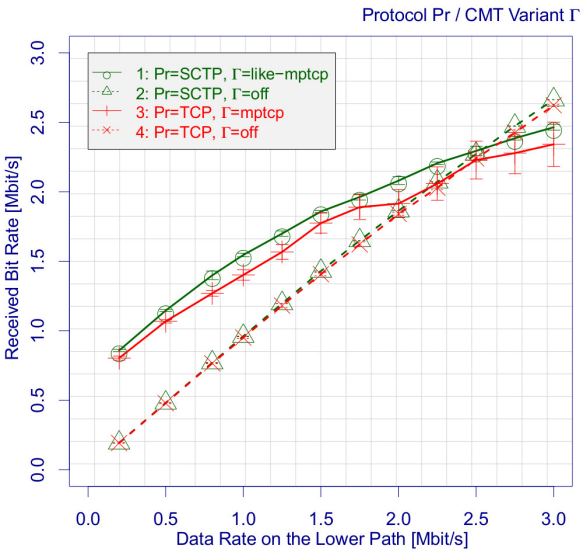


Figure 5.16: Zoom on the lower bandwidth of the testbed experiment with 200 ms delay difference

focuses again on the Internet scenario. As mentioned, first measurements have shown a significant difference between MPTCP and CMT-SCTP. As both congestion control and CPU load can be ruled out as reasons, the different path management strategies can be identified as predominant factor. As expected, the choice of the strategy has a significant impact in more complex Internet topologies.

Again, for CMT-SCTP, the selection of the source/destination address pair for the initial handshake determines the first "primary" path and consequently also the options left for the additional ones. If unfavorable combinations are chosen, this choice may have a significant impact on the achievable throughput (see Subsection 5.1.2.2).

A set of measurements was performed to verify and quantify the impact of the path management, where each of the four possible address pairs has been used to initiate the MPTCP and CMT-SCTP connections. The measurements were performed using the parameters described in Subsection 5.3.2 and repeated 50 times over a period of several weeks to get representative results independent from the current state of the Internet. But even though a significant difference in the choice of the initial path as in our routing experiment (see Figure 5.8) was expected, the results of the testbed show a homogeneous behavior with less than the expected throughput for CMT-SCTP over all source/destination address variations.

Figure 5.17 shows that the throughput of the MPTCP connection significantly exceeds the maximum throughput measured for singlepath connections (see Table 5.5) confirming the benefits of multipath transfer. Furthermore, the throughput is significantly higher in all cases than for CMT-SCTP, where the throughput for CMT-SCTP is less than for the best singlepath address pair in the worst case. This demonstrates that the use of the additional paths by MPTCP actually provides significant advantages.

The measurement confirmed that the MPTCP throughput does not significantly depend

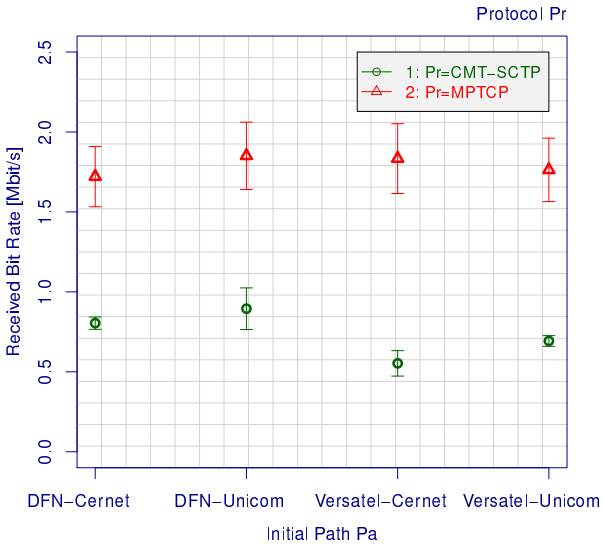


Figure 5.17: Intercontinental testbed scenario

on the address pair used to set up the connection initially and SCTP throughput does. The performance of MPTCP was not reached at any time with any primary path setup by CMT-SCTP although this could be expected from the discussion of Subsection 5.2.2. Even more, the results surprise with respect to the performance of CMT-SCTP, where even the choice of the initial address pair did not achieve in any case the expected throughput. The reason can be guessed to be bottlenecks in the network, as they are, e.g., provided by traffic shaping middleboxes which limit the throughput per flow. Thus, not alone asymmetric access links have a negative impact on the CMT-SCTP extension, also real world limitations on link level. Here the network prevents an optimal performance of CMT-SCTP.

Summarized, none of the load sharing protocol extensions are usable with a real benefit on a challenging Internet setup, like an intercontinental connection that spans the globe. Only MPTCP has, due to its path definition, the ability to adapt to the resources provided by the network. Even worse, by choosing the wrong initial path CMT-SCTP performs less than a default singlepath SCTP connection applied on the strongest link.

5.4 Conclusion

In this chapter, the path management strategies of MPTCP and CMT-SCTP have been identified and analyzed. Furthermore, they have been discussed in a theoretical analysis and evaluated in a real world scenario. As an additional challenge in this evaluation, the first ever intercontinental testbed which supported both IETF load sharing extensions has been set up. This pioneer work provides a basis for upcoming large scale multipath testbed platforms like NorNet [Dre13].

During the theoretical analysis, two different test topologies have been identified, depending on the goal of the evaluation scenario. The research community has not considered this difference between path and network adaptation so far.

Also the dependencies of the IETF load sharing extensions with focus on the routing task have been discussed. Topologies not supported by CMT-SCTP have been identified. Furthermore, for MPTCP it has been demonstrated that the routing approach is not as universal as expected. Two different approaches have been discussed to avoid this issue. Likewise, it has been demonstrated that the CMT-SCTP approach – as straightforward extension of SCTP – lacks the ability to adapt to non-disjoint topologies and does not support asymmetric multi-homing scenarios in every case as good as MPTCP. A similar problem became apparent in the evaluation of the real world scenario, where CMT-SCTP was not able to adapt to the network conditions.

It was shown that both approaches are deployable in the intercontinental setup, but only MPTCP adapts in a way that addresses the goal of load sharing to increase throughput. Furthermore, the real world scenario was investigated in detail, so these results build an excellent base for future testbed and simulation setups.

Chapter 6

Revisiting fairness

The fair allocation of resources among multiple end-to-end connections is an important task of a transport protocol. In particular in the context of the IETF standardization process this is a hard requirement for singlepath transport protocols (see also Subsection 3.2.2) and there is no doubt that this fair allocation is also a hard requirement for the load sharing extensions of TCP and SCTP. Thus, a load sharing approach on the transport layer cannot be discussed without this main IETF requirement.

Today, the mechanisms to achieve a fair allocation with a multipath flow are based on the Resource Pooling (RP) idea (see Subsection 3.2.2.2) and are denoted as coupled congestion controls (CCC). Currently three different CCCs are in the discussion. Two have been developed by the Universities of London [RWH09] and Berlin [Ram12] in the context of MPTCP and one has been co-developed as preparatory work for this thesis [DBPR10a]. In contrast to the first two solutions, the solution of [DBPR10a] has been developed first for CMT-SCTP and adapted to MPTCP later. First evaluation studies [RWH09, Ram12, DBPR10a, Dre12a] had shown that these new multipath mechanisms in principle achieve the goals of the multipath fairness in the way the fairness is discussed the IETF. However, it should be mentioned, that these first individual evaluation studies have been done only in selected scenarios under incomparable conditions. Until now there exists no systematic evaluation of all CCCs under uniform framework conditions and no comprehensive evaluation of MPTCP and CMT-SCTP. These mechanisms are quite new and there exists no common and comparable implementation for all CCCs up to now. The only exception is the simulation environment created in the context of this thesis (see Section 4.1). The simulation model provides the first basis for a real comparison of all available coupled congestion controls implemented in both load sharing extensions.

However, besides the IETF view also other specific perspectives on the fair allocation of resources among multiple end-to-end connections exist. As discussed in this chapter, the alternative perspectives on multipath fairness goals are not always the same as for the IETF. A major goal of this chapter is to bring more aspects into the IETF fairness discussion and to discuss the fairness beyond the limits of the IETF multipath fairness goals.

Altogether, the fairness aspect is discussed in this chapter on two levels. First the origin of the IETF fairness goals is revisited and critically questioned. Second, the proposed CCCs are evaluated for the first time in comparable simulation scenarios. Significant parts of the following discussion have been published in [BDAR12b].

6.1 Multipath fairness goals

As discussed in Subsection 3.2.2 the initial IETF fairness goal has its root in a TCP-compatible behavior [Flo00, BCC+98], where no flow[1] of a multipath connection is allowed to be more aggressive than a flow of a singlepath TCP connection under comparable conditions (see Subsection 3.2.2). However, this fairness definition focuses on flow level and leads to the requirement that a multipath flow has to be TCP-compatible to a singlepath TCP flow. The expected share of the link capacity ρ can also be described by this goal. Thus, if a multipath TCP connection is in competition with a singlepath TCP connection on a shared bottleneck link, the expected capacity allocation for the multipath link ($\rho_{multipath}$) must be less than or equal to $\rho_{singlepath}$:

$$\rho_{multipath} \leq \rho_{singlepath} \tag{6.1}$$

This is a dilemma for the multipath connection, because the multipath connection wants to use as much link capacity as possible to increase its throughput. But the goal of increased throughput sometimes affects the goal to behave TCP-compatible. The issue becomes particularly obvious in the so-called shared bottleneck scenario as discussed in Subsection 3.2.2.1.

The initial conservative idea to prevent a destructive impact of the load sharing extensions was to postulate a TCP-compatible behavior on flow level with the intention to prevent that a multipath flow harms competing singlepath connections or in the worst case even breaks the stability of the network. In the start phase of the IETF load sharing standardization process for TCP there was a broad agreement that the deployment of multipath connections is only acceptable if a backward TCP-compatible behavior can be ensured. The Resource Pooling idea addresses this initial fairness requirement in particular with respect to the shared bottleneck scenario (see Subsection 3.2.2.2). The primary goal of TCP-compatible behavior is described as *do not harm* goal [FRHB13]. The general principle of the Resource Pooling idea is to let the overall flow of a complete multipath connection – using multiple disjoint or joint paths through the network – behave like the flow of a singlepath connection. The Resource Pooling idea achieves this goal by weighting a multipath connection and a singlepath connection as equal in the shared bottleneck scenario. But this behavior is only desirable in specific scenarios where a multipath connection, in contrast to a singlepath connection, uses multiple paths on a shared bottleneck link.

However, during the introduction of the Resource Pooling idea and the definition of the initial *do not harm* goal, two additional goals have been defined to address the characteristics of the Resource Pooling idea. The initial IETF multipath fairness goal was extended with the additional *improve throughput* and *balance congestion* goals (see Subsection 3.2.2.2 and [FRHB13]). The initial idea behind these two additional goals was to describe limits for an implementation of the Resource Pooling idea (see Subsection 3.2.2.2). These limits evolved over time into IETF multipath fairness goals, which today define a fair behavior of a multipath flow in the network.

The initial idea of the *improve throughput* goal was to ensure a minimum performance of the load sharing extensions. This minimum performance was defined by a performance not less than the performance of a singlepath connection assigned to the strongest path. The third goal addresses the initial load balancing goal with *balance congestion* as discussed in Section 3.2. As RFC6356 [RHW11] points out, the achievement of the *balance congestion* goal

[1] A flow describes the sum of all data that is transmitted via one or more paths through the network in the context of one end-to-end connection.

should improve the robustness and overall throughput. But besides these IETF multipath fairness goals there also exist other specific perspectives on a fair allocation of resources which were not considered in the IETF discussion so far.

6.1.1 Perspectives on multipath fairness goals

Besides the IETF, which provides the rules, also the Internet service providers (ISP) and their customers should be involved in the fairness discussion, because they provide the hardware and determine the operation on the network.

The customer of the ISPs, as user of an application, chooses the load sharing extensions of the transport protocols with the primary goal to increase the throughput as much as possible. He has no inside view of the network and the mechanisms used. From his point of view, it is fair to get as much or even more throughput as an alternative user who provides fewer resources to access the same network.

The ISP is motivated to utilize its networks in an optimal way. Thus, an ISP is not that much interested to transfer traffic to another provider as the IETF is interested to achieve a better network utilization across provider borders. It is fair for the ISP to keep as much traffic in its network as in a singlepath connection scenario.

Even if the IETF proposes the multipath fairness goals, this institution is not able to control these goals and prevent misuse. Just to define them is not enough, especially if the user of the multipath connection has no real motivation to achieve these multipath fairness goals in the shared bottleneck, because it will decrease his share of the available resource. But it is only under the control of the user to support these goals and only the ISP can control this behavior, although the ISP is not aware of the allocation of resources among multiple end-to-end connections in other network domains. Therefore, reasons can be identified that lead to misuse and no detection can be ensured and the deployment of the load sharing extensions can fail completely if the ISPs and their customers will be treated unfairly.

6.1.2 Revisiting the IETF multipath fairness goals

Until now all efforts spent to achieve a fair allocation of resources among multiple multipath and singlepath end-to-end connections had a focus on the shared bottleneck problem (see Subsection 3.2.2.2). This shared bottleneck problem is rated very important in the IETF. As consequence, first the Resource Pooling idea was proposed and then the corresponding fairness goals were defined which reflect the needs of the Resource Pooling idea. However, this focus on the shared bottleneck problem and the Resource Pooling idea has possible side effects on the fair allocation of resources in scenarios without a shared bottleneck. A more detailed discussion is needed to understand the dependencies.

6.1.2.1 Impact of the IETF multipath fairness goals

The IETF multipath fairness goals, i.e., *do not harm*, *improve throughput* and *balance congestion*, are the result of the shared bottleneck discussion and the corresponding requirements of the Resource Pooling idea. Figure 6.1 illustrates a similar scenario like the one used by [WHB08] to demonstrate the load balancing goal and the Resource Pooling idea. Figure 6.1 illustrates a network with three independent senders ($S0$ to $S2$) with corresponding receivers ($D0$ to $D2$). Every receiver has the same access link setup as the corresponding

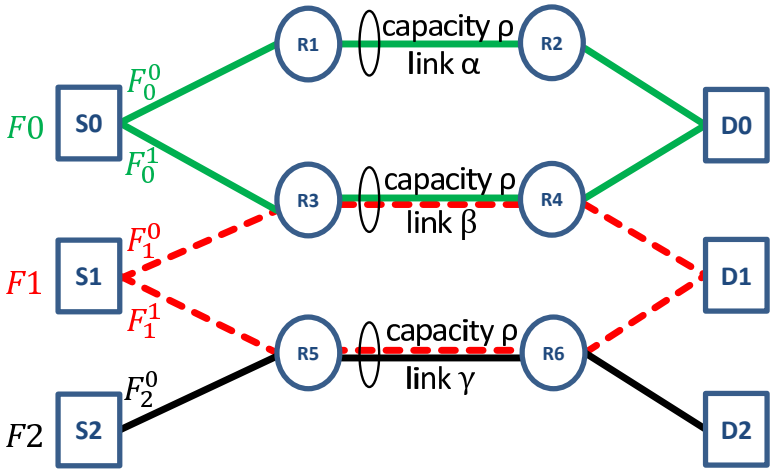


Figure 6.1: Example of balance congestion (is based on [WHB08])

sender, where the access links always provide more capacity than the connecting links between the routers. The links connecting routers are named α, β and γ. Every sender generates one flow ($F0$ to $F2$) through the network. The flows of the senders $S0$ and $S1$ each consist of two subflows[2] routed via disjoint paths, the sender $S2$ uses a singlepath flow $F2$. All links provide the same capacity ρ which is lower than the access link capacity. The path used for subflow F_0^0 is assigned to the link α. The subflows F_0^1 and F_1^0 are competing for capacity on link β and subflow F_1^1 and F_2^0 are in competing on link γ. Thus, because the flow of sender $S1$ is in competition with the flows of $S0$ and $S2$ all paths used in this network setup must be added to the resource pool to share. Therefore, for this setup the "resource to share" has to be calculated by taking into account all possible bottlenecks of the network, i.e. links α, β and γ. Therefore, for the three equivalent flows in this example the sum of all limiting links provides the "resource to share" such that every flow is allowed to allocate a capacity equal to ρ. It is important to understand that this shared resource consists of a complete pool of all paths used by the flow and, furthermore, all other paths – with related links – of all other competing flows (see [WHB08]).

So in this scenario the mechanism to achieve the *balance congestion* goal becomes particularly important, because the singlepath flow F_2^0 of sender $S2$ must allocate the complete capacity ρ on link γ to achieve its fair share of the pooled resource. In RFC6356 [RHW11] it is clearly said that the multipath flow should move as much traffic as possible off its most congested path to adhere to the Resource Pooling idea. This constraint leads to a specific behavior of the flows and to a specific capacity allocation on the links.

The flow $F1$ has to react to the behavior of the singlepath flow $F2$ on link γ. Thus, only the sender $S1$ is able to move traffic from the congested path (link γ) to the alternative link β. Here a new competition scenario exists with the lower subflow F_0^1 of $S0$. Because $S0$ has the ability to utilize 100% of the capacity of the upper path via link α, $S0$ is able to move traffic from the congested path on link β. This leads to the situation where every flow can use a path exclusively and can achieve a capacity allocation of ρ.

Thus, in the context of the Resource Pooling idea the definition of the "resource to share" has been extended from a link view to a complete network view in the context of the IETF

[2]The subflow can be identified by the superscript.

multipath standardization [KV05, WHB08] without explicitly addressing this in the documents. The *balancing congestion* goal was defined to support this, even if was argued for the support of the load balancing idea.

This example makes clear that the fairness on the single link has to step back behind the goal of the Resource Pooling idea and the corresponding *balance congestion* goal. This fairness interpretation should be not mixed up with the TCP-compatible behavior requirement that can also be achieved by another resource allocation. An example can be to allow every flow a capacity proportional to the number of paths it has through the network. Thus, the complete resource pool can be defined as three times ρ where $S0$ and $S1$ with two access links get $\frac{2}{5}$ of the pooled resource each and $S0$ with one access link gets $\frac{1}{5}$ of the pooled resource. Thus, in this case the singlepath resource gets $\frac{3}{5}$ of ρ which is more than $\frac{1}{2}$ ρ, the result of a competition with an alternative singlepath TCP flow.

Although the *balance congestion* goal is claimed as an important argument to place the load balancing feature on the transport layer, it should not be interpreted as a straightforward requirement to share the resources with respect to the Resource Pooling idea. Furthermore, the fairness definition is somehow misleading, because it does not reflect the effort and cost spent by each sender to connect to the network and provides no real benefit for the multipath flows. Therefore, this interpretation does not fit to the expectation of the user, it only addresses the goal to achieve optimal network utilization from the IETF perspective. But even this argument is not valid in this scenario, because it is not a real load balancing function. In the end the system shifts traffic from one path to another until every path is in the same congestion state – as it also would be without the *balance congestion* goal. The overall system performance stays the same, the only difference when using the Resource Pooling approach is that it privileges the throughput of the singlepath flow as shown in this example by increasing the allocated capacity from $\frac{3}{5}$ ρ to 1ρ without any valid reason.

6.1.3 Revisiting fair resource allocation

If the fair sharing of resources is the goal, the first major step is to identify the "resources to share". After this, rules for a fair resource allocation (FRA) can be defined. But even if a rule is functional in the theory, it must be usable in a real world setup as well. Also a controllable and predictable behavior in the network must be a goal, otherwise the system has no chance to identify a misuse.

From the historical view, the "resource to share" was always the link capacity ρ of the bottleneck link l that is shared by the paths used for the end-to-end connection. There exists an one-to-one relation between a subflow assigned to a path and the flow transferred for a singlepath connection. This relation changes for a multipath end-to-end connection. The entity that occupies resources remains the same – here the flow – but the number a of usable paths P through the network (see Subsection 3.2.1) is increased. If more than one path (P_1, ..., P_a; $a > 1$) is used on the same bottleneck link l, a rule to share the capacity ρ must exist. In principle there are two alternatives to define the entities m which have a claim for a fair resource allocation.

- **Subflow**
  Every subflow is defined as independent and has the same claim on the resource. Thus, m is the number of paths and corresponding subflows assigned to the bottleneck link.

- **Flow**
  Here every flow has the same claim on the resource. Thus, m is equal to the number of flows using the bottleneck link. Subflows of the same end-to-end connection must be coupled, because they are used for the same flow.

Therefore, the entity definition of m can in general be based on subflow and flow. This link centric fairness view with respect to subflow and flow can be used for the current singlepath Internet [Bri07] and causes no problems in the context of the IETF. The link share is for a singlepath connection identical for the flow and subflow based fairness, but not in case of multipath. The TCP-compatibility was also consensus for the multipath flow. The multipath link centric fairness view on flow level leads to the shared bottleneck issue (see Subsection 3.2.2.1). The sender is responsible to detect the coupled subflows on the shared bottleneck. However, there was no mechanism identified to achieve this in the Internet, therefore the Resource Pooling idea was introduced to bypass the shared bottleneck problem. But the Resource Pooling idea does not address the link centric fairness view on flow level, rather it addresses a network centric fairness view on flow level. Therefore, the "resource to share" shifts from a link definition to a network definition. Thus, the link centric fairness view with respect to subflow and flow level and the network centric fairness view with respect to the flow level are in the focus of this thesis [BDAR12b].

With this in mind, a first fair resource allocation (FRA) can be calculated for the link centric fairness view on flow and subflow level by:

$$FRA = \frac{\rho(l)}{m} \tag{6.2}$$

In contrast to the link centric fairness, the definition of the "resource to share" is much more complex for the network fairness view. Here, the resource is defined as the sum of the capacities of all bottleneck links used by subflows of flows competing on at least one of these bottleneck links. Thus, in an extreme scenario the whole network – like e.g. the Internet – becomes the "resource to share". The idea of a fair resource allocation (FRA) can be described in this case in a first step based on a fully-meshed network with identical link characteristics. Let there be n flows, all sharing the same paths $P_1, \ldots, P_a$. Then, considering the new IETF multipath fairness goals, a fair resource allocation (FRA) is defined as:

$$FRA = \frac{\left(\sum_{1 \leq i \leq a} \rho(P_i)\right)}{n}. \tag{6.3}$$

However, besides the question if the fair sharing of the resources should be defined by using coupled or decoupled subflows, the practical impact of this resource calculation should be discussed, first. The complexity of the calculations increases in case of different coupled as well as decoupled subflows competing for a variety of different resources in a heterogeneous distribution. It cannot be assumed that the ISPs or their customers are able to perform this calculation. Thus, no monitoring and controlling is possible. In the worst case a user allocates 100% of a bottleneck link capacity and the ISP cannot judge whether this is legitimate or not. A misuse of this can be expected.

With all the focus on the transport layer, it should not be forgotten that a coupling of subflows is already possible today without taking into account the IETF fairness goals by coupling the subflows on the application layer. It might be surprising that the required TCP-compatible behavior on the transport layer is not required on the application layer. Of course,

this requires individual implementation efforts, but in the end the application can provide an individual solution on top of the well-known TCP connections. As an example, if two senders $S0$ and $S1$ are both multi-homed and sharing a bottleneck this leads to an interesting capacity share. In this pure multi-homed scenario the sender $S0$ uses an application with one TCP connection per source/destination address and sender $S1$ uses MPTCP in a cross path setup. Thus, the IETF fairness definition identifies 5 equivalent flows in this setup (4 x singlepath TCP and 1 x MPTCP), where the application of sender $S0$ gets $\frac{4}{5}$ of the available resources and the application on sender $S1$ gets only $\frac{1}{5}$. The user of the multipath flow provides the same access resources but gets only $\frac{1}{20}$ bottleneck capacity per provided access resource instead of $\frac{1}{8}$. The only reason for this share is that the organization of the overall flow is managed for sender $S0$ on the application layer and for sender $S1$ on the transport layer. This behavior is not transparent for the ISP and the application. The IETF requires this by the definition of a TCP-compatible behavior on flow level for a transport protocol, although the impact of this definition is not well-known.

It should be mentioned that only this network centric flow fairness brings the high complexity into the protocol design. Thus, developers are forced to replace the old well-known singlepath congestion controls with new unknown coupled congestion controls to achieve this goal.

If the IETF requirement of a TCP-compatible behavior could be defined on subflow level, the transport protocol would just interact in the same way as a multipath protocol extension on the application layer. The only question left is, are there other potential benefits that favor the complex, uncontrollable fairness interpretation of the Resource Pooling idea.

6.1.3.1 Revisiting the Resource Pooling idea

As discussed in Subsection 6.1.2.1, the *balance congestion* goal prefers a singlepath flow, even if the benefits of the load sharing feature are not clear for the user and the ISP. It should be discussed if other universal goals legitimate this hard coupling of the *balance congestion* goal with the fairness goal.

The authors of [WRGH11] motivate the added value of the *balance congestion* goal with a specific multipath dominated scenario. This scenario is illustrated in Figure 6.2. This multipath dominated scenario includes three different bottleneck links, where each bottleneck link provides a capacity of 12 Mbit/s. Furthermore, three different multipath connections – each multipath connection with two subflows – share this network setup. Every multipath connection shares every bottleneck link with all other multipath connections, so that in the end three subflows of three connections share one link.

The *balance congestion* goal has the intention to shift traffic from a more congested link to a less congested one. Every multipath connection uses one path with one bottleneck link and one path with two bottleneck links. Thus, the path with two bottlenecks can be assumed as path with more congestion. Deploying a mechanism to achieve the *balance congestion* goal leads to a specific share of the network resources and improves the network utilization. In detail, without the *balance congestion* goal the link capacity is divided into three shares of 4 Mbit/s resulting in a multipath connection throughput of 8 Mbit/s. But mechanisms to achieve the *balance congestion* goal bypass the second bottleneck link, so that every multipath connection is able to utilize one path exclusively and does not share the link capacity anymore. Thus, by shifting traffic according to the *balance congestion* goal, a real load balancing benefit will be achieved. A multipath connection is able to achieve a throughput of around 12 Mbit/s

instead of 8 Mbit/s. Without any question this is a really good argument for the *balance congestion* goal and the corresponding Resource Pooling idea.

So it is easy to understand that the authors of the multipath approaches push this argument to argue in favor of the Resource Pooling idea and the corresponding IETF fairness goals. Anyway, until now it was ignored in the discussion that this behavior of Resource Pooling – to adapt to the network topologies – also has negative aspects. Similar to the example in [WRGH11] an alternative scenario has been created in the context of this thesis to demonstrate the drawbacks.

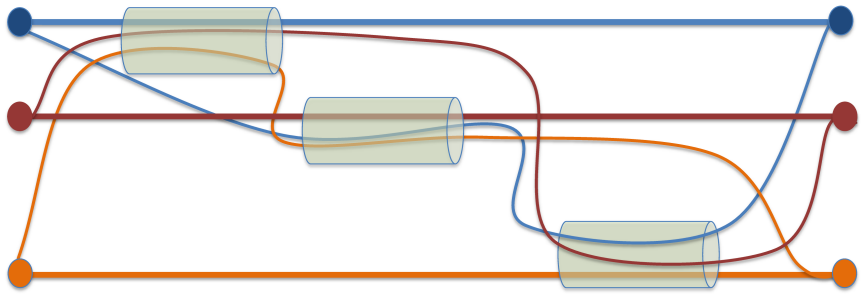


Figure 6.2: Resource Pooling idea: Benefit of adapting to topology

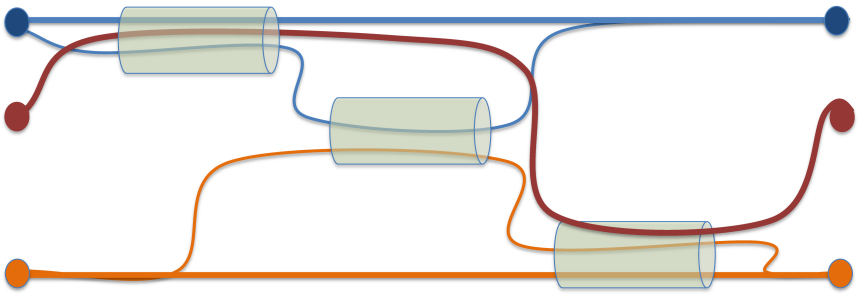


Figure 6.3: Resource Pooling idea: Drawback of Resource Pooling

Figure 6.3 shows a similar base scenario as Figure 6.2. Here two multipath connections and one singlepath connection are sharing the network. The first multipath connection has two subflows via the upper bottleneck link and the second multipath connection has two subflows via the lower bottleneck link. Both multipath flows share one path on the middle bottleneck link. Furthermore, a singlepath connection completes the scenario. The singlepath connection shares the resources with the first multipath connection on the upper bottleneck link and with the second multipath connection on the lower bottleneck link.

The drawback of the Resource Pooling idea is discussed in a comparison to the link centric subflow fairness view. First the resource share in the network centric flow fairness for the Resource Pooling idea is discussed. The multipath connection and the singlepath connection each allocate 50% of the link capacity on the upper shared bottleneck. The same behavior can be observed for the relationship of the second multipath connection and

the singlepath connection on the lower shared bottleneck. The resource allocation on the middle bottleneck link is implicitly the result of the share on the upper and lower bottleneck and no real competition exists. Finally, the singlepath connection gets 6 Mbit/s like the multipath connections as result of the path coupling and all three connections together achieve a throughput of 18 Mbit/s (3 * 6 Mbit/s).

For link centric subflow fairness each subflow gets $\frac{1}{3}$ of the bottleneck link capacity. Thus, the singlepath flow gets 4 Mbit/s and the multipath flows each get 8 Mbit/s. Therefore, the addition leads to a total throughput of 20 Mbit/s (2 * 8 Mbit/s + 1 * 4 Mbit/s). So if the first scenario of Figure 6.2 gives the argumentation for the Resource Pooling idea, Figure 6.3 provides a comparable argument against the deployment. While the first scenario argues for a multipath fairness goal to achieve a TCP-compatible behavior on flow level according to the Resource Pooling idea, the second argues for a fair sharing on subflow level preserving the simple, well known mechanisms used today.

6.1.3.2 Conclusion

The potential drawbacks of the Resource Pooling idea should not be underestimated, because with the Resource Pooling idea a completely new type of congestion controls is introduced without having a solid knowledge of their impact on the network. Furthermore, the Resource Pooling idea does not reflect the motivation of the users nor the ISPs. Of course the bottleneck issue exists as long as a multipath connection has to share the capacity on flow level, but it should be kept in mind that an implementation on the application layer also is possible and nobody can claim a risk that the network will break under these conditions.

The fair sharing on subflow level in a link centric fairness view is much easier to control by, e.g., the ISP and is backed by 30 years of experience to ensure a stable network behavior. Furthermore, the link centric fairness on subflow level also increases the motivation of the users to provide more hardware. The IETF multipath fairness is hard to calculate and to monitor and has the potential to decrease the network utilization in certain scenarios. Furthermore, the IETF multipath fairness prefers a singlepath connection with the *balance congestion* goal in different scenarios without any valid reason.

Altogether, the conservative requirement of a TCP-compatible behavior on flow level causes the Resource Pooling idea with the corresponding new IETF multipath fairness definition. This new fairness definition requires new, unpredictable, coupled congestion controls. The existing uncoupled singlepath congestion controls are not able to achieve this new fairness definition. Thus, because of this conservative IETF requirement, the well-known uncoupled singlepath congestion controls have to be replaced by new, unpredictable, coupled congestion controls. Therefore, the initial conservative design goal for the load sharing extensions of SCTP and TCP mandates the deployment of new, not systematically investigated coupled congestion control (CCC) proposals with a potentially high impact on the overall system performance.

Therefore, even if the initial documents have been published as RFCs, a new discussion in the IETF should be initiated. But as long as there is no rethinking in the IETF community, the behavior of the coupled congestion controls has to be investigated.

6.2 Variants of coupled congestion controls

Although many good reasons argue for a link centric fairness, the network centric fairness with the corresponding Resource Pooling idea was defined by the IETF. Therefore, there exists a strong motivation to systematically investigate the impact of the Resource Pooling idea. Furthermore, it must be evaluated if the corresponding IETF multipath fairness goals [RHW11] can even be achieved. Currently, three different coupled congestion control (CCC) proposals on the basis of the Resource Pooling idea exist in IETF. This section investigates their behavior in a first step by step analysis in three very basic scenarios for CMT-SCTP and MPTCP to demonstrate their pros and cons.

From the historical point of view, the first CCC was introduced by [RWH09]. Today this CCC is known as Linked Increases Algorithm (LIA) and is used by the MPTCP reference implementation on Linux. As preparatory work of this thesis, a second CCC [DBAR11b] was introduced as a very basic and straightforward implementation of the Resource Pooling idea. Today, an improved version of this first idea is known as Resource Pooling Multipath Version 2 (RP-MPv2) [BDAR12b]. The newest CCC was introduced as Opportunistic Linked Increases Algorithm (OLIA) by [Ram12], which was claimed to be an optimized version of LIA.

Before going into the description of the congestion controls some basic notations have to be introduced to discuss the details of the *congestion window* c calculation of the path P. The path congestion window c_P defines the upper limit for the number of outstanding bytes on path P. The calculation of c_P is done differently in a *slow start* phase and a *congestion avoidance* phase. A threshold triggers the transition from *slow start* to *congestion avoidance*. This so-called *slow-start threshold* s_P of the path P controls the growth rate of c_P:

- In the *slow start* phase ($c_P \leq s_P$) the c_P increases exponentially.
- The *congestion avoidance* phase ($c_P > s_P$) only allows a linear growth.

This behavior was already discussed for the New Reno congestion control (see Subsection 2.2.4.1). The New Reno congestion control is also deployable for the multipath extensions of CMT-SCTP and MPTCP on subflow level and is denoted in this context as CMT-Reno. CMT-Reno represents an uncoupled congestion control according to the link centric fairness on subflow level. In addition, it should be mentioned that even if SCTP and TCP share the same basic idea of applying a window based congestion control, the window of SCTP is maintained in bytes and the window of TCP in MSS. Therefore, the CCCs have to be adapted to the load sharing approaches of TCP and SCTP.

This thesis provides the first complete adaptation of all three CCCs for both the load sharing extensions of TCP and SCTP. For LIA this had already be done as cooperation in the context of [DBPR10a]. The results of the adaptation work for OLIA and RP-MPv2 are discussed exclusively in this thesis. An overview of all variables used in the following subsections is given in the Table 6.1.

6.2.1 Resource Pooling Multipath version 2 (RP-MPv2)

This variant introduced by [DBAR11b] applies the idea of Resource Pooling – to couple the congestion controls of the paths – in a very strict manner. The *increase factor* $\hat{i}_P$ is applied to increase c_P on α acknowledged bytes on path P in a fully utilized congestion window. $\hat{i}_P$

Parameter	Description
α	Acknowledged bytes
$\hat{a}$	Aggressiveness factor
c	Congestion window
c_P	Congestion window of path P
$\hat{d}_P$	Decrease factor of path P
$\hat{i}$	Increase factor
$\hat{i}_P$	Increase factor of path P
i	Path number
max_i	Maximum value of all paths
MSS_P	Maximum segment size of path P
MTU_P	Maximum transmission unit of path P
P	Path
p_P	Partially acknowledged bytes of Path P
RTO	Retransmission timeout
RTT	Round trip time
RTT_P	Round trip time of path P
s_P	Slow-start threshold of path P
SEG_P	Data unit on path P for calculation: For MPTCP SEG is MSS, for CMT-SCTP SEG is an MTU.

Table 6.1: Overview of parameters used to describe CCC proposals

represents the bandwidth share of path P and is based on the relationship between c_P and the RTT_P of the path. The partially acknowledged bytes p_P are used to ensure that the complete congestion window is acknowledged, before the window is increased again.

$$\hat{i}_P = \frac{\frac{c_P}{\mathrm{RTT}_P}}{\sum_i^n \frac{c_i}{\mathrm{RTT}_i}}. \tag{6.4}$$

$$c_P = c_P + \begin{cases} \lceil \hat{i} * \min\{\alpha, \mathrm{SEG}_P\} \rceil & (c_P \leq s_P) \\ \lceil \hat{i} * \mathrm{SEG}_P \rceil & (c_P > s_P \wedge p_P \geq c_P) \end{cases}. \tag{6.5}$$

The ceiling function ensures a congestion window growth of at least one byte in order to preserve the AIMD behavior. The *decrease factor* $\hat{d}_P$ is applied to reduce c_P on a packet loss on path P:

$$\hat{d}_P = \max\left\{\frac{1}{2}, \frac{1}{2} * \frac{\sum_i^n \frac{c_i}{\mathrm{RTT}_i}}{\frac{c_P}{\mathrm{RTT}_P}}\right\}. \tag{6.6}$$

$$s_P = \max\left\{c_P - \lceil \hat{d}_P * c_P \rceil, \mathrm{SEG}_P\right\}, \tag{6.7}$$

$$c_P = \begin{cases} s_P & \text{(Fast retransmission)} \\ \mathrm{SEG}_P & \text{(Timer-based retransmission)} \end{cases}. \tag{6.8}$$

$\hat{d}_P$ represents the factor by which the bandwidth of path P should be reduced in order to halve the total flow bandwidth. That is, c_P may decrease to SEG_P. If $\hat{d}$ would reduce c_P to a smaller value (prevented by the max function), the path P could not be used for further data transmissions during the time of one RTO on path P [APB09].

6.2.2 Linked Increases Algorithm (LIA)

The LIA congestion control was proposed by [RWH09] to support TCP-fairness for MP-TCP [FRH+11]. Like RP-MPv2, the ceiling function ensures an increase of at least one byte. $\hat{a}$ denotes the per flow aggressiveness factor.

c_P calculation for MPTCP:

$$c_P = c_P + \begin{cases} \text{MSS}_P, & (c_P \leq s_P) \\ \min\left\{\left\lceil \frac{\hat{a}*\alpha*\text{MSS}_P\}}{\sum_i^n c_i} \right\rceil, 1\right\}, & (c_P > s_P \wedge p_P \geq c_P) \end{cases}. \tag{6.9}$$

$$\hat{a} = \left(\sum_i^n c_i\right) * \frac{\max_i \left\{\frac{c_i}{(\text{RTT}_i)^2}\right\}}{\left(\sum_i^n \frac{c_i}{\text{RTT}_i}\right)^2}. \tag{6.10}$$

c_P calculation for CMT-SCTP:

$$c_P = c_P + \begin{cases} \min\left\{\left\lceil \frac{c_P*\hat{a}*\min\{\alpha,\text{MTU}_P\}}{\sum_i^n c_i} \right\rceil, \min\{\alpha, \text{MTU}_P\}\right\}, & (c_P \leq s_P) \\ \min\left\{\left\lceil \frac{c_P*\hat{a}*\text{MTU}_P}{\sum_i^n c_i} \right\rceil, \text{MTU}_P\right\}, & (c_P > s_P \wedge p_P \geq c_P) \end{cases}. \tag{6.11}$$

$$\hat{a} = \left(\sum_i^n c_i\right) * \frac{\max_i \left\{\frac{c_i/\text{MTU}_i}{(\text{RTT}_i)^2}\right\}}{\left(\sum_i^n \frac{c_i/\text{MTU}_i}{\text{RTT}_i}\right)^2}. \tag{6.12}$$

Formula 6.11 and 6.12 are based on [RWH09], but have been adapted from a congestion window given in MSS to a congestion window given in bytes for CMT-SCTP. Furthermore, the congestion window decrease behavior has been modified slightly. In case of a retransmission (i.e. fast or timer-based) on path P, s_P and c_P are reduced as follows:

$$s_P = \max\left\{c_P - \frac{1}{2} * c_P, \text{SEG}_P\right\}, \tag{6.13}$$

$$c_P = \begin{cases} s_P & \text{(Fast retransmission)} \\ \text{SEG}_P & \text{(Timer-based retransmission)} \end{cases}. \tag{6.14}$$

The minimum value of c_p in case of a timer-based retransmission in MPTCP is less than for singlepath TCP as the c_P may decrease to MSS_P instead of $4 * \text{MSS}_P$. The minimum value decreases to one MTU for CMT-SCTP.

6.2.3 Opportunistic Linked Increases Algorithm (OLIA)

This congestion control was introduced by [Ram12]. The aggressiveness factor per flow $\hat{a}$ includes an RTT weighting like for RP-MPv2. Furthermore, the approach provides different modi for the calculation of the aggressiveness factor $\hat{a}$. Therefore, $\hat{a}$ depends on the path characteristics of the subflows. c_P will be re-calculated for each ACK on path P.

c_P calculation for MPTCP:

$$c_P = c_P + \left(\frac{\frac{c_P}{\mathrm{RTT}_P}}{\sum_i^n \frac{c_i}{\mathrm{RTT}_i}} + \frac{\hat{a}}{c_p} \right) * \mathrm{MSS}_P * \alpha\} \tag{6.15}$$

c_P calculation for CMT-SCTP:

$$c_P = c_P + \left(\frac{\frac{c_P}{\mathrm{RTT}_P} * \min\{\alpha, \mathrm{MTU}_P\}}{\sum_i^n \frac{c_i}{\mathrm{RTT}_i}} + \hat{a} * \min\{\alpha, \mathrm{MTU}_P\} \right) \tag{6.16}$$

The term $\min\{\alpha, \mathrm{MTU}_P\}$ in equation 6.16 could also be excluded, but this would increase the risk of rounding errors. The impact of $\hat{a}$ is very important because it has a significant influence on the aggressiveness level. The calculation of $\hat{a}$ depends on the status of the currently calculated P_i. Different groups for P_i are defined in [KGPB14] with different impact on the congestion window calculation:

> all_paths: The set of all the paths established by the MPTCP connection.
>
> max_w_paths: The set of paths in all_paths with largest congestion windows.
>
> collected_paths: The set of paths in all_paths that are presumably the best paths but do not have largest congestion window (i.e. the paths within the set of best paths that are not in max_w_paths).

This leads to following calculation of $\hat{a}$:

- if P_i is in *collected_paths*: $\hat{a} = \frac{1}{collected\_paths * n}$
- if P_i is in *max_w_paths* and *collected_paths* is not empty: $\hat{a} = \hat{a} - \frac{1}{max\_w\_paths * n}$
- Otherwise: $\hat{a} = 0$

Even if the authors of OLIA have confirmed the observations about LIA - which will be discussed in the following - and claimed LIA as not pareto-optimal [Ram12], this does not change the fact that LIA has been standardized as congestion control for CMT. Therefore, if current standard CMT implementations will be deployed, the LIA approach will be used.

6.2.4 Resume of coupled congestion controls

Summarized, all multipath coupled congestion controls have been developed with the main goal to solve the shared bottleneck issue by implementing the Resource Pooling idea. But even if they have been designed for the same goal and all apply the AIMD behavior, they interpret the increase and decrease (AIMD) in different ways. In the next sections the impact of this different interpretation will be discussed.

6.3 Evaluation of coupled congestion controls

The quality criteria for the CCC proposals mainly relate to three different aspects:

- The mechanism used by the proposals has to fulfill the IETF fairness goals, in particular with respect to the shared bottleneck scenario.
- The result of the CCC proposals must be predictable for all involved parties to create a controllable and comparable basis.
- The mechanisms of the coupled congestion controls have to be accurate. Thus, a measured value hast to achieve the expected and predicted value.

The congestion control has to achieve the different goals with the additional challenge to adapt to the different link characteristics. Different straightforward scenarios will be used for the evaluation to keep the effects easy to predict.

First, the CCC proposals have to work in setups with just one link like the well-known singlepath congestion controls. Without knowing if multipath transfer is provided or not, an application has to choose whether a single-homed or a multi-homed endpoint should be configured during connection setup. There is no dispute that the Internet is still single-homing dominated, even if multi-homing becomes more and more important. Therefore, a simple singlepath scenario gives the first basic scenario and allows a first general fairness discussion.

In a second step the "resource to share" stays the same, but the endpoints switch from pure single-homed senders to a combination of single-homed and multi-homed. This leads to the shared bottleneck scenario (see Subsection 3.2.2.1) and is used to analyze whether the IETF multipath fairness goals can be achieved or not.

The third scenario keeps the endpoint characteristics of the shared bottleneck scenario, but changes the topology and, therefore, the number of disjoint paths. Thus, the bottleneck does not consist of one shared bottleneck anymore, it is replaced by two disjoint bottlenecks. This scenario demonstrates the impact of the topology on the fairness behavior of the multipath protocol extensions.

In the last step, the impact of an increased number of multipath flows is discussed.

It should be mentioned that all access links for the different setups provide enough bandwidth so they will not create bottlenecks. In the simulation setup 1 Gbit/s access links were used with zero delay and error rate. If this not the case, it will explicitly mentioned. This setup ensures that the bottleneck is always located in the connecting network.

6.3.1 Scenario 1: Singlepath

Currently single-homed access is still the regular connection case. Thus, multipath congestion controls have to adapt to the conditions of a singlepath scenario, even if multipath transfer is configured. The behavior of MPTCP and CMT-SCTP is discussed in this first scenario, with the secondary goal to introduce the different fairness views and corresponding notation in more detail.

6.3.1.1 Simple singlepath model and theoretical discussion

The first basic scenario is shown in Figure 6.4. Here, two singlepath connections are used to transfer data via a shared bottleneck. This is a straightforward scenario to demonstrate how

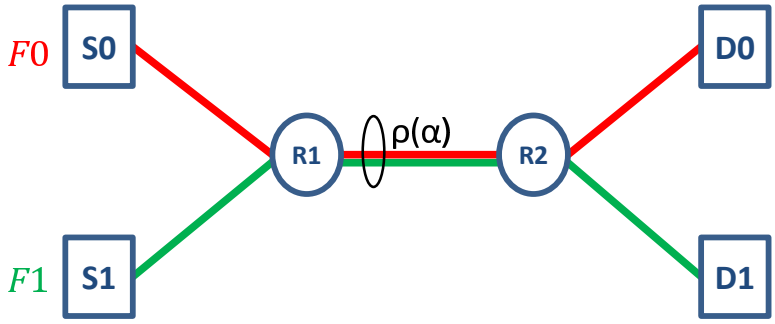


Figure 6.4: Fairness Scenario 1: Singlepath

current congestion controls achieve a fair sharing.

The complete capacity of the bottleneck link α is denoted as $\rho(\alpha)$. The flow between $S0$ and $D0$ is denoted as $F0$ and is composed of only one subflow[3] F_0^0. The capacity allocated to $F0$ is denoted as $B0$ and the capacity allocated to F_0^0 is denoted as B_0^0. It is assumed that a protocol uses the link capacity as efficiently as possible. Thus, in case of link centric subflow fairness this leads to the following network capacity allocation according to equation 6.2:

$$B0 = B_0^0 = \frac{\rho(\alpha)}{2} \quad ; \quad B1 = B_1^0 = \frac{\rho(\alpha)}{2}. \tag{6.17}$$

The network centric flow fairness leads to the same result according to equation 6.3. Here, the network only consists of a single bottleneck link α shared by all flows. The fairness is obvious to determine and can be visualized in the curve shown in Figure 6.5. This curve has also been used for example by [Wel05] to illustrate the convergence to a fair sharing. The line shows that for each link capacity the same throughput ($B1 = B0$) should be achieved for both flows. The convergence to this fair allocation is denoted as optimal (fairness) point and is achieved by the discussed AIMD mechanisms.

6.3.1.2 Evaluation of the singlepath scenario

A simulation setup was configured to evaluate this with $\rho(\alpha)$ varying from 5 Mbit/s to 100 Mbit/s for the singlepath bottleneck. Both senders and all access links were configured to be able to use the complete bottleneck capacity $\rho(\alpha)$. Every setup was realized with TCP and SCTP and their multipath extensions with four different competition scenarios, where a singlepath New Reno flow is in competition with

1. An alternative singlepath New Reno flow
   The goal of this experiment was to demonstrate the behavior singlepath congestion controls used in the Internet today. The results of this experiment give the benchmark for the link centric flow fairness. However, given the fact that in this scenario the link centric and the network centric flow fairness are identical, this line is usable as benchmark for the CCC proposals, too.

2. A multipath flow using the LIA CCC
   LIA represents the default congestion control for MPTCP, in particular for Linux. The results of LIA represent the currently standardized approach.

[3] F_a^b: a describes the flow number and b describes the subflow number.

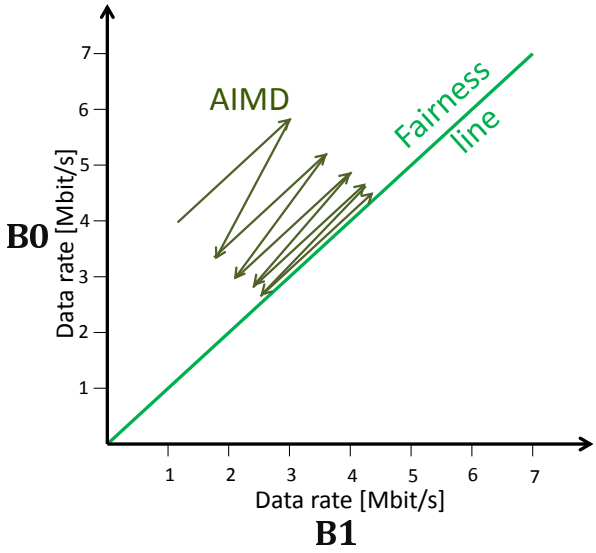


Figure 6.5: Well-known theoretic singlepath fairness curve

3. A multipath flow using the OLIA CCC
   This represents the latest CCC developed for MPTCP.

4. A multipath flow using the RP-MPv2 CC
   This represents the latest CCC developed for CMT-SCTP.

CMT-Reno is identical to New Reno in the singlepath setup and, therefore, not added as additional experiment. The flow $S1$ always used TCP with the New Reno congestion control (see singlepath TCP congestion control in Subsection 2.2.4.1). $S0$ varied over singlepath SCTP und CMT-SCTP in Figure 6.6 and over TCP and MPTCP in Figure 6.6. A complete configuration overview is given in the Table 6.2 for the Figure 6.6 and in the Table 6.3 for the Figure 6.7. The curves of both figures always show the difference between the singlepath TCP flow $F1$ and the alternative $F0$ and, therefore, the deviation from the optimal fairness point. The results with absolute values are also added in the Appendix.

The benchmark Curve #1 in Figure 6.6 and in Figure 6.7 demonstrates impressively how accurate today's singlepath congestion controls work. As expected, the results represent the optimal fairness point for every capacity selection. There is no difference whether a singlepath TCP flow is in competition with an alternative TCP flow (Curve #1 in Figure 6.6) or with an SCTP flow (Curve #1 in Figure 6.7). The deviation of both singlepath flows results to zero.

In contrast to the singlepath congestion control, LIA starts to suffer at around 50 Mbit/s (Curve #2 in both figures). The reason for this CCC behavior can be identified in the general design of the CCC proposals, where every flow is allowed to perform only in maximum as good as a default singlepath TCP flow. Thus, the multipath flow can achieve as much as a singlepath flow only in the optimal case. Even small rounding errors or a bit of loss can change this behavior.

OLIA behaves similar to LIA, even if the effect is not that drastic (Curve #3 in both figures). It is obvious, that with OLIA improvements in the CCC design were achieved, even if the optimal fairness points in the scenario were not achieved in every case.

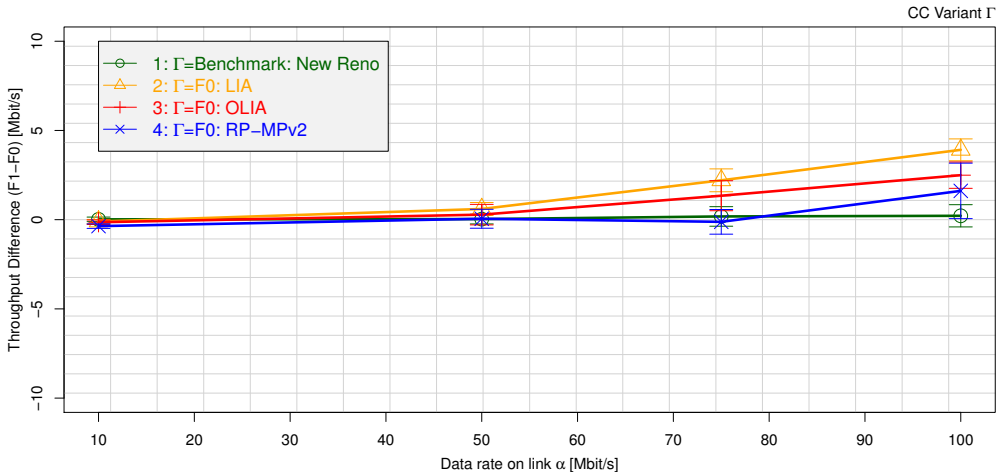


Figure 6.6: Accuracy of fairness in a singlepath setup with SCTP and CMT-SCTP

Experiment	$F0$ protocol	$F0$ CC	$F0$ subflows	$F1$ protocol	$F1$ CC
1	SCTP	New Reno	1	TCP	New Reno
2	CMT-SCTP	LIA	1	TCP	New Reno
3	CMT-SCTP	OLIA	1	TCP	New Reno
4	CMT-SCTP	RP-MPv2	1	TCP	New Reno

Table 6.2: Configuration for Figure 6.6

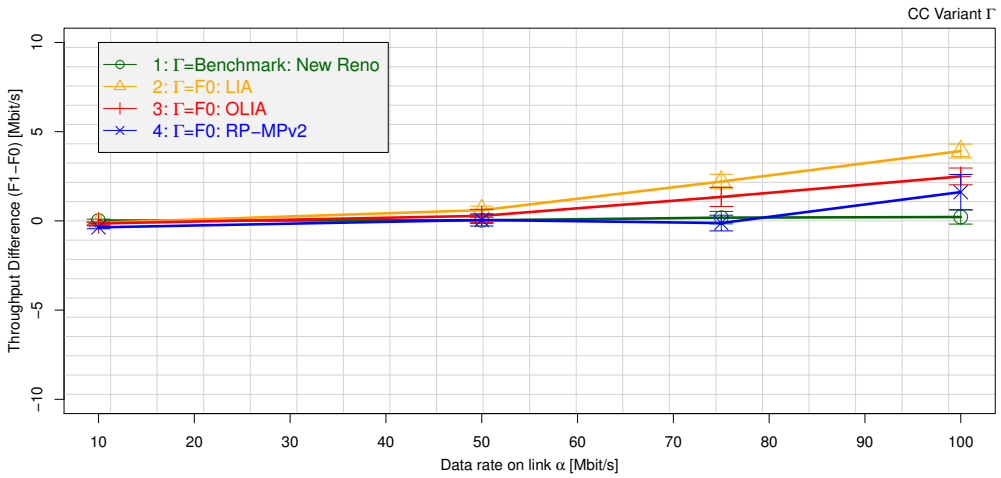


Figure 6.7: Accuracy of fairness in a singlepath setup with TCP and MPTCP

Experiment	$F0$ protocol	$F0$ CC	$F0$ subflows	$F1$ protocol	$F1$ CC
1	TCP	New Reno	1	TCP	New Reno
2	CMT-SCTP	LIA	1	TCP	New Reno
3	CMT-SCTP	OLIA	1	TCP	New Reno
4	CMT-SCTP	RP-MPv2	1	TCP	New Reno

Table 6.3: Configuration for Figure 6.7

Anyway, in this setup the RP-MPv2 CCC proposal performed best compared to the OLIA or LIA proposal (Curve #4 in both figures). The optimal fairness points were achieved up to 75 Mbit/s.

Summarized, it can be observed that none of the multipath flows achieved a performance comparable to a singlepath congestion control. All CCCs showed a non optimal behavior with increasing link rate, even if the impact distinguishes them. Anyway, a deviation of less than 5% is not very accurate, but acceptable.

6.3.2 Scenario 2: Shared bottleneck

The next step extends the simple singlepath test setup. $S0$ and $D0$ are designed as multihomed with an additional access link as shown in Figure 6.8. This change leads to the important shared bottleneck scenario, as discussed in general in Subsection 3.2.2.1. The challenge in this scenario is clear: both flows share the same bottleneck and the three Resource Pooling goals[4] for the IETF multipath fairness have to be fulfilled.

In this scenario the different number of the subflows per flow has to be considered. As a result of the discussion in Chapter 5, the routing applied on the network layer may support two kinds of path management strategies. In Subsection 5.1.2.1 the strategies were defined as cross path setup and linear path setup. The supported path management strategy has a direct impact on the useable subflows of the multipath extensions, as pointed out in Subsection 5.1.1.2. Therefore, in the optimal case CMT-SCTP can use two different subflows as demonstrated in the in Figure 6.8, whereas MPTCP is able to establish four subflows.

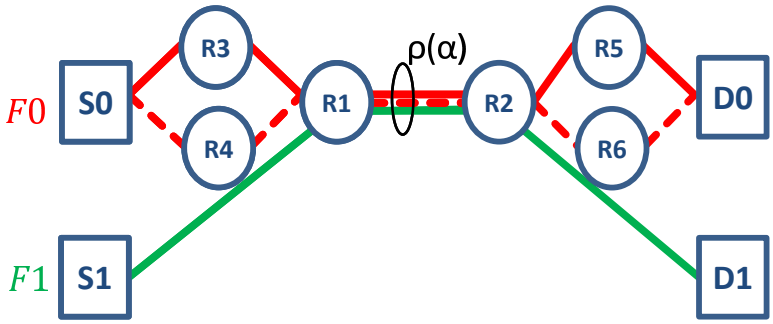


Figure 6.8: Fairness scenario 2: Shared bottleneck

6.3.2.1 Shared bottleneck model and theoretical discussion

The impact of the link and network centric flow fairness is the same. The multipath flow has to allocate the same resources as the singlepath flow. Therefore, fairness can be described for both approaches – like in Subsection 6.3.1 for the singlepath network – as:

$$B0 = \sum_{i}^{n} B_0^i = \frac{\rho(\alpha)}{2} \quad ; \quad B1 = B_1^0 = \frac{\rho(\alpha)}{2}. \tag{6.18}$$

In the IETF multipath fairness definition all subflows of a connection are only allowed to allocate as much capacity as a singlepath flow on the shared bottleneck link. Figure 6.5

[4] "do not harm", "be fair" and "balance congestion" see Subsection 3.2.2.2

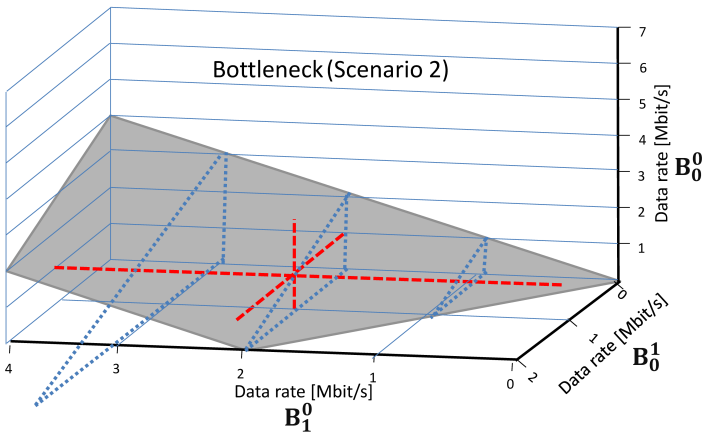


Figure 6.9: Simple fairness curve

demonstrates the fairness line on flow level also for this scenario. On subflow level this becomes more complex, that is in this setup for CMT-SCTP n is in maximum 2 and for MPTCP n in maximum 4. The "fairness curve" for CMT-SCTP on subflow level is illustrated in Figure 6.9. Here, the "fairness curve" transforms for CMT-SCTP with two subflows to a fairness plane for the capacity allocation of F_0^0, F_0^1 and F_1^0 [BDAR12b]. In addition, the figure demonstrates an optimal fairness point in a Cartesian coordinate system by using a low link capacity example. The example is illustrated in Figure 6.9 for a 4 Mbit/s shared bottleneck link capacity. While F_1^0 is allowed to occupy 2 Mbit/s, the subflows of flow $F0$ are allowed to use between 0 Mbit/s and 2 Mbit/s, as long the sum of both capacity allocations does not exceed 2 Mbit/s. An example of a symmetric share – both subflows of $F0$ with 1 Mbit/s – is illustrated by the red dotted lines. The overall system works inaccurate or unfair if the fairness plane is not hit by the congestion control. This also is valid if the path the subflow is assigned to is limited. If for example the path with the subflow F_0^1 is limited to 2 Mbit/s, the plane is cut off for a shared bottleneck link capacity of 6 Mbit/s ($B_1^0 = 3$ Mbit/s), like illustrated in the Figure 6.9. Here the fairness plane does not cover every combination of B_0^0 and B_0^1 summing up to 3 Mbit/s. The subflows of $F0$ have to share the 3 Mbit/s left. As a result, F_0^0 is allowed to take between 1 Mbit/s and 3 Mbit/s and F_0^1 is allowed to allocate between 0 Mbit/s and 2 Mbit/s[5].

The IETF fairness interpretation just focuses on the TCP flow definition, where a TCP flow is identified by a unique information tuple (see also discussion of TCB in Subsection 5.1.1.1). The resources provided by the user/endpoint are not reflected in this fairness definition. Thus, the allocation depends on the number of the flows instead of subflows provided. In the shared bottleneck there exists no relation between hardware characteristics (like bandwidth) provided by the user and the achieved performance of the flows. Therefore, it might be questioned if the sender $S0$ has a real motivation to spend efforts in the infrastruc-

[5]Of course, as long the sum of both does not exceed 3 Mbit/s.

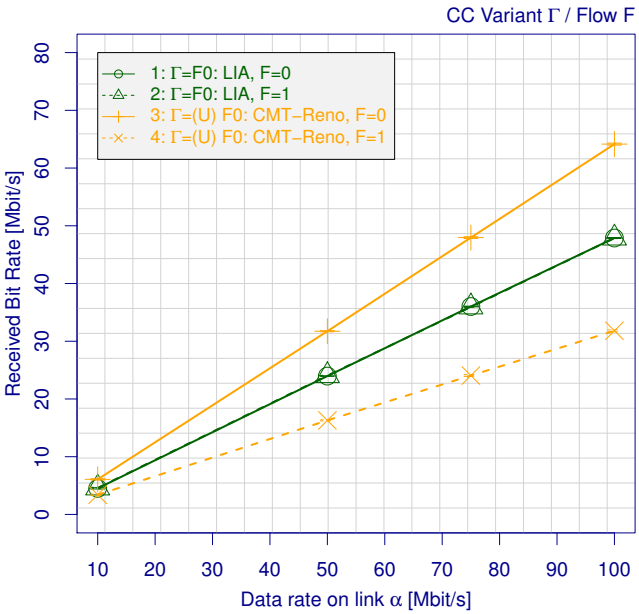


Figure 6.10: Shared bottleneck: Fairness on subflow and flow level

ture. Just to support the load balancing[6] without getting any personal benefit might not provide an adequate motivation.

6.3.2.2 Evaluation of the shared bottleneck scenario

The shared bottleneck scenario provides challenges on different levels. First the impact of fairness on flow level and on subflow level will be discussed. After that, the impact of different network characteristics are in the focus of this section, in particular with respect to link capacity, error rate and delay.

Subflow and flow fairness

Two simple experiments with the shared bottleneck scenario (see Figure 6.8) were carried out to demonstrate the difference between flow level and subflow level fairness in this setup. Both experiments used two flows. Flow $F1$ was a singlepath TCP flow with applied New Reno congestion control and flow $F0$ was CMT-SCTP with applied CMT-Reno CCC in the first experiment and LIA CCC in the second. The capacity of the shared bottleneck was varied from 5 Mbit/s to 100 Mbit/s. Figure 6.10 presents the results of this experiment. Curve #1 and Curve #2 represent the outcome of the flow fairness[7] achieved by the LIA CCC. Curve #3 and Curve #4 illustrate the resource allocation for the link centric subflow fairness view achieved by the CMT-Reno CCC. In case of LIA, both flows allocate the same fair share ($\frac{1}{2}\ \rho(\alpha)$). This allocation on flow level is in general expected by the Resource Pooling idea. Thus, it represents the expected behavior of LIA, OLIA and RP-MPv2. In contrast, CMT-Reno allocates as much link capacity as the flow provides subflows. Thus,

[6] It should be always kept in mind that the load balancing goal, with the corresponding balance congestion goal, is a strong motivation for the load sharing extension in the context of the IETF.

[7] It does not matter whether it is link or network centric view.

in case of CMT-SCTP the singlepath flow (F=1; Curve #4) gets $\frac{1}{3}$ of the capacity and the multipath flow (F=0 ; Curve #3) gets $\frac{2}{3}$.

In this context it should be mentioned that all coupled congestion controls achieved a fair allocation of the shared bottleneck link capacity in line with the Resource Pooling idea on flow level[8] in corresponding experiments. The next step is to discuss how accurate the goals of the Resource Pooling idea were achieved in this setup.

Capacity share

Again four experiment setups were created to discuss the accuracy of the CCC proposals with respect to their ability to achieve the fairness on flow level. Thus, $F1$ was always a singlepath TCP flow and $F0$ varied between TCP/MPTCP and SCTP/CMT-SCTP with the corresponding congestion controls, in particular New Reno, CMT-Reno, LIA, OLIA and RP-MPv2. The shared bottleneck resource was defined as one link with $\rho(\alpha)$ that varies the capacity from 5 Mbit/s to 100 Mbit/s. As mentioned before, all senders used 1 Gbit/s access links. The links themselves were connected by the routers $R1$ to $R6$ configured with RED queues as defined in Subsection 2.2.3. Detailed configuration overviews are given by the Table 6.4 for the results in the Figure 6.11 and by the Table 6.5 for the Figure 6.12.

All figures show the throughput difference $B1$-$B0$, i.e. the region with negative values on the y axis defines a sharing which is defined as unfair to TCP. Positive values indicate that the multipath flow is less aggressive than allowed. While this is still TCP fair, it is inefficient from the point of view of the multipath flow.

Two additional experiments with the uncoupled CMT-Reno congestion control were added, one for CMT-SCTP and one for MPTCP. Of course, a flow level fairness cannot be expected from CMT-Reno, but the outcomes of Curve #5 in both the Figures 6.12 and 6.11 show the difference between CMT-SCTP and MPTCP. This difference is the result of using two subflows for CMT-SCTP and four subflows for MPTCP. This subflow configuration for MPTCP and CMT-SCTP is used for all other multipath experiments, too. It should be mentioned that the observed accuracy of the CCC proposals depends more on the number of subflows as on the protocol[9] used in the setup. Thus, if MPTCP would be used with two subflows the same results as for CMT-SCTP with two subflows could be expected.

The optimal fairness points on flow level are again represented by two benchmarks with two competing singlepath flows, where $F0$ used SCTP in Figure 6.11 and TCP in Figure 6.12. The CCCs are not able to achieve a completely accurate behavior in every case, but a deviation up to 5% can be defined as acceptable, similar to the results of the singlepath setup.

LIA is unable to achieve the optimal fairness for higher link rates. Already for a link rate of 50 Mbit/s a deviation can be observed for CMT-SCTP. This deviation increases for higher link capacities. Thus, for 100 Mbit/s a deviation of 5% can be observed, which is acceptable. LIA for MPTCP is always unfair and even more unfair as for CMT-SCTP. Thus, for 100 Mbit/s a deviation of more than 5% can be observed.

Curve #3 shows the behavior of OLIA in the shared bottleneck scenario. The OLIA CCC works accurate for CMT-SCTP with two subflows (see Figure 6.11). A higher deviation can be identified in the experiments with MPTCP and four subflows (see Curve #3 and Curve

[8]The Figures A.3, A.4, A.5 and A.6 in the Appendix represent the results of the comparison with absolute values for the same experiment setup. They build the base for comparison with Figure 6.10.

[9]The figures with the absolute values in the Appendix – Figures A.3, A.4, A.5 and A.6 shows here more details

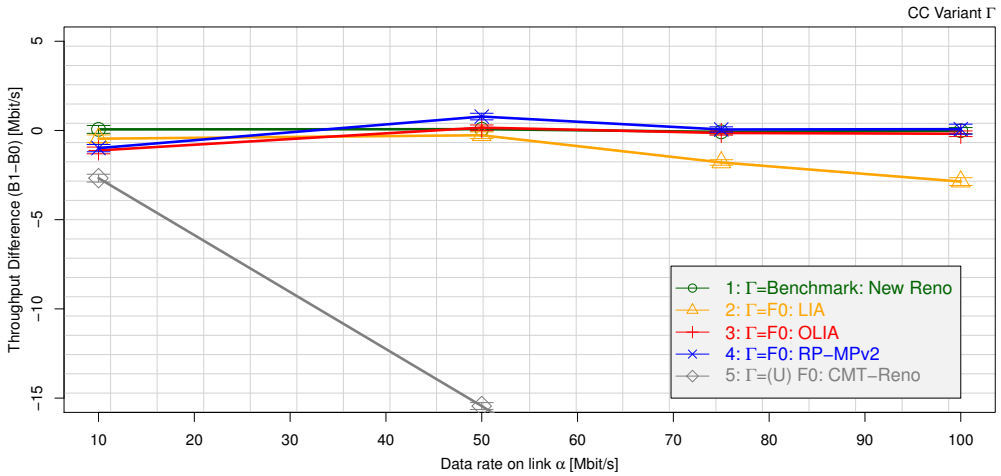


Figure 6.11: Shared bottleneck: SCTP and CMT-SCTP will all CCC proposals

Experiment	$F0$ protocol	$F0$ CC	$F0$ subflows	$F1$ protocol	$F1$ CC
1	TCP	New Reno	1	TCP	New Reno
2	CMT-SCTP	LIA	2	TCP	New Reno
3	CMT-SCTP	OLIA	2	TCP	New Reno
4	CMT-SCTP	RP-MPv2	2	TCP	New Reno
5	CMT-SCTP	CMT-Reno	2	TCP	New Reno

Table 6.4: Configuration for Figure 6.11

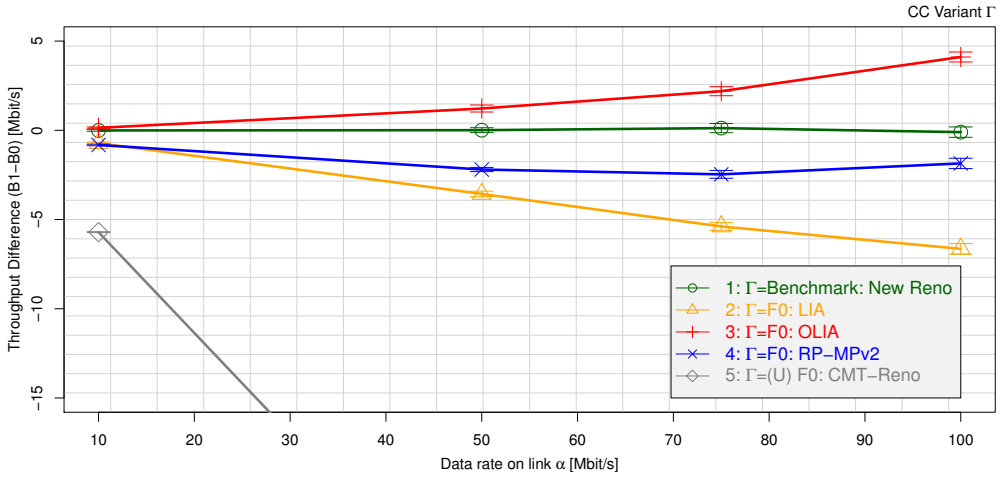


Figure 6.12: Shared bottleneck: TCP and MPTCP with all CCC proposals

Experiment	$F0$ protocol	$F0$ CC	$F0$ subflows	$F1$ protocol	$F1$ CC
1	TCP	New Reno	1	TCP	New Reno
2	MPTCP	LIA	4	TCP	New Reno
3	MPTCP	OLIA	4	TCP	New Reno
4	MPTCP	RP-MPv2	4	TCP	New Reno
5	MPTCP	CMT-Reno	4	TCP	New Reno

Table 6.5: Configuration for Figure 6.12

#4 Figure 6.12). OLIA does not perform as aggressive as it is allowed, thus the throughput is suboptimal for the MPTCP flow.

Curve #4 reflects the behavior of RP-MPv2. Like OLIA, RP-MPv2 works accurate for CMT-SCTP with two subflows but for MPTCP RP-MPv2 shifts to a too aggressive behavior like LIA. However, RP-MPv2 is less aggressive than LIA and, therefore, in acceptable borders (in maximum 2.5%).

This first real comparison on a common and comparable platform showed that all coupled congestion controls worked in the scenario they were designed for in an acceptable range, even if there were small deviations. However, this fairness behavior was achieved under nearly perfect conditions[10]. The next step is to investigate the behavior of the CCC proposals under less optimal conditions, by varying the delay and error rate.

Delay

Delay is an important network characteristic for the congestion controls. The delay has a direct impact on the RTT experienced before an acknowledgement can increase the congestion window. Furthermore, with increased delay there is more time required for fast retransmissions or to detect congestion. This has an impact on the CCC proposals and is, therefore, an important topic for evaluation. The same shared bottleneck topology as before (see Figure 6.8) is used to discuss the impact of the delay on the CCC proposals. The only difference is a variation of the delay on the shared bottleneck link α.

The LIA congestion control was chosen to discuss the general behavior in this scenario. The behavior of the OLIA CCC is comparable to LIA[11]. RP-MPv2 shows specific abnormalities with respect to the number of subflows, therefore it is discussed separately after the LIA results.

Figure 6.13 and Figure 6.14 show the impact of delay δ varying on the bottleneck link from 1 ms to 80 ms. Figure 6.13 shows the results for the LIA CCC in CMT-SCTP and Figure 6.14 in MPTCP. The different path management strategies for CMT-SCTP and MPTCP should be kept in mind during the discussion.

The impact of delay is for LIA – and also for OLIA – in principle similar for CMT-SCTP (see Figure 6.13) and for MPTCP (see Figure 6.14). Thus, the number of subflows has no fundamental impact for varying delays.

Both figures demonstrate an inaccurate behavior over the whole parameter range. While the experiments show an unfair behavior for LIA in CMT-SCTP and MPTCP with low delay (Curve #1 and #2 with δ=1 ms and 2 ms), they show a suboptimal multipath performance for high delays (Curve #4 and #5 with δ=60 ms and 80 ms). But again, the behavior is within the acceptable range.

One reason for this inaccurate behavior is of course the increased RTT. The longer it takes to increase the congestion window by an acknowledgment, the longer there is no room for an additional segment. This is caused by the aggressiveness factors of the CCC proposals (see in Subsection 6.2.2). Thus, there exists the situation in the sending process, where the data in flight is equal to the congestion window. Therefore, the sender stops the sending process in expectation of an acknowledgment to confirm sent data in a first step and to increase the congestion window in a second. If the calculation of the congestion window leads to

[10] Relating to delay (2 ms), error rate (0%) and nearly unlimited buffer.

[11] The corresponding results of the OLIA CCC are placed in the Appendix as Figure A.7 for the CMT-SCTP and as Figure A.8 for MPTCP.

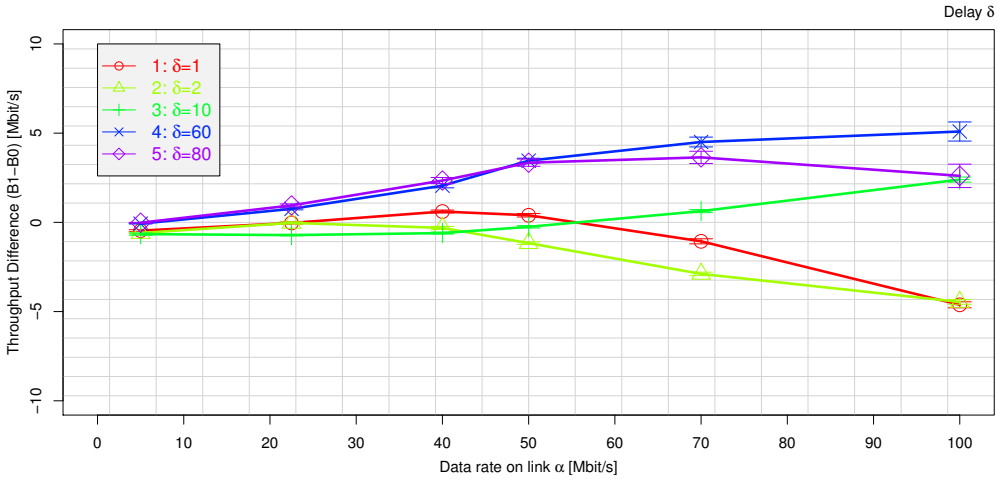


Figure 6.13: Impact of increased common delay for LIA in CMT-SCTP

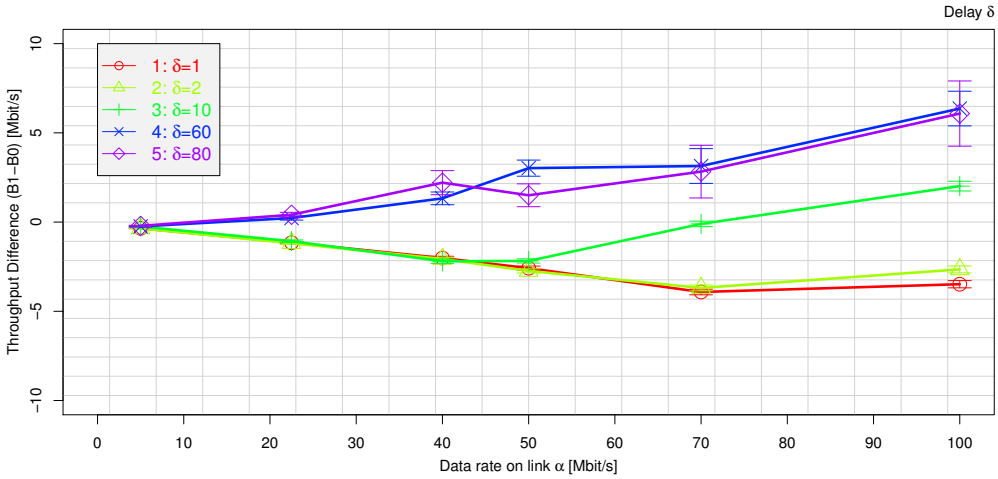


Figure 6.14: Impact of increased common delay for LIA in MPTCP

smaller values as in the optimal case, the window is not increased to the theoretically allowed maximum. For MPTCP this leads to higher fragmentation of data and in case of CMT-SCTP even no new segment will be sent until the window is large enough. The system has to wait for the next acknowledgments until the congestion window is increased enough that there is free window space for a new message. With low delays this effect is decreased, therefore behavior comparable to a singlepath connection can be expected. The longer the sender has to wait for the next acknowledgment, the less aggressive is the multipath flow.

However, the multipath flow can also be more aggressive than the singlepath flow if the calculation of the aggressiveness factor is inaccurate. This inaccurate aggressiveness calculation can already be caused by small rounding errors.

In contrast to LIA and OLIA, the path management strategy had a severe impact on the RP-MPv2 CCC. While RP-MPv2 performed in acceptable borders for CMT-SCTP (see Figure 6.15), the performance suffered a lot for MPTCP (see Figure 6.16). In case of a link capacity of 100 Mbit/s on link α, a high delay caused a deviation up to 15%. There should

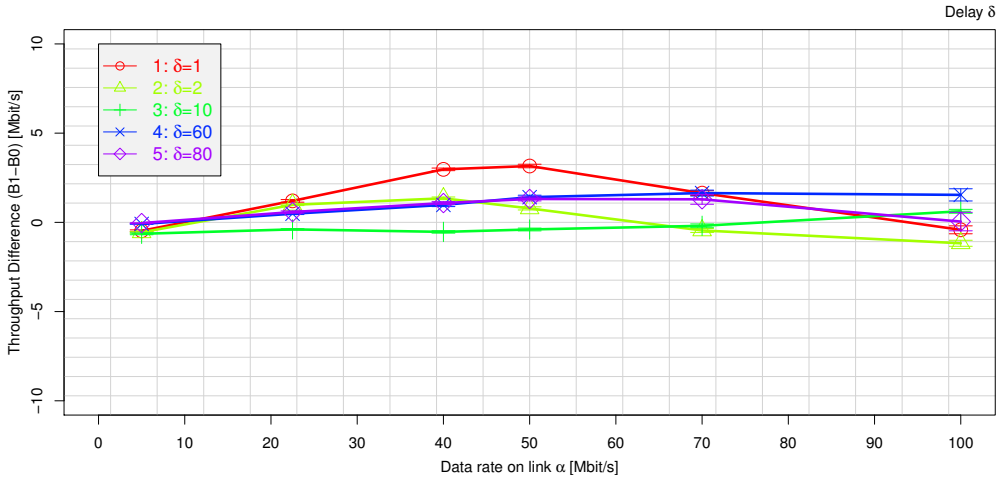


Figure 6.15: Impact of increased common delay for RP-MPv2 on CMT-SCTP

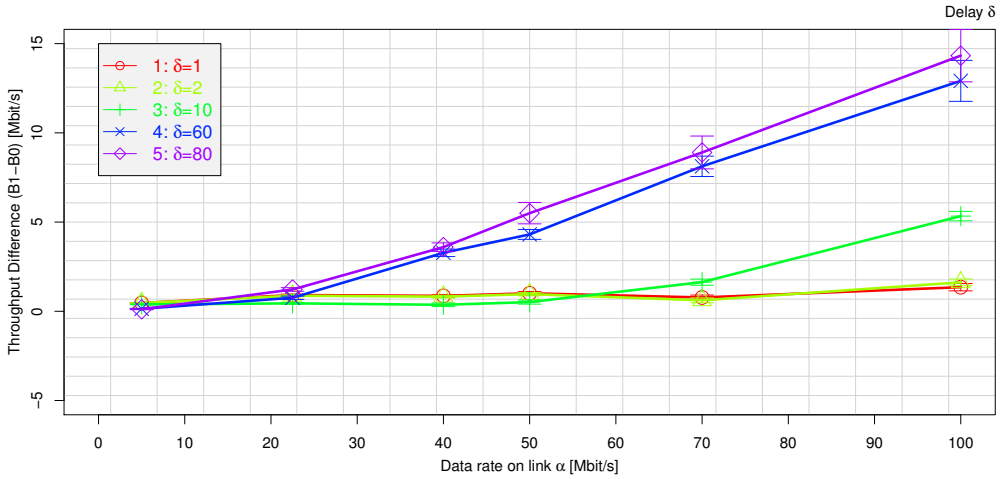


Figure 6.16: Impact of increased common delay for RP-MPv2 on MPTCP

be a broad agreement that a deviation up of 15% from the perfect fairness point is hard to accept, even if this less aggressive behavior is TCP-compatible with respect to the IETF fairness goals. The RP-MPv2 CCC applied to MPTCP preferred the singlepath flow much more than needed and, therefore, occupied less than a singlepath TCP flow would do in the same MPTCP scenario. Therefore, this does not fulfill to the *improve throughput* goal of the IETF multipath fairness (see Subsection 3.2.2.2).

One reason is the algorithm for the congestion window reduction [BDAR12b]. The RP-MPv2 CCC always leads to more window reduction than in case of OLIA or LIA. For RP-MPv2 the decrease and the increase are directly coupled to the congestion window. Chances are high that a congestion window decrease would lead to a congestion window size below one MSS which is prevented by the max function (see Subsection 6.2.1). In case of RP-MPv2, a path will not be used for any new data during a time span of one RTO if a loss occurs. The other subflow in CMT-SCTP interacts in this period like a singlepath flow and, therefore, as aggressive. But this behavior can only be observed for the CMT-SCTP scenario with two

subflows. In the MPTCP setup (see Figure 6.16) four subflows are used. Thus, if a loss occurs only one subflow is blocked during one RTO and the other three subflows are still in competition. The three subflows cause more congestion and, therefore, a decreased coupled congestion window. To block more than one path as an adaption of the algorithm is also not possible, because the endpoint is not aware about the routing and the bottlenecks. An endpoint cannot detect the cause of an error. If this would be possible this would describe a bottleneck detection. With a bottleneck detection approach no Resource Pooling would be needed.

Error rate

The new coupled congestion controls are all loss based congestion controls. Thus, every event that causes loss in the endpoint or in the network has a direct impact on the congestion control window. The ability to deal with different loss rates was tested by varying the packet error rate on the bottleneck from 0% to 0.4%. The same shared bottleneck setup as before was used. Again two experiments were carried out to demonstrate the impact on CMT-SCTP with two

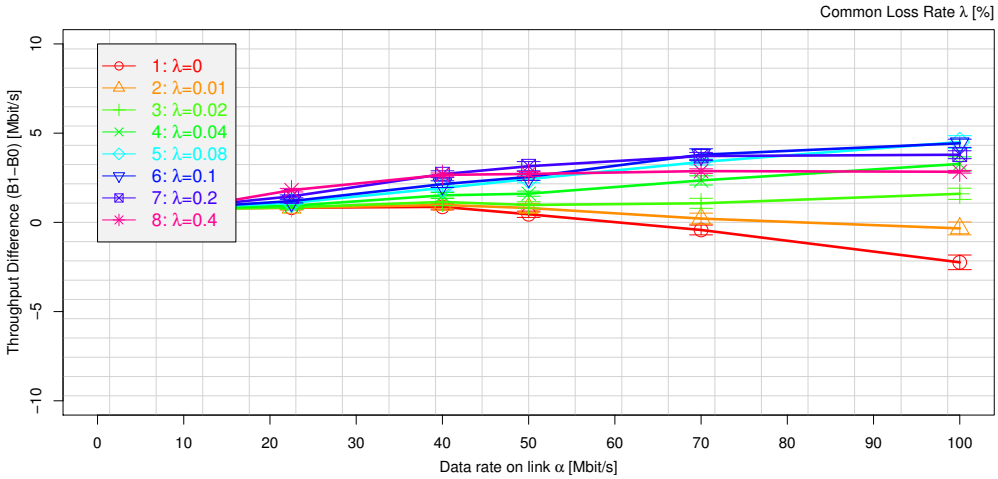


Figure 6.17: Impact of increasing error rate on LIA and CMT-SCTP

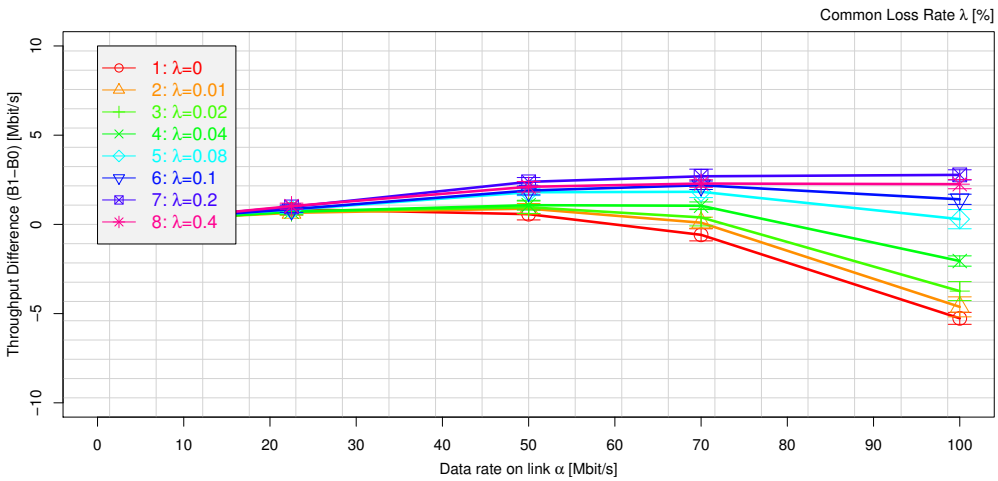


Figure 6.18: Impact of increasing error rate on LIA and MPTCP

subflows and MPTCP with four subflows. The flow $F1$ was a singlepath TCP flow with New Reno as congestion control and the flow $F0$ used LIA as CCC. The experiment results for OLIA and RP-MPv2 are also attached in the Appendix as Figure A.9, A.11, A.12 and A.10, but they show less impact on the accuracy than LIA. LIA showed a deviation in acceptable borders with respect to an increased error rate for both multipath protocol extensions (for CMT-SCTP in Figure 6.17 and for MPTCP in Figure 6.18). It could be observed that the general tendency was independent of the number of subflows, even if the deviation was more significant for MPTCP than for CMT-SCTP. Thus, while SCTP showed an unfair deviation of 3% for a 100 Mbit/s link rate, a deviation of 5% was observed for MPTCP.

The behavior was not perfect for LIA, but can be argued as acceptable in comparison to other the scenarios discussed in the following. RP-MPv2 and OLIA are more close to the ideal share of the link capacity.

6.3.2.3 Comparable conditions for a fair sharing

So far, the fairness analysis focused on the shared bottleneck link characteristics, in particular with respect to error rate, delay and bandwidth. Goal of this analysis was a comparison of the CCCs and the protocol extensions under "comparable conditions" (see Subsection 3.2.2).

However, the CCC mechanisms are reflecting the Resource Pooling idea. Thus, the performance of a multipath flow is the result of the summed up subflow throughputs and it makes no difference which subflow provides the throughput as long as the multipath flow in sum can provide it.

Thus, the "comparable conditions" are defined on flow level for the CCCs. The capacity allocation to each multipath subflow in a shared bottleneck does not have to be equal and, therefore, also not the bandwidth characteristics of the subflow paths. The goal of this subsection is to investigate, whether the capacity aggregation can have an impact on the fairness behavior of the CCCs.

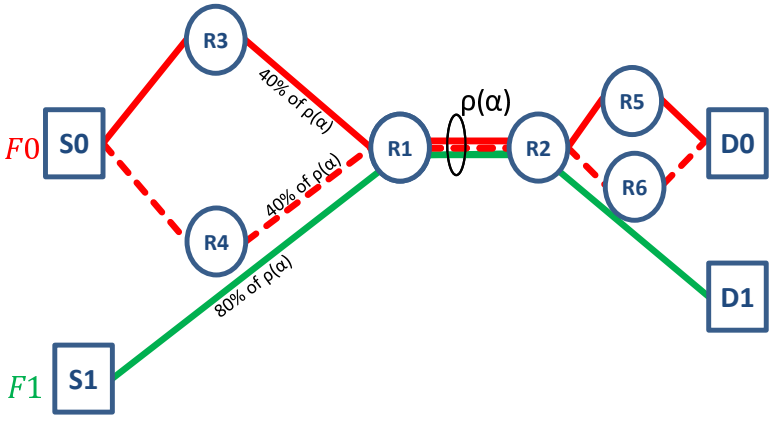


Figure 6.19: Fairness scenario 3: Full bottleneck (with limited access links)

The shared bottleneck scenario has two routers ($R3$,$R4$) to support the access via different paths to the network as illustrated in Figure 6.19. The access links were connected via these routers and were used for the experiment to manipulate the path characteristics for the subflows of flow $F0$. The capacities of the links (R3↔R1) and (R4↔R1) were configured to 40% of the capacity of $\rho(\alpha)$ each. The capacity of the bottleneck link in the experiment design

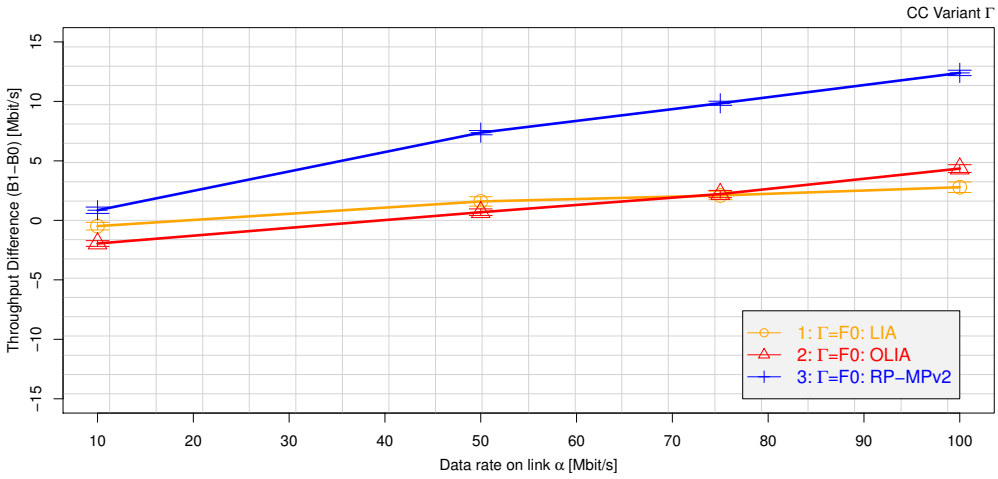


Figure 6.20: Limited access links for CMT-SCTP

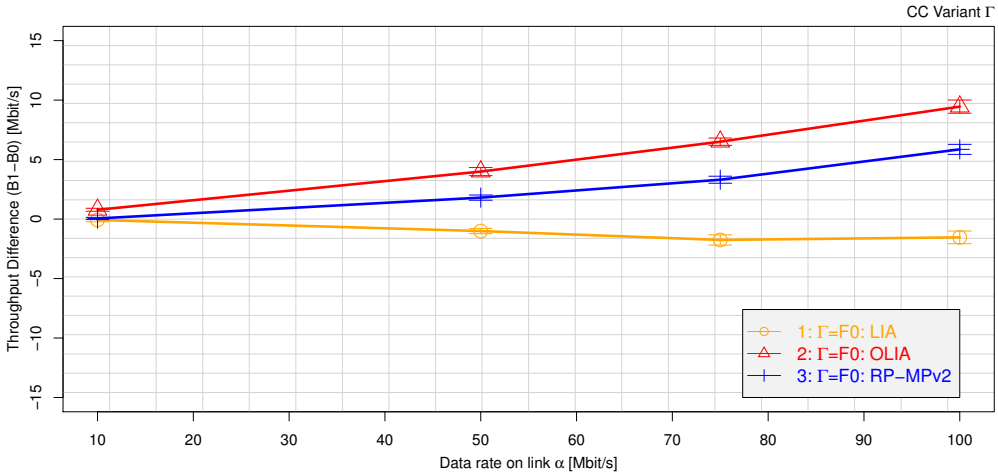


Figure 6.21: Limited access links for MPTCP

ranged from 5 Mbit/s to 100 Mbit/s. The capacity of one access link of the multipath sender alone was not able to fill the bottleneck up to 50%, but the complete multipath flow was able to achieve in theory 80% of the capacity $\rho(\alpha)$. The capacity of the access link between $S1$ and $R1$ was decreased to 80% of $\rho(\alpha)$ to achieve the "comparable conditions". Clearly, both flows could achieve in sum the same maximum bandwidth on the shared bottleneck. In all experiment setups RED queues were used in the routers. All CCC proposals were deployed on CMT-SCTP and MPTCP to demonstrate the effects. Figure 6.20 shows the results of this experiment for CMT-SCTP and Figure 6.21 for MPTCP.

It must be understood that a transport protocol with a loss-based congestion control has the tendency to fill the system with as much data as possible, as long as it remains within the limits of the window mechanism. That means, the transport protocol also has the tendency to fill the queues in the system in front of a bottleneck link. Therefore, the behavior of the CCC in general might depend on the limited link configuration as it causes a new potential bottleneck. Therefore, the routers preceding the bottleneck have to buffer the segments. If

this buffer is limited this has a possible impact on the CCCs.

The LIA CCC performed close to the optimal fairness points in case of CMT-SCTP with two subflows (see Curve #1 in Figure 6.20) and for MPTCP with four subflows (see Curve #1 in Figure 6.21). LIA always prefers the congestion window of the strongest link based on the throughput estimation during the aggressiveness factor calculation. Thus, only one subflow on one path was filled with data, such that the subflow was only able to fill the queue at either router $R3$ or $R4$. As a result, there was no difference if two subflows or four were used, because only the best one was able to increase the congestion window with the chance to fill the queues. After the window of the strongest subflow was reduced, the send process continued with the alternative subflow. Even if the chance was high that an alternative subflow became the new strongest path, it was again only one subflow that filled the queue. This led to flapping among all available subflows, but this decreased the aggressiveness only slightly. The more subflows, the higher the frequency of this flapping was, but in the end the LIA CCC interacted always like one queue filling subflow.

In principle the behavior of the LIA CCC could also be observed for the OLIA CCC in the CMT-SCTP setup (Curve #2 in Figure 6.20). OLIA privileges subflows as LIA, but OLIA prefers groups of subflows instead of one specific subflow (see Subsection 6.2.3). However, in the CMT-SCTP setup with two subflows this made no difference. Both subflows were assigned to different groups and, therefore, one subflow was preferred more and the other less similar to the behavior of the LIA CCC. The send behavior with one preferred subflow was comparable to a "normal" singlepath subflow.

However, this behavior changed in the MPTCP experiment. Curve #2 in Figure 6.21 shows a deviation of 10% for a bottleneck link capacity of 100 Mbit/s. The performance of the multipath flow suffered significantly because in the MPTCP setup more than one flow tried to increase its congestion window. As consequence, one subflow per path was preferred and, therefore, at least two subflows filled the queues and caused loss in different queues. More loss caused more cycles with decreased congestion window and that in sum decreased the aggressiveness.[12]

RP-MPv2 exhibits an inaccurate behavior beyond 10% in the CMT-SCTP setup (see Curve#2 in Figure 6.20), which is comparable to the behavior of the OLIA CCC in MPTCP. Even if RP-MPv2 does not prefer any subflow in contrast to OLIA, this led to the same result in case of CMT-SCTP because two subflows tried to expand their windows as much as possible and filled two queues at the same time. Therefore, more loss was observed than for OLIA in MPTCP. The effect for RP-MPv2 in MPTCP was less significant, because the strongly coupled congestion window was decreased by every loss. Because for MPTCP the queues were filled by four subflows the congestion occurred – more or less – at the same time and this led to a decreased congestion window. It might surprise that this decreased coupled congestion window had a positive effect on the overall system behavior, but the increase factor of RP-MPv2 is calculated based on this reduced congestion window. Thus, the lower the aggressiveness factor is, the lower is the congestion window growth between the send cycles. Thus, less packets are enqueued in the router queues. Therefore, more time is provided to send without loss caused by the router queues and in sum this leads to an increased throughput. A limited congestion window has, therefore, a positive impact on the overall system. This effect of the scheduling process is discussed in more detail in Section 7.6.

In theory this setup provides "comparable conditions" on the shared bottleneck, but it

[12]The Figure A.14 for MPTCP and the FigureA.13 for CMT-SCTP can be found in the Appendix.

also shows different results depending on the deployed CCCs. Even if the behavior is fair in the context of the IETF multipath goals[13] it is not very accurate. Thus, even if it fits the IETF multipath fairness goals, the results show neither a link centric nor a network centric flow fairness.

6.3.3 Scenario 3: Half bottleneck with single- and multipath flow

In the singlepath and shared bottleneck scenarios, the fairness definition focused on the relationship of flows, subflows and capacity. The link centric and network centric flow fairness view led to the same capacity share. The fair sharing depended only on the question whether the fairness is based on the number of flows or subflows. The next scenario, which is denoted as half bottleneck, is going to demonstrate the ambiguity related to multi-homed configurations with respect to link centric and network centric flow fairness. A new scenario was created to demonstrate this as shown in Figure 6.22. The senders $S0$ and $S1$ and the access links to the shared network stay the same as in the shared bottleneck scenario, but the connecting topology is changed. Now, the sender $S0$ transfers data to the receiver $D0$ via two links (R1↔R2 and R3↔R4). The subflow $F1$ belongs to the connection between $S1$ and $D1$ and shares the network segment R3↔R4 defined as new bottleneck link β. The link α is used by one subflow of $S0$ in case of CMT-SCTP and by two subflows in case of MPTCP. The competition exist on the link β, which in case of CMT-SCTP is used by one subflow of $S0$ and one subflow of $S1$ and in case of MPTCP by two subflows of $S0$ and one subflow of $S1$. The subflow distribution for CMT-SCTP is illustrated in Figure 6.22. One major goal of the following experiments was to identify the impact of the *balance congestion* goal.

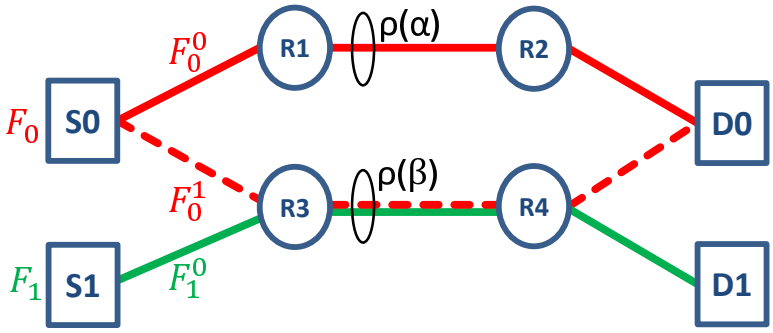


Figure 6.22: Fairness Scenario 3: Half bottleneck for singlepath

6.3.3.1 Half bottleneck model and theoretical discussion

The main challenge of this scenario is a fair sharing of $\rho(\beta)$ under the constraints of $\rho(\alpha)$ for the network centric fairness view on flow level. The fairness regarding the bandwidth distribution can be considered as hard to identify and to monitor, in particular with respect to the *balance congestion* goal. Before discussing the results for these mechanisms, the fairness goals have to be defined. The following definitions are valid as long as a subflow i is an element of a set of subflows that are routed via link α and a subflow j is an element of the set of

[13]The CCC achieves in sum more performance than a singlepath flow but also is less aggressive than the singlepath flow

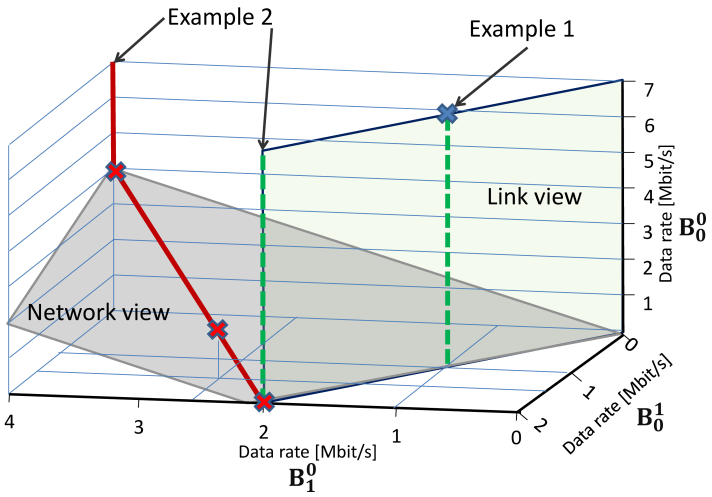


Figure 6.23: Fairness example

subflows routed via β. In the link centric flow fairness, the allocation is:

$$\sum_{i}^{n} B_0^i = \rho(\alpha) \quad ; \quad \sum_{j}^{n} B_0^j = B_1 = \frac{\rho(\beta)}{2}. \tag{6.19}$$

The green plane in Figure 6.23 illustrates the result of this allocation for CMT-SCTP. Here, the fairness plane is based on the behavior of $F0$ and $F1$ with respect to the link capacity of β. Thus, only the link capacity of β has to be shared among the subflows F_0^1 and F_1^0. The behavior of F_0^0 has no impact. The allocation of F_0^0 is always 100% of $\rho(\alpha)$. As Example #1 a specific point is marked with a blue cross in Figure 6.23. This cross represents the fairness point for a topology setup where $\rho(\alpha)$ is 7 Mbit/s and $\rho(\beta)$ is 2 Mbit/s. A link centric flow fairness for flow $F0$ is achieved when the throughput of the subflow assigned to α (B_0^0) is 7 Mbit/s and the throughput of the second subflow (B_0^1) – in competition with $F1$ – is half of $\rho(\beta)$ ($2*\frac{1}{2}=1$). Therefore, the flow $F1$ gets 1 Mbit/s and the flow $F0$ gets 8 Mbit/s. The allocation of the capacity $\rho(\beta)$ to F_0^1 and to flow $F1$ remains the same as long as $\rho(\beta)$ is fixed, even if $\rho(\alpha)$ will be increased or decreased.

In contrast to the link centric flow fairness, the network centric flow fairness considers the complete network characteristics in this scenario. Thus, network centric flow fairness must consider the same plane as in the shared bottleneck scenario. The only difference to the shared bottleneck scenario is, that the "resource to share" consists of two instead of one link and, furthermore, the resulting resource distribution depends on the difference between the capacities $\rho(\alpha)$ and $\rho(\beta)$. That is, the multi-homed flow between S0 and D0 has to restrain its resource usage on the bottleneck, when the upper path (which is used exclusively) gets more capacity. In general three different cases can be distinguished:

- $\rho(\alpha) = 0$: The flow $F0$ can then be considered as a singlepath flow with respect to its behavior; the bandwidth allocation is then identical to the singlepath scenario (see Subsection 6.3.1.2).

- $\rho(\alpha) \geq \rho(\beta)$: In this scenario, the multipath flow can be considered as a singlepath flow on link α. The capacity allocation for the singlepath flow $F1$ is $\rho(\beta)$ and of the multipath flow $F0$ is $\rho(\alpha)$.
- $0 < \rho(\alpha) < \rho(\beta)$: In this case, the third IETF multipath goal ("balance congestion", see Subsection 3.2.2.2) has to be taken into account. Here, the multipath flow should move as much traffic as possible off its most congested path. With a growing $\rho(\alpha)$, more and more bandwidth has to be moved from F_0^1 to F_0^0 to fulfill the condition of $F_0^0 + F_0^1 = F1$. The fairness can be described in general by the following calculation, as long as a subflow i is an element of a set of subflows that are routed via link α and a subflow j is an element of the set of subflows routed via β:

$$\sum_i^n B_0^i = \rho(\alpha) \ ; \ \sum_j^n B_0^j = \max\left\{0, \frac{\rho(\beta) - \sum_i^n B_0^i}{2}\right\} \ ; \ B_1 = \rho(\beta) - \sum_i^n B_0^i. \quad (6.20)$$

In order to demonstrate the difference between the link centric and the network centric flow fairness, Example 2 for CMT-SCTP is illustrated in Figure 6.23. This example is based on the assumption that $\rho(\beta)$ has 4 Mbit/s and $\rho(\alpha)$ increases from 0 Mbit/s to 7 Mbit/s. A symmetric share of all subflows on all paths is considered here, in order to simplify the illustration.

Figure 6.23 shows two lines for Example 2, the green dotted line represents the link centric flow fairness and the red solid line represents the network centric flow fairness. For the green dotted line the fair sharing is like in Example 1. The focus is only on the fair sharing of $\rho(\beta)$ between B_0^1 and B_1^0 which results in 2 Mbit/s independent from $\rho(\alpha)$.

However, the ratio for all subflows is different for the network centric flow fairness in the red line. At $\rho(\alpha) = 0$ the subflow F_0^0 gets 0 Mbit/s and $\rho(\beta)$ can be shared for the other subflows like in the singlepath scenario, thus F_0^1 and F_1^0 get each 2 Mbit/s (see lower red cross in Figure 6.23). Another example for $\rho(\alpha)$=1Mbit/s is illustrated by the middle red cross where F_0^0 gets 1 Mbit/s, F_0^1 gets 1.5 Mbit/s and F_1^0 gets 2.5 Mbit/s. Thus, in sum both flows get 2.5 Mbit/s. The highest red cross illustrates the point where $\rho(\alpha)$=$\rho(\beta)$, from this point the subflow F_0^1 gets no more additional resources. Thus, the resources allocated to $F0$ are equal to $\rho(\alpha)$ (for the red cross 4 Mbit/s) and for $F1$ equal to $\rho(\beta)$.

To be clear here, the shared bottleneck scenario (see Figure 6.22) and the half bottleneck scenario do not differ in the resource capacity provided, they only differ with respect to the calculation. The challenge of the network centric view on flow level in this scenario is not to adapt to the topology only, it also has to consider the limitations of the usable links. Therefore, the CCCs implementing the Resource Pooling idea to achieve network centric flow fairness have to adapt to the network characteristics even in this scenario. This is a much more complex task as implemented in the singlepath congestion control which addresses a link centric fairness on subflow level and also a different task compared to the shared bottleneck scenario.

It should be noted again that the estimation of the available resources of all possible paths and their combination is much easier in this simple example scenario than in the Internet and no endpoint or ISP is aware of all the connection details. Thus, a real control of the achieved results is not possible by any entity of the system. This should be discussed as a real drawback of the IETF fairness definition. This thesis does not argue that the fairness interpretation is wrong, it only argues that it is hard to control even if it is deployed correctly. If it is deployed correctly is discussed in the next subsection.

6.3.3.2 Evaluation of half bottleneck with singlepath flow

First it should be discussed which impact the *balance congestion* goal has on the CCC proposals and what kind of performance can be expected. The topology of Figure 6.22 is used to demonstrate this in different experiments. For these experiments the capacity of α was static and set to 20 Mbit/s.

Figure 6.24 demonstrates the expected results for the link centric and network centric flow fairness. Both fairness views are illustrated by two flows, a singlepath flow (dashed curves) and a multipath flow (solid curves). The blue colored curves reflect the link centric fairness, where the capacity sharing of β defines the gradient of both curves. The singlepath flow starts with 0 Mbit/s and the multipath flow curve starts with an offset of the static $\rho(\alpha)$ on the y axis.

The green curves illustrate the network centric flow fairness with respect to both cases where $\rho(\alpha) > \rho(\beta)$ and $\rho(\alpha) \leq \rho(\beta)$. For $\rho(\alpha) > \rho(\beta)$ the capacity of α is occupied by the multipath flow (solid line) and the capacity of β by the singlepath flow (dashed line). Therefore, if both bottlenecks provide in sum a capacity of 30 Mbit/s ($\rho(\beta)$=10) the singlepath flow gets 10 Mbit/s and the multipath flow gets 20 Mbit/s. In contrast, in case of $\rho(\alpha) \leq \rho(\beta)$ both flows have to occupy exactly the same capacity. Thus with, e.g., $\rho(\beta)$=60 Mbit/s both flows have to share the complete capacity (80 Mbit/s) with an allocation of 40 Mbit/s to each flow.

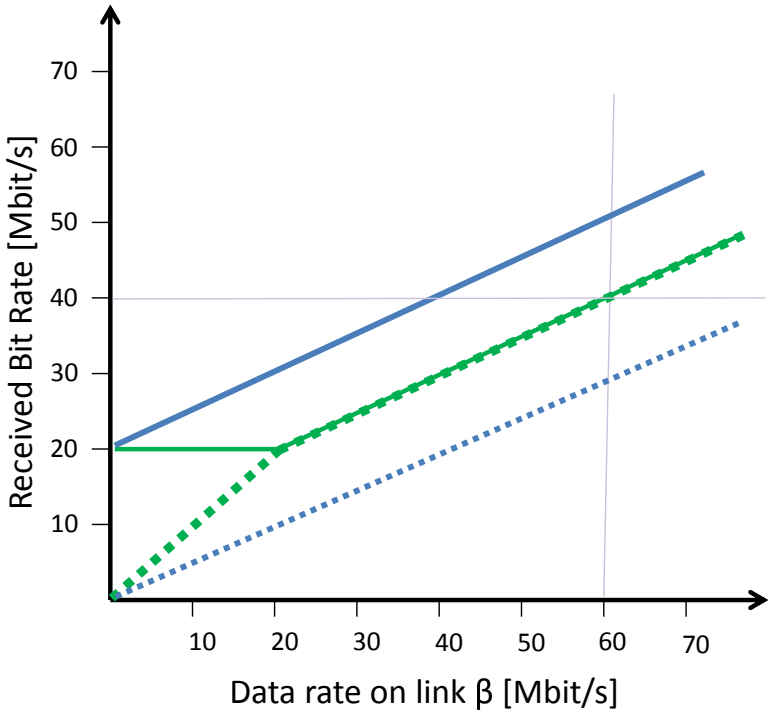


Figure 6.24: Fair sharing example for $\rho(\alpha)$=20 Mbit/s

Figure 6.25 shows the results for CMT-SCTP and 6.26 for MPTCP by using the same capacity configuration (α=20 Mbit/s) in the simulation setup. Four experiment setups are presented in every figure, where the sender $S1$ always use a singlepath TCP flow $F1$ and the

sender $S0$ a multipath flow $F0$. The sender $S0$ used different CCC proposals for every experiment. In this discussion for every experiment again two result curves have to be considered for the multipath flow and the singlepath flow. The dashed curves with the even numbers represent the multipath flow $F0$, whereas the solid curves with the odd numbers represent the singlepath flows $F1$.

Curve #7 and #8 represent the behavior of the CMT-Reno congestion control which represents the baseline for the link centric fairness on subflow and flow level, which is comparable to the blue line in Figure 6.24. The curves illustrate the behavior as it can be expected for this fairness view.

Curve #1 and #2 represent the behavior of LIA, Curve #3 and #4 of OLIA and Curve #5 und #6 for RP-MPv2. None of the CCC proposals shows the expected behavior for the network centric flow fairness. Even worse, with increased $\rho(\beta)$ the fairness behavior shifts more and more to a reversed link centric fairness behavior, where the singlepath flow connection gets more of the shared link capacity than the multipath flow (compare with link centric flow fairness of Curve #7 and #8).

In case of $\rho(\beta)$ <20 Mbit/s an acceptable behavior can be observed for OLIA and RP-MPv2 deployed on CMT-SCTP and MPTCP. Both CCC proposals were able to detect the capacity distribution and allocate the resources as the links α and β provide them. OLIA and RP-MPv2 are able to balance the congestion and to provide the whole capacity of link β to the singlepath flow. Here the network centric flow fairness is achieved. The LIA CCC demonstrates this ideal behavior only in the MPTCP setup (see Curves #1 and #2 in Figure 6.26). In the CMT-SCTP setup (see Figure 6.25) the multipath flow $F0$ allocates more than the 20 Mbit/s.

In case of $\rho(\beta)$ ≥20 Mbit/s the behavior of all CCC proposals becomes very inaccurate with respect to the network centric flow fairness view. For every CCC proposal in both protocols an allocation can be observed where the singlepath and multipath flow cross each other. But except for this point a fair sharing of the resource is not achieved as illustrated in Figure 6.24.

Even if the multipath flow can use the additional access link, the singlepath flow is able to allocate e.g. in case of LIA nearly twice as much capacity of the shared link as the multipath flow. With this ratio neither the link nor the network centric flow fairness view is addressed, it only fits the IETF fairness definition to be equal or better than the strongest singlepath flow and less aggressive than TCP.

And even worse, none of the CCCs – in particular not RP-MPv2 – was able to saturate the link for an increased $\rho(\beta)$ like the CMT-Reno congestion control. Figure 6.27 illustrates the overall throughput of as sum of the MPTCP flows $F0$ and the TCP $F1$. While the uncoupled congestion control CMT-Reno (see Curve #4 in Figure 6.27) is able to utilize the complete network capacity, the CCC proposals with the additional singlepath flow just achieve an overall capacity allocation of around 85% for $\rho(\alpha)$=20 Mbit/s und $\rho(\beta)$=80 Mbit/s in case of RP-MPv2. The insufficient behavior is not as drastic in case of LIA in Curve #1 and in case of OLIA in Curve #2, but even here hard to accept.

Even if the preference for the singlepath flow and the decreased network utilization fit the fairness definition of the IETF, the experiments demonstrate an inaccurate and unpredictable behavior in a competition with a singlepath flow. This behavior might be acceptable from the IETF perspective, but it decreases the network utilization for the ISP and brings in the worst case no benefit for the user who provides – and pays – multiple access links.

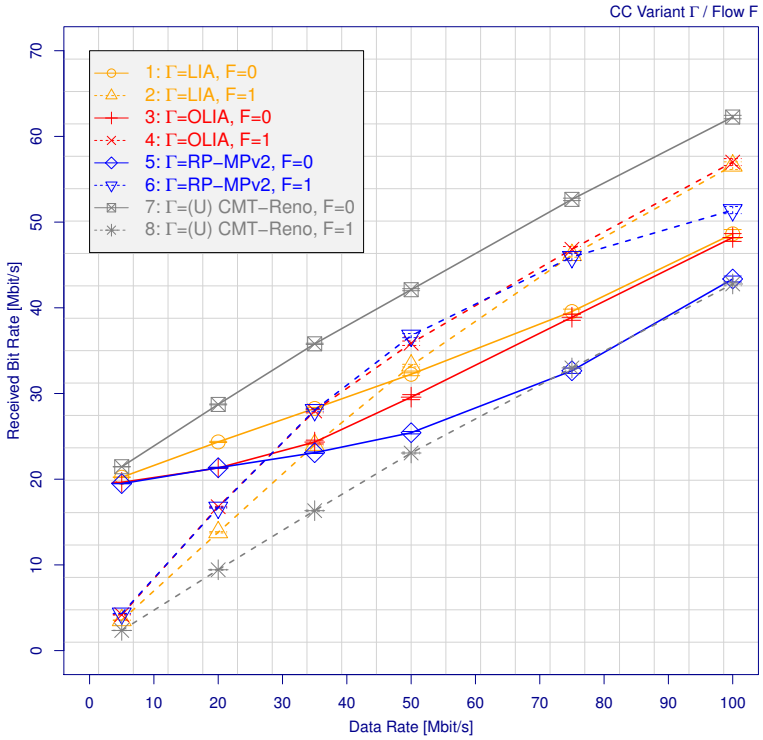


Figure 6.25: Scenario 4: TCP vs. CMT-SCTP

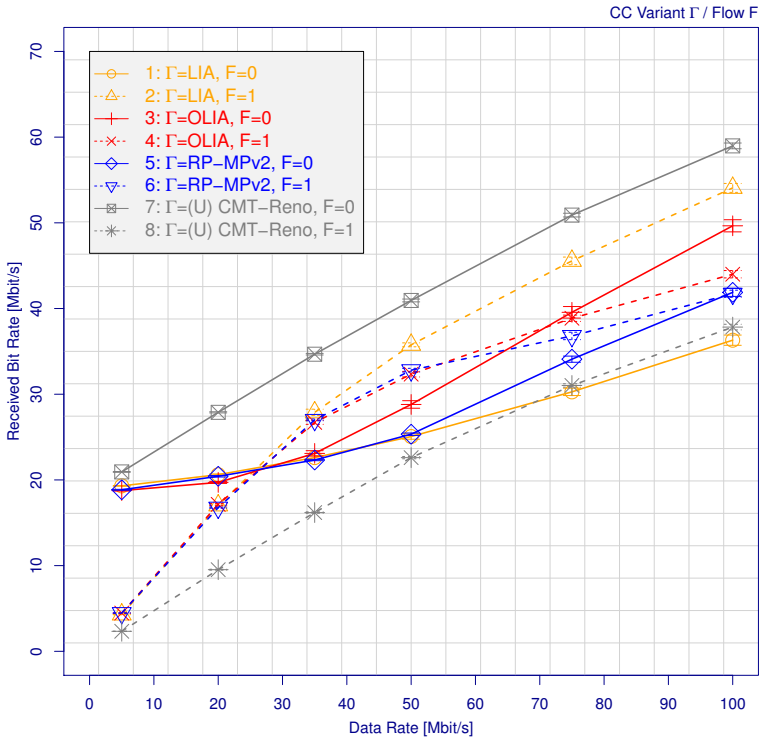


Figure 6.26: Scenario 4: TCP vs. MPTCP

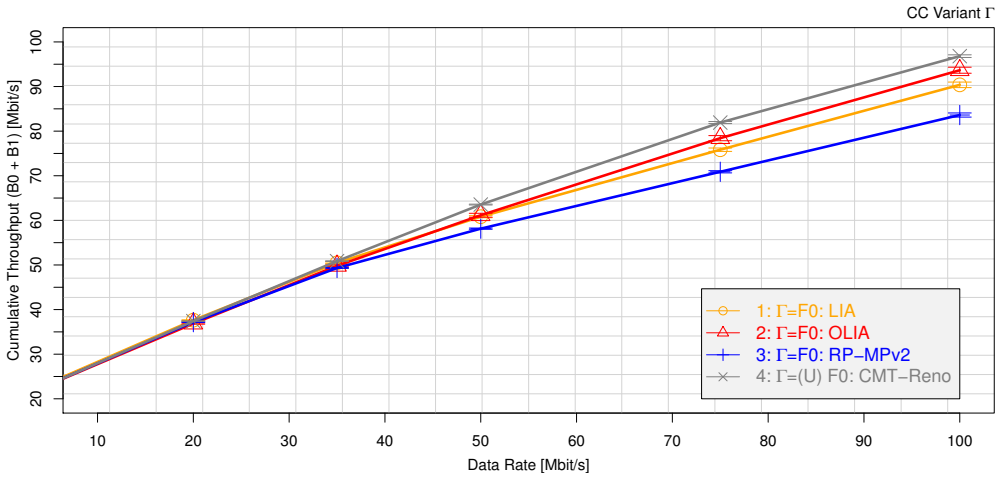


Figure 6.27: Overall performance of both flows

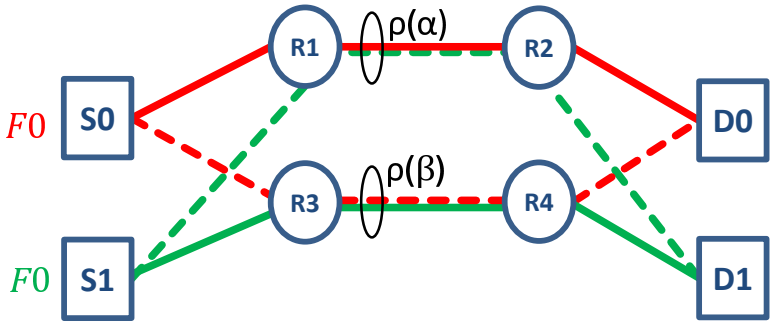


Figure 6.28: Fairness Scenario 4: Half Bottleneck for multipath

6.3.3.3 Evaluation of half bottleneck with multipath flows

This evaluation investigates if an important fairness goal is missed in the current IETF fairness discussion focusing on singlepath TCP connections only. An experiment was designed to make the open issue visible by replacing the singlepath flow by an additional multipath flow in the half bottleneck setup. Thus, the setup shifts to a multipath dominated network and the fairness issue with the singlepath TCP flow does not exist anymore. Figure 6.28 illustrates the experiment setup connecting two multi-homed endpoint pairs via the same network. Thus, sender $S0$ and $S1$ both transmit one subflow via the network by using independent access links. Both endpoints support a routing in a way that in every case the same number of subflows is routed via the bottlenecks between R1↔R2 (bottleneck α) and R3↔R4 (bottleneck β). The capacity of α is configured static as $\rho(\alpha)$=20 Mbit/s and the capacity of β is ranging from 5 Mbit/s to 100 Mbit/s. By using the CCC proposals the goal for both protocols and both fairness views – network or link centric flow fairness – leads to the same expected behavior, where two flows share the resources in a 50:50 manner. The experiment covers the behavior in case of $\rho(\alpha) \neq \rho(\beta)$ and $\rho(\alpha) = \rho(\beta)$.

Figure 6.29 and Figure 6.30 represent the deviation from the optimal fairness point for the

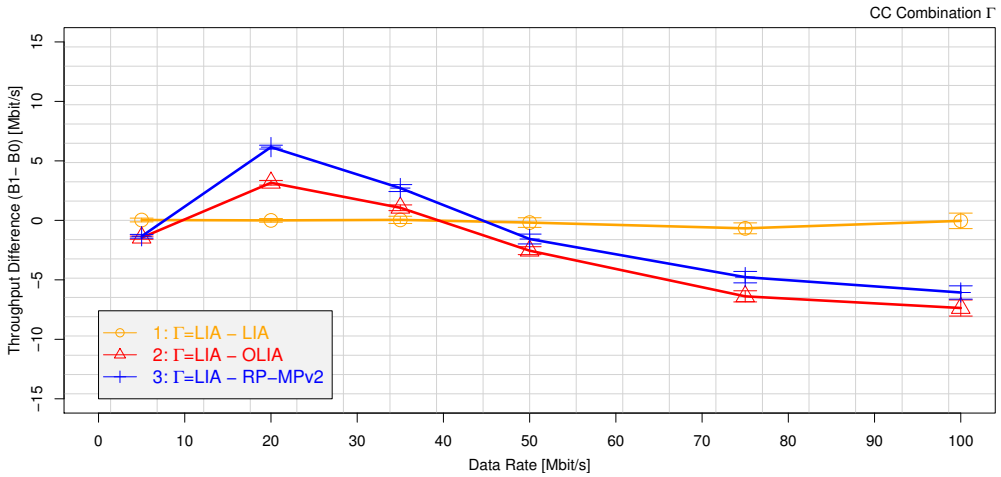


Figure 6.29: Multipath fairness depending on $\rho(\beta)$: LIA in comparison with all CCCs for CMT-SCTP

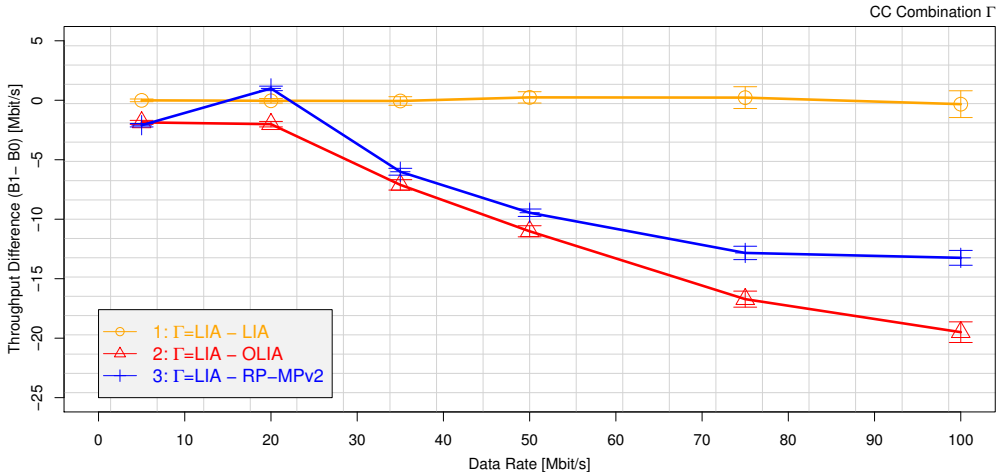


Figure 6.30: Multipath fairness depending on $\rho(\beta)$: LIA in comparison with all CCCs for MPTCP

example of LIA as basic CCC[14]. The Curve #1 in Figure 6.29 for CMT-SCTP and Figure 6.30 for MPTCP represents the baseline for this experiment by using LIA in competition with another LIA flow. Both figures show that the LIA CCC is fair to itself for MPTCP and CMT-SCTP. Independent of the link rate each multipath flow with the LIA CCC is able to allocate $\frac{1}{2}$ of the provided capacity at the bottlenecks. This behavior is changed drastically if the congestion control of flow $F0$ is changed to OLIA or RP-MPv2. While for CMT-SCTP (see Figure 6.29) the deviation oscillates around the optimal fairness point in a range of $\pm 10\%$ depending on $\rho(\beta)$, the deviation for MPTCP (see Figure 6.30) exceeds a deviation of 15% for RP-MPv2 (Curve #3) and 20% for OLIA(Curve #2). Obviously, the CCC proposals are unfair to each other. A specific multipath aggressiveness cannot be described just by the definition of a less aggressive behavior against singlepath flows only. The current fairness

[14]The Figures with the absolute values are attached in the Appendix as A.15 and Figure A.16.

definition must be redesigned to consider this in the network centric fairness view on flow level.

The observations for LIA also can be applied to the behavior of OLIA and RP-MPv2. The CCC proposals are fair to themselves but show an unfair behavior towards other CCC proposals. The result curves with absolute values can be found in the Appendix.

Altogether, this raises a new question with respect to multipath fairness that is not part of the IETF fairness definition and was not discussed so far. The question is open how a multipath flow is supposed to share the resources with another multipath flow using a different CCC. And the results show that this question matters because all known CCC proposals show a different aggressiveness on the link, which leads to an unfair use of the resources.

If the Resource Pooling idea is not able to provide a fair multipath behavior, the benefits of Resource Pooling idea should be questioned in this context.

6.4 Conclusion

The evaluation of the existing coupled congestion controls leads to different conclusions. First the IETF fairness discussion results in fairness behavior that is, in the worst case, completely decoupled from the resources provided in the network. All experiments with singlepath flows show that the singlepath flow will be preferred in the current IETF interpretation mostly without any valid reason. The cause for this behavior is the hard coupling of the *balance congestion* goal with the *fairness* goal in the Resource Pooling idea.

No entity in the network is able to control the behavior. Even worse, the goal of a better network utilization cannot be achieved in all scenarios. Even if the Internet community will accept the interpretation of the ambivalent multipath fairness definition also in the future and the deviations from the fairness line will be tolerated, the idea of a fair sharing of resources will not be achieved for multipath flows themselves. A multipath flow fairness is not covered by the current IETF multipath fairness goals. Here it can be expected that history is repeating itself. Like the definition of singlepath flow fairness was created with respect to a TCP-compatible behavior, multipath flow fairness will be defined by an MPTCP-compatible behavior. Thus, also for CMT-SCTP it can be expected that the discussion shifts to an MPTCP-compatible behavior, where LIA currently sets the baseline. Even if OLIA, RP-MPv2 or future congestion controls would perform better, they will not be deployed, because they are unfair to the LIA CCC. The community has to discuss if this should really be the goal.

The “fair sharing” of the resources is an important task for the load sharing extensions of SCTP and TCP and will decide about the success of both extensions. But even if it is assumed that the mechanisms used in the future will be developed based on the Resource Pooling idea and work in acceptable borders, the motivation for the multi-homed sender is not always clear. To provide the resources for the own flow but to achieve less than a singlepath flow is hard to understand. The impact of this motivation is not predictable. In the current status, the IETF multipath fairness goals are not complete, inconsistent and not user friendly.

Altogether, the mechanisms addressing the Resource Pooling idea do not behave very accurate and are not easy to predict, to monitor or to control. And in the end, none of the involved parties in the network will be motivated to interact as required by the IETF multipath fairness definition, as long as use cases can be defined that result in less performance for everyone. Furthermore this thesis argues for a link centric subflow fairness, with all its benefits as an alternative solution.

Chapter 7

Revisiting scheduling

Chapter 5 has discussed the direct impact of the "path" definition on the overall throughput of MPTCP and CMT-SCTP. Furthermore, Chapter 6 has demonstrated that the ambiguous fairness definition not always causes a straightforward sharing of the network resources. All mechanisms addressed by these chapters have a significant impact on the defined resources and, therefore, on the throughput.

In addition to this, however, a link as basis entity has own specific characteristics with respect to the usable bandwidth, the error rate and the delay. These link characteristics are based on physical limits. A transport protocol has to deal with these path characteristics which are very different in a network such as the Internet and can vary greatly over time.

To support an effective utilization of these path resources, different decisions have to be taken for a subflow such as:

- Timing of the send process,
- Amount of data to send,
- The resources (path, buffer, window, etc.) to use.

This task is assigned to a so-called scheduler instance [SM11, DSTR10, SGTG+12]. Until now it has not been methodically investigated which degrees of freedom a multipath scheduler for MPTCP and CMT-SCTP has. Of course, there exist some exisiting solutions in the reference implementations of FreeBSD and Linux, but it has not been discussed if they have been designed for the network requirements like those of the Internet.

The contribution of this chapter is to revisit the way the data of a connection is distributed over the paths to achieve the goals of the load sharing extensions. This chapter identifies the weaknesses of the currently used approaches and proposes improvements for the extensions of TCP and SCTP.

The discussion starts with a detailed look into the challenges of a multipath scheduler design on the transport layer, in particularly taking into account the requirements of SCTP, TCP and their extensions. Afterwards the mechanisms of the multipath scheduling design are adapted to the multipath protocol extensions and evaluated. In addition this chapter introduces a completely new approach for MPTCP to decrease the effects of buffer blocking. This new approach is based on a real-time protocol extension of [BDB+13], which was done as preparatory work for this thesis. Furthermore, this chapter identifies new blocking issues for MPTCP and CMT-SCTP and proposes approaches to decrease their impact on the

throughput. Before starting with the design, the multipath scheduling process itself has to be analyzed.

7.1 The multipath scheduler

Although it is an important task, the IETF leaves the scheduler discussion open to the implementers. Therefore, nothing related is defined in the Internet drafts. Only the current reference implementations of MPTCP and CMT-SCTP (see Subsection 3.2.3.4) give a first implementation guideline and solution approach. This section discusses and analyzes the reference implementations in FreeBSD and Linux and the related work to demonstrate the benefits, the drawbacks and the requirements. The reference implementation for MPTCP [RPB+12] is in Linux and for CMT-SCTP in FreeBSD [Fre12] (see Subsection 3.2.3.4).

7.1.1 Goals of scheduling

Every application has its specific requirements concerning the transport protocols based on the services provided. Some applications need a high throughput and other applications require a low delay. Also many applications have a mix of delay and throughput constraints, like a live video streaming service.

However, the IETF transport protocols do not address these requirements with such a fine granularity. The application developer has the choice among different service bundles, represented by the IETF transport protocols TCP, UDP, SCTP and DCCP (see Section 2.1). Therefore, the design goals of a multipath scheduler are not a question of application goals in the first place, rather they are a question of the goals of the service bundle represented by the core protocols as defined in the RFCs.

As discussed in Subsection 2.2.1, SCTP and TCP provide a connection-oriented, unicast and congestion-controlled transport service. The main difference is that TCP is based on a byte stream transfer and SCTP uses a message format to organize user data. In addition, SCTP provides the possibility to divide the application flow into multiple independent logical streams, where each stream is a logical unit similar to a TCP flow (see Subsection 2.2.2).

The MPTCP architecture RFC [FRH+11] is also really clear with its goals and defines *improved throughput* and *improved resilience* as primary tasks (see Subsection 5.1.1.1). The CMT-SCTP Internet draft [ABD+13] describes primarily the *improved throughput* goal. Nevertheless, an additional goal of CMT-SCTP is to become more effective with respect to the availability check[1]. The better availability check goal in case of CMT-SCTP confirms once more the focus of the SCTP/CMT-SCTP authors on the reliability feature.

Furthermore, a theoretical throughput has to be identified to judge if the currently described multipath extensions even achieve the throughput that can be expected and what is defined as goal. A lot of effort has been spent for TCP and SCTP to create models for theoretical throughput calculation. An often-cited result for TCP is the formula of Mathis [MSM97]. However, there also exist other approaches for TCP like [PFTK98]. A similar approach exists for SCTP in the work of [Rün09]. All approaches are based on similar assumptions. The basic idea of all approaches is to calculate the throughput based on a period of time called

[1] Availability check with real data instead of heartbeat messages.

"send cycle" of the connection. A send cycle is typically given by the round trip time of the singlepath connection "path" in TCP and SCTP.

These approaches work quite well for singlepath transport protocols like SCTP and TCP. It is important to mention that all the calculation models focus on the *congestion avoidance* phase (see Subsection 2.2.4.1). The *slow start* phase is not in the focus of these calculation models, because it does not have a relevant impact on the singlepath system behavior, particularly with focus on the long-term behavior. Therefore, the first straightforward expectation for the load sharing approaches is a summed throughput calculation over all subflows. Some experiments, like the high throughput experiment setup in [Chr13], have shown that such a throughput can be achieved under perfect conditions. This confirms the argumentation of the authors of [SGTG+12] who claimed that if there are no limiting resources in a setup with similar links, the scheduling has no relevant impact on the throughput of the multipath connection. It is only important that the sender is saturated and that the system provides enough resources to support a continuous send process of data. This leads to a straightforward calculation of the expected throughput T as sum of the throughputs of all available paths P.

$$T = \sum_{i=1}^{P} \frac{\text{data per cycle}_i}{\text{time per cycle}_i} \tag{7.1}$$

However, neither CMT-SCTP nor MPTCP are deployed only in a perfect world scenario. The Internet is a target network with imperfect conditions, it is highly heterogeneous and characterized by unpredictable combinations of links and paths. Most impact have dissimilar path characteristics. In this thesis four kinds of path characteristics are distinguished:

- **"Fast" path**
  This defines a path with a low delay. A subflow using a fast path is denoted as fast subflow. The "fastest" subflow is assigned to the path with the lowest RTT.

- **"Slow" path**
  A slow path provides a high delay and a subflow using a slow path is characterized as slow subflow.

- **"Strong" path**
  This identifies a path that supports a high throughput. Therefore, a strong path provides a combination of high bandwidth, low error rate and low delay. A subflow assigned to a strong path is denoted as strong subflow and the strongest subflow achieves the best throughput over a period of time.

- **"Weak" path**
  A weak path uses links with poor characteristics with respect to bandwidth, error rate and delay. A subflow which transfers data via this path is called weak subflow.

7.1.2 Scheduler setup

The Internet provides a challenging environment with nearly endless combinations of link and path characteristics. The resources provided are diverse and cover different media like for example fiber, satellite link or copper cable. Every network component provides specific characteristics of its own and has to provide the resources to different flows. In this thesis the experiments are performed on a 3G-like and a WiFi-like link to demonstrate the effect of

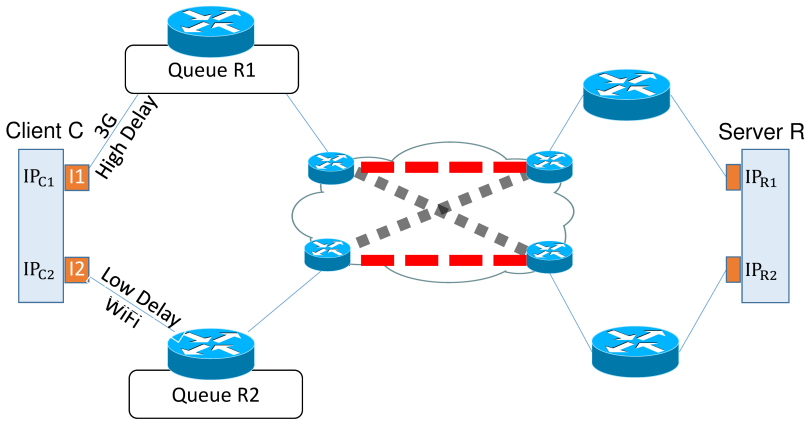


Figure 7.1: Experiment setup

asymmetric, heterogeneous network topologies. This setup was chosen because mobile devices used already today are mostly connected via these two access links.

The experiment setup becomes even more relevant because such a scenario is also used by the authors of [RPB+12]. They are also co-authors of the MPTCP standard and, therefore [RPB+12] demonstrates their achievements related to load sharing schedulers and buffer blocking. This allows a direct comparison with respect to related work done by the MPTCP research community. Figure 7.1 illustrates the experiment setup with the usage of the two mentioned access technologies. The path via the WiFi link is the fast and the strong path and the 3G link is the slow and weak path. The characteristics distinguish the subflows using the paths and require a send behavior at the endpoint that includes mechanisms to adapt to the path characteristics to use the provided resources in an optimal way. Even if the following experiment setup is close to the work of [RPB+12] the experiment setup was extended in this thesis by the results of the Chapter 5. The connecting network was defined in two different ways with respect to the path management strategy (see Subsection 5.1.2.1):

- **Disjoint path setup**
  With this path management strategy (represented by the red dotted lines in the cloud of Figure 7.1) a topology and routing is addressed that provides links which are disjoint and do not share a source or destination address. This model is valid for CMT-SCTP and MPTCP.

- **Cross path setup**
  This path management strategy is the default in case for MPTCP and provides a cross table of all path combinations – as illustrated in the Figure 7.1 by the cloud with four possible paths (red and grey dotted lines) instead of two.

Of course, the capacity of these links is an important characteristic to calculate the throughput that can be achieved, but also the latency and the error rate of the links have a direct impact. The resources have to be shared by every flow and corresponding subflow of a connection. The fair sharing of the path resources in the network was discussed in Chapter 6.

In general, if not enough memory or CPU capacity can be provided by the routers or endpoints the continuous send process and, therefore, the overall throughput will suffer. However, in the following discussion particularly the coupled resources with respect to available memory in endpoints and routers are in the focus. Memory is represented in networks like the Internet by different queues and buffers in the endpoints and the routers. A queue has a queue length – in bytes or packets – and an applied queueing discipline (see Subsection 2.3.2). Furthermore, these queues are often used as buffer to organize data for the next process step. Different queues are used for different mechanisms like for the send, receive or routing process. Buffer space is required for example for SCTP and TCP and the corresponding multipath extensions to support reliable and ordered data transfer. Therefore, there exists a strong dependency among the support of reliable and ordered data transfer, the send and receive mechanisms and the send and receive buffer provided on the endpoints.

7.1.3 Challenges for a multipath scheduler

It is obvious that the path characteristics have a direct impact on a multipath scheduler instance and, therefore, on the overall protocol performance. In particular the increased complexity in the re-ordering process and the buffer provided to organize this process must be addressed by the scheduling instance. Asymmetric, heterogeneous path characteristics often result in a poor throughput for TCP and SCTP when using the initial multipath extensions without any further improvements. The work of [SGTG+12] gives a detailed overview about this problem and shows that several publications address this "re-ordering" issue, like [GTGB11] or [NZNP11]. In addition, preparatory work done for this thesis in [DBRT10] can be mentioned and also the authors of the MPTCP IETF Internet standard address the issue in [RPB+12]. Furthermore, [DBRT10] (for CMT-SCTP) and [NZNP11] (for MPTCP) have demonstrated that the throughput in heterogeneous networks can even be worse when multiple subflows are used instead of the strongest subflow alone. It is necessary to discuss the reasons for this behavior in heterogeneous networks in a first step and to develop mechanisms to avoid this insufficient performance in a second. The following discussion of the reasons for this behavior is an extract of different discussions done as preparatory work for this thesis in [DBRT10, DBAR11b, VBOMT13a, ADB+11].

7.1.3.1 Information gaps

As [SGTG+12] pointed out, schedulers adapting to network conditions have a much better performance than those based on simpler scheduling mechanisms, like e.g., Round Robin. [SGTG+12] highlights in particular propagation delay and link capacity as aspects with high impact. Information about the network is relevant for an optimized scheduling process without any question. Thus, information is required for the scheduler which has to be measured or estimated. But the measurement of this information leads to another major issue in the context of multipath scheduler development. The measured – or estimated – data describes only a previous state of the network and not the current state at send time. The current state in a dynamic network like the Internet is unpredictable. Furthermore, as a special case, it should be kept in mind that in the starting phase of a connection only the information generated during the handshake is available for the first scheduling decision.

Many scheduler approaches assume a stable network behavior in the design process and take into account only the congestion avoidance phase – like for example the approach

in [SBL+13] –, i.e. they do not address the startup phase. This leads to the expected benefits but does not cover all the requirements of the deployment in the Internet. The startup phase has to be considered for example – as will be shown later in Section 7.4 – because it has a major impact on the flow throughput with limited buffer queues. The lack of consideration will lead to false assumptions and afterwards the scheduler will not be able to adapt to the conditions of the network in an optimal way anymore. Only heuristics can help to decrease the impact of these false assumptions [DBRT10, RPB+12]. However, these heuristics are not exact and lead to additional transmissions in the worst case, as will be discussed in Subsection 7.3.2. In particular the heuristics with respect to MPTCP do not have the positive effect expected and, in the worst case, decrease the throughput as will be discussed later.

Altogether, it can be expected that a scheduler does not achieve the perfect allocation of the resources all the time in a network like the Internet, because the scheduler process lacks a correct, up-to-date database for the decisions to make.

7.1.3.2 Interaction with protocol mechanisms

TCP is 30 years old and was subject to continuous changes. In order to achieve throughput optimization, several new mechanisms were introduced over the years. These mechanisms have a significant impact on the send and receive process. The interaction of the scheduler with these mechanisms is an underestimated challenge. A multipath scheduler has to measure – or to estimate – information about the available resources in a nearly perfect way and has to adapt to the established and used mechanisms. The dependencies here are complex and cannot be easily identified.

As an example, a multipath scheduler can have a negative impact on path related mechanisms like the fast retransmission mechanism of TCP (see Subsection 2.2.3). A fast retransmission only occurs if a DupAck is reported by an acknowledgment or SACK at least three times via the same path. This assignment of the DupAck to a path works well in a singlepath TCP environment (see Subsection 2.2.3). In MPTCP, this assignment remains the same and the receiver will send a DupAck via the path on which the segment has arrived. A scheduler behavior is not defined and, therefore, a scheduler is free to choose another path with free resources for the next segments. If, therefore, the send process will be interrupted on a specific path after the first or second acknowledgment, a fast retransmission cannot be triggered. Thus, a timer-based retransmission will occur instead of a fast retransmission. However, this timer-based retransmission will decrease the congestion window more than necessary (see Subsection 2.2.4.1) and, of course, the overall system throughput, too.

Figure 7.2 describes this new kind of blocking – caused by mechanism interaction – as a simplified example for MPTCP by using SACKs (see Subsection 2.2.3). Figure 7.2 illustrates a multipath connection with two subflows on two disjoint paths. Every subflow has its own sequence number space maintained independently and a shared flow sequence number space as coupled resource. While the subflow sequence number space ensures the reliable and ordered transfer on subflow level, the flow sequence number space provides the same service on flow level (see Subsection 3.2.3.2). In case the segments arrive without any gaps and ordered, the data can be delivered to the application and be removed from the receive queue. But an individual gap on flow level prevents the ordered delivery to the application for both subflows and, therefore, also the removal from the send and receive queue. Back to the example, Figure 7.2 shows on the left side an incoming weak subflow that covers the sequence numbers from #1 to #3 of the flow sequence number space and subflow sequence number space. On

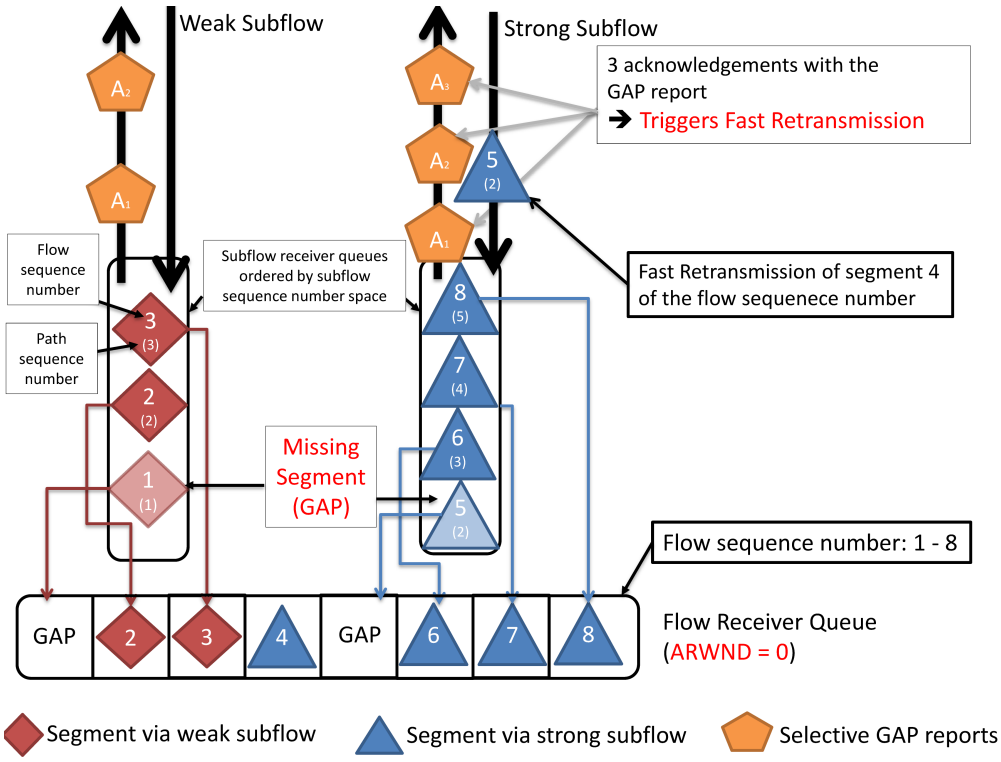


Figure 7.2: Scheduler issue

the right side of Figure 7.2 an alternative strong subflow of MPTCP is illustrated. This strong subflow covers the flow sequence number space from #4 to #8 and uses the subflow sequence numbers from #1 to #5. The receiver queue of the weak subflow waits for the first in order-transmitted segment with the flow and subflow sequence number #1. The second and the third segment of the subflow have been transferred with success and placed in the subflow receiver queue in the order from #2 to #3. It should be mentioned again that all queues share the same receive buffer.

Singlepath TCP would trigger a fast retransmission after the transmission of the 4th segment of the flow sequence number space to close this gap, but in this example the scheduler prevents this. The multipath scheduler decides to send all further segments via the strong subflow, until the scheduler recognizes that the advertised receiver window (ARWND) is closed (see flow control Subsection 3.2.3.3) and the receiver queue is filled. The first segment (flow sequence number #4 and path sequence number #1) on the strong subflow is transferred successfully to the flow receiver queue, because it arrived in order. But the second segment (flow sequence number #5 and path sequence number #2) on the strong subflow gets lost during the transmission. Therefore two gaps exist which prevent the ordered delivery to the application, even if the third, the fourth and the fifth segment on path level again arrive on the strong path as expected. The mechanisms to repeat the lost segments distinguish the gaps. The receiver sends acknowledgments for the segments delivered on the strong subflow, with a gap report that identifies the loss three times. This is for the sender a loss indicator that triggers a fast retransmission of the lost segment on the strong path. However, since nothing happens on the weak subflow, the send process is still blocked. Only after the retransmission

timer will expire, the weak subflow will retransmit the segment with the missing subflow sequence number #1 which was assigned to the timer. In this scenario, only an additional retransmission on the weak subflow helps to avoid this situation.

The potential drawback that the mechanisms cannot interact as expected exists for every mechanism that expects continuous sending of messages. But this continuous send behavior on path level is an assumption about the subflow behavior that is not true anymore. And this assumption is common for many mechanisms, i.e., the delayed acknowledgment feature [Bra89] or mechanisms like Nagle's algorithm [Nag84].

7.1.3.3 Sender side buffer blocking

Many reasons for sender side buffer blocking are known from the singlepath world and some are just trivial. It is obvious that the sender is not able to saturate all available links if not enough data is available for sending or not enough buffer to support the send process. Another common reason is that the CPUs provide only insufficient processing power for the send or routing process. However, the multipath approaches bring in new challenges. First the new challenges were described for CMT-SCTP in [IAS05] and, later, investigated in more detail as preparatory work for this thesis in [DBRT10]. [DBRT10] also claims the relevance for both IETF approaches, which was indirectly confirmed three years later by [RPB+12], which proposes a mechanism to avoid buffer blocking for MPTCP. The proposed approach of [RPB+12] is based on the same principles as in [DBRT10], even if the terminology used is different (see detailed discussion in Subsection 7.3.3). The two challenges identified for sender side buffer blocking are:

- **Transmission induced send buffer blocking**
  If nothing else is defined, a scheduler tries to send as much data as possible to a path with free resources. This data has to be stored in a buffer on the sender side to support reliable transfer, in particular with respect to retransmissions. The mechanism which is responsible for the buffer management has to decide how much buffer every path is allowed to occupy. Wrong decisions can have a severe impact, because by allocating buffer for a subflow causes blocked memory until the transmission is acknowledged and, furthermore, it is not ensured that this memory is blocked for the most effective subflow. However, based on wrong assumptions it is possible that memory is blocked for an ineffective subflow which just fills router queues in the network – like known from the bufferbloat context [Get11a] – without an added value because this data that blocks the send buffer does not increase the bytes sent per send cycle. Therefore, in some cases a potentially weak subflow is able to allocate more buffer than a strong subflow, even if the strong subflow has more potential to increase the throughput. This leads to an unbalanced distribution of buffer size over time, where ineffective buffer allocation decreases the throughput. It should be the goal that the strong subflow always has enough available buffer and it should be prevented that available buffer is allocated to the weak subflow if it could be used more effectively. The preparatory work in [DBRT10] demonstrates that this buffer blocking even occurs during an unordered transfer.

- **Gap induced send buffer blocking**
  Different issues, such as the loss of a fast retransmission, cause this kind of blocking. The main issue is that sender and receiver do not have the same information in every case and the sender cannot be sure if buffered segments are still required for the retransmission

process or not. Even if buffer can theoretically be freed, because data was already acknowledged by the gap reports of a SACK, the sender is not allowed to do so, because the gap reports of a SACK are defined as not reliable [NEY+08, YA13]. Thus, the receiver is allowed to revoke[2] these gap reports any time. This issue exists for both load sharing approaches. This can lead to a complete buffer blocking, where the transmission on all paths is suspended until a successful timer-based retransmission of data and the reception of a new higher cumulative acknowledgment has taken place.

The sender side buffer blocking issues have a huge impact on the throughput because they cannot be avoided by heuristics in a perfect way. Therefore, a detection system or mechanisms to decrease the effects of sender side buffer blocking must be applied.

7.1.3.4 Receiver side buffer blocking

The scenario is well-known from a singlepath connection, too. The receiver is able to limit the sender by means of the advertised receiver window (ARWND). In fact, this mechanism was designed to protect the receiver if the receiver has limited resources. A closed[3] receiver window is in the singlepath context an indication that the buffer at the receiver side is too small or the CPU is too slow to process the data. However, the reasoning for a closed ARWND has to be extended by two additional aspects in the multipath context:

- **Reordering induced receiver buffer blocking**
  This is the most straightforward buffer blocking issue, which is caused by the loss of packets during the transmission. These gaps are also caused by a weak subflow which fragments the receiver buffer. A continuous send process cannot be ensured in this case.

- **Advertised window induced receiver buffer blocking**
  The advertised receiver window (ARWND) is a shared resource of all subflows of an overall multipath flow (see Subsection 3.2.3.3). Thus, receiver queues which are also used as buffer for reliable and ordered transfer share the buffer size on flow level. Therefore, if one subflow needs a lot of the receiver buffer size to organize its sending, it increases the risk that there is not enough buffer space for the alternative subflows. The goal should be always to support the most effective subflow with the highest throughput per send cycle, because otherwise the buffer blocking caused by the less effective links will close the ARWND in the worst case. Every subflow needs an open ARWND to support a continuous send process. The overall throughput suffers more than necessary, if this condition is not ensured for the most effective subflow.

The scheduler on the sender side has to implement a receiver side buffer blocking detection or a mechanism to mitigate the effects of receiver side buffer blocking. The flow control mechanism has to be considered to detect critical situations and to get the current state of the receiver buffer on sender side. Thus, an effective mechanism to detect or avoid buffer blocking at the sender can only operate successfully after the window sizes are (re-)calculated.

[2] Besides the term of an revokeable SACK sometimes also the term renegable SACK can be found.
[3] "closed" means here that the allowed amount of data is decreased to zero.

7.2 Architectural aspects

In theory – from top-down – the application schedules a logical ordered data flow to the protocol by utilizing the connection interface. Now the scheduler can make its decision depending on its context and has to map the flow to the paths. It is important to understand how the architecture of the singlepath transport protocols and their extensions currently support the scheduling of user data with respect to the IETF multipath goals.

7.2.1 General scheduling decisions

A scheduler implemented on the sender side of a transport protocol works on the data provided by the application and decides about the sending order and the timing. The flow must be prepared for a packet switched network (see encapsulation process in Subsection 2.2.1) and decisions have to be made to organize the shared resources of the sender. Decisions for the sender also have a direct impact on the receiver, e.g., if the scheduler on the sender side prefers a specific subflow. This also requires corresponding resources on the receiver side for this preferred subflow to achieve the expected throughput. This decision process is described in this thesis as a process in three steps for the IETF load sharing approaches of TCP and SCTP:

- **Identification of usable paths**
  On every send impulse a multipath scheduler has to identify the paths usable for transmission. The MPTCP specification defines nothing about the identification of usable subflows, so it must be assumed that in a first step all established subflows could be used for sending by the scheduler. CMT-SCTP offers more support to identify usable resources by supporting a heartbeat mechanism (see Subsection 2.2.1). As also discussed in RFC4960 [Ste07], the heartbeat mechanism is used by a default SCTP association to probe alternative paths.

- **Selection of paths**
  If useable paths have been identified and there exists more than one option, a path for transmission must be chosen from the pool (see definition of path for CMT-SCTP and MPTCP in Chapter 5). But again, no detailed procedure is mentioned in the IETF documentation. Therefore, it is not surprising that there already exist different interpretations in the different reference implementations of CMT-SCTP and MPTCP. CMT-SCTP in the FreeBSD kernel uses a mapping of segments to subflows by applying Round Robin among the subflows. MPTCP in Linux supports a weighted scheduling based on the smallest RTT (see Subsection 3.2.3.4).

- **Allocation of path resources**
  If a usable path is chosen, the next decision is about the amount of "resources to use" during one scheduling decision. In the end, the resources in this context are defined by the amount of data affected by one decision. The amount of data has various impacts, e.g. on the buffer size or the CPU load. The smallest granularity is byte based, but this granularity is not very effective and impractical. A byte based granularity increases the costs for the encapsulation and decapsulation process and makes the ratio of user data and control information sent by one segment ineffective. It is well known that a use of partially filled packets decreases the throughput. An effective usage is ensured if the

amount of control data is small compared to the amount of user data and the IP packet can be filled completely. The most effective usage is achieved for a transport protocol if a completely filled MTU can be achieved.

An effective scheduler has the goal to reduce organization overhead as much as possible. But besides that there exist transport protocol limitations given by the design which the scheduler has to consider, too. Thus, for CMT-SCTP a scheduler has to take into account the SCTP message size [Ste07] – as long as it is smaller than an MTU – and for MPTCP the scheduler should not use more than the MSS [Pos81b].

In [YAE13] similar decision steps are described, although all three scheduling steps differ in their interpretation. Just to focus on the congestion control as limiting factor for example – as described in [YAE13] – to identify usable resources is not enough. Just checking the window of flow and congestion control, in particular with respect to the open buffer blocking issues does not characterize a usable subflow.

Furthermore, the scheduler should not be defined by assuming a high degree of freedom in the selection of the resources. It will be demonstrated in this thesis that it is more a decision about "to use" or "to ignore" a free resource in particular with respect the support of a saturated sender[4]. Thus, if there is no bottleneck in the network or limitation caused by the window size, the bottleneck is caused at the sender or receiver endpoint by the depletion of shared resources. A bottleneck can be caused for example by limited buffer space to organize the user data for the reliable and ordered transfer. Already blocked buffer space can only be freed by an incoming acknowledgment. Therefore, if a "buffer blocked" state exists and an acknowledgment arrives, the scheduler can only react on this new trigger and can free the corresponding buffer space. The scheduler task should focus only on an effective allocation of the resources, i.e. the buffer space represented by receive and send queues. In contrast to the claim in [YAE13] a real choice between the alternative free paths at the same time is not a usual situation in this state. Most of the time, potential alternative resources are blocked by the sending mechanism itself.

7.2.2 Scheduling subflow to path

It is known from the path definition in Chapter 5 that there exist different levels of mapping a data flow to a path. If it is assumed that CMT-SCTP has only one stream per flow[5], MPTCP and CMT-SCTP have the same "mapping" challenge. The scheduler instance has to decide for the data flow of both protocols how the data has to be mapped to the available resources of the subflows. However, even if it is the same problem statement for MPTCP and CMT-SCTP, the buffer architecture for both schedulers is different. First, the different path definition approaches (see Chapter 5) lead to a different number of send queues in the architecture and second – with more consequences – the protocols use the sequence numbers to organize the reliable and ordered transfer of user data in different way. Thus, both approaches have to be analyzed separately.

Furthermore, the focus on the shared resources of the subflows is not enough. The congestion control and receiver side flow control are an important part of the scheduling process and their impact on the scheduler is often underestimated. In the worst case the scheduler,

[4]Saturated means in this context that the sender tries to send as much data as possible through the network, only limited by the capacity of the links and the window sizes of congestion and flow control.

[5]The stream concept has no specific impact on this issue as discussed in the following for CMT-SCTP.

e.g., enqueues data on a subflow send queue, which will not be dequeued for a long time if the congestion control prevents it. Furthermore, it can cause buffer blocking on the receiver side, if the receiver waits for data which is enqueued in the send process but not covered by the congestion window size. All of these aspects have a significant impact on the MPTCP and CMT-SCTP scheduler design process as discussed in the following sections.

7.2.2.1 MPTCP

TCP provides a reliable, ordered transfer of user data and so does MPTCP. In case of MPTCP the flow sequence number – denoted as data sequence signal (DSS, see Subsection 3.2.3.2) by the IETF – is used to achieve this goal on flow level. The re-ordering of the flow segments is organized in MPTCP in two steps: first, for the reordering of the subflow sequence number space and second the reordering of the DSS. Thus, where CMT-SCTP has no path-specific sequence number space, MPTCP has. This has a potential drawback, since the "data to send" has to be divided into an unspecified number of different sequence number spaces and subflow send queues in case of MPTCP. Figure 7.3 illustrates the simplified context of flow and subflow send queues for MPTCP. The figure shows that the incoming data is first stored in a general send queue for the complete flow. In this queue the data gets an ordered flow sequence number – MPTCP DSS – which organizes the data order within the complete flow. After this sequence number assignment is done, a scheduler decides to which path the data should be assigned. Each path can be interpreted as one or more subflows which use the chosen path[6].

The one-to-one relationship has a direct impact on the target subflow queue length of MPTCP. The subflow queue levels should always be equal or less than the minimum of the congestion window (CWND) and the advertised receiver window (ARWND) reduced by the outstanding bytes per subflow. Every byte on top of that has the potential to cause a problem of enqueued segments which cannot be sent. Thus, the MPTCP scheduler has to check the subflow send queue fill level by the following condition:

$$Queueable_i \leq max(min(CWND_i, ARWND_i) - Outstanding_i, 0) \tag{7.2}$$

At the time data is enqueued in subflow send queues, the subflow sequence number will be assigned to the segment. This has consequences, because the enqueued segments will be sent immediately under the conditions of Equation 7.2. Data entered in a send queue of a subflow (with a corresponding assignment of a path sequence number) has to be kept in the subflow send queue until the delivery on subflow level is acknowledged. Thus, a correction of the scheduling decision or a rescheduling of the same data on flow level is not possible anymore without copying the segment. The segment has to be delivered on subflow level in every case, even if it is not required for the overall flow sequence number space anymore. A subflow has to support reliable transfer on the subflow level to avoid middlebox issues. This is a potential drawback of the sequence number space per subflow, where the subflow is assigned to a specific path given by a specific source/destination address pair.

Summarized, the main buffer for MPTCP segments in the protocol stack is the flow send queue. The sizes of the subflow send queues should be as small as possible, in maximum equal to the data in flight on a subflow.

[6]To simplify the setup, it is assumed in this thesis that only one subflow per source/destination address combination exists, although the MPTCP RFC does not forbid the setup of more than one.

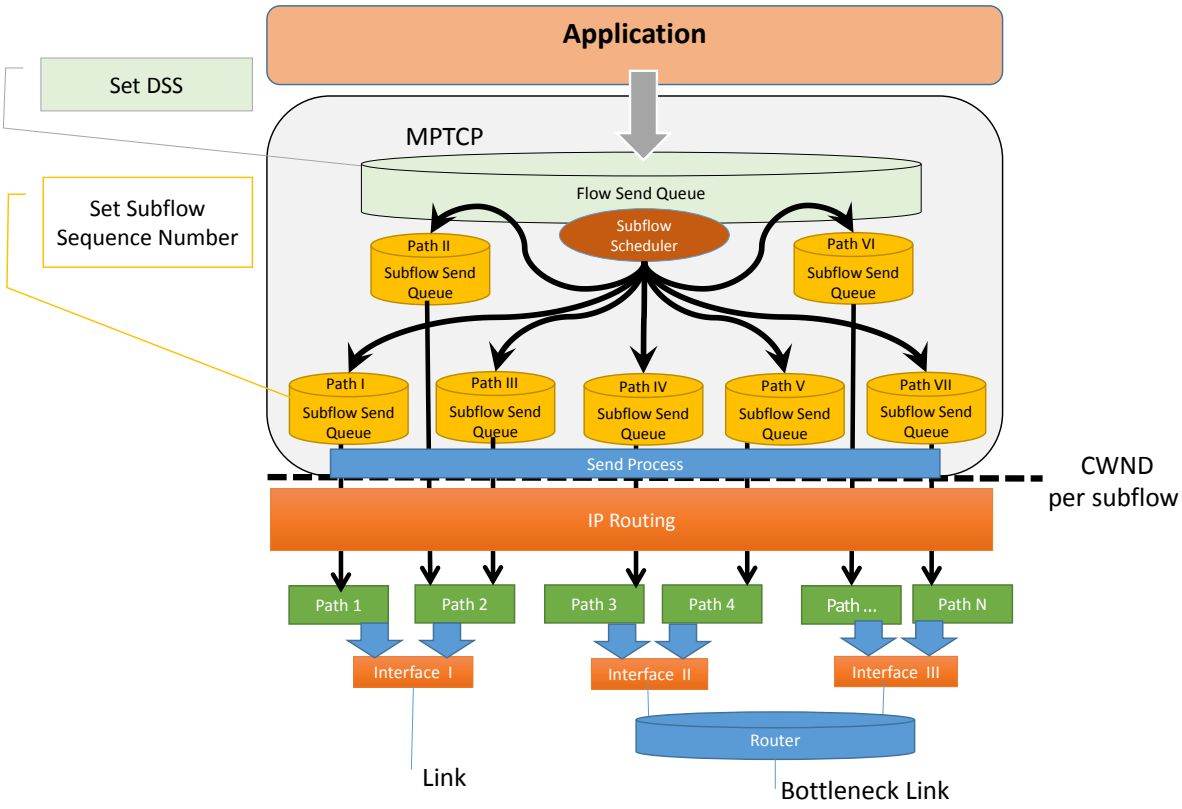


Figure 7.3: Send queue MPTCP

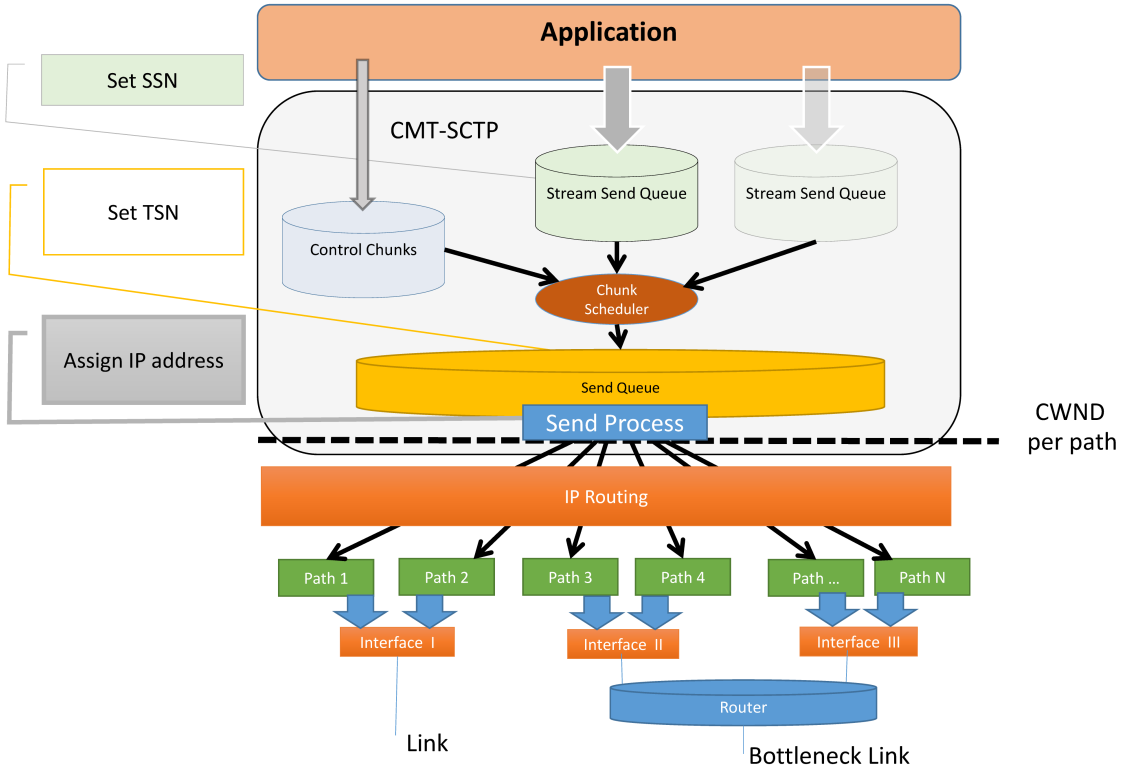


Figure 7.4: Send queue (CMT-)SCTP

7.2.2.2 CMT-SCTP

Figure 7.4 illustrates that in case of CMT-SCTP the data will be first enqueued by the application into a specific stream send queue. Here, the Stream Sequence Number (SSN) will be assigned to an SCTP DATA chunk. This SSN is not comparable with a subflow sequence number because it is not assigned to a specific subflow/path. After that, the sequenced DATA chunk is provided to the "Chunk Scheduler". The tasks of the Chunk Scheduler are:

1. To create an SCTP common header structure.
2. To check for control chunks and to add existing control chunks to the common header.
3. To control the sequence in which the stream queues are emptied. This step is also often denoted as "stream scheduler".
4. To add as much DATA chunks as possible to the existing common header and control chunks. During this step, SCTP provides the possibility to bundle or to fragment DATA chunks to achieve the maximum size of one MTU per SCTP segment.
5. To enqueue the new SCTP segment in the general send queue. Here also the TSN as flow sequence number is assigned.

There exists no specific subflow send queue. Therefore no sequence number is assigned to a specific path and every chunk can be sent or retransmitted arbitrarily via every path.

It should be mentioned that CMT-SCTP provides the stream concept (see Subsection 2.2.2) like singlepath SCTP which fills the "general" send queue from different stream send queues. This might raise the question, if the mapping of the application data to streams and, furthermore, the mapping of the streams to the paths has an impact on the results of this chapter. These two steps are discussed as follows:

- **Application data to stream**
  A scheduling instance for SCTP exists which is responsible for the assignment of messages to logical streams. This instance is by default located in the application, because only the application is aware of the logical structure of the overall flow. Although this mapping is an interesting topic and researchers spent efforts on this challenge, like [STR10], this mapping it is not a specific multipath issue. The achievable throughput for the complete multipath connection will not be increased or decreased as long as the flow send process will not be interrupted or limited. The overall throughput is the same as long as the "general" send queue is filled with DATA chunks.

- **Stream to path**
  If there exists more than one stream – which is possible for CMT-SCTP – the question should be raised how the streams themselves should be mapped to the available paths. [DSTR10] investigated this CMT-SCTP issue of mapping streams to paths and presented some interesting results. They claim that by mapping each stream to a certain path, the buffer size can be kept small. Furthermore, [DSTR10] showed that the per-stream message delay can be decreased by an optimized scheduling variant. Although the increased buffer usage or the increased per-stream message delay are very important for specific SCTP applications, this does not address the general multipath challenge, i.e., how the data itself should be scheduled to utilize the available resources in an optimal way.

Summarized, a stream scheduler is a scheduler with limited impact on the overall throughput of a multipath SCTP flow as long as the send process of the complete flow is not interrupted or limited in functionality.

7.2.3 Location of the scheduling process

So far there exists no well defined location for a multipath scheduler, i.e. it is not an explicit part of the existing multipath architecture. Furthermore, the scheduler cannot be described as an instance or module, rather as a process in two steps.

The first step in the scheduling process is the assignment of the overall flow sequence number. This is necessary for the ordered and reliable transfer of the overall flow. In CMT-SCTP the first step is done by the chunk scheduler (see Figure 7.4). No hard restrictions can be identified, as long the sender is able to send saturated. In MPTCP the subflow scheduler does the first step. The subflow scheduler fills the subflow send queues under the restrictions identified above (see Figure 7.3).

The second step is done by the instance which organizes the transfer to the network layer. This instance is denoted as "send process" for both protocol extensions in the following. The degrees of freedom of the scheduler process at the send process are decreased in case of blocked resources. The send process instance only has the choice to

- **Use free resources immediately**.
  This is the straightforward behavior to use newly freed resources immediately if there exists a chance to allocate them. Anyway, it becomes obvious in the buffer blocking discussion that this strategy alone only leads to a poor performance.

- **Delay the use of free resources**.
  Thus, even if there exist free resources in the system, the scheduler decides not to use them immediately. It makes sense if the whole system expects a better information base in the next decision cycle and, furthermore, this delay leads to a pool of free resources. This pool in turn leads to a real choice of free resources.

- **Skip free resources**.
  At first sight it may seem surprising to skip free resources, but to use free resources for the wrong subflow leads to decreased throughput. Sometimes it makes more sense to use just the strongest subflow, rather than all available subflows mapped to the different paths and links. For example, if an MPTCP connection is used with two subflows – one via a path with a bandwidth of 10 Gbit/s and 1 ms delay and one via a path of 56 Kbit/s with a delay of 1 minute – it is obvious that it makes no sense to block buffer resources for the weak subflow, if this prevents a perfect transmission on the strong subflow. This example might be extreme, but it gives the right idea of the important points. It is not an optimal solution to allocate every available resource to the send process on the sender side. However, the sender is not aware of the network conditions and this is a challenge in particular with respect to the startup phase – until the sender got feedback from all paths.

The scheduler process is assigned to two different locations in the transport protocol architecture and has a reduced degree of freedom in case of a saturated sender. It should be

mentioned that the authors of [SGTG+12] have done a similar classification of implementing a scheduler for MPTCP by defining the possibilities of *push data*, *pull data* and *hybrid push-pull data*. *Push data* leads to a direct allocation of a data packet to a subflow. The *pull data* approach stores the generated data packets in a send buffer shared by all subflows. The *hybrid push-pull data* approach combines the two methods.

However, the push strategy – even if it is a possible scenario for a scheduler with focus on prioritization – does not consider the possibility of receiver side buffer blocking as discussed before (see Subsection 7.1.3.4) and does not check the condition of Equation 7.2. Thus, the push strategy is only in the best case able to fulfill the goal of high throughput. The same is true for the hybrid push-pull strategy. However, the pull strategy describes a scheduler, which is able to consider the minimum subflow queue levels.

7.3 Multipath scheduler process chain

It is one major goal to increase the throughput for the load sharing extensions of MPTCP and CMT-SCTP. This major goal should be extended by an additional task. In the Internet, the multipath scheduler is facing changing network conditions and information gaps with respect to the state of the paths. This has to be addressed by the design process. Two steps can define the tasks with respect to an optimal multipath scheduling mechanism:

- Compensate information gaps
  In this step the mechanisms try to adapt the scheduler to the network conditions. Here mechanisms are addressed which work proactively like the NR-SACK mechanism discussed in the following Subsection 7.3.2.

- Reduce the impact of information gaps
  Here the mechanism accepts the negative side-effects and attempts to decrease their impact. Typical approaches are chunk rescheduling or opportunistic retransmission as they will be discussed in Subsection 7.3.3.

Furthermore, until now three decision tasks, namely the identification, the selection and the allocation of resources have been identified for a multipath scheduler and must be covered by a process chain. All decisions are based on the resource (e.g. path, buffer, packet size) specified by the protocol and the conditions given by the existing mechanisms like flow, congestion control or retransmission mechanisms.

7.3.1 Adaptation to network conditions

A multipath scheduler starts to work after the first successful handshake of the transport protocol. Thus, the sender is allowed to transfer data by using a subflow. For both protocols – MPTCP and CMT-SCTP – not all subflows are available after the first handshake. MPTCP requires additional MP_JOIN handshakes (see Subsection 3.2.3) for additional subflows and CMT-SCTP has to test every alternative path with an initial heartbeat exchange. This requirement for CMT-SCTP is defined in the singlepath SCTP RFC [Ste07]. Therefore, both multipath protocols start with the first scheduling decision like a singlepath protocol.

New user data causes the initial trigger to send data. Furthermore, this trigger is valid as long as no resource limitations occur during the sending process. It is just important that there is enough data enqueued for a continuous send process. In detail, if the ARWND

– shared among all paths – is open for new user data, the multipath scheduler is able to choose a valid path for sending. It is in the responsibility of the chosen scheduler how this selection is organized. If the system detects a buffer limitation, the send process stops. The trigger by new user data alone will not lead to further sending events anymore. Only a new cumulative acknowledgment leads to a new sending cycle. All free resources have been allocated and the system has to wait until the situation will change. So a newly established subflow only leads to a new sending cycle if enough sender and receiver buffer is available. In a "blocked" send process state the scheduler has only limited decision options since it is restricted to the recently freed resources. The multipath scheduler only decides to send data immediately/delayed or to skip the send trigger entirely. Wrong decisions in the starting phase of the connection will lead to throughput degradation, which cannot be detected or corrected easily in the later phase, where only acknowledgements can be used to start a new send cycle. This is completely different from the singlepath world experiences, where the impact of the start phase can be neglected.

Furthermore, asymmetric, heterogeneous paths lead to a high need of buffer and cause, therefore, buffer blocking [DBRT10]. A mechanism is mandatory to detect and mitigate the effects of buffer blocking. Clearly, limited buffer becomes more an issue for a multipath than for singlepath protocol and, therefore, must be discussed as next.

7.3.2 Mechanisms to avoid buffer blocking

The so-called NR-SACK [NEY+08] mechanism is one of the oldest mechanisms used to avoid buffer blocking. The NR-SACK is an enhancement of the SACK mechanism (see Subsection 2.2.3). NR stands for Non-Renegable and assures the sender that segments acknowledged by NR-SACK can be removed from the send buffer without any risk. This mechanism decreases the required send buffer size and is also known from singlepath SCTP. Furthermore, it is an answer to the gap induced send buffer blocking (see [DBRT10] and Subsection 7.1.3.3). NR-SACK was also introduced for MPTCP in [YA13]. A detailed discussion about the impact can be found in [Dre12a].

New mechanisms to minimize buffer blocking for multipath transfer in CMT-SCTP and MPTCP have also been developed with buffer splitting and chunk-rescheduling as preparatory work for this thesis in [DBRT10]. In this context only the main functionality will be discussed. A more detailed analysis was done in [ADB+11, Dre12a].

Buffer splitting was developed to mitigate the effects of transmission induced send buffer blocking (see Subsection 7.1.3.3) and advertised window induced receiver buffer blocking (see Subsection 7.1.3.4). The idea behind this mechanism is quite simple and requires the implementation of a new threshold to control the booking of the buffer. By means of this threshold a maximum buffer size which can be allocated to one individual subflow is defined. A similar mechanism was denoted three years later by [RPB+12] as penalizing for MPTCP with the same goals as CMT-SCTP buffer splitting. The penalizing mechanism has the goal to decrease the congestion window (CWND) of the blocking subflow once per RTT if a buffer blocking problem is detected. The basic idea behind this behavior is to prefer the subflows with the most effective send behavior during the next scheduling cycles. This causes a lower buffer allocation to the blocking subflow. In contrast to the buffer splitting approach, this mechanism focuses on the reduction of the effects rather than avoiding them completely. The buffer splitting mechanism just assigns a fixed amount of buffer to a subflow.

In addition, there is a strong need for a new retransmission mechanism to avoid reordering

induced receiver buffer blocking (see Subsection 7.1.3.4). A first approach was introduced in [DBRT10] as a preparatory work for this thesis and is called "chunk rescheduling". The idea behind this mechanism is rather simple. The sender tries to support continuous ordered delivery to the receiver by repeating outstanding data on the alternative subflow. Three years later the main authors of MPTCP confirmed this mechanism independently in [RPB+12] and denoted their approach for MPTCP as opportunistic retransmission (Opp-Rtx). The only difference between both approaches is that the chunk rescheduling approach has an own threshold parameter as trigger for the retransmission process which is depending on the buffer status. The opportunistic retransmission mechanism is only triggered if the buffer blocking event occurs which is caused by a closed ARWND. This threshold parameter for MPTCP makes sense, because every segment which is enqueued in a send queue must be transmitted via this path, even if it is not necessary anymore. This would cause unnecessary traffic. In CMT-SCTP this risk does not exist. Anyway, in the end both mechanisms are similar except for this threshold parameter.

However, the mechanisms to avoid or decrease the effects of buffer blocking can only be applied after the calculation of the flow control window, because only this value gives an overview about the receiver buffer status. Anyway, if the buffer blocking avoidance mechanism shows that the usable subflow has not been chosen properly, this process has to be repeated for another subflow before the subflow sequence number will be assigned. This can be tried for all available subflows. If no subflow can be identified with enough free receiver buffer space and without any buffer blocking evidence no send process should be started. It should be mentioned that it requires significant effort to identify correct evidences in order to avoid or decrease the effects of buffer blocking, especially in the starting phase where no or limited information about the connection is available. The efforts for the alternative approach to collect the required information by an initial measurement phase would be too high to be practically usable in a real world implementation.

It is important to mention that besides the idea to decrease the effects of buffer blocking there also exist other pro-active approaches. These ideas are based on path delay compensation as discussed in [SGTG+12], [SBB10] or [SBL+13]. The basic mechanism for these pro-active approaches always is the estimation of the path RTT or smoothed RTT (sRTT) (see Subsection 2.2.3). The measured RTT is used to perform an educated guess on the best distribution of the segments to the subflows in order to compensate for higher path delay dissimilarity. The primary goal is to organize the sending in a way that the data arrives at the receiver at the right time. A major drawback of this approach is that it assumes a relatively constant network behavior. Furthermore, the RTT difference among the paths can be too high. In this case, the throughput of a strong subflow is not sufficient to fill the existing gap caused by the delay. For example, a link delay difference of 2 seconds on a 1 Gbit/s link requires 220 MiB user data and send buffer to keep the data synchronized on the links. Of course, these mechanisms also introduce an additional delay for packets sent via otherwise low-latency paths. The compensated path delay makes the handling of multipath transfer easier, but also increases the overall transmission delay [BDB+13]. In the end the mechanisms to compensate the path delay are not robust enough to be used in unpredictable networks like the Internet.

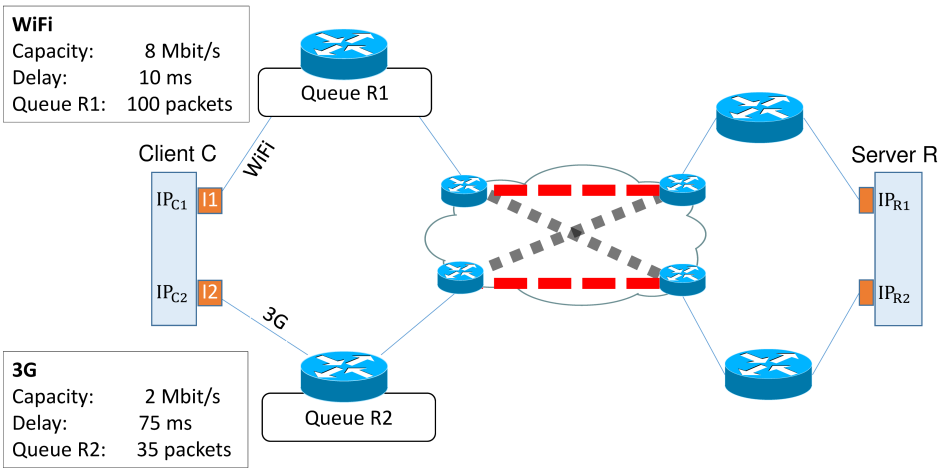


Figure 7.5: Experiment setup details

7.3.3 Analysis of deployed multipath schedulers

It is known from preparatory work in [DBRT10] and later work of [RPB+12] that a homogenous, symmetric topology has no negative impact on the throughput of a multipath flow. Furthermore, it was discussed that if there exists no bottleneck in the network, the bottleneck is caused by a shared resource on the endpoint, e.g. CPU or memory. Therefore, one goal should be to optimize the usage of memory, considering the conditions of the network to avoid buffer blocking issues caused by a heterogeneous, asymmetric topology.

There already exist different approaches to mitigate the effects of the known buffer blocking issues for MPTCP and CMT-SCTP. The experiment setup of the scenario of Subsection 7.1.2 was chosen to demonstrates the effects. Figure 7.5 illustrates the setup with more details. This thesis uses the same parameter setup as the authors of [RPB+12]:

- WiFi link with 8 Mbit/s capacity and a base RTT of 20 ms. The connecting access router was configured with a FIFO queue length of 100 packets.
- 3G link with 2 Mbit/s capacity and a base RTT of 150 ms and a configured FIFO queue size of 35 packets in the connecting access router.

All other links have no relevant impact on the path characteristics. This setup provides with the WiFi and 3G access links two paths to the network. According to the discussion of Subsection 7.1.1 it is expected that the maximum throughput of the subflows depends on the combined resources of the paths.

Therefore, under perfect conditions with lossless paths a combined bandwidth of 10 Mbit/s can be expected. If furthermore a protocol efficiency of 97% – by using an MTU of 1492 Bytes – is assumed, the theoretical maximum throughput on application level (goodput) could be expected around 9.7 Mbit/s. In the context of the IETF a multipath flow should achieve in minimum the same throughput as the strongest subflow alone. In the experiment setup the goodput of the strongest subflow alone is 7.7 Mbit/s.

Furthermore a TCP connection requires a minimum receiver buffer size. The receiver buffer has to be sized for TCP to twice the bandwidth delay product (BDP) of a path. This

provides for the sender an advertised receiver window (ARWND) large enough to ensure a continuous send process (see Subsection 5.1.2.2). Assuming a 10 ms delay for the strong path and 75 ms for the weak one a BDP of around 30 Kbytes for both paths in sum can be calculated. Therefore, already in a perfect world setup without a re-ordering issue a minimum receiver queue size of 60 Kbytes is required to support a continuous send process in sequential order.

But as also discussed, the buffer requirements of the multipath extensions are higher to be able to support the re-ordering process on flow level. Thus also for the strong path the delay of the weak path must be assumed to calculate the bandwidth delay product of both coupled path in the worst case (10 Mbit/s * 75ms). This BDP is called "virtual" BDP in the following discussion. Therefore, a new "virtual" BDP of around 90 Kbytes can be calculated for both flows and, therefore, a new minimum receiver buffer size of 180 Kbytes is required for the experiments to support a goodput of 9.7 Mbit/s. It should also be mentioned that the strongest subflow alone only requires in minimum a receiver buffer size of 20 Kbytes to achieve a goodput of 7.7 Mbit/s,

7.3.3.1 MPTCP

Figure 7.6 shows the baseline results for MPTCP as described in the IETF documents in the orange Curve #1. Furthermore, Figure 7.6 shows the results for opportunistic retransmission mechanism (Opp-Rtx Ω=on) in the red Curve #2 and for the additional penalizing mechanism (σ=on) in the green Curve #3. These extensions were proposed by the co-authors of the MPTCP draft in [RPB+12]. The LIA CCC was used for all these experiments as multipath congestion control mechanism (see Subsection 6.2.2).

Figure 7.6 shows in one graph the same results as [RPB+12] in its Figures 4a, 4b and 4d for a disjoint path setup and confirms again that the behavior of the OMNeT++/INET implementation of MPTCP (see Section 4.1) correctly models the behavior of the Linux reference implementation. The outcome confirms furthermore that MPTCP underperforms if no mechanisms to avoid or mitigate the effects of buffer blocking are applied (see Curve #1), and, that the opportunistic retransmission mechanism alone is not sufficient for a performance better or even equal the strongest subflow alone in every case (see red Curve #2). The targeted overall throughput can only be achieved if both mechanisms to mitigate all potential buffer blocking issues are applied (see green Curve #3). Both mechanisms in combination achieve a goodput of 7.7 Mbit/s for smaller receiver buffer sizes ($<$ 200 Kbytes) and close to 9.7 Mbit/s goodput for larger receiver buffer sizes ($\geq$ 200 Kbytes). At first sight, no optimization is needed for this mechanism bundle to be part of the multipath scheduler. But it is important to know that the authors tested their mechanism just for the disjoint path setup scenario (see Subsection 7.1.2), i.e. with just two subflows, rather than four as needed for the cross path setup scenario.

Figure 7.7 shows the result of the same experiment setup as for Figure 7.6. The only difference is that the complete cross table of all possible source/destination address pairs was used (see Chapter 5). The deployed mechanisms to avoid buffer blocking do not have such a positive impact anymore, in particular for smaller receiver buffer sizes (smaller than 2000 Kbyte) the throughput suffers a lot. The routing in the cross path setup is defined for 4 subflows on 4 paths, where 2 subflows are routed via the strong WiFi link and 2 via the weak 3G link. Therefore, MPTCP established two strong subflows and two weak subflows. The problem observed was caused by the second weak subflow routed via the 3G link. As only one

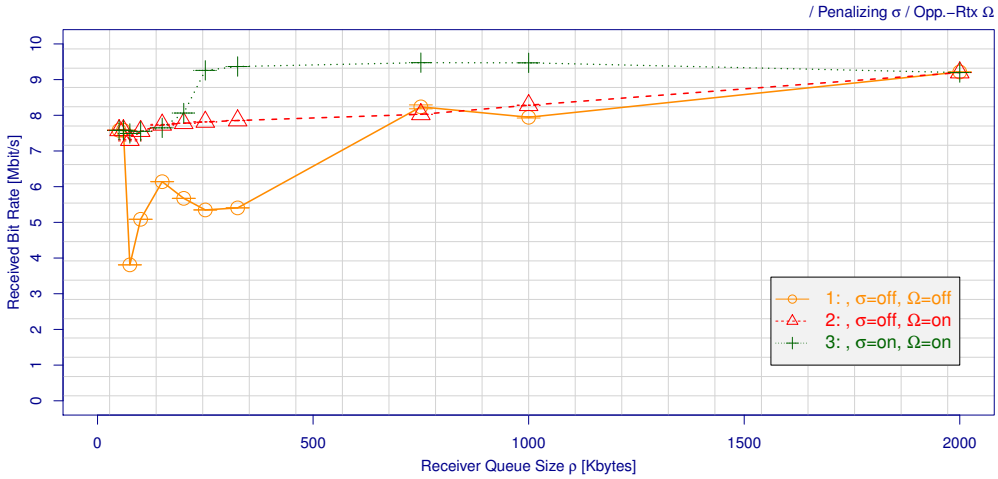


Figure 7.6: MPTCP with penalizing and Opp-Rtx in a disjoint path setup

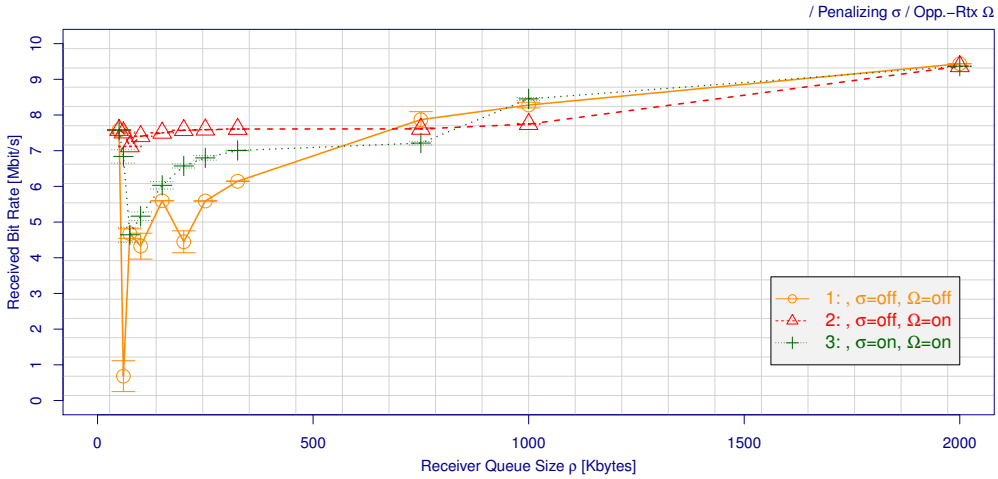


Figure 7.7: MPTCP with penalizing and Opp-Rtx in a cross path setup

link can cause the buffer blocking and can be penalized, all other subflows which cause similar problems are not handled by this mechanism. That means, the system was throttled by this second weak subflow which could not be identified by the MPTCP scheduler. Thus, even for receiver buffer sizes above 200 Kbytes up to 2000 Kbytes the overall system was not able to achieve the optimum throughput (see Curve #3 in Figure 7.7). Even worse, for receiver buffer sizes up to 800 Kbytes both deployed mechanisms to avoid buffer blocking in combination did not achieve as much throughput as the strongest subflow alone. Only the opportunistic retransmission without the penalizing mechanisms achieved a throughput comparable to the scenario with two paths (see Curve #2 in Figure 7.7). This could be expected, because the strong subflows repeat all segments of the weak subflows. However, these retransmissions for the opportunistic retransmission cause a lot of overhead[7] and are only helpful as long as the strong link has the free resources to support the opportunistic retransmission for all subflows with the potential to block the continuous send process. Figure 7.7 shows clearly

[7]Retransmissions cause copies of user data.

that much more buffer size is needed to ensure a continuous send process if both mechanisms are deployed on the receiver side to cover the gaps caused by the asymmetric, heterogeneous paths. In this example the asymmetric topology required up to 10 times more buffer as needed for the continuous send process in normal case, i.e. in detail 2000 Kbytes instead of 200 Kbytes. It is an option to turn off the cross path setup behavior[8] for MPTCP, but this would also avoid the benefits provided by this cross path setup approach as demonstrated in Section 5.3 for real world setups. Thus, the cross path setup is an important setup for the mechanisms currently proposed. Therefore, to deploy MPTCP in an Internet scenario with a real risk of buffer blocking requires further efforts to support a throughput just as good as the strongest subflow alone can achieve.

7.3.3.2 CMT-SCTP

The same scenario as before for MPTCP was used to demonstrate the effects for CMT-SCTP. This scenario was adapted to reflect the behavior of the CMT-SCTP reference implementation on FreeBSD (see Subsection 3.2.3.4). Thus, instead of the opportunistic retransmission and penalizing mechanisms, the buffer splitting and chunk rescheduling mechanisms were used. The scheduler was configured as Round Robin scheduler.

The Figures 7.8 to 7.10 impressively demonstrate the impact of the mechanisms to mitigate the effects of buffer blocking for CMT-SCTP applying the CCCs LIA, OLIA and RP-MPv2 (see Section 6.2). The Curve #1 of each figure shows the behavior without any mechanism to decrease the effects of buffer blocking as described in the current Internet draft of the IETF. The curves demonstrate that the throughput of CMT-SCTP applied on asymmetric, heterogeneous networks is poor for small receiver buffer sizes. Even worse, the throughput is less than it can be expected from a singlepath connection via the strong 8 Mbit/s link alone. Curve #1 in all figures demonstrates that a large receiver buffer size is required to achieve a throughput beyond 8 Mbit/s. This is in the LIA scenario with 750 Kbytes – similar to MPTCP – many times higher as 2 * "virtual" BDP[9]. Furthermore, the Curve #1 in all figures shows that the choice of the congestion control has a direct impact on the overall throughput of CMT-SCTP. Whereas OLIA and RP-MPv2 need less receiver buffer size (> 300 Kbytes) to achieve nearly the minimum targeted goodput (> 7.7 Mbit/s), LIA needs twice as much. Anyway, all coupled congestion controls need more as expected. That leads to the observation that the CCC and the path configuration leads to different interactions. With focus on the optimal goodput (close to 9.7 Mbit/s), the OLIA CCC (see Figure 7.12) and the RP-MPv2 CCC (see Figure 7.13) require around 1000 Kbytes receiver buffer size, whereas the LIA approach needs 1500 Kbytes (see Figure 7.11).

An optimized chunk rescheduling approach as the buffer blocking avoidance mechanisms[10] shows a positive effect for all congestion control mechanisms. The impact is illustrated in the Figures 7.8 to 7.10 by the Curve #2. At a receiver buffer size $\geq$ 200 Kbytes a throughput close to the optimal throughput can be observed for all CCCs. However, the positive effect of the buffer blocking avoidance mechanisms decreases drastically for smaller receiver buffer sizes. In comparison to MPTCP, the throughput decreases much more than expected and achieves a lower throughput compared to the strongest subflow alone (7.7 Mbytes) even for receiver buffer sizes above 60 Kbytes (which is enough buffer to saturate the strongest

[8] Like it was done for the high performing experiment in [Chr13].
[9] Calculation based on highest delay (75 ms) of the 3 G link (180 Kbytes).
[10] Ψ=on: on represents activation on both endpoints.

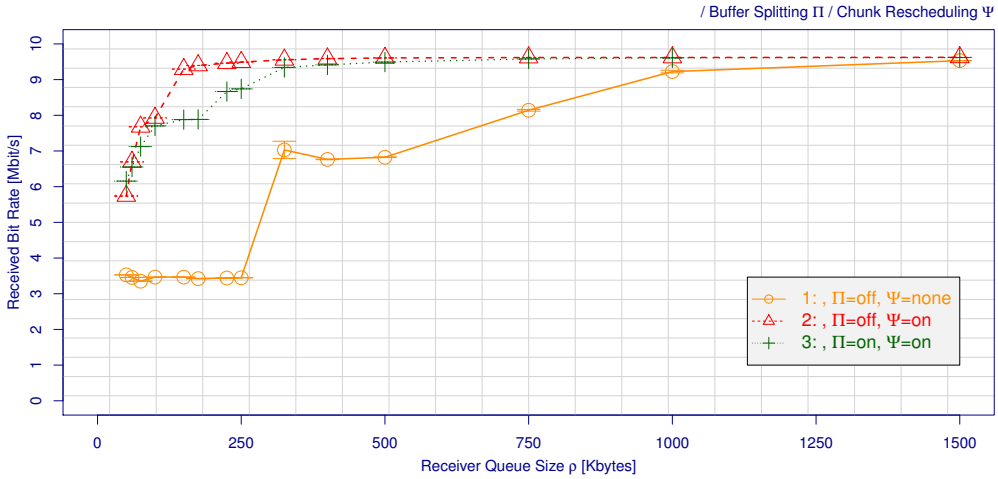


Figure 7.8: Throughput of CMT-SCTP with LIA

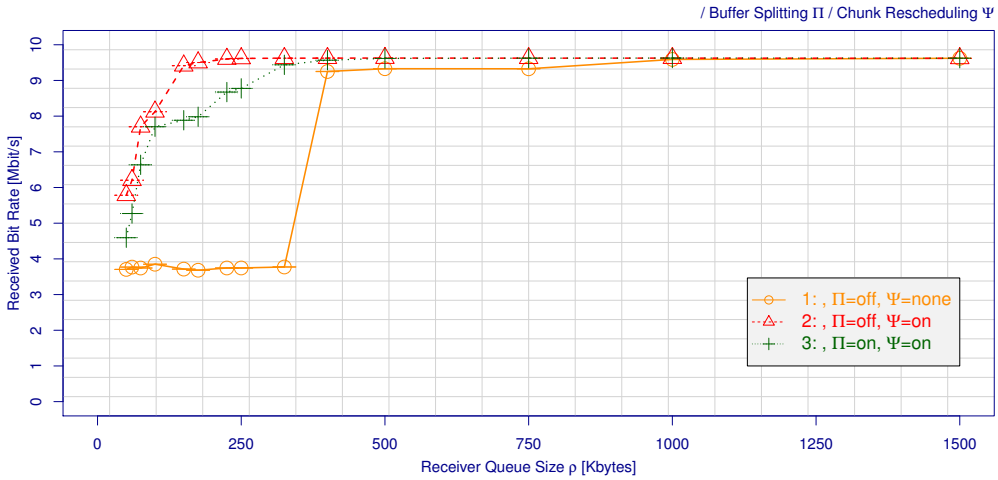


Figure 7.9: Throughput of CMT-SCTP with OLIA

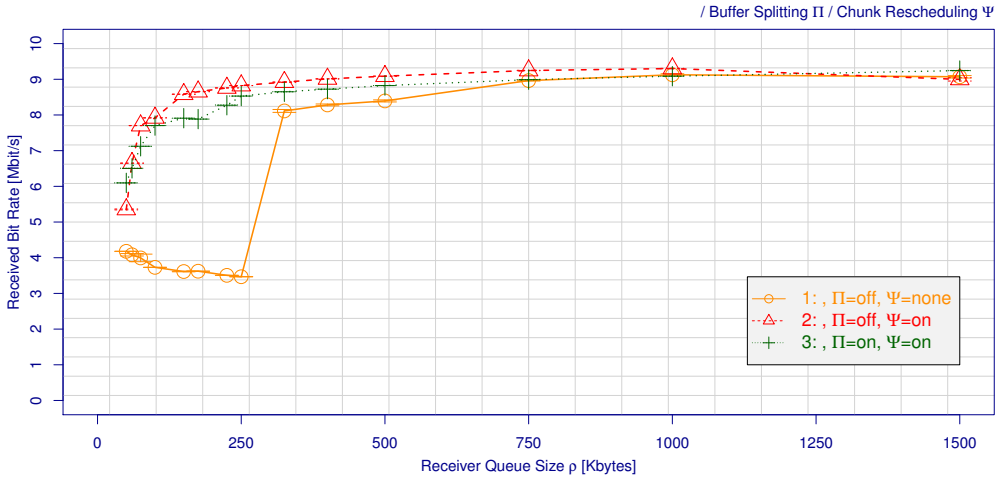


Figure 7.10: Throughput of CMT-SCTP with RP-MPv2

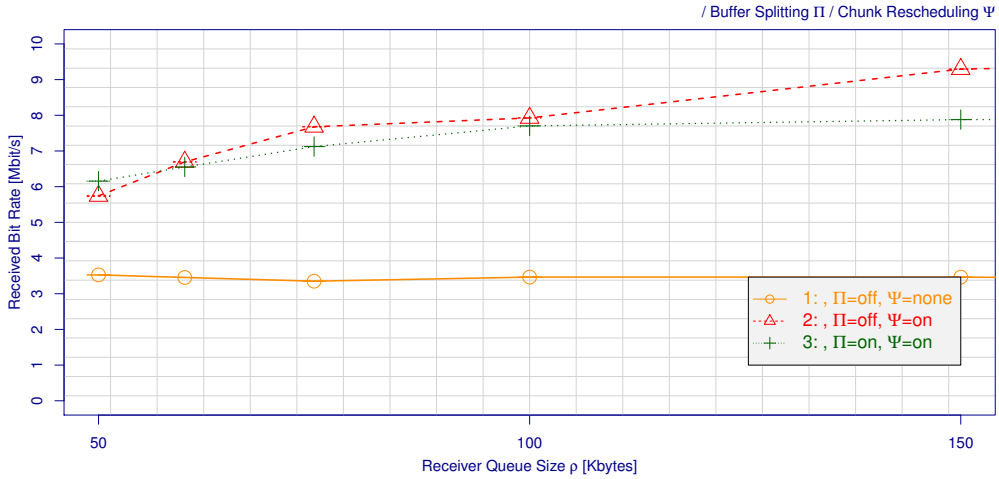


Figure 7.11: Throughput of CMT-SCTP with LIA for small buffer size

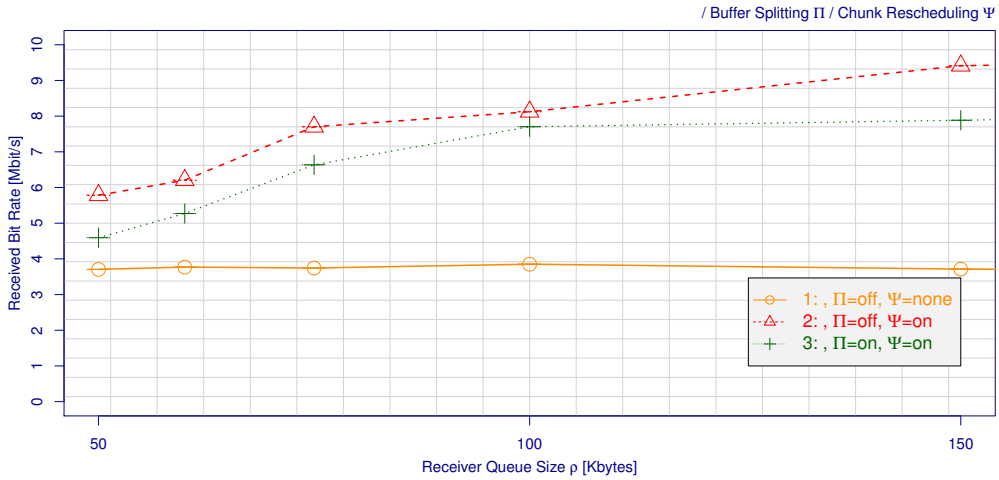


Figure 7.12: Throughput of CMT-SCTP with OLIA for small buffer size

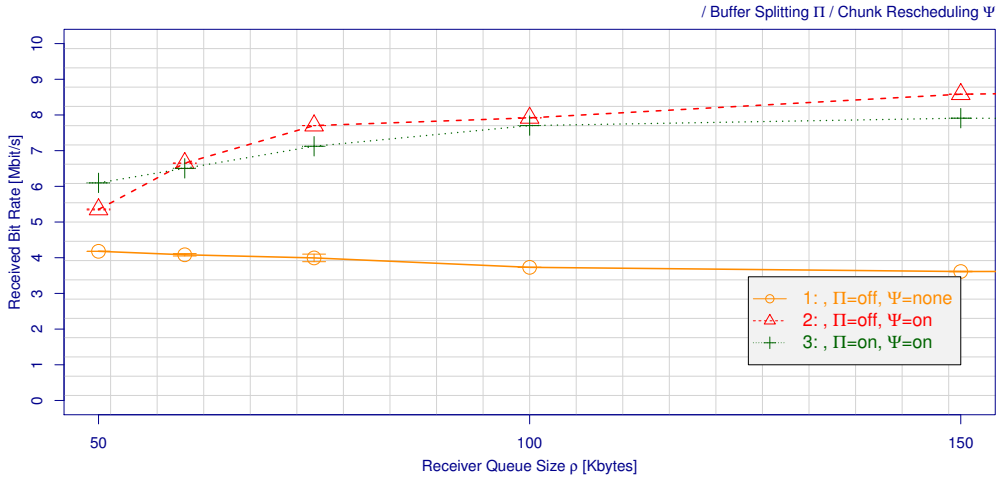


Figure 7.13: Throughput of CMT-SCTP with RP-MPv2 for small buffer size

subflow). Therefore, the target throughput cannot be achieved with mechanisms to decrease the effects of buffer blocking for small receiver buffer sizes in CMT-SCTP. Until now only experiments with a receiver buffer size large enough to cover twice the "virtual" BDP, like e.g. in [Dre12a] were designed for CMT-SCTP. The behavior below this limit was not in the focus so far, even if smaller buffer sizes decrease the positive effect of buffer splitting[11] below the throughput of the strongest subflow alone. However, this buffer size range is important for the practical use and matters for system administrators to design effective, scalable and usable systems on base of the straightforward buffer size calculation. Wrong assumptions will lead to a poor overall system performance, even by activating every known mechanism.

Figures 7.11 to Figure 7.13 give a more detailed look on the usability of the mechanisms to avoid or decrease the effects of buffer blocking in combination with the LIA, OLIA or RP-MPv2 congestion control for small buffer sizes. It clearly shows that the Curve #2 – where only chunk rescheduling was enabled – is in all cases above Curve #3 where all mechanisms were activated. None of the Curves #3 in all three figures achieves an overall throughput comparable to the strongest singlepath flow on the strongest path (see Figure 7.6). This effect is most noticeable if the OLIA congestion control is used (see Figure 7.9 Curve #3). By using OLIA as congestion control, the throughput decreases for smaller buffer sizes ($<$ 100 Kbytes) by 20% compared to the strongest subflow alone. To be clear here, this load sharing behavior does not even achieve the goals of the IETF specification. The reason for this poor behavior can be found in the startup phase of the association. In the startup phase the CMT-SCTP multipath scheduler allocates buffer for both subflows. This leads to the situation where there does not exist enough buffer space to saturate the strong subflow. To fill the gaps between the subflows requires a lot of buffer as explained in the discussion about receiver side buffer blocking issues (see Subsection 7.1.3.4). Furthermore, the buffer splitting mechanism makes the situation even worse for small receiver buffer sizes, because in the basic setup this mechanism reserves parts of the buffer for the weak subflow, even if the buffer could be used more effectively for this strong subflow.

Furthermore, it is important to mention that the mechanisms to decrease the effect of buffer blocking in combination with the congestion control RP-MPv2 do not achieve the optimal goodput of 9.7 Mbit/s at all. The reason is the coupled window behavior as also observed in Section 6.3. Whenever a buffer blocking issue could be detected, the window growth for all paths was limited by the mechanisms. This led to a limited window, which led to a less aggressive increase factor and a smaller congestion window size (see Subsection 6.2.1).

7.3.4 Conclusion

Altogether, the implementation of a multipath scheduler without mechanisms to avoid or mitigate the effect of buffer blocking makes no sense. However, none of the mechanisms was able to achieve the expected results for smaller buffer sizes in the disjoint path setup of CMT-SCTP. Even worse, the buffer splitting mechanism of CMT-SCTP had a negative impact on the throughput with small receiver buffer sizes. Also, particularly in a cross path setups, MPTCP did not achieve the same throughput as the strongest subflow alone – which is a major goal of the multipath approaches. Here alternatives must be developed to improve the scheduler with mechanisms that are really able to achieve increased throughput.

[11] Π=on: on represents activation on both endpoints

7.4 Optimized scheduling variant for CMT-SCTP

CMT-SCTP does not perform as expected. A possible solution for this CMT-SCTP issue is to learn from MPTCP. Therefore, a first step is to deploy a similar weighted scheduling mechanism for CMT-SCTP as for MPTCP (see Subsection 3.2.3.4) described as a successful scheduler strategy by [SGTG+12].

This scheduler changes the protocol behavior in two ways. First the scheduler only schedules segments on the fastest subflow, and second even if there exists free send or receive buffer on an alternative path, the resources are not used as long as the fastest subflow is not limited by e.g. a falling congestion window or loss. Therefore, the alternative paths are not used as long as the fastest path provides continuous sending, even if the fastest subflow has the filled congestion window in flight. Thus, a completely filled congestion window is not defined as a blocking resource. This provides a real benefit for a scenario with limited buffer size – like demonstrated in Figure 7.14 – because the resources are allocated to the strongest link and will not be blocked by alternative subflows. In the worst case this allows the strong subflow to use the resources completely and the whole system performs like the fastest subflow.

The positive effect will be explained by using Round Robin. If Round Robin is used to schedule the segments via the subflows, the receiver buffer will be fragmented by the weak subflow. The weak subflow is not able to provide all messages fast enough for the ordered delivery to the application layer. Therefore, gaps exist in the receiver queue, which block the buffer. The strong subflow is not aware of these gaps, therefore the strong subflow sends as normal and continuously fills the receiver queue until all space is blocked. This fragmentation should be avoided, therefore the strongest subflow should be preferred until the strongest subflow is blocked. But this is not possible directly after the first handshake, when the resources are allocated for the first time. No information about the network exists, and therefore it is unknown which subflow should be preferred. Here a heuristic is required.

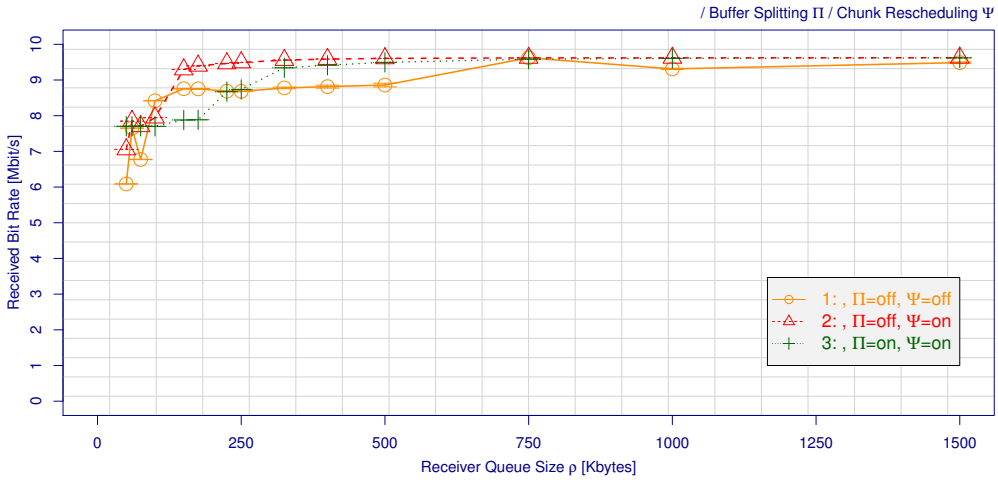


Figure 7.14: Weighted RTT based scheduling with LIA

Experiments with MPTCP show the benefits of a good heuristic by preferring the fastest subflow. Thus all receiver buffer resources are allocated to the fastest subflow during the startup phase as long as no other information exists. It is important to note that the focus on

the RTT is not always a clever idea, because there exists no reliable relationship between a low RTT and the overall throughput of the subflow during a send cycle. Thus, the throughput will suffer if the strongest subflow does not have the lowest RTT. However, with this heuristic at least the chance exists to achieve an optimal throughput, even if this chance decreases with the number of alternative paths.

Figures 7.15, 7.16 and 7.17 show the results for all coupled congestion controls mechanisms – LIA, OLIA and RP-MPv2 – with the focus on small buffer size if a weighted RTT based scheduler is used. First, using a weighted scheduler had the same positive effect for the CMT-SCTP in the experiment setup as for MPTCP. Even more, without any buffer blocking mechanism (see Curve #1 of all figures) the load sharing extension showed at least a better throughput than for MPTCP also for small buffer sizes.

Second and as expected, no CCC was able to achieve a minimum target throughput with a receiver buffer size of < 60 Kbyte without further efforts as can be observed looking the first measurement point of Curve #1 in the figures. The second measurement point of Curve #1 reflects an acceptable behavior by achieving a throughput that can be expected as minimum for the multipath flow. However, the third measurement point of Curve #1 shows a decrease again. This was caused because the receiver buffer size provided was larger than needed to support a continuous sending on the strongest subflow. Therefore, all 100 simulation runs aggregated in the third measurement point of Curve #1 reflect the impact of the advertised window induced receiver buffer blocking (see Subsection 7.1.3.4). No equilibrium could be achieved for smaller receiver buffer sizes in the setup, where the amount of receiver buffer allocated to the weak subflow prevented a continuous send process during a send cycle. Data was missing in the re-ordering process during a send cycle of the strong subflow and has not allowed the needed freeing of receiver buffer. Thus, even if more data was sent at first sight during the first send cycle, the data on the weak link has decreased the amount of data for the second send cycle and that had an impact on the throughput in this setup.

The figures demonstrate that there is a strong need for the existing mechanisms to decrease the effects of buffer blocking, if the resources of the strongest link cannot be saturated (see Curve #2 and Curve #3). The more data is sent via the weak subflow the more receiver buffer is needed as known from the Round Robin scheduler (see Curve #1). In this situation the chunk rescheduling mechanism can help, because this mechanism retransmits the missing segments on the strongest path and achieves in minimum the throughput of the strongest path alone (see measurement point 2 in Curve #2 of all figures). But also the chunk rescheduling mechanism alone is not enough for buffer sizes up to 750 Kbytes as can be seen in Figure 7.14. The effect becomes particularly obvious by the difference between Curve #2 and Curve#3 in the receiver buffer range above 250 Kbytes. Only the combination of chunk rescheduling and buffer splitting is able to achieve an overall throughput close to the optimum.

The third result concerns the negative impact of the mechanisms to decrease the effects of buffer blocking, in particular with respect to the configuration of the buffer splitting mechanism. Again, the strongest path needs in minimum twice its BDP as receiver buffer. But the buffer splitting mechanism is not aware of this minimum size. Therefore, the buffer is just allocated in equal parts to the subflows. So two effects come in for the Curve #3 in Figures 7.15, 7.16 and 7.17 for the receiver buffer range between 20 Kbytes and around 250 Kbytes. First, the strongest subflow is limited by the buffer splitting, because it halves the usable ARWND for the strong path. Second, the opportunistic retransmission compensates this limitation by retransmitting segments transmitted first on the weak subflow. Thus, the buffer splitting mechanisms should be configured very carefully in setups with limited receiver

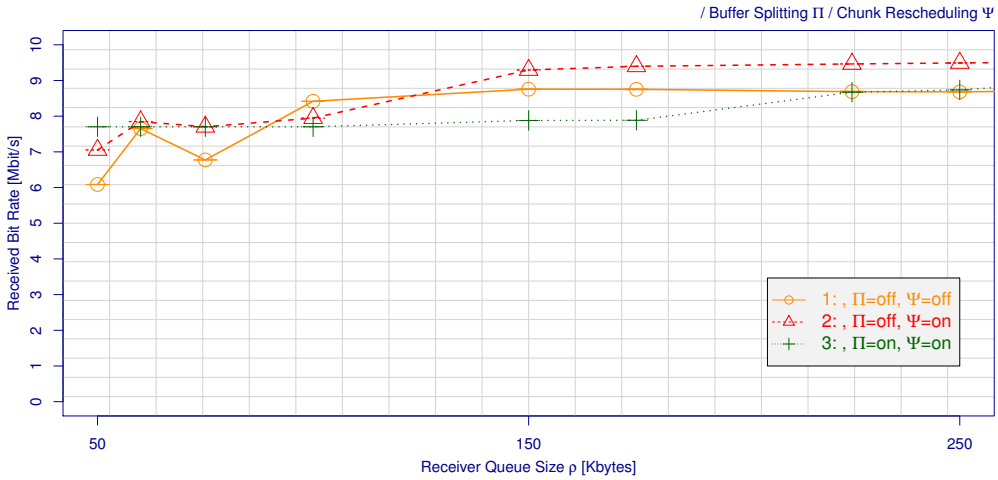


Figure 7.15: Weighted RTT based scheduling with LIA and focus on small buffer size

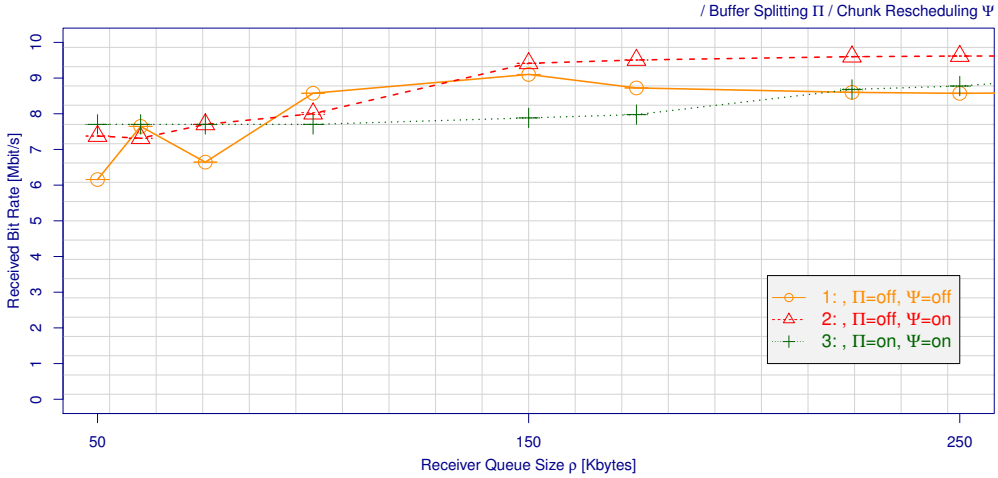


Figure 7.16: Weighted RTT based scheduling with OLIA and focus on small buffer size

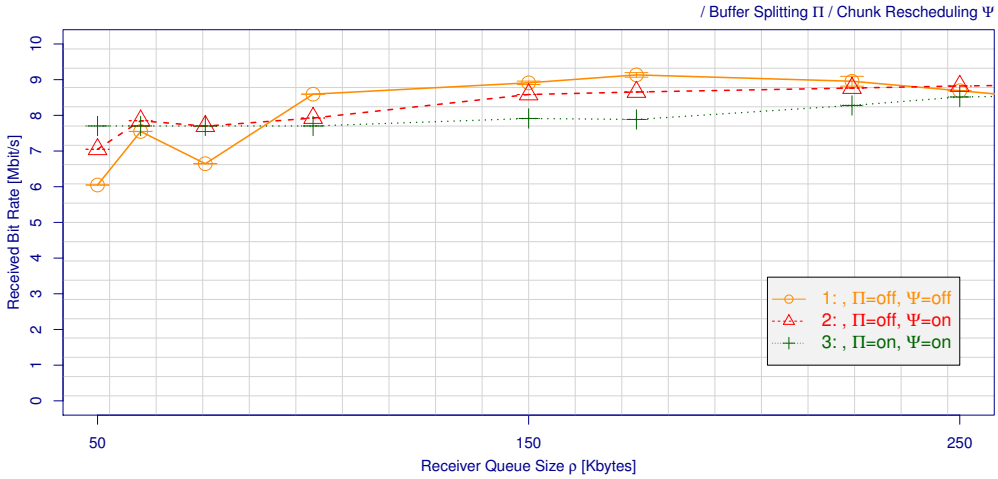


Figure 7.17: Weighted RTT based scheduling with RP-MPv2 and focus on small buffer size

buffer in order not to limit the strongest path.

The fourth result of this analysis is that, the CCC has an impact on the overall system throughput. It is surprising that for RP-MPv2 and a receiver buffer size between 100 Kbytes to 250 Kbytes the best throughput can be achieved without a mechanisms to decrease the effects of buffer blocking. The different aggressiveness increase factor of the congestion controls has an impact here that allows for example for RP-MPv2 a preference of the strongest flow with respect to the receiver buffer space because it is not limited in growth.

Altogether, the existing bundle of mechanisms to mitigate buffer blocking for CMT-SCTP is able to achieve a throughput comparable to MPTCP as long as the scheduler gives the system the chance to avoid buffer blocking at startup time[12].

7.5 Optimized scheduling variant for MPTCP

As discussed above, the combination of opportunistic retransmission and penalizing as mechanisms to reduce the effects of buffer blocking are not enough in the cross path setup. The throughput of the MPTCP flow suffers much more as it can be expected from the strongest subflow alone. This effect is not valid for smaller receiver buffer sizes in a disjoint path setup and, therefore, not visible in such experiments. However, the cross path setup can be expected to be important in the Internet (see Chapter 5) and every mechanism has to be evaluated using the cross path setup for MPTCP, too.

Of course, it is not possible to achieve the maximum throughput in every case, but it should be possible to achieve a throughput comparable to the strongest subflow alone. In this section a new approach is introduced for MPTCP that replaces the opportunistic retransmission as the main mechanism to reduce the effects of buffer blocking. This alternative approach is denoted in this thesis as number.

7.5.1 Idea behind the confluent sequence numbers approach

Before starting the design process for an alternative mechanism to avoid or to decrease the effects of buffer blocking, the drawbacks of the current approaches are discussed.

The main problem with the opportunistic retransmission approach is the unknown state of the network. As discussed before (see Subsection 7.2.2.1) a segment resent via an alternative subflow by this mechanism has a potentially high impact. Data retransmitted by the opportunistic retransmission mechanism has to be transferred on subflow level even if the transmission is not necessary on flow level anymore. This can happen because a retransmission via an alternative path always uses a copy of the segment (see Subsection 7.2.2.1). The scheduler is not free to revise this decision once the data is enqueued in the subflow send queue and a subflow sequence number was assigned to the segment. Therefore, opportunistic retransmission for MPTCP can lead to a duplication of packets without any benefit. Thus, opportunistic retransmission has the potential to fail due to two different reasons:

- **Using the wrong path**
  This can happen easily for small receiver buffer sizes, since an opportunistic retransmission choses an alternative subflow for retransmission when a specific path is blocked. But the problem is not avoided if the newly selected subflow for the retransmission also has the additional potential to block the send process. Even worse, the performance

[12] In optimal case the application uses the best path for first connection setup.

of the second weak subflow is decreased more than needed, because this weak subflow retransmits copied segments without a benefit. This problem is caused by the fact that the subflow with the best statistics not always represents the strongest subflow, i.e. the strongest subflow can be limited by competition or reduced receive buffer size. Thus, the measurement or the estimation of the path characteristics provides the basis for these wrong assumptions.

- **Early retransmissions**
  This occurs if, e.g., the segment that blocks the buffer has actually arrived, but the sender was not notified yet by a corresponding acknowledgment. Thus, the buffer is possibly freed on the receiver side if the segments have arrived ordered, but the sender buffer is still reserved until the cumulative acknowledgment has arrived. Even if there is no need of a retransmission, the mechanism to avoid buffer blocking will be triggered as long as the acknowledgment is in transmission. This has two consequences: first the buffer is not available to send new data and second the blocking segment will be sent as retransmission via an alternative path (see Subsection 3.2.3.3). This effect can occur often with small buffer sizes, because all subflows are in competition for the limited buffer space and try to allocate as much as possible. Clearly, one subflow is always the sender of the last missing segment and will be penalized if the remaining available buffer size is too small. That means that the free resources are wasted for an unnecessary retransmission and a potential strong subflow will be penalized without any reason. Even worse, this kind of "false positive" retransmission on a weak subflow can increase the potential buffer blocking even more.

The reason for both shortcomings is the same, i.e. the assumption that all data has to be transmitted ordered to the receiver. The dissimilar paths reorder the segments anyway, although they have been kept ordered during the send process on subflow level[13]. Even if the network may reorder the segments, the send process is organized with a strictly ascending order on flow and subflow level. But a useful transport by the paths in strictly ascending order on flow level cannot be supported without exact knowledge of the network. On subflow level an ordered transport can be assumed like for the singlepath transfer, but on flow level a re-ordering is more the default as the exception and causes only problems. A strong requirement for a strictly ascending order on flow level does not exist and, therefore, this thesis argues for a send process with a strictly ascending order on subflow level only. The only requirement is a numbering of the application data in strictly ascending order with flow level sequence numbers. With this in mind the send process provides unused degrees of freedom.

The main goal of the IETF load sharing approaches is to increase throughput. The efforts for multipath transfer only make sense if there is more data available than the strongest subflow alone can process. Thus, there is more data in the flow send queue than can be assigned to one subflow send queue. Therefore, the scheduler can subdivide the enqueued data in the subflow send queue into two parts, namely the data that can be queued in the subflow send queue of the strongest subflow and data that is ready for the send process but cannot be queued for the strongest path.

For example, if a simple file transfer is the use case, the file size should be bigger than twice the BDP (see Subsection 2.2.3) of the strongest path to argue for multipath transfer. If the file is less than this size the use of the load sharing extension makes no sense. Thus,

[13] No re-ordering of segments sent on the same path.

the user data can be divided into at least two ranges for two subflows. The first range has the size of the bandwidth delay product of the strongest path, the second range consists of the rest and in maximum the current window size of the congestion and flow control of the alternative path. It would make sense to schedule the first range via the subflow assigned to the strongest path and the second via the weak. The benefits are:

- The receiver gets ordered blocks of the transmitted data and that requires less effort in the reordering process.
- The buffer blocking problem will not occur in this scenario.
- The second range of the data has more time to arrive at the receiver in time, because the first block has to completely arrive first.

Of course no protocol implementation has unlimited memory to buffer complete files of arbitrary size and an absolute limit for the sender has to be defined. This limit of data that can be sent is given with the advertised receiver window (ARWND). The advertised receiver window (ARWND) is used to report the free receiver buffer space to the sender. Thus, if the focus is again on the file example, the ARWND and the starting flow sequence number space (DSS) (see Subsection 3.2.3.2) identify start and end point of a "block" to transfer. However, because of the information gaps discussed in Subsection 7.1.3.1, the calculation for splitting the data in such blocks is hard in practice. Anyway, the mechanism introduced in this thesis enables a natural behavior by using confluent sequence numbers on flow level, which avoids the need of a bandwidth calculation. The details of this approach are discussed in the following section.

7.5.2 Confluent sequence numbers (ConSN)

The basic idea of confluent sequence numbers as such is simple, but the integration into the existing protocol infrastructure introduces a significant complexity.

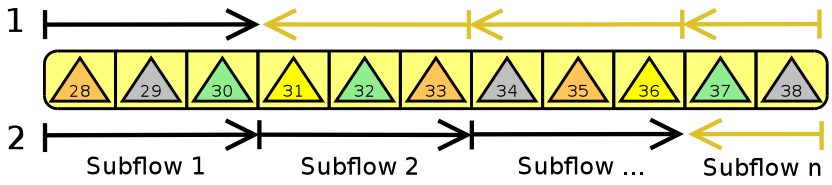


Figure 7.18: The principle of the ConSN strategy

Figure 7.18 illustrates the principle: instead of sending the flow sequence numbers in strictly ascending order, the blocks of the send buffer are split up among the n subflows. Two scenarios are possible:

1. Subflow #1 gets the segments in ascending order and the other $n-1$ subflow in descending order, or
2. Subflow #1 to #$n-1$ get the segments in ascending order and subflow #n in descending order.

It is obviously advantageous if the strongest subflow is always used to send the first range of the data enqueued in the flow send queue, but it is not a strict requirement for the scheme. In the worst case the average delay of the flow will be increased by the average delay of the chosen subflow. However, the key idea of this strategy is that the flow sequence number is used confluent (hence the name confluent sequence numbers). Thus, if there exists a problem – either a jump of the delay or a complete link failure – on a subflow $\#j$ $(1 \leq j < n)$, the transmitted flow sequence numbers run towards the missing flow sequence number. Thus, no coordination is needed. Therefore, a successful retransmission of the missing segments on a strong path can easily solve a blockage caused by a weak subflow.

7.5.2.1 Example for two subflows

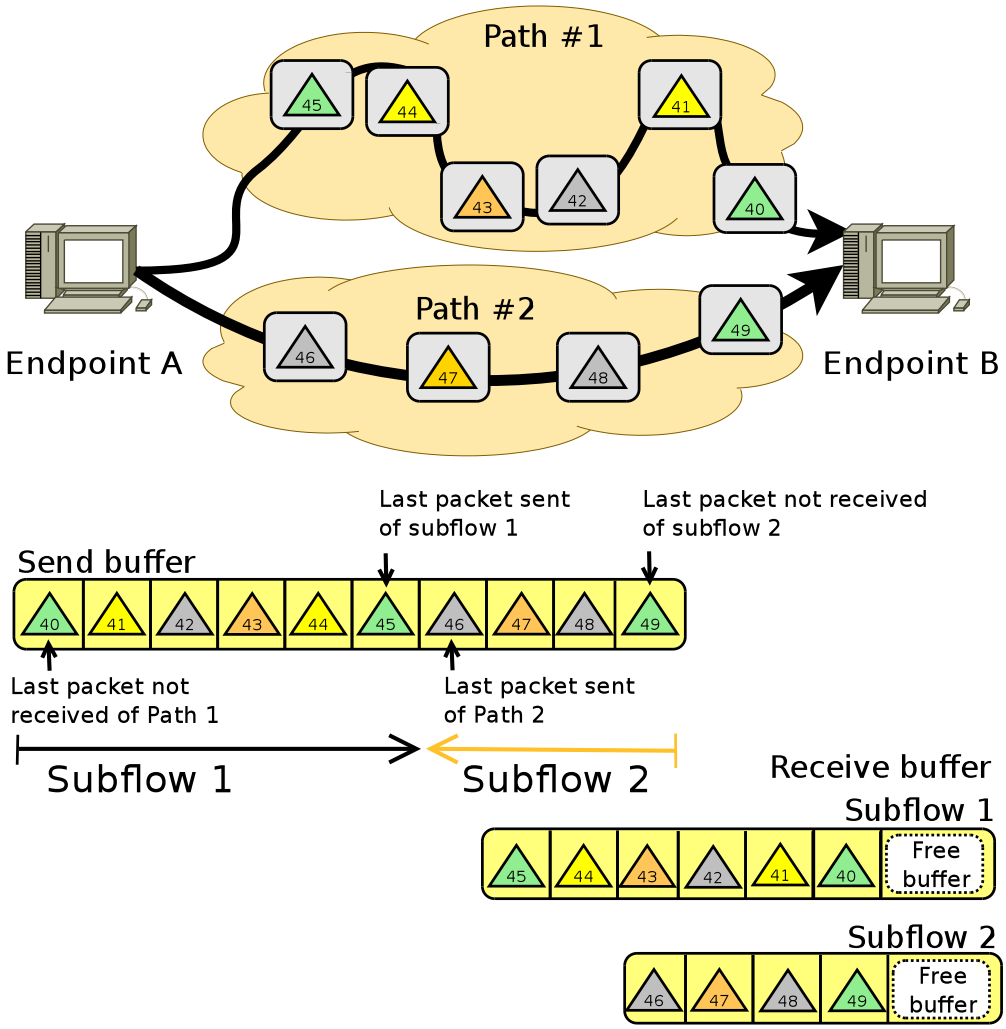


Figure 7.19: An example for the usage of ConSN

Figure 7.19 presents an example with only two subflows where both subflows are assigned to a specific path to make the algorithm as implemented in the MPTCP module of OMNeT++/INET more clear. Thus subflow #1 is assigned to path #1 and subflow #2 is

assigned to path #2. Segments #40 to #45 are transmitted on subflow #1 and the segments #46 to #49 are sent – in descending order – on subflow #2. All segments have to be delivered to the remote application in sequence. A loss of segment #45 (on subflow #1) would block the segments #46 to #49 (on subflow #2) in the receiver buffer, due to the need for in-sequence delivery. The successful retransmission of segment #45 on subflow #2 as described in the following allows to release the whole receiver buffer size allocated to path #2.

ConSN leads to consecutive blocks of missing segments for subflow #1 or subflow #2 as illustrated by the receiver queues of both subflows in Figure 7.19. This allows an efficient signaling of missing segments by the receiver, e.g., by providing the first missing flow sequence number and the length of the block, instead of listing many scattered segment sequence numbers (see SACK in Subsection 2.2.3). Furthermore, having consecutive blocks instead of scattered missing segments reduces the buffer blocking issues described in [ADB+11].

In addition, ConSN can improve the handover performance in mobility scenarios by just having to retransmit some consecutive segment blocks in break-before-make scenarios. However, dynamic path management – adding and removing subflows during a connection – is not in the focus of this thesis. It should nevertheless be mentioned in this context that this approach also provides a potential benefit for this kind of scenarios.

Furthermore, with ConSN it is possible to perform retransmissions preventively, in order to avoid buffer blocking. Given the example in Figure 7.19, subflow #1 can continue with segments #46 and higher after having sent segments #40 to #45. In this case, possibly missing segments higher than #46 are recovered by subflow #1. On the other hand, subflow #2 can continue with segments #45 and lower after having sent segments #49 to #46. These overlapping – and therefore redundant – segment transmissions on both subflows reduce the delay[14] at the cost of bandwidth usage.

In the worst case, the strongest subflow will retransmit all data of the weak subflows assigned to paths with poor characteristics with respect to bandwidth, error rate or delay. This will reduce the effective throughput to the throughput of the strongest subflow alone. Furthermore, since the sender tries to saturate all subflows at startup in the same way, this scheduler behavior can be used as a first measurement to identify the strongest link, because it uses all subflows from the beginning and increases the congestion windows for sending during this phase. This is the behavior required to support the weighted scheduler approach of [SGTG+12].

An important aspect of ConSN is an effective usage of the send blocks which are denoted as ConSN blocks. The first ConSN block depends on the ARWND and the initial flow sequence number which is calculated during the initial handshake of MPTCP. The first block for the ConSN can be calculated when there is user data enqueued in the flow send queue for the first time. The start of the ConSN block is denoted as *block_snd_una* and is in the first step equal to the initial flow sequence number. The end of the ConSN block is denoted as (*block_snd_max*) and is calculated by adding the size of the ARWND to the initial flow sequence number.

The next step is the assignment of the subflow send windows to data blocks of the overall ConSN block. There is no information about an adequate assignment after the initial handshake so it can be done very straightforward by considering the RTT. Subflow #1 as fastest subflow sends the block in ascending order and subflow #2 in descending order.

The next step is to prevent that both subflows send the complete ConSN block. An abort

[14] By avoiding the need for retransmissions.

criterion denoted as *block_abort_subflow* is defined that works as *snd_max* for both subflows. A subflow must transmit all segments of its send window first before a new send window can be assigned. It is important in this context to understand that a window can be "be sent" or "be acknowledged". In detail, *snd_una* to *snd_max* defines the maximum window of the subflow. *snd_una* to *snd_nxt* represents the data sent but not acknowledged. Therefore, in the ConSN a new send window can only be assigned to a subflow if *snd_nxt* is equal to *snd_max*. This is independent of whether the send window is fully acknowledged (*snd_una*=*snd_max*) or not. In the initial step, the ConSN block is divided by two to calculate *block_abort_subflow*[15]. This block building process is denoted in this thesis as major block building process. For subsequent major block buildings an adaption of the abort criterion should be provided by a adjustment value. This adjustment value should be calculated with respect to the number of acknowledged segments sent via the subflows. For this thesis, the adjustment value was defined as the difference of the acknowledged bytes of both subflows between two major block building events.

Figure 7.20 illustrates a specific Time *A* after the first major block building process was done. In this figure a blue rectangle represents the amount of user data provided by the application layer. The edge on the left side represents the start of the ConSN block and is therefore identified by *block_snd_una*. The ConSN Block is smaller than the enqueued user data and, therefore, the other end is defined by *block_snd_max* according to the ARWND. The *block_abort_subflow* line reflects the calculation of $\frac{block\_snd\_max - block\_snd\_una}{2}$. The sending order of subflow #1 can be identified by the black arrow and the one of subflow #2 by the orange. The sequence number space between the start point of subflow #1 and the abort criterion defines Block 1 and the difference of the abort criterion to the start point of subflow #2 defines Block 2. The sequence numbers allocated to the blocks also define the send windows for the respective subflows. Both flows have already sent data, but until Time A no acknowledgements have arrived via the subflows. It is assumed that the subflow #1 is

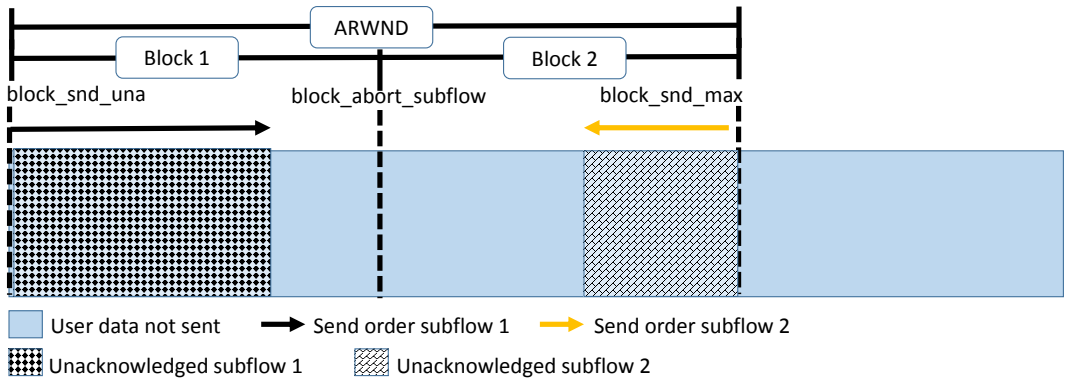


Figure 7.20: Time A - After major block building

the strongest subflow. Thus, subflow #1 will fill its block first and, thus, exhausts its send window. As discussed, it is mandatory to provide a continuous send process for the strongest

[15] A corresponding calculation is also used if there are more than 2 subflows to define the subflow windows for all flows.

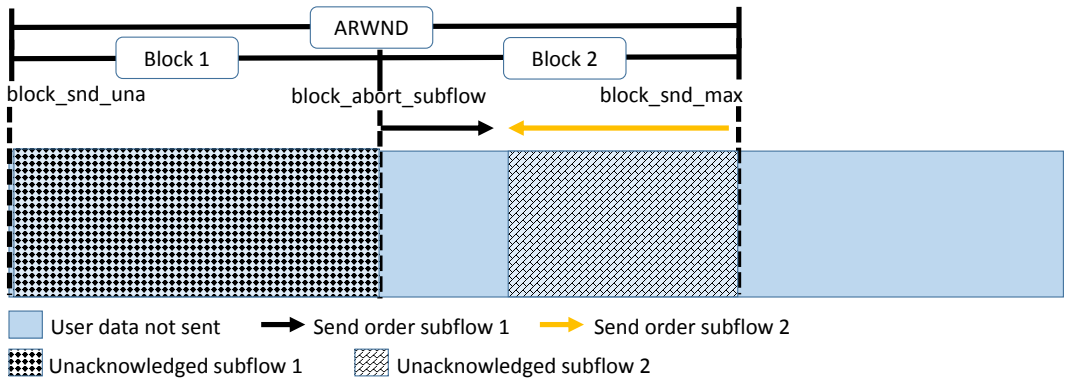


Figure 7.21: Time B-1 - After complete sent Block #1 and no acknowledgments arrived

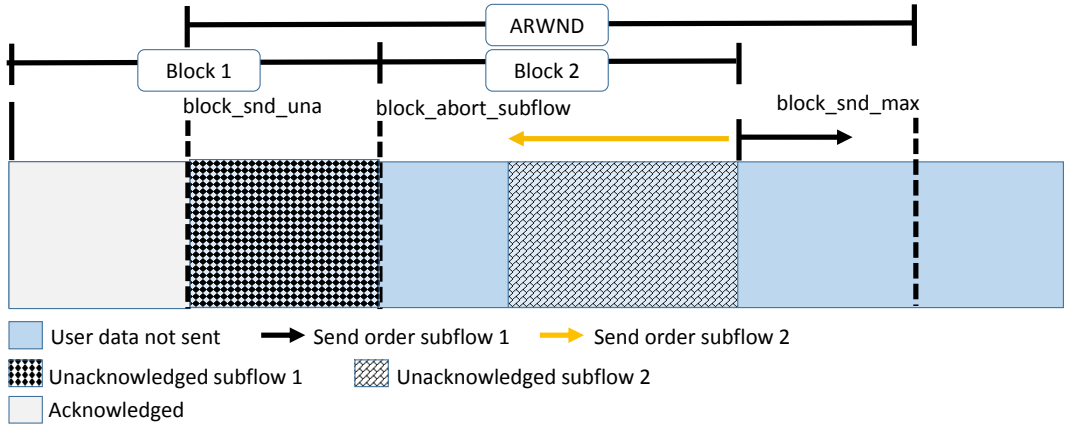


Figure 7.22: Time B-2 - After complete sent Block #1 and acknowledgments arrived

subflow. Therefore, a new part of the ConSN block must be assigned to the strongest subflow #1. Here, two different cases can be distinguished:

- **No acknowledgements have arrived via subflow #1 (B-1)**
  Figure 7.21 illustrates the conditions after the subflow #1 has filled its send window and no acknowledgments have arrived. Therefore, no buffer has been freed and the remaining buffer is still allocated to subflow #2. Thus, the subflow #1 has no other alternative as to work on the block assigned to subflow #2. The only difference is, that the subflow #1 sends this block in ascending order. Subflow #1 only stops this send process either when the combined acknowledgements of both flows have fully acknowledged the complete block or after subflow #1 has also successfully transmitted the complete Block 2. In this case the subflow #1 has sent the complete ARWND like a normal singlepath flow. The subflow #2 stops sending if the Block 1 is completely acknowledged and Block 2 is completely sent by subflow #2 or if the complete block is acknowledged by both subflows combined. If the Block 1 is not completely acknowledged subflow #2 starts

to transmit Block 1 in descending order to avoid gaps caused by a connection abort of subflow #1.

- **Acknowledgments have arrived via subflow #1 (B-2)**
  Figure 7.22 illustrates the situation after subflow #1 has sent the complete Block 1 and a number of acknowledgments have arrived. These acknowledgements of the strong subflow #1 freed blocked buffer, increased *block_snd_una* and, therefore, *block_snd_max*. Thus there is new free buffer space available that is assigned to the subflow #1. This has the positive effect that more time is provided to subflow #2 to fill its Block 2. The new additional block of subflow #1 should be sent in ascending order to make a future major block building process easier. This assignment of freed buffer to subflow #1 can be repeated until Block 1 is completely acknowledged.

So far a continuous send process for subflow #1 was ensured by the assignment of additional send windows. If the situation occurs that both blocks are completely sent by subflow #1 and subflow #2 and no acknowledgments have arrived something is wrong with the system setup. This situation can only occur if not enough ARWND is provided for the send process or both subflows are disturbed – normally new acknowledgments can be expected. ConSN uses as major trigger that all acknowledgments have arrived for Block 1. After that the following two situations can be distinguished according to the discussion above.

- **Subflow #1 re-used buffer equal to the size of Block 1 (C)**
  Here the allocation of freed buffer to subflow #1 was performed as often as possible until the complete space of Block 1 has been reused by subflow #1. In this situation the subflow #1 again has no other alternative as to repeat the send window of subflow #2 in ascending order. Figure 7.23 reflects the conditions after such a situation. Nearly the complete ARWND was already sent except the parts of Block 2 assigned to subflow #2. Subflow #1 and subflow #2 work on this gap until Block 1 and Block 2 are fully acknowledged or subflow #1 has sent the whole Block 2. After that, a complete new major block building process can be performed.

- **Subflow #1 re-used buffer less than the size of Block 1 (D)**
  Subflow #1 has sent additional data on top of block 1. The additional send window was defined as Block 2 (see B-1) or as reused buffer space smaller than the size of Block 1 (see B-2). ConSN uses the free buffer size for a new major block building process, where the subflow #1 remains the strong subflow. Figure 7.24 reflects the situation at a specific Time D, where the new block building process has been done after the Block 1 was completely acknowledged. The only difference to a "normal" major block building process is, that acknowledged data could be part of the block. But this has no negative effect, because these segments will not be enqueued in the subflow queue, if they are already acknowledged on flow level. During the new calculation process of *block_abort_subflow* a adjustment value has been considered to optimize the block building process.

The case when the subflow #2 sends faster than subflow #1, i.e. the wrong subflow was selected, was not discussed so far. Here two cases have to be distinguished:

- **Block 1 has not been sent completely**
  This case is similar to the case described in Figure 7.21 with the difference that subflow

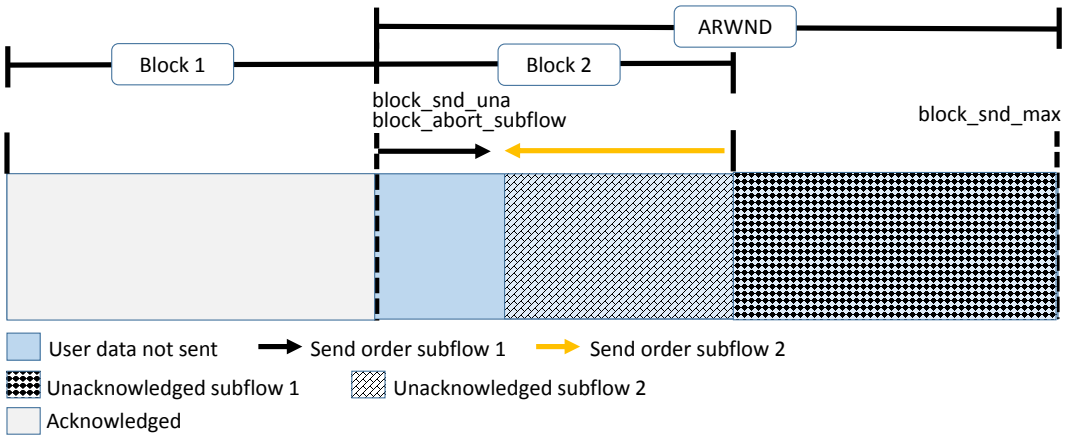


Figure 7.23: Time C - After complete sent Block #1 and no more receiver buffer

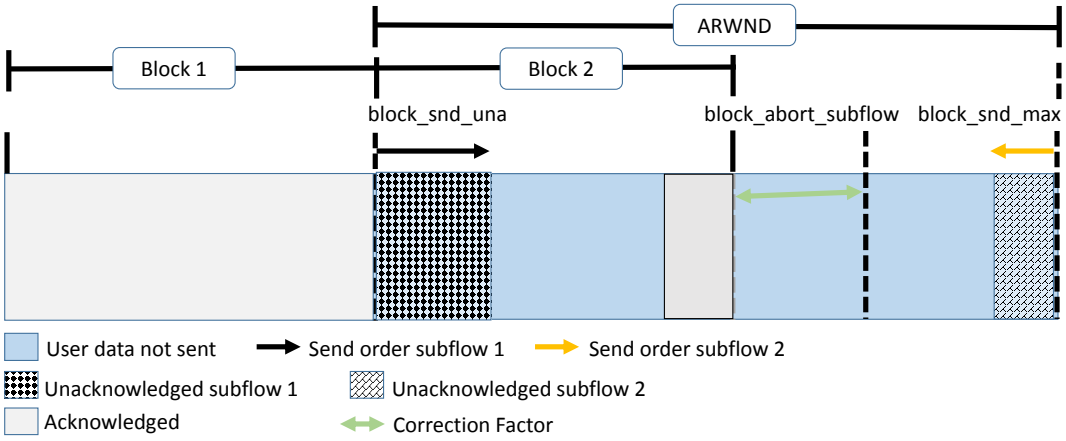


Figure 7.24: Time D - After major block building

#2 works on the send window of subflow #1 in descending order. The major block building process has to be triggered after the Block 1 is completely acknowledged or subflow #2 sent the complete Block 1. This is the case if subflow #1 is not the strongest path or a major problem occurred on path #1.

- **Block 1 has been sent but not acknowledged completely**
  Figure 7.25 describes the situation after the Block 2 was filled by subflow #2. In this situation the subflow #2 started to repeat the unacknowledged segments of Block 1 in descending order where the start point was defined by $block\_abort\_subflow$. This overlap is not necessary in every case but ensures continued sending even in case of a subflow #1 connection abort. An alternative approach would be to deploy a flow level retransmission mechanism. If it is not the goal to retransmit the Block 1 with subflow

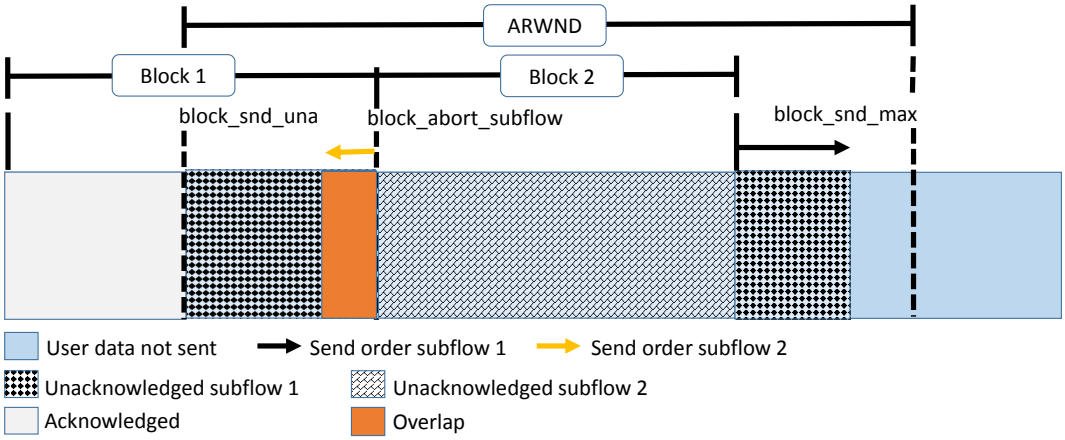


Figure 7.25: Time E - After completely sent Block #2

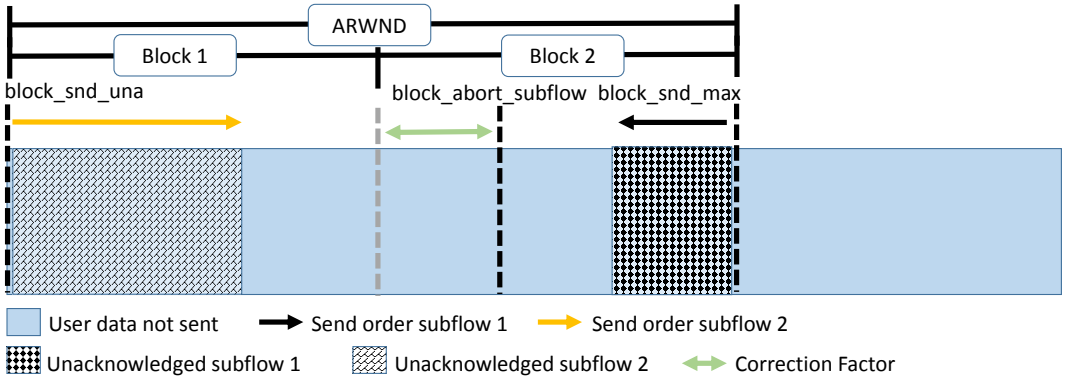


Figure 7.26: Time F - After an optimized major block building

#2 a major block building process similar to Figure 7.24 can be performed.

A new major block building process should always be performed by considering the throughput difference between subflow #1 and subflow #2 in a adjustment value or by considering indications for connections errors. Figure 7.26 illustrates the situation, where the subflow #2 achieved in sum more acknowledged segments as subflow #1. Therefore the send order is reversed compared to the initial major block building process. Furthermore the abort criterion is shifted to the right, to provide as much time as possible for incoming acknowledgments via subflow #2. This is a self clocking mechanism that demonstrated adequate behavior for the disjoint and cross path setup in the simulation. However, in the first step it is also a good heuristic to keep the subflow #1 as strongest subflow, if the subflow #1 was able to reuse buffer released after incoming acknowledgments.

7.5.2.2 Simulation

The ConSN approach was implemented in the simulation environment OMNeT++/INET. The experiment setup with the WiFi and 3G access links of Subsection 7.1.2 was used for the experiments. In the disjoint path setup (see Figure 7.27) the implemented ConSN approach (ϵ=on) shows positive results for the LIA CCC. The Curve #4 of Figure 7.27 with activated ConSN shows good throughput without any other mechanisms to avoid buffer blocking. Furthermore, Curve #4 demonstrates better throughput than the opportunistic retransmission mechanism alone (Opp-Rtx Ω=on, Curve #2) over the whole range. However, even ConSN shows for the disjoint path setup a great performance, it performes not as good as both existing mechanism to avoid the effects of buffer blocking. In detail the disjoint path setup the Curve #3 shows even a small benefit compared to ConSN indicated by the difference to Curve #4 in the receiver buffer range between 300 to 400 Kbytes. In this receiver buffer range an overlap of the subflow blocks in the ConSN approach has occurred and had a small negative impact. The issue here was that the ARWND was not always a multiple of the MSS and there existed an overlap around the abort criterion during the block building phase. This overlapping segment has an effect if the ARWND is small. If the ARWND becomes larger (in this example around 400 Kbyte) the throughput of ConSN is comparable to the throughput of the currently proposed mechanisms for the disjoint path setup. The observation is also true for OLIA and RP-MPv2. However, this overlapping effect during the send block building process of ConSN is negligible compared to the benefits in the cross path setup of the experiment, here the existing mechanism show much weaker behavior.

The cross path setup is a realistic use case in the Internet as discussed in (see Chapter 5) and the ConSN provides as the only approach a real benefit in this setup. Figures 7.28, 7.29 and 7.30 for the LIA, OLIA and RP-MPv2 CCC show impressively the added value of ConSN for smaller buffer sizes ($\leq$ 2000 Kbytes). In contrast to enabled opportunistic retransmission (Opp-Rtx Ω=on) and penalizing (σ=on) in Curve #3, the ConSN approach always achieves at least the throughput that could be expected from the strongest subflow alone.

Most notable is that the ConSN in the Curve #4 of all figures is also able to provide a

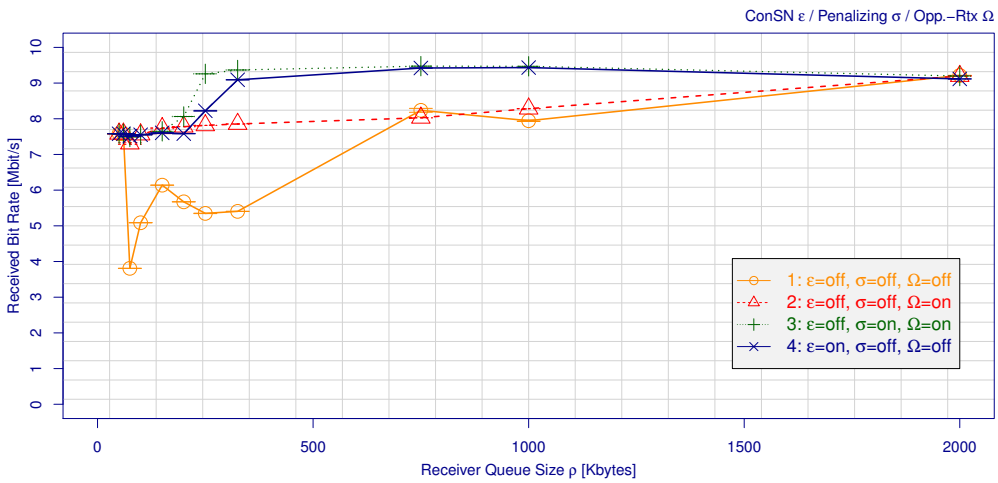


Figure 7.27: ConSN vs penalizing and Opp-Rtx with LIA in a disjoint path setup

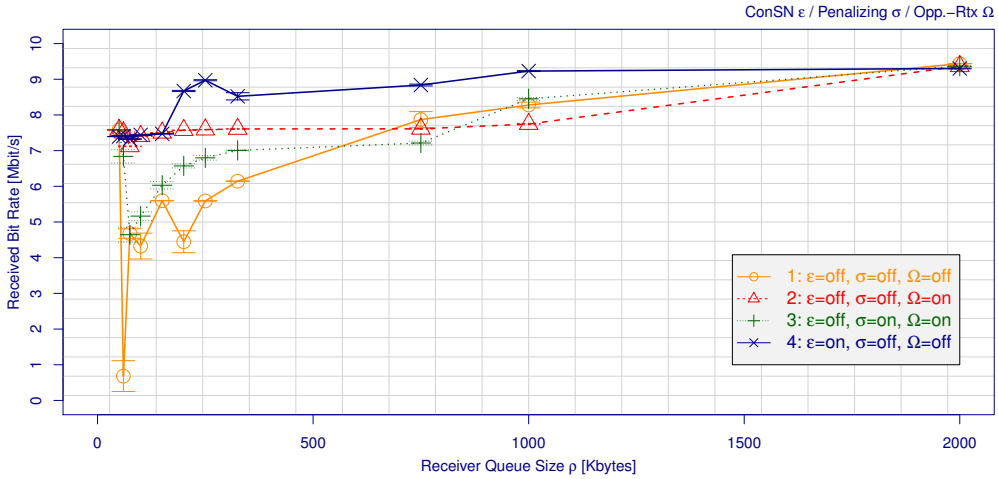


Figure 7.28: ConSN vs penalizing and Opp-Rtx with LIA in a cross path setup

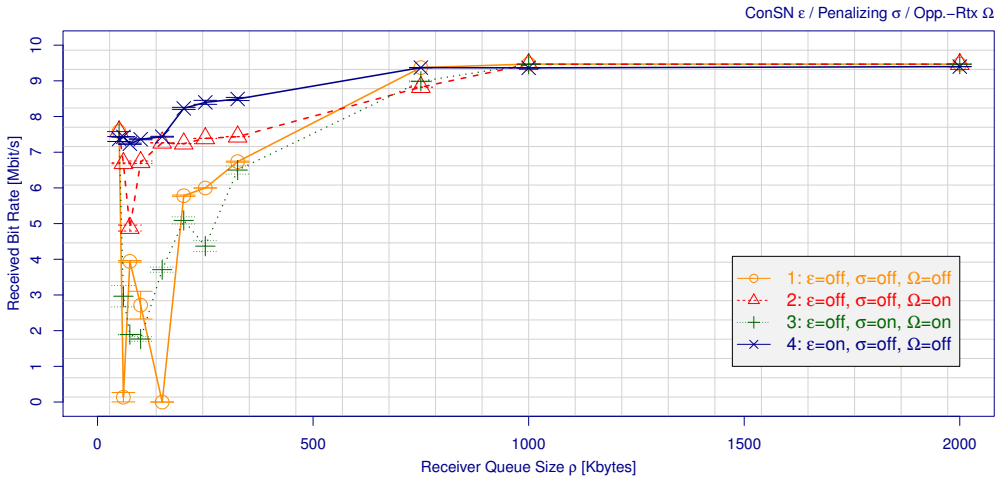


Figure 7.29: ConSN vs penalizing and Opp-Rtx with OLIA in a cross path setup

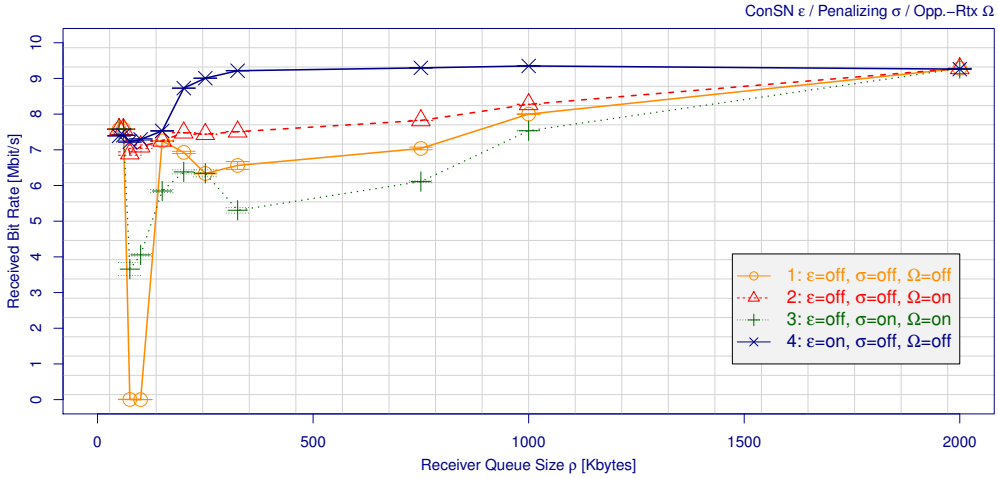


Figure 7.30: ConSN vs penalizing and Opp-Rtx with RP-MPv2 in a cross path setup

real added value for relatively small receiver buffer sizes (50 Kbytes to 1000 Kbytes), even if the benefits are different for the chosen congestion controls. To understand the differences among the Curve #4 in the Figures 7.28, 7.29 and 7.30 for the receiver buffer range from 50 Kbytes to 1000 Kbytes, it must be understood that there exists a dependency between the time needed to transfer segments via the weak subflow and the time needed to allocate all available receiver buffer to the strong subflow. The faster the receiver buffer is allocated by a specific subflow (ARWND - $outstanding\ bytes = 0$), the faster the retransmission phase starts in the ConSN approach and decreases the effective usage of the ARWND. In the worst case, this retransmission phase leads to a retransmission of duplicate data that occupies send and receive buffer but does not increase the throughput. Thus, the more the congestion control prefers a strong subflow the faster the retransmission phase will start, and the higher is the potential risk that data will be transferred twice without need.

OLIA always prefers the best subflow as known from the fairness discussion in Chapter 6, while LIA prefers the subflow with the best congestion window and RP-MPv2 – based on the initial load sharing idea – searches an equilibrium among all subflows. Thus, in theory the RP-MPv2 CCC should show the best results compared to LIA and OLIA.

The throughput for LIA in Figure 7.28 with less than 1000 Kbytes receiver buffer can be divided in 3 regions. Region 1 describes the receiver buffer size between 50 Kbytes and 100 Kbytes, where only the subflow of the path assigned to the strongest link determines the overall throughput. Beyond that the throughput increases, because the weak subflow had time to also provide new segments on flow level. In this second phase, the throughput up to the increased optimum is achieved as long as the receiver buffer size is not high enough to allow a huge asymmetric window size of the congestion control. Thus, a first share of the blocks has allowed all subflows to transfer usable segments on flow level and the retransmission phase of ConSN was not triggered. Anyway, one of the strong subflows starts to suffer, due to the competition to the alternative strong subflow on the same strong path. Again, in this experiment setup the routing of the cross path strategy resulted in an assignment of two subflows to each paths using a WiFi access link and a 3G access link. The resources are not shared equally anymore between the two strong subflows in the buffer range from 300 Kbytes to 1000 Kbytes, because one subflow was preferred. This leads to the situation where the subflow with the larger congestion control window finishes its block faster and allocates more receiver buffer and starts to retransmit segments on the alternative subflows earlier. This decreases the overall throughput. In case of larger receiver buffer size also the second strong subflow is able to occupy enough send and receive buffer and this also ensures a continuous sending for this subflow. Thus, even if one strong subflow will be preferred this has no negative impact on the other subflows as long enough receiver buffer space for all subflows can be provided and no overlapping has an impact.

Figure 7.29 shows only the phases 1 and 3 for OLIA, because OLIA prefers groups of subflows from the start. In this scenario that leads to one preferred subflow per access link. Thus, the second subflow assigned to the strong path does not get the receiver buffer space that is necessary to achieve the optimal throughput. Furthermore, the preferred subflow starts to retransmit the segments of the block assigned to the alternative subflow on the same strong path.

As expected, RP-MPv2 provides the most positive behavior in the experiment because RP-MPv2 distributes the data among all subflows in a manner that the retransmission phase is triggered as late as possible. Therefore, only phase 1 and 2, as described for the LIA CCC, can be observed in this setup. Here the Curve #4 in Figure 7.30 shows a nearly perfect

throughput comparable to the results of the disjoint path setup. RP-MPv2 provides the same resource for every subflow[16] and no subflow is preferred, therefore both strong subflows are not limited and perform equal and close to the optimum.

Altogether, the preference for a specific subflow decreases the time until the retransmission phase for LIA and OLIA starts and shows a suboptimal behavior for smaller receiver buffer sizes (here smaller than 1000 Kbyte). But ConSN provides in general more throughput than penalizing and opportunistic retransmission (alone or in combination under every condition) in the cross path setup. And as only mechanism ConSN achieves the minimal goals even for small buffer ranges in the cross path setup.

7.6 Other side effects with RED queues

As discussed, a user of an IETF load sharing extension cannot expect the best throughput in every case for small receiver buffer sizes. Therefore, a strategy of the user can be to set the buffers as large as possible, because the experiments discussed so far indicate that large buffers lead to high throughput. However, even if this behavior can be observed for the experiments used so far, it is not generally valid. The following experiment demonstrates that larger receiver buffer sizes can decrease the throughput and that small receiver buffer sizes can increase it.

The Figure 7.31 demonstrates the throughput for the same disjoint path setup as used in the ConSN experiments on the topology with the WiFi and 3G link as defined in Subsection 7.1.2. The only difference to Figure 7.27 is that RED queues (see Subsection 2.3.2) are applied in the routers and not FIFO queues. Applying opportunistic retransmission (Opp-Rtx Ω=on) and penalizing (σ=on) in Curve #3 or applying ConSN in Curve #4 of Figure 7.27 show bot the same suboptimal throughput for large receiver buffer sizes.

The throughput curves can be divided again – like for Figure 7.28 – in three regions. In the first region (from 50 Kbytes to 100 Kbytes) Curves #3 (opportunistic retransmission and penalizing) and #4 (ConSN) are able to achieve a throughput that can be expected from a singlepath connection established via the strongest path (goodput close to 7.7 Mbit/s). In the second region the throughput is increased to a maximum (goodput close to 9.7 Mbit/s at around 500 Kbytes receiver buffer size) that can be expected from a load sharing solution using the capacity of all links[17]. However, the overall throughput of the MPTCP flow is in every case below the optimal throughput curve if the receiver provides more than 500 Kbytes receiver buffer. Even if all mechanisms to decrease the effects of buffer blocking are enabled as for Curve #2, #3 and #4, the optimal result cannot be achieved.

The reason for this unexpected behavior is an effect caused by the load sharing mechanism of the scheduler, which is able to balance the congestion up to certain levels. First some facts have to be mentioned to explain this behavior, e.g. that packet loss has the potential to decrease the throughput. A congestion control like OLIA, LIA or RP-MPv2 reacts on loss by decreasing the send rate. Queues in routers can cause this loss and can, therefore, cause a limited send rate. If the congestion window allows to fill a link with a high send rate, the sender will fill the queues in the routers. Therefore, a different queue behavior has a direct impact on the CCCs, the CCCs have a direct impact on the send rate and the send rate has a direct impact on the fill level of the router queues. Red queues increase the probability of

[16] Here two every subflow of paths assigned to the strongest link.
[17] Using 2 Mbit/s and 8 Mbit/s access links.

loss depending on their fill level. In the experiment setup of Figure 7.31, the probability for a packet drop starts at 30% of the queue size which is 100 packets in this experiment. With a fill level of 30 packets in the queue there possibly exists a risk that a packet drop will occur. If the fill level is lower than 30 packets no loss caused by the router queues will be observed. Furthermore, it is not ensured that the packet drop always affects segments sent for the first time, also the duplicate packets resulting from a fast retransmission will be dropped from the queues in the routers. A dropped fast retransmission causes a timer-based retransmission in the worst case. This blocks the transmission until the timer for the retransmission will expire and leads to a decrease of the congestion control window to a minimum value (here one 1 MSS). Thus, if there exists enough free receiver buffer space, the subflow will fill the link and, therefore, the queues of the connecting routers. If a subflow starts to fill the queues at the router, the probability of a packet drop will be increased. Furthermore, the send rate can also be limited by the ARWND and can prevent a filling of the router queues up to a specific level. The ARWND is depending on free receiver buffer size.

Figure 7.32 and Figure 7.33 are provided to demonstrate the effect of filled links. Figure 7.32 shows the congestion control window of the subflow on the path assigned to the strong link. In this case the subflow is not limited by the receiver buffer size and is able to increase the congestion control window. With the growth of the congestion control window the queue of the first router of the strong path[18] will also be filled up to a maximum of 100 packets. Thus, the probability of drop will increase to 100% for incoming packets. About 220 seconds after the connection establishment the first timer-based retransmission occurs, the congestion control window decreases to one MSS and the congestion control restarts with a slow start phase. During this time the fill level of the router queue on the strong path decreases to zero.

Here the effect can be observed that the more the queue is filled, the more likely the RED queue mechanism will drop packets. This effect does not occur throughout the complete transmission, but often enough to decrease the throughput.

In comparison to this negative router queue overflow scenario, Figure 7.34 and Figure 7.35 illustrate the behavior of the router queue and the corresponding congestion window for the case when a smaller receiver buffer limits the send rate. The congestion window of the subflow on the path assigned to the strongest link is not able to increase to more bytes with a receiver buffer of 500 Kbytes than needed to fill the buffer on the path to a reasonable level (here to around 30 packets). The risk is low that the sender drops segments and retransmissions occur in this scenario. If no packets are dropped in the routers, no timer-based retransmissions will occur, which in sum increases the overall throughput.

This example illustrates that more receiver buffer does not necessarily lead to an increased throughput or the maximum possible throughput in every case for an end-to-end load sharing approach. If the user of the system has knowledge about details of the overall system, like queue management, bandwidth or delay, he is able to optimize the systems for the transport protocol much more by adapting individual parameters than the end-to-end mechanisms are able by adaptation, measurement or estimation. A scheduler on transport protocol level is not able to detect all these details of, e.g. router queue fill level, fast and exactly enough to use the resources perfectly in every case.

[18] with the WiFi link.

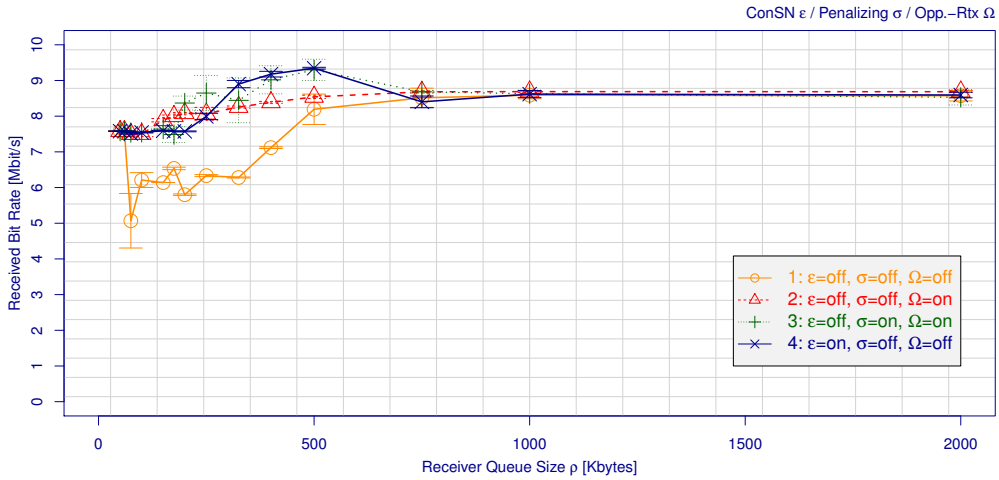


Figure 7.31: Throughput using RED Queues with queue size of 100

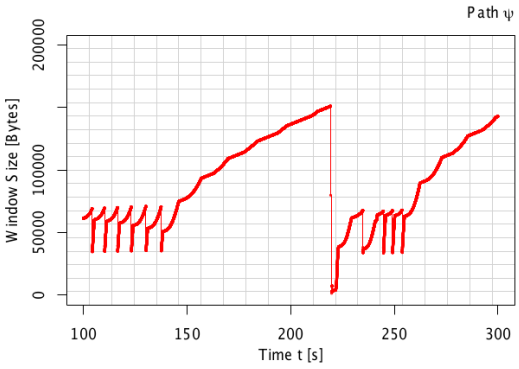


Figure 7.32: Congestion control window with large receiver buffer size

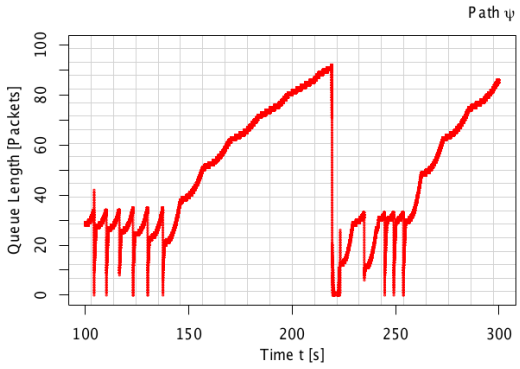


Figure 7.33: RED Queue with huge receiver buffer

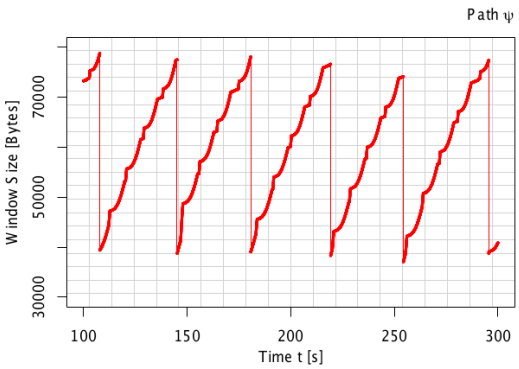


Figure 7.34: Congestion control with small receiver buffer size

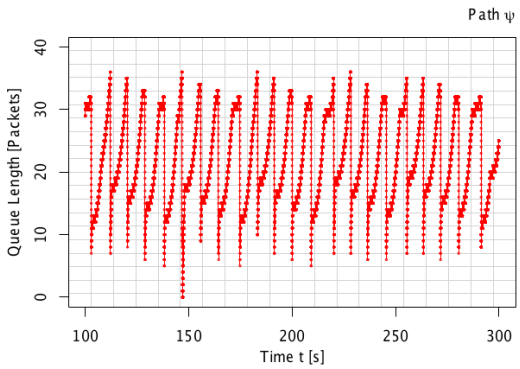


Figure 7.35: RED Queue with small receiver buffer size

7.7 Conclusion

The development of a high performing multipath scheduler is not an easy task. The discussion in this chapter identifies different influencing factors such as the flow control, retransmission mechanisms or the queueing discipline deployed at the router queues in the network. The discussion of this chapter furthermore clarifies that buffer blocking has a huge impact on a multipath scheduling system.

One major contribution of this chapter is the identification of further potential buffer blocking problems, in particular for small buffer scenarios for both IETF load sharing approaches. The analysis of the scheduler showed that the root cause is the starting phase of a multipath connection and the sharing of limited resources like the receiver buffer, which was not considered so far.

Another important contribution of this chapter is the analysis of the scheduling process which identified the limited degrees of freedom for decisions in case of saturated sending. Furthermore, issues with existing protocol mechanisms were identified and demonstrated.

A goal of this chapter was to increase throughput in a common (dissimilar) network topology typical for the Internet. Optimizations were proposed for CMT-SCTP and, furthermore, for MPTCP in particular for MPTCP a completely new mechanism to decrease the effects of buffer blocking has been proposed.

Chapter 8

Consequences for the future

The results of the Chapters 5, 6 and 7 are useful for the current IETF standardization process, but also for a discussion about a completely redesigned Internet ("Future Internet"). Thus, the results of the analysis lead to different conclusions depending on the time frame, i.e., whether a short, mid or long term time frame is considered. Short and mid term solution approaches focus on the current network architecture using the IP-based transport protocols. A new congestion control can be introduced very quickly, e.g., in less than a year, as long as the community supports the proposal. The replacement of protocols and hardware in the network cannot be achieved that quickly[1]. However, short and mid term efforts are interesting for the standardization process in the IETF.

The Future Internet community focuses on new clean slate approaches to replace the current Internet technology with new approaches and technologies, which needs much more time – if it is even possible. Thus, researchers who focus on this topic are taking new and increasing demands in terms of security, QoS, mobility support, privacy, sustainability and scalability [Fel07] into account.

Anyway, for every research community – current IETF standardization or Future Internet – the results of this thesis are relevant, but with different consequences. This chapter gives some suggestions for the different communities.

8.1 Results relevant for the standardization process

Mid and short term approaches have to be translated to standards if a wide deployment is the goal. Short term goals address updates or extensions of current drafts, where mid term goals lead to completely new standardization efforts.

8.1.1 Short term

In particular with respect to the CMT-SCTP draft [ABD+13], the results of this thesis have a potentially high short term impact, because the draft is not completed so far. In the context of MPTCP, only updates and extensions to the existing RFC6824 [FRHB13] are possible.

[1] As example, the introduction of IPv6 required more than 15 years up to now and the goal is still not achieved.

8.1.1.1 Path management

First, the MPTCP routing approach should address the open issues of the initial path choice (see Subsection 5.1.2.1). The multipath SYN retransmission can be introduced without any changes to the core MPTCP extension, as a small update for the RFC6824 [FRHB13]. Thus, it should be included as optional feature, because it is a valid approach to avoid a missing SYN with $MP\_CAPABLE$ option caused by middleboxes. Furthermore, SYN retransmission can be implemented on sender side only. The maximum of number testable destination addresses should be configurable[2]. That allows a scalable configuration by the user.

The path definition issue of CMT-SCTP is more complex. The path definition of MPTCP cannot be adapted to CMT-SCTP, this would be more than a straightforward extension, it would require a change of the basic protocol definition of SCTP. Even if it seems to be the straightforward way to improve the CMT feature of SCTP, this change would cause lot of problems for backward compatibility. Introducing a new path definition would change the structure of the TCB of SCTP. This has impact on every calculation, condition, order of events and optimization done so far for the transmission of segments. Therefore, also for the developer this causes issues, because it is close to a complete redesign and recoding of an existing SCTP model, resulting in a loss of the protocol layer independency and stability. It can be expected that these issues would prevent the deployment of the load sharing extension of SCTP in the worst case instead of supporting it. It is also not possible to introduce the new path definition as an additional feature, because backward compatibility has to be provided.

It can be expected that most implementations will use CMT-SCTP as straightforward enhancement to provide the benefit of load sharing in a setup similar to the SIGTRAN use case (see Section 2.2). Thus, the improvement of the throughput should be supported within the architectural limitations, but the standardization process should be kept focused on the other benefits of SCTP (e.g. high reliability). This is also a benefit for the users who, with this focus on specific tasks, have the choice between two specialized transport protocols which provide optimized behavior for specific multipath use cases and goals (see Subsection 3.1.1). The reliability goal for the redundant network setup, which is still the undisputed strength of SCTP, should be in the main focus of CMT-SCTP. On the other hand, load sharing to increase the throughput in the Internet should be a main task of MPTCP. This should also be mentioned in the standardization documents to give an orientation to the users.

8.1.1.2 Fairness

The fairness discussion is still an open issue. As so far only an incomplete fairness definition is provided that does not consider the fairness to another multipath TCP flow, there is a valid argument to restart the fairness discussion. The current definition just provides a framework to support the Resource Pooling idea, not more. Furthermore, the coupled congestion controls, which are based on this Resource Pooling idea, do not work very accurately in many cases and prefer a singlepath flow without any specific reason. The Resource Pooling idea gives an inadequate response to the bottleneck issue and should be reconsidered. In particular, the shift from the load balancing goal to the balance congestion goal is not straightforward in its argumentation and leads to an uncontrollable behavior. Scenarios can easily be created that demonstrate drawbacks of the Resource Pooling idea which question the benefit in general. The current approaches provide several drawbacks, e.g.:

[2]For Unix and Linux with a *sysctl* [SFR03].

- Hard to control.
- Hard to monitor.
- No motivation to follow this standard.

A return to the original link centric fairness is strongly advised and if a flow level fairness cannot be achieved only a sublevel fairness is a real practical option. This thesis proposes the use of the already existing uncoupled congestion controls. There are good reasons to revert to a link centric subflow fairness which can be summarized as:

- Link centric subflow fairness can be monitored and controlled by a network provider.
- Link centric subflow fairness allows to consider number and costs of access links in the fairness calculation.
- The existing mechanisms and fairness metrics can be re-used on subflow level (as is not possible on flow level).

A link centric subflow fairness will not create new risks for the service of the Internet, because it can already today be emulated easily on the application layer without any problems (see Subsection 6.3.2.1). Moreover, the deployment of the coupled congestion controls provides a higher risk for the Internet due to the unpredictable behavior of these mechanisms. Thus, as long as the congestion control mechanisms are not able to detect the topology – with functional and practical bottleneck detection – a flow based comparison of a singlepath flow and a multipath flow is not productive. Furthermore, the IETF community should consider the impact of the resources, efforts and costs on the fairness rules in their discussion.

8.1.1.3 Scheduling

The discussion of the scheduling issue makes clear that the load sharing extensions for TCP and SCTP are functional and well designed, but that the current specifications are not able to achieve the self-set goals of "do not harm", "balance congestion" and "improve throughput" without further efforts and in every use case. Additional efforts for the scheduler process are needed. The IETF load sharing standard documents of MPTCP do not address the buffer blocking issue and are therefore in the initial version unusable in the Internet. An own group of standardization documents should address this topic – like done for the fairness issue (see Section 1.3 of RFC6182 [FRH+11] and Section 6.1 of the CMT-SCTP draft [ABD+13]). Thus, for the IETF standardization process, it should be a main goal to bring the scheduling requirements into the next version of the architecture drafts and also to address the need to support more than two subflows concurrently.

As a result of this discussion, the ConSN approach (see Subsection 7.5.2) can be provided as alternative algorithm to address the buffer blocking issue. This work can lead to an additional draft for MPTCP in the same manner as it was done for congestion control mechanisms. Chunk rescheduling and buffer splitting were already added to the CMT-SCTP Internet draft [ABD+13]. However, also the negative impact of the known mechanisms to avoid the impact of the buffer blocking issue should be addressed. It should be noted as an advice to deactivate e.g. buffer splitting during the first startup phase. Furthermore it should be added to the MPTCP and CMT-SCTP implementation notes that ConSN for MPTCP and the RTT-weighted scheduler for the disjoint path setup are identified as best option so far (see Section 7.4).

8.1.2 Mid term

The discussion of Chapter 6 demonstrates that the fair sharing of resources has not been solved adequately. The load sharing feature on the transport layer cannot be discussed isolatedly from the network layer as shown in Subsection 5.1.2. Thus, if new approaches to solve the issues are discussed, also the network layer should be considered. This is true for path management, scheduling and fairness issues.

Anyway, changes in infrastructure protocols, like routing protocols, have a high impact on the hardware used. However, new technologies come up with, e.g. Open Flow [MAB+08] that make the network more dynamic and intelligent and allow shifting functionalities to the network, like for example bottleneck avoidance.

In the context of this thesis, a cooperation with the University of Technology Ilmenau was done [VBOMT13b] that investigated the possibilities on the transport layer. The main idea of this cooperation was the usage of the new "hierarchical routing management system" of Ilmenau [VMT12] to avoid bottleneck issues. An additional goal was the avoidance of high path dissimilarities with respect to delay or error rate by providing a QoS infrastructure. A key concept was the combination of existing infrastructure and protocols with new routing mechanisms in the network. The results are described in [VBOMT13b]. A new cross layer communication was introduced to allow a transport protocol the exchange of control data among the network components to support a QoS-oriented routing. This cross layer communication between transport protocol mechanisms and routing requires meta data. SCTP was extended in a first version for the proof of concept, even if approaches, e.g., with additional IPv4 options or IPv6 hop-by-hop extension headers are also possible. As discussed in Subsection 2.2.1, an SCTP segment includes a "common header", and is first filled with Control chunks and afterwards with DATA chunks. Control chunks are introduced to exchange control information between connection endpoints. Part of the new approach was an additional chunk type that was denoted as "network" control chunk. This network control chunk was designed as extension of the currently used control chunks to signal to all entities along a route. So, in case an intermediate router supports network control chunks, the router reacts on the requirements of the sending application instance. This signaling provides a more sophisticated routing than implemented by today's networks. In order to allow a fast parsing of such extended packets in hardware a hierarchy of messages in CMT-SCTP was introduced that replaces the current definition [Ste07] and places network control chunks after the SCTP common header and before any following chunk. The results of using the hierarchical routing was very straightforward and supported a flow setup without any bottleneck issues.

Thus, for mid term standardization plans for new routing protocols, the support of load sharing approaches should be considered. The signaling to network components has also drawbacks with respect to security if these control chunks are encrypted in an end-to-end connection. However, even here first approaches are in the discussion that allow the operation on encrypted control data [SBJR13].

8.2 Long term (Future Internet)

This thesis demonstrates that load sharing protocols are designed to support the reliable and ordered transfer of data. This will also be necessary for future networks. Networks provide different multipath connectivity characteristics with respect to error rate, delay or bandwidth. A scheduler has to adapt to the topology also in a Future Internet to address the

re-ordering issue, because the physical link characteristics cannot be changed, even though aggregation of equivalent paths, or the reservation of specific resources for specific flows can help. Just scheduling data via different paths without considering the reordering issue or the buffer blocking issues will decrease the performance, as it can be observed even for unordered transfer in [DBRT10]. A first architectural approach was proposed with an Encapsulated Responsibility Centric Architecture model (ERiCA) [BDAR12a] to demonstrate the requirements in the Future Internet. This model provides a process to normalize, cluster and aggregate network (NCA) information to support an optimized load sharing service with a defined information exchange among all involved network components. Thus, link characteristics have to be provided to the scheduler instance in a normalized form. A correct and up to date database allows a scheduler instance to use and cluster links that provide usable characteristics which allows an optimal aggregation of network resources. Just measurement or estimation of network information on end-to-end level is not exact and complete enough as, e.g., demonstrated by the RED queue example in Subsection 7.6. However, the complete information exchange chain needs an ability to exchange data across the complete Internet with a centralized aggregation point. Therefore, a completely new structure and architecture have to be introduced.

Chapter 9

Conclusion and outlook

In this chapter the main results of revisiting the TCP and SCTP load sharing extensions are recalled. Although the preparatory work for this thesis provides important improvements, too, this chapter focuses on the results that are provided exclusively by this thesis. Furthermore, an outlook on further research is given based on the results that will become important for the current and the Future Internet.

9.1 Achieved results

The analysis in Chapter 5 shows that the currently discussed IETF load sharing extensions have the same initial intention with the improvement of throughput. However, only the load sharing extension of MPTCP provides a design that is able to adapt to the conditions of the Internet on path and network level. The benefits of MPTCP are not caused by a better load sharing design, they are a result of different design decisions made for singlepath SCTP. These existing singlepath design decisions prevent an optimal increased throughput for CMT-SCTP in the Internet (see Subsection 5.1). MPTCP is able to use all possible address combinations as usable paths and, therefore, to establish subflows. This completely different resource base can be exploited by an MPTCP scheduler if the buffer blocking problems can be mitigated or avoided. This thesis demonstrates that the path definition of MPTCP provides a real benefit when using multiple paths, even if it comes with scaling limitations. Only the cross path setup strategy can increase the throughput in an Internet setup with asymmetric access links, which was proven in a real Internet setup.

The proof that the cross setup model matters in the real world is also a contribution of this thesis. The experiments in this thesis have shown that the current approaches are deployable in a large scale real Internet experiment and fully functional in challenging long distance setups (see Section 5.3). But only MPTCP is able to adapt completely to the conditions. The experiences gained provide an important base for upcoming large scale multipath testbeds, as they are created for, e.g., in the NorNet project [FODA14].

The real world testbed also changed the view on the relevance of the test and evaluation models used so far. It was identified as insufficient to evaluate the load sharing extensions on the transport layer just based on the disjoint path setup model (see Section 5.1.2). Also, the cross path setup model must be considered. The choice of the test model is in particular important with respect to a deployment on asymmetric, heterogeneous topologies. Without considering the cross path setup model, the weaknesses of the existing mechanisms to decrease

the effects of buffer blocking do not become visible. This was demonstrated for MPTCP in particular with respect to opportunistic retransmission and penalizing.

Another important contribution of this thesis addresses the challenges of the network layer. By defining different common use cases for the load sharing protocols (see Section 5.2) the weaknesses of both protocol extensions were identified. In the first step, this thesis discussed different scenarios with respect to the applicability of MPTCP and CMT-SCTP and in the second with respect to the support by the network layer. It was pointed out that MPTCP in combination with the existing routing support provides more supportable use cases than CMT-SCTP. But also the approach of MPTCP has its drawbacks (see Subsection 5.1.2.1) as was identified with the MPTCP routing issue. However, also solutions were provided to address these challenges, like for example the "multipath SYN retransmission" approach for the MPTCP routing issue.

Furthermore, this thesis analyzed different coupled congestion controls (CCC) in Chapter 6. All CCCs are based on the Resource Pooling idea. This thesis introduced for the first time a common platform for all CCCs applied to MPTCP and CMT-SCTP. The MPTCP model in OMNeT++/INET was designed from scratch to achieve this. This common platform is for simulation and real systems until now unique.

An additional contribution of Chapter 6 is the first analysis of all congestion controls – LIA, OLIA and RP-MPv2 for MPTCP and CMT-SCTP – under comparable conditions (see Section 6.2). The analysis demonstrates that the multipath fairness goals defined by the IETF cannot be achieved in every case. It is an important observation of this thesis that there exists also a grey area in the definition of comparable conditions (see Subsection 6.3.2.3).

One of the most important contributions of the Chapter 6 is the fact that the multipath fairness definition is not complete and exact enough to cover the requirements of a real world setup (see Subsection 6.3.3.2). The currently used definition does not achieve a fair behavior among multipath flows themselves. The "resource to share" has to be defined to provide an exact multipath fairness definition, and here the IETF has to decide whether a link centric fairness or a network centric fairness is the goal. This thesis argues for a link centric fairness, because it is, e.g., more predictable and can be monitored and controlled (see Subsection 6.1.3.1). Furthermore, Chapter 6 demonstrates that – caused by the Resource Pooling idea – a singlepath flow is preferred compared to the multipath flow even when it is not necessary (see Subsection 6.1.3 and 6.3.3.1). The argumentation for this is given by the balance congestion goal that provides benefits for the network utilization. However, this thesis demonstrates that in the same manner arguments against the Resource Pooling idea in general can be identified (see Subsection 6.1.3.1). Anyway, in the end, only the IETF community can decide about the future of the Resource Pooling idea and the corresponding network flow fairness, but it is a contribution of this thesis to provide new arguments for this discussion.

The scheduling task evaluated in Chapter 7 was not discussed adequately for the IETF multipath extensions so far. A main contribution of this chapter is the insight that the scheduler cannot be defined as freely as expected at the beginning of the standardization process. It is strongly advised to adapt the scheduler to different constraints, because otherwise the achievement of the minimum IETF goals is simply not possible. This thesis has clearly demonstrated that mechanisms to avoid or to decrease buffer blocking have to be part of the MPTCP and CMT-SCTP designs. Furthermore, this thesis has demonstrated that the approaches implemented in FreeBSD and Linux provide a good starting point, but that they are insufficient if small receiver buffer sizes were used (see Subsection 7.3.3). Furthermore,

this thesis has shown that the complete life cycle of a connection has to be taken into account for scheduler process. An optimized scheduler behavior right from the start is required to provide at least the chance for an optimum performance.

Based on these results, the scheduler process for CMT-SCTP and MPTCP was optimized in a second step (see Section 7.4). This thesis proposes the same weighted scheduler mechanism for CMT-SCTP as the reference implementation of Linux provides for MPTCP. But the analysis has furthermore shown that the mechanisms to avoid or decrease buffer blocking have to be deactivated until the window of the "fastest" path reaches its maximum during the first scheduling phase, even if the MPTCP scheduling approach is used.

An important optimization was done for MPTCP. A completely new buffer blocking avoidance mechanism was designed and implemented that redefines the degrees of freedom to schedule data via the subflows of an MPTCP flow (see Section 7.5). The new *confluent sequence numbers* (ConSN) scheduling approach decouples the flow sequence number from the subflow sequence number space (see Subsection 7.5.2). The MPTCP scheduling process uses with this approach new degrees of freedom in the send process. Only the result of this optimization work achieved acceptable results for MPTCP in the cross path setup (see Subsection 7.5.2.2). In addition the interaction of router queues, ARWND, CCC and multipath extensions was demonstrated. This analysis of the interaction has shown that an end-to-end multipath protocol will not be able to identify all relevant information of the network fastly and exactly enough to provide an optimal behavior in the Internet.

Besides the more technical discussion with focus on the protocol extensions themselves the Chapter 8 discussed the contributions of this thesis in the context of the IETF und the Future Internet community. Chapter 8 gives a possible roadmap for short and long term goals and a first initial idea of how they can be achieved. Here, in particular the combination with new routing mechanisms should be highlighted and the first proof of concept done to support a cross layer communication.

Last but not least many improvements were done for the TCP simulation model of the OMNeT++/INET environment and extensions created for different INET modules and the SimProcTC toolchain.

9.2 Future work

The results of this thesis provide a solid base for future research. The real world behavior of multipath transport protocols was only tested for relatively small scale network setups until now. New research networks like NorNet are coming up that provide the possibility to create multipath experiments via the Internet under more complex conditions. The behavior of the protocol extensions should also be analyzed under a more dynamic path management. It is still open whether the current extensions can address those requirements and can provide a new mobility solution. But there exists a strong need, because already first approaches have been identified in real products like in Apple's speech assistant Siri in iOS7[1].

The open fairness discussion leaves room for further research. The questions concerning a fair resource sharing raised up in this thesis are part of a new "Deutsche Forschungsgemeinschaft" (DFG) project in cooperation with the University of Bremen. Besides the goal to create a new fairness definition proposal for the IETF, the design of new congestion control mechanisms is addressed. A deeper analysis is necessary of the dependencies of existing

[1]MPTCP for Siri: http://perso.uclouvain.be/olivier.bonaventure/blog/html/2013/09/18/mptcp.html.

singlepath mechanisms and new multipath mechanisms to understand the direct and indirect dependencies. The common simulation model improved by this thesis provides a basis to compare the load sharing extensions of MPTCP and CMT-SCTP.

And perhaps most important, it is not clear what will happen with a network, if the default use case switches from singlepath to a more multipath dominated network. The scheduler instance has to be investigated in more detail, in particular with respect to the use cases with more than two access links. Even if one or two interfaces per device are currently the most deployed use case in the Internet, the others are relevant and important as well in particular for the future.

Appendix A

Appendix

A.1 Evaluation of the singlepath scenario

Absolute values of the singlepath scenario experiment in Subsection 6.3.1.2.

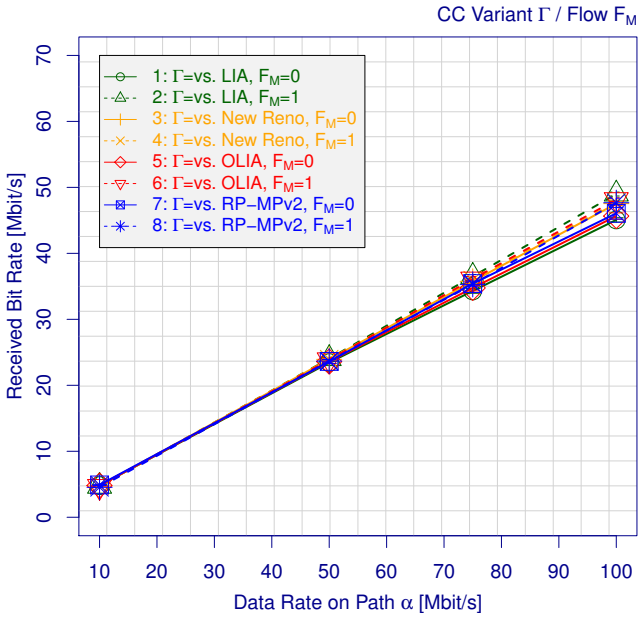


Figure A.1: Fairness line in a singlepath setup in comparison to SCTP

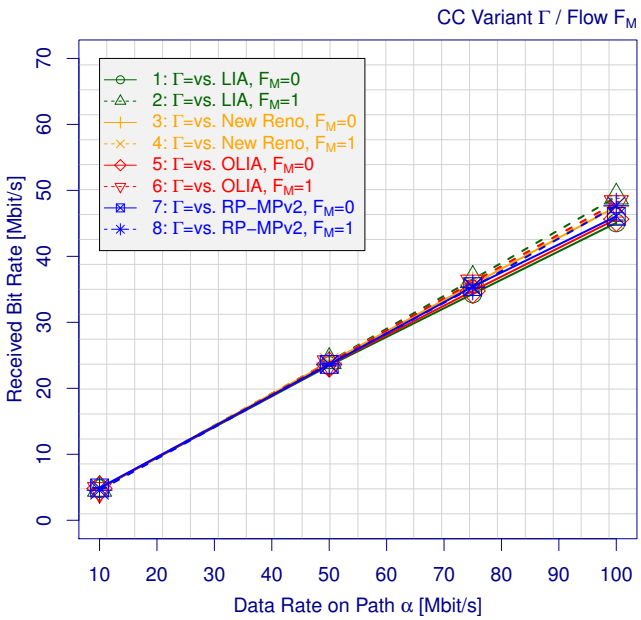


Figure A.2: Fairness line in a singlepath setup in comparison to TCP

A.2 Evaluation of the shared bottleneck scenario

A.2.1 Capacity share

Absolute values of the shared bottleneck experiment with focus on the fair share of the link capacity in Subsection 6.3.2.2.

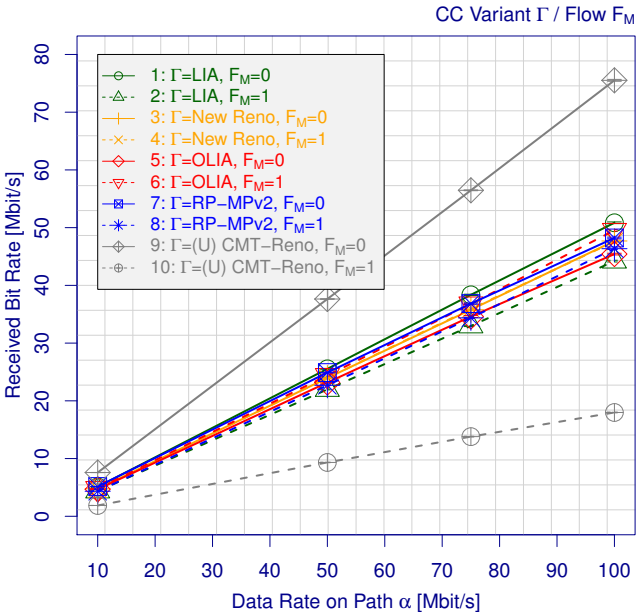


Figure A.3: Full Bottleneck: TCP vs. all MPTCP CCC variants

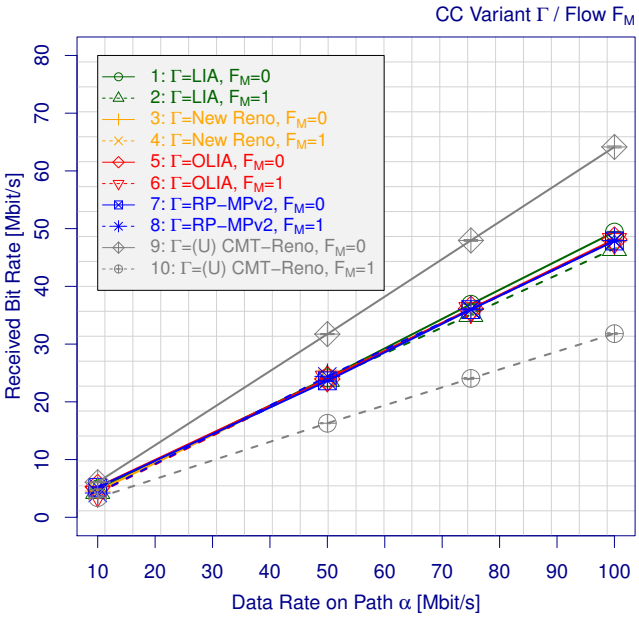


Figure A.4: Full Bottleneck: SCTP vs. all CMT-SCTP CCC variants

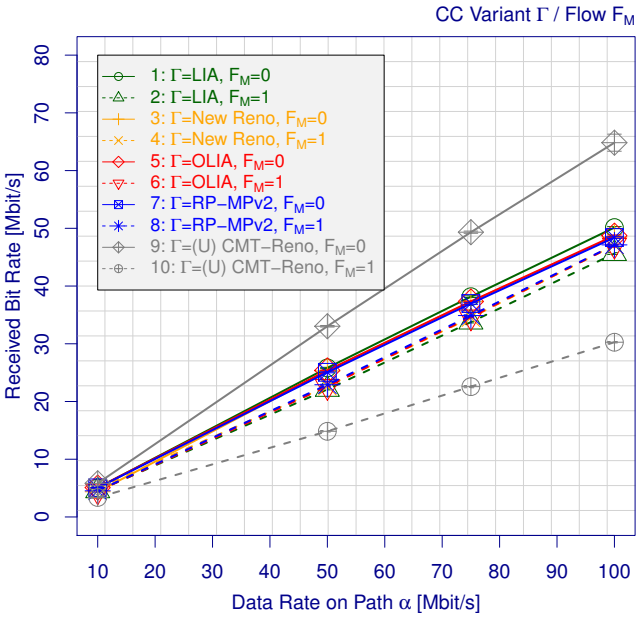


Figure A.5: Full Bottleneck: TCP vs. all CMT-SCTP CCC variants

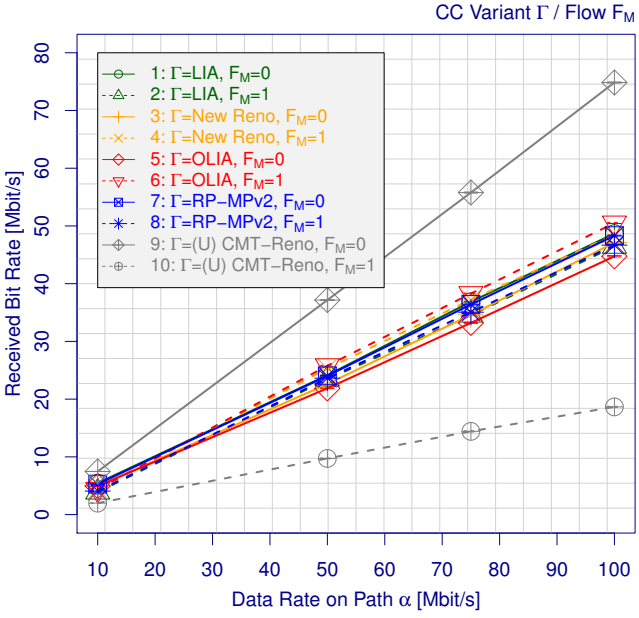


Figure A.6: Full Bottleneck: SCTP vs. all MPTCP variants

A.2.2 Delay

Alternative delay measurements of the shared bottleneck experiment 6.3.2.2.

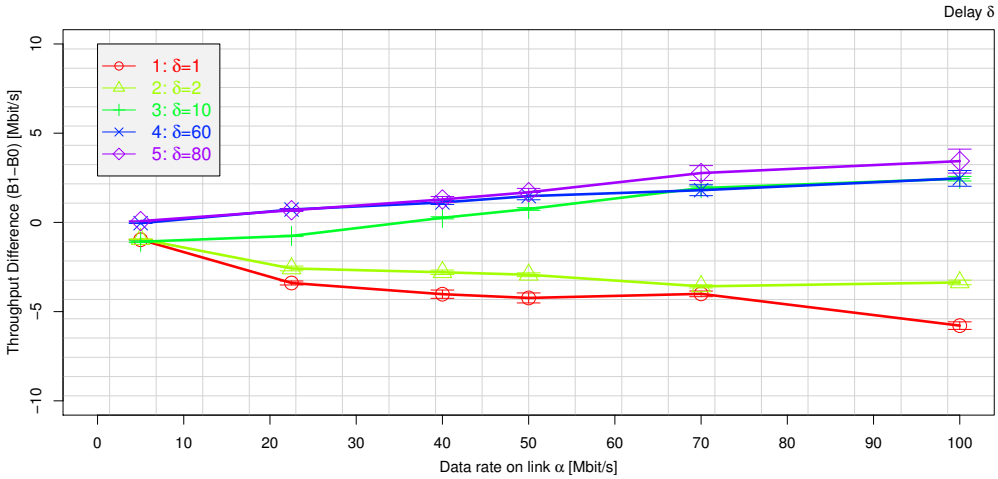


Figure A.7: Impact of delay for OLIA on CMT-SCTP

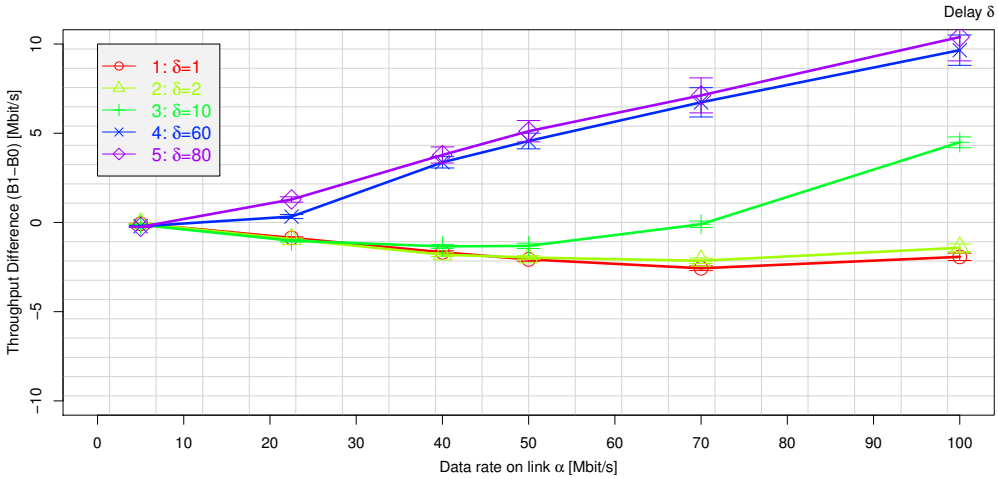


Figure A.8: Impact of delay for OLIA on MPTCP

A.2.3 Error rate

Alternative error rate measurements of the shared bottleneck experiment for the discussion in Subsection 6.3.2.2.

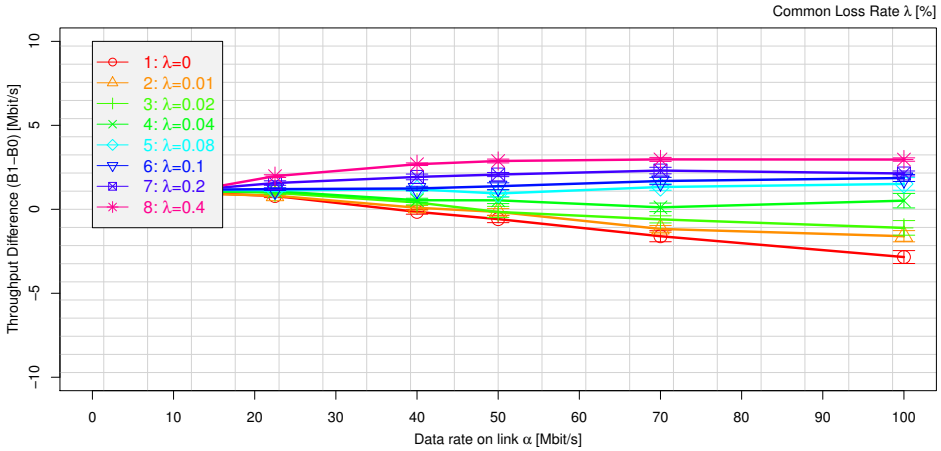


Figure A.9: Impact of increasing error rate on OLIA and CMT-SCTP

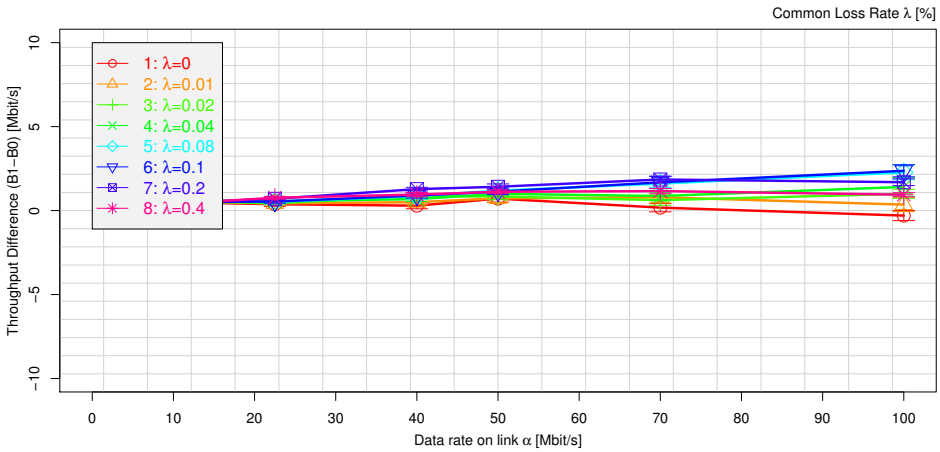


Figure A.10: Impact of increasing error rate on RP-MPv2 and CMT-SCTP

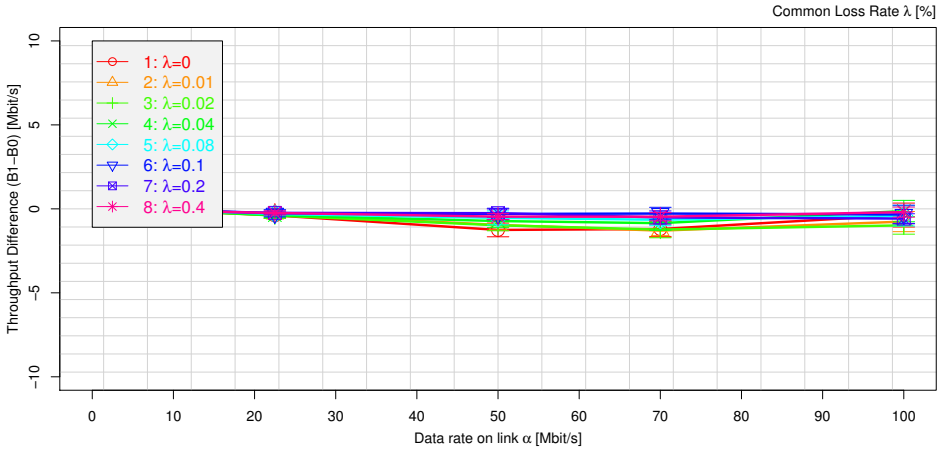


Figure A.11: Impact of increasing error rate on OLIA and MPTCP

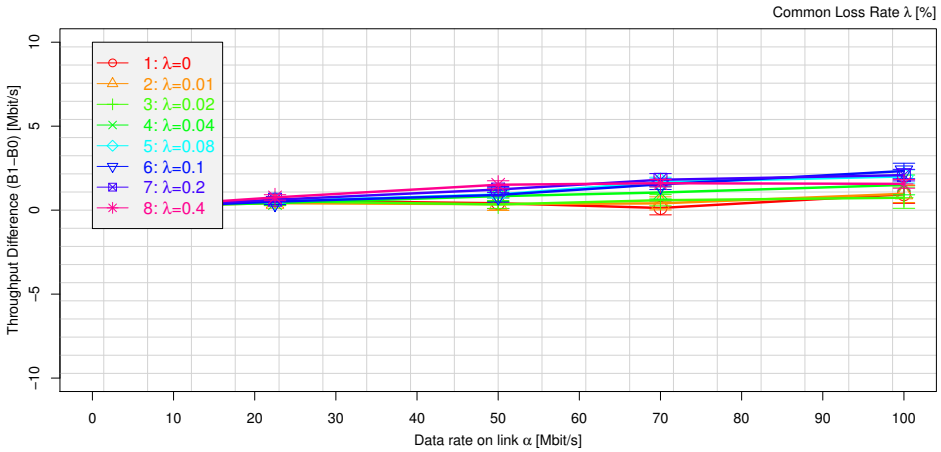


Figure A.12: Impact of increasing error rate on RP-MPv2 and MPTCP

A.3 Comparable conditions for a fair sharing

Absolute values of the shared bottleneck experiment under comparable conditions in Subsection 6.3.2.3.

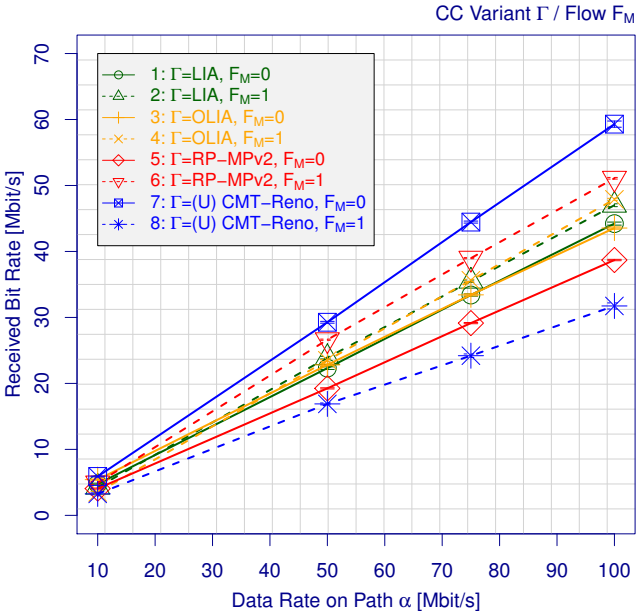


Figure A.13: Comparable conditions in CMT-SCTP setup

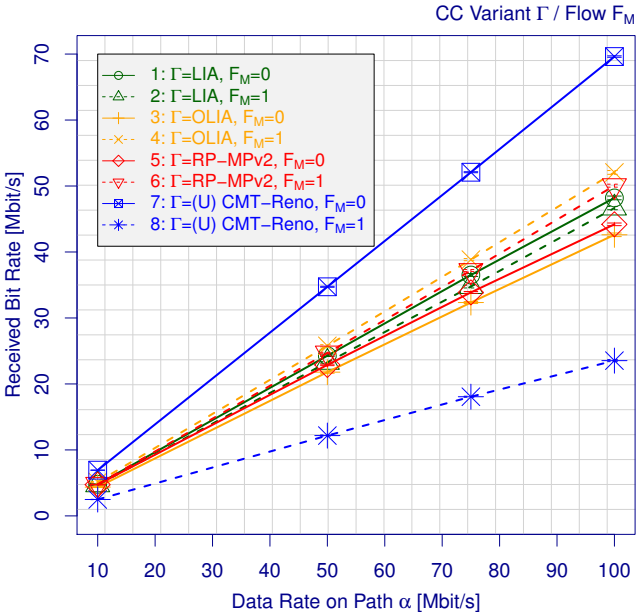


Figure A.14: Comparable conditions in MPTCP setup

A.4 Evaluation of half bottleneck with multipath flows

Alternative multipath fairness measurments for MPTCP and CMT-SCTP discussed in Subsection 6.3.3.3.

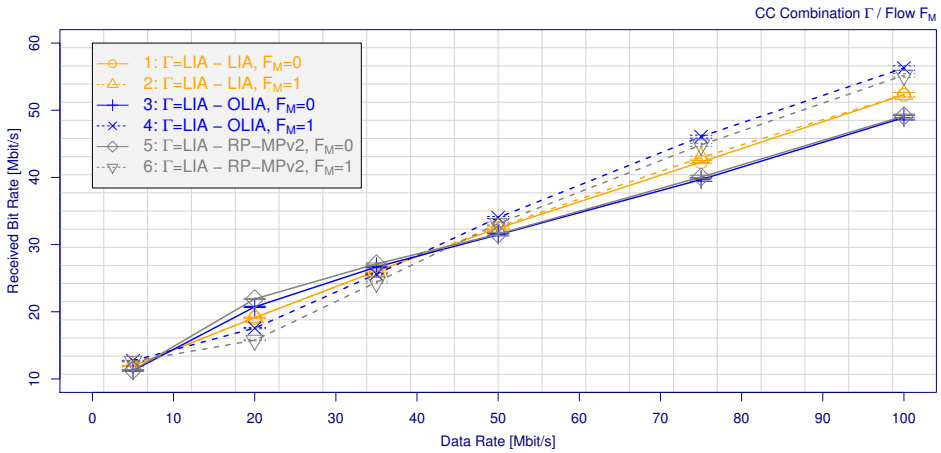


Figure A.15: LIA in comparison with all CCCs on SCTP

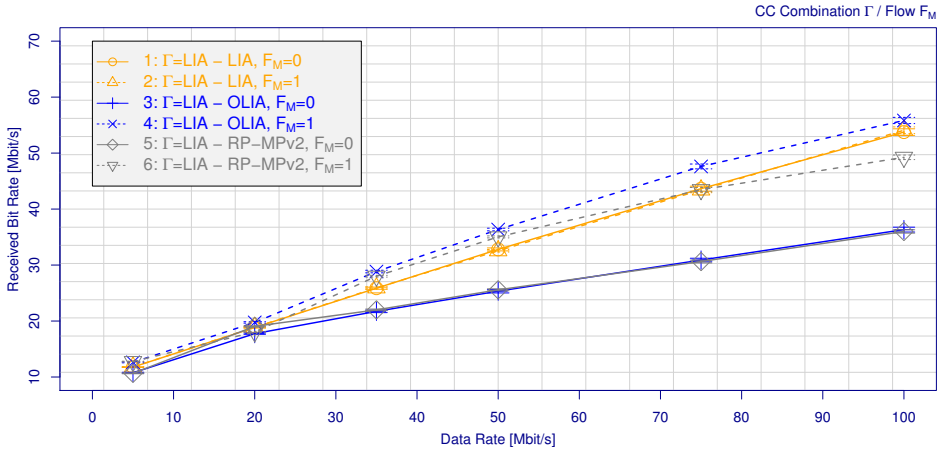


Figure A.16: LIA in comparison with all CCCs on TCP

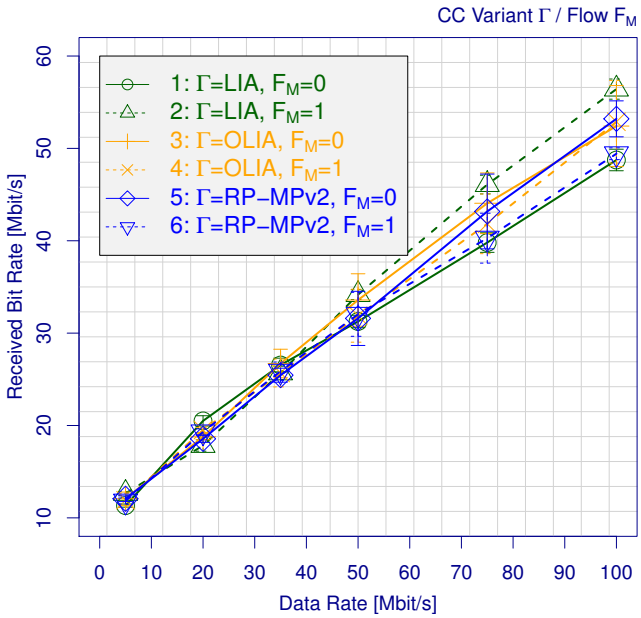


Figure A.17: OLIA in comparison with all CCCs on SCTP

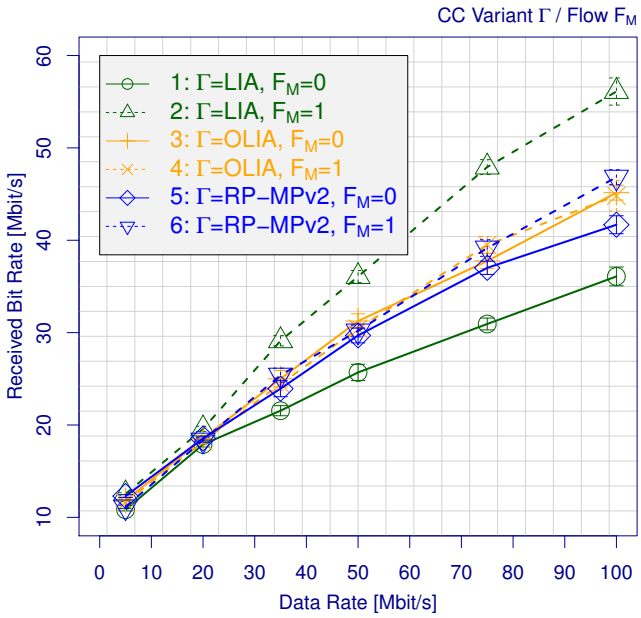


Figure A.18: OLIA in comparison with all CCCs on TCP

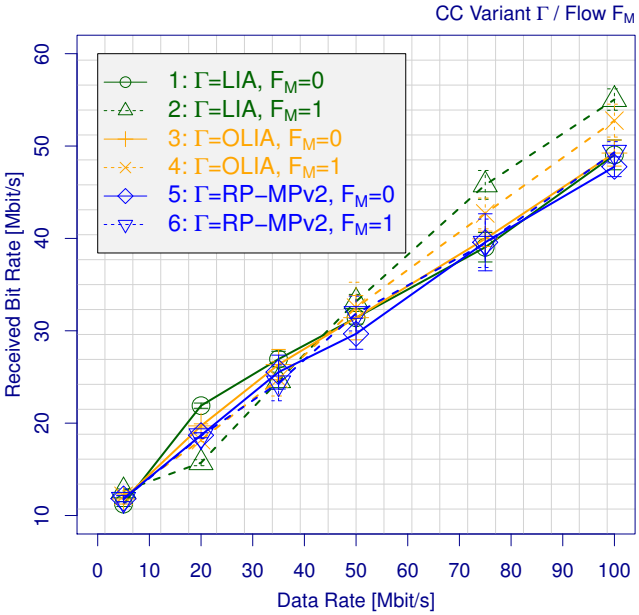


Figure A.19: RP-MPv2 in comparison with all CCCs on SCTP

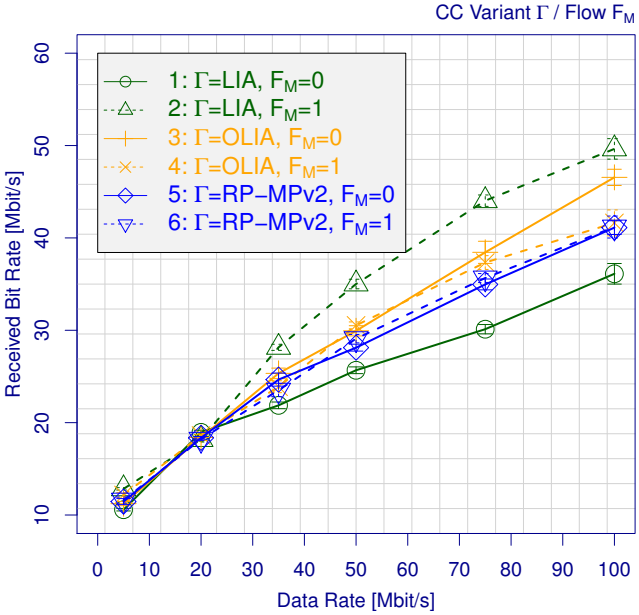


Figure A.20: RP-MPv2 in comparison with all CCCs on TCP

List of Figures

List of Tables

Bibliography

[ABD+13] Paul D. Amer, Martin Becke, Thomas Dreibholz, Nasif Ekiz, Janardhan R. Iyengar, Preethi Natarajan, Randall R. Stewart, and Michael Tüxen. Load Sharing for the Stream Control Transmission Protocol (SCTP). Internet Draft Version 08, IETF, Network Working Group, October 2013. draft-tuexen-tsvwg-sctp-multipath-07.txt, work in progress. URL: https://tools.ietf.org/id/draft-tuexen-tsvwg-sctp-multipath-08.txt. 1.1, 3.2.3.2, 4, 5, 5, 5.1.1.1, 5.1.1.3, 5.3.2, 7.1.1, 8.1.1, 8.1.1.3

[ADB+11] Hakim Adhari, Thomas Dreibholz, Martin Becke, Erwin Paul Rathgeb, and Michael Tüxen. Evaluation of Concurrent Multipath Transfer over Dissimilar Paths. In *Proceedings of the 1st International Workshop on Protocols and Applications with Multi-Homing Support (PAMS)*, pages 708–714, Singapore, March 2011. ISBN 978-0-7695-4338-3. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/SCTP/Paper/PAMS2011.pdf, doi:10.1109/WAINA.2011.92. 1.2, 4.2, 5, 5.3, 5.3.2, 7.1.3, 7.3.2, 7.5.2.1

[APB09] Mark Allman, Vern Paxson, and Ethan Blanton. TCP Congestion Control. Standards Track RFC 5681, IETF, September 2009. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc5681.txt. 2.1, 2.2.4, 2.2.4.1, 5.1.1.1, 6.2.1

[APS99] Mark Allman, Vern Paxson, and W. Richard Stevens. TCP Congestion Control. Standards Track RFC 2581, IETF, April 1999. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc2581.txt. 2.1, 5.1.1.1

[AWDR14] Hakim Adhari, Sebastian Werner, Thomas Dreibholz, and Erwin Paul Rathgeb. LEDBAT-MP – On the Application of Lower-than-Best-Effort for Concurrent Multipath Transfer. In *Proceedings of the 4th International Workshop on Protocols and Applications with Multi-Homing Support (PAMS)*, Victoria, British Columbia/Canada, May 2014. to be published. 5

[Bag11] Marcelo Bagnulo. Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6181 (Informational), March 2011. URL: http://www.ietf.org/rfc/rfc6181.txt. 3.3

[Bar11] Sébastien Barré. *Implementation and assessment of Modern Host-based Multipath Solutions*. PhD thesis, UniversitÈ catholique de Louvain, 2011. 4.1.2.2, 6, 4.1.2.2

[BAR+13] Martin Becke, Hakim Adhari, Erwin Rathgeb, Fu Fa, Xiong Yang, and Xing Zhou. Comparison of Multipath TCP and CMT-SCTP based on Intercontinental Measurements. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2013)*, Atlanta/USA, December 2013. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/Forschung/GLOBECOM2013.pdf. 1.1, 1.2, 5.3.1.2

[BCC+98] Bob Braden, David D. Clark, Jon Crowcroft, Bruce Davie, Stephen E. Deering, Deborah Estrin, Sally Floyd, Van Jacobson, Greg Minshall, Craig Partridge, Larry Peterson, K. K. Ramakrishnan, Scott Shenker, John Wroclawski, and Lixia Zhang. Recommendations on Queue Management and Congestion Avoidance in the Internet. Informational RFC 2309, IETF, April 1998. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc2309.txt. 2.3.2, 3.2.2, 6.1

[BDAR12a] Martin Becke, Thomas Dreibholz, Hakim Adhari, and Erwin P. Rathgeb. A future internet architecture supporting multipath communication networks. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 639–642, April 2012. doi:10.1109/NOMS.2012.6211975. 8.2

[BDAR12b] Martin Becke, Thomas Dreibholz, Hakim Adhari, and Erwin Paul Rathgeb. On the Fairness of Transport Protocols in a Multi-Path Environment. In *Proceedings of the IEEE International Conference on Communications (ICC)*, pages 2666–2672, Ottawa, Ontario/Canada, June 2012. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/SCTP/Paper/ICC2012.pdf, doi:10.1109/ICC.2012.6363695. 3.2.1, 5, 5.1.2.1, 5.3, 6, 6.1.3, 6.2, 6.3.2.1, 6.3.2.2

[BDB+13] Martin Becke, Thomas Dreibholz, Andreas Bayer, Markus Packeiser, and Erwin Paul Rathgeb. Alternative Transmission Strategies for Multipath Transport of Multimedia Streams over Wireless Networks. In *Proceedings of the 12th IEEE International Conference on Telecommunications (ConTEL)*, pages 147–153, Zagreb/Croatia, June 2013. ISBN 978-953-184-175-7. URL: https://simula.no/publications/Simula.simula.2000/simula_pdf_file. 5.3, 7, 7.3.2

[BDRF11] Martin Becke, Thomas Dreibholz, Erwin Paul Rathgeb, and Johannes Formann. Link Emulation on the Data Link Layer in a Linux-based Future Internet Testbed Environment. In *Proceedings of the 10th International Conference on Networks (ICN)*, pages 92–98, St. Maarten/Netherlands Antilles, January 2011. ISBN 978-1-61208-002-4. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/SCTP/Paper/ICN2011.pdf. 4.2

[BFH03] Robert Braden, Ted Faber, and Mark Handley. From Protocol Stack to Protocol Heap – Role-Based Architecture. *ACM SIGCOMM Computer Communication Review*, 33:17–22, January 2003. ISSN 0146-4833. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.1741&rep=rep1&type=pdf, doi:10.1145/774763.774765. 2

[BPB11a] Sébastien Barré, Christoph Paasch, and Olivier Bonaventure. MultiPath TCP: From Theory to Practice. In *Proceedings of the 10th International IFIP Networking Conference*, pages 444–457, Valencia/Spain, May 2011. ISBN 978-

3-642-20756-3. URL: http://inl.info.ucl.ac.be/system/files/networking-mptcp.pdf, doi:http://dx.doi.org/10.1109/JPROC.2010.2093850. 5.1.2.1

[BPB11b] Sébastien Barré, Christopher Paasch, and Olivier Bonaventure. MultiPath TCP - Guidelines for implementers. Internet Draft Version 00, IETF, Network Working Group, March 2011. draft-barre-mptcp-impl-00.txt, work in progress. URL: https://tools.ietf.org/id/draft-barre-mptcp-impl-00.txt. 5.1.2.1, 5.1.2.1, 5.1, A.4

[Bra89] Robert Braden. Requirements for Internet Hosts – Communication Layers. Standards Track RFC 1122, IETF, October 1989. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc1122.txt. 5.1.1.1, 7.1.3.2

[Bra96] Scott Bradner. The Tao of IETF: A Novice's Guide to the Internet Engineering Task Force. Informational RFC 2026, IETF, October 1996. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc2026.txt. 2.1

[Bri07] Bob Briscoe. Flow Rate Fairness: Dismantling a Religion. *ACM SIGCOMM Computer Communication Review*, 37:63–74, March 2007. ISSN 0146-4833. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.8436&rep=rep1&type=pdf, doi:10.1145/1232919.1232926. 3.2.2, 6.1.3

[BRW+13] Martin Becke, Erwin P. Rathgeb, Sebastian Werner, Irene Rüngeler, Michael Tüxen, and Randall R. Stewart. Data channel considerations for RTCWeb. *IEEE Communications Magazine*, 51(4):34–41, 2013. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=06495758&isnumber=. 4.1.2.4, 5.1.2.2, 5.3.1.2

[BS11] Marc Blanchet and Pierrick Seite. Multiple Interfaces and Provisioning Domains Problem Statement. Technical Report 6418, IETF, November 2011. ISSN 2070-1721. URL: http://www.ietf.org/rfc/rfc6418.txt. 5.1.2

[CA11] Brian Carpenter and Shane Amante. Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels. Technical Report 6438, IETF, November 2011. URL: http://www.ietf.org/rfc/rfc6438.txt. 5.1.2

[CB02] Brian Carpenter and Scott Brim. Middleboxes: Taxonomy and Issues. Technical Report 3234, IETF, February 2002. URL: http://www.ietf.org/rfc/rfc3234.txt. 2.3.1

[CCR+03] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. PlanetLab: An Overlay Testbed for Broad-Coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3), July 2003. ISSN 0146-4833. URL: https://www.planet-lab.org/files/pdn/PDN-03-009/pdn-03-009.pdf, doi:10.1145/956993.956995. 5.3.1

[Chr13] Christoph Paasch, Gregory Detal, Sébastien Barré, Fabien Duchene, Olivier Bonaventure. Automatic configuration on Gentoo, 2013. URL: http://multipath-tcp.org/pmwiki.php?n=Main.50Gbps. 5, 5.1.2.1, 5.1.2.2, 7.1.1, 8

[DABR12a] Thomas Dreibholz, Hakim Adhari, Martin Becke, and Erwin Paul Rathgeb. NetPerfMeter – A Versatile Tool for Multi-Protocol Network Performance Evaluations. Omnet++ code contribution, University of Duisburg-Essen, Institute for Experimental Mathematics, February 2012. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/SCTP/Paper/NetPerfMeterSim2012.pdf. 5.3.1.2

[DABR12b] Thomas Dreibholz, Hakim Adhari, Martin Becke, and Erwin Paul Rathgeb. Simulation and Experimental Evaluation of Multipath Congestion Control Strategies. In *Proceedings of the 2nd International Workshop on Protocols and Applications with Multi-Homing Support (PAMS)*, Fukuoka/Japan, March 2012. ISBN 978-0-7695-4652-0. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/SCTP/Paper/PAMS2012.pdf, doi:10.1109/WAINA.2012.186. 4.1.2.4, 5, 5.3

[DBAR11a] Thomas Dreibholz, Martin Becke, Hakim Adhari, and Erwin Paul Rathgeb. Evaluation of A New Multipath Congestion Control Scheme using the NetPerfMeter Tool-Chain. In *Proceedings of the 19th IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, pages 1–6, Hvar/Croatia, September 2011. ISBN 978-953-290-027-9. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/SCTP/Paper/SoftCOM2011.pdf. 4.2, 5.3.1.2, 5.3.1.2

[DBAR11b] Thomas Dreibholz, Martin Becke, Hakim Adhari, and Erwin Paul Rathgeb. On the Impact of Congestion Control for Concurrent Multipath Transfer on the Transport Layer. In *Proceedings of the 11th IEEE International Conference on Telecommunications (ConTEL)*, pages 397–404, Graz, Steiermark/Austria, June 2011. ISBN 978-953-184-152-8. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/SCTP/Paper/ConTEL2011.pdf. 5.1.2.1, 5.3, 6.2, 6.2.1, 7.1.3

[DBPR10a] Thomas Dreibholz, Martin Becke, Jobin Pulinthanath, and Erwin Paul Rathgeb. Applying TCP-Friendly Congestion Control to Concurrent Multipath Transfer. In *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 312–319, Perth, Western Australia/Australia, April 2010. ISSN 1550-445X. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/SCTP/Paper/AINA2010.pdf, doi:10.1109/AINA.2010.117. 4.1.2.4, 5, 5.3, 6, 6.2

[DBPR10b] Thomas Dreibholz, Martin Becke, Jobin Pulinthanath, and Erwin Paul Rathgeb. Implementation and Evaluation of Concurrent Multipath Transfer for SCTP in the INET Framework. In *Proceedings of the 3rd ACM/ICST International Workshop on OMNeT++*, Torremolinos, Málaga/Spain, March 2010. ISBN 978-963-9799-87-5. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/SCTP/Paper/OMNeT_Workshop2010-SCTP.pdf, doi:10.4108/ICST.SIMUTOOLS2010.8673. 4.1.2.4

[DBRT10] Thomas Dreibholz, Martin Becke, Erwin Paul Rathgeb, and Michael Tüxen. On the Use of Concurrent Multipath Transfer over Asymmetric Paths. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*, Miami, Florida/U.S.A., December 2010. ISBN 978-1-4244-5637-6. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/SCTP/Paper/Globecom2010.pdf, doi:10.1109/GLOCOM.2010.5683579. 1.1, 1.2, 4.1.2.4, 5, 5.1.2.1, 5.3, 5.3.2, 7.1.3, 7.1.3.1, 7.1.3.3, 7.3.1, 7.3.2, 7.3.3, 8.2

[DH98a] Stephen E. Deering and Robert M. Hinden. Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. Standards Track RFC 2463, IETF, December 1998. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc2463.txt. 5.3.1.2

[DH98b] Stephen E. Deering and Robert M. Hinden. Internet Protocol, Version 6 (IPv6). Standards Track RFC 2460, IETF, December 1998. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc2460.txt. 1, 2.1

[DHB+13] Gregory Detal, Benjamin Hesmans, Olivier Bonaventure, Yves Vanaubel, and Benoît Donnet. Revealing middlebox interference with tracebox. In *Proceedings of the 2013 ACM SIGCOMM conference on Internet measurement conference*. ACM, October 2013. 2.3.1

[Dre07] Thomas Dreibholz. *Reliable Server Pooling – Evaluation, Optimization and Extension of a Novel IETF Architecture*. PhD thesis, University of Duisburg-Essen, Faculty of Economics, Institute for Computer Science and Business Information Systems, March 2007. URL: http://duepublico.uni-duisburg-essen.de/servlets/DerivateServlet/Derivate-16326/Dre2006_final.pdf. 4.1.3.1

[Dre12a] Thomas Dreibholz. *Evaluation and Optimisation of Multi-Path Transport using the Stream Control Transmission Protocol*. Habilitation treatise, University of Duisburg-Essen, Faculty of Economics, Institute for Computer Science and Business Information Systems, March 2012. URL: http://duepublico.uni-duisburg-essen.de/servlets/DerivateServlet/Derivate-29737/Dre2012_final.pdf. 2.2, 3.2.1, 3.2.2.1, 3.2.3.2, 4.1.2, 4.1.2.1, 4.1.2.2, 4.1.2.3, 4.1.2.4, 4.1.3, 4.1.3.1, 6, 7.3.2, 7.3.3.2

[Dre12b] Thomas Dreibholz. SimProcTC – A Simulation Processing Tool-Chain for OMNeT++ Simulations, 2012. URL: http://www.iem.uni-due.de/~dreibh/omnetpp/. 4.1.3

[Dre13] Thomas Dreibholz. The NorNet Core Testbed – Introduction and Status. In *Proceedings of the 1st International NorNet Users Workshop (NNUW-1)*, Fornebu, Akershus/Norway, September 2013. URL: https://simula.no/publications/Simula.simula.2124/simula_pdf_file. 5.4

[DSTR10] Thomas Dreibholz, Robin Seggelmann, Michael Tüxen, and Erwin Paul Rathgeb. Transmission Scheduling Optimizations for Concurrent Multipath Transfer. In *Proceedings of the 8th International Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT)*, volume 8, Lancaster, Pennsylvania/U.S.A., November 2010. ISSN

2074-5168. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/SCTP/Paper/PFLDNeT2010.pdf. 5, 7, 7.2.2.2

[Dun94] Jay Duncanson. Inverse multiplexing. *Communications Magazine, IEEE*, 32(4):34–41, 1994. 3.1

[DZR09] Thomas Dreibholz, Xing Zhou, and Erwin Paul Rathgeb. SimProcTC – The Design and Realization of a Powerful Tool-Chain for OMNeT++ Simulations. In *Proceedings of the 2nd ACM/ICST International Workshop on OMNeT++*, pages 1–8, Rome/Italy, March 2009. ISBN 978-963-9799-45-5. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/ReliableServer/Publications/OMNeT__Workshop2009.pdf, doi:10.4108/ICST.SIMUTOOLS2009.5517. 4.1.3

[Ear14] Philip Eardley. Survey of MPTCP Implementations. Internet Draft Version 2, IETF, Individual Submission, January 2014. draft-eardley-mptcp-implementations-survey-02.txt, work in progress. URL: https://tools.ietf.org/id/draft-eardley-mptcp-implementations-survey-02.txt. 4.1.2.2

[Edd07] Wesley M. Eddy. TCP SYN Flooding Attacks and Common Mitigations. Informational RFC 4987, IETF, August 2007. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc4987.txt. 2.2.2, 2.2.2

[EF94] Kjeld Borch Egevang and Paul Francis. The IP Network Address Translator (NAT). Informational RFC 1631, IETF, May 1994. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc1631.txt. 2.3.1

[ELZ86] Derek L. Eager, Edward D. Lazowska, and John Zahorjan. Adaptive load sharing in homogeneous distributed systems. *Software Engineering, IEEE Transactions on*, (5):662–675, 1986. 3.1

[Fel07] Anja Feldmann. Internet Clean-Slate Design: What and Why? *ACM SIGCOMM Computer Communication Review*, 37:59–64, July 2007. ISSN 0146-4833. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.98.8874&rep=rep1&type=pdf, doi:10.1145/1273445.1273453. 8

[FJ93] Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993. ISSN 1063-6692. URL: https://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.128.5092&rep=rep1&type=pdf, doi:10.1109/90.251892. 2.3.2

[Flo97] Sally Floyd. RED: Discussions of Setting Parameters, November 1997. URL: http://icir.org/floyd/REDparameters.txt. 2.3.2

[Flo00] Sally Floyd. Congestion Control Principles. Technical Report 2914, IETF, September 2000. Updated by RFC 7141. URL: http://www.ietf.org/rfc/rfc2914.txt. 2.1, 3.2.2, 6.1

[FODA14] Simone Ferlin-Oliveira, Thomas Dreibholz, and Özgü Alay. Tackling the Challenge of Bufferbloat in Multi-Path Transport over Heterogeneous Wireless Networks. In *Proceedings of the IEEE/ACM International Symposium on Quality*

of Service (IWQoS), Hong Kong/People's Republic of China, May 2014. ISBN 978-1-4799-4852-9. URL: https://www.simula.no/publications/Simula.simula.2722/simula_pdf_file. 1.2, 9.1

[Fre00] Ned Freed. Behavior of and Requirements for Internet Firewalls. Technical Report 2979, IETF, October 2000. URL: http://www.ietf.org/rfc/rfc2979.txt. 5.1.1.1

[Fre12] FreeBSD Documentation Project. *FreeBSD Handbook*, July 2012. URL: http://ftp.freebsd.org/pub/FreeBSD/doc/en/books/handbook/book.pdf.bz2. 7.1

[FRH+11] Alan Ford, Costin Raiciu, Mark Handley, Sébastien Barré, and Janardhan R. Iyengar. Architectural Guidelines for Multipath TCP Development. Informational RFC 6182, IETF, March 2011. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc6128.txt. 5.1.1.1, 5.1.1.2, 5.2, 6.2.2, 7.1.1, 8.1.1.3

[FRHB13] Alan Ford, Costin Raiciu, Mark Handley, and Olivier Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. Standards Track RFC 6824, IETF, January 2013. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc6824.txt. 1.1, 3.2.3, 3.2.3.1, 4, 5, 5.1.1.1, 6.1, 8.1.1, 8.1.1.1

[GÉA13] GÉANT. GÉANT Project Home, 2013. URL: http://www.geant.net/. 5.3.1.2

[Get11a] Jim Gettys. Bufferbloat – Dark Buffers in the Internet, January 2011. URL: https://www.bufferbloat.net/attachments/9/BufferBloat11.pdf. 7.1.3.3

[Get11b] Jim Gettys. What is Bufferbloat, Anyway?, 2011. URL: https://gettys.wordpress.com/what-is-bufferbloat-anyway/. 1

[GHJV95] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns*. Addison Wesley, Reading, MA, 1995. 4.1.2.2

[Gro02] Dan Grossman. New Terminology and Clarifications for DiffServ. Informational RFC 3260, IETF, April 2002. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc3260.txt. 2.3.1

[GTGB11] Carmelita Goerg, A. Timm-Giel, and M. Thomas-Ralf Banniza. Performance evaluation of multipath tcp linux implementations. In *Wuerzburg Workshop on IP: Joint ITG and Euro-NF Workshop Visions of Future Generation Networks*, 2011. 7.1.3

[Hah91] Ellen L. Hahne. Round-Robin Scheduling for Max-Min Fairness in Data Networks. *IEEE Journal on Selected Areas in Communications*, 9:1024–1039, September 1991. URL: https://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.21.2114&rep=rep1&type=pdf. 3.2.2

[HFGN12] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida. The NewReno Modification to TCP's Fast Recovery Algorithm. Technical Report 6582, IETF, April 2012. ISSN 2070-1721. URL: http://www.ietf.org/rfc/rfc6582.txt. 5.1.1.1

[HS02] Hung-Yun Hsieh and Raghupathy Sivakumar. pTCP: An End-to-End Transport Layer Protocol for Striped Connections. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, pages 24–33. IEEE Computer Society, 2002. URL: http://dblp.uni-trier.de/db/conf/icnp/icnp2002.html. 3.2

[Hui95] Christian Huitema. Multi-homed TCP. Internet Draft Version 01, IETF, Network Working Group, May 1995. draft-huitema-multi-homed-01,.txt, work in progress. URL: https://tools.ietf.org/id/draft-huitema-multi-homed-01. 3.2

[IAS05] Janardhan R. Iyengar, Paul Amer, and Randall Stewart. Receive Buffer Blocking in Concurrent Multipath Transfer. In *Proceedings of the IEEE GLOBECOM*, pages 121–126, St. Louis, Missouri/U.S.A., November 2005. ISBN 978-1-4244-1707-0. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.145.7496&rep=rep1&type=pdf, doi:10.1109/GLOCOM.2005.1577365. 7.1.3.3

[IAS06] Janardhan R. Iyengar, Paul D. Amer, and Randall Stewart. Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-End Paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964, October 2006. ISSN 1063-6692. URL: http://www.fandm.edu/jiyengar/papers/cmt-ton2006.pdf, doi:10.1109/TNET.2006.882843. 3.1, 3.2.3.2, 5.1.1.3, 5.3.2

[Ins13] Institute of Information and Electronics Communication Technologies and Applied Mathematics (ICTEAM). MultiPath TCP – Linux Kernel Implementation, 2013. URL: http://mptcp.info.ucl.ac.be. 4.1.2.2

[JBB92] Van Jacobson, Robert Braden, and David A. Borman. TCP Extensions for High Performance. Informational RFC 1323, IETF, May 1992. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc1323.txt. 2.2.3, 4.1.2.1

[Jun05] Andreas Jungmaier. *Das Transportprotokoll SCTP: Leistungsbewertung und Optimierung eines neuen Transportprotokolls.* PhD thesis, University of Duisburg-Essen, Institute for Experimental Mathematics, 2005. 3.2.3.2

[Kel97] Frank Kelly. Charging and Rate Control for Elastic Traffic. *European Transactions on Telecommunications*, 8:33–37, January 1997. URL: https://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.144.6615&rep=rep1&type=pdf. 3.2.2

[KGPB14] Ramin Khalili, Nicolas Gast, Miroslav Popovic, and Jean-Yves Le Boudec. Coupled Multipath-Aware Congestion Control. Internet Draft Version 04, IETF, Network Working Group, February 2014. draft-ietf-mptcp-congestion-07, work in progress. URL: https://tools.ietf.org/id/draft-khalili-mptcp-congestion-control-04. 6.2.3

[KHF06] Eddie Kohler, Mark Handley, and Sally Floyd. Datagram Congestion Control Protocol (DCCP). Standards Track RFC 4340, IETF, March 2006. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc4340.txt. 2.1

[Koz05] Charles Kozierok. *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*. No Starch Press, 2005. ISBN 978-1593270476. URL: http://www.tcpipguide.com/free/index.htm. 2, 2.2, 2.2.1, 5.1.1.1

[KR02] James F. Kurose and Keith Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2002. 5.2.1

[Kri13] Kristian Evensen. Automatic configuration with MULTI, 2013. URL: https://github.com/kristrev/multi. 5.1.2.1

[KSc+11] Michael Kleis, Abbas Siddiqui, Irfan Şimşek, Martin Becke, Dirk Hoffstadt, Alexander Marold, Christian Henke, Julius Müller, Cristian Varas, Thomas Magedanz, Paul Müller, and Erwin Rathgeb. Cross-Layer Security and Functional Composition for a Future Internet. In *Proceedings of the Joint ITG and Euro-NF Workshop on Visions of Future Generation Networks (EuroView2011)*, Würzburg, Bayern/Germany, August 2011. Demo Presentation at the G-Lab Status Meeting. URL: http://www.euroview2011.com/fileadmin/content/euroview2011/abstracts/abstract_kleis.pdf. 5.3.1

[KSW+08] Andreas Köpke, Michael Swigulski, Karl Wessel, Daniel Willkomm, P.T. Klein Haneveld, Tom Parker, Otto Visser, Hermann Simon Lichte, and Stefan Valentin. Simulating wireless and mobile networks in omnet++ – the mixim vision. In *OMNeT++ 2008: Proceedings of the 1st International Workshop on OMNeT++ (hosted by SIMUTools 2008)*, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 4.1.2

[KV05] Frank Kelly and Thomas Voice. Stability of End-to-End Algorithms for Joint Routing and Rate Control. *ACM SIGCOMM Computer Communication Review*, 35(2):5–12, April 2005. ISSN 0146-4833. URL: http://ccr.sigcomm.org/archive/2005/april/SIGCOMM-CCR-V35-N2.pdf, doi:10.1145/1064413.1064415. 1.3, 3.2, 6.1.2.1

[LOTD08] Peter Lei, Lyndon Ong, Michael Tüxen, and Thomas Dreibholz. An Overview of Reliable Server Pooling Protocols. Informational RFC 5351, IETF, September 2008. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc5351.txt. 4.1.3.1

[MAB+08] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38(2):69–74, March 2008. URL: http://doi.acm.org/10.1145/1355734.1355746, doi:10.1145/1355734.1355746. 8.1.2

[Mal93] Gary Scott Malkin. Traceroute Using an IP Option. Standards Track RFC 1393, IETF, January 1993. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc1393.txt. 5.3.1.2

[Man10] William Manning. *Basic Administration for Citrix NetScaler 9.0: 1Y0-A11 Exam Certification Exam Preparation Course in a Book for Passing the Basic*

Administration for Citrix ... On Your First Try Certification Study Guide. Emereo Pty Ltd, London, UK, UK, 2010. 4.1.2.2

[Max13] MaxMind. IP Geolocation and Online Fraud Prevention, 2013. URL: https://www.maxmind.com/. 5.3.1.2

[MK01a] Luiz Magalhaes and Robin Kravets. Transport level mechanisms for bandwidth aggregation on mobile hosts. In *Network Protocols, 2001. Ninth International Conference on*, pages 165–171. IEEE, 2001. 3.1

[MK01b] Luiz Magalhaes and Robin Kravets. Transport Level Mechanisms for Bandwidth Aggregation on Mobile Hosts. In *Proceedings of the Ninth International Conference on Network Protocols*, pages 165–171. IEEE Computer Society, 2001. 3.2

[MLE+99] Louis Mamakos, Kurt Lidl, Jeff Evarts, David Carrel, Dan Simone, and Ross Wheeler. A Method for Transmitting PPP Over Ethernet (PPPoE). Informational RFC 2516, IETF, February 1999. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc2516.txt. 2.2.1

[MMD91] Ravi Mazumdar, Lorne G. Mason, and Christos Douligeris. Fairness in Network Optimal Flow Control: Optimality of Product Forms. *IEEE Transactions on Communications*, 39:775–982, May 1991. ISSN 0090-6778. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=87140&isnumber=, doi:10.1109/26.87140. 3.2.2

[MMFR96] Matt Mathis, Jamshid Mahdavi, Sally Floyd, and Allyn Romanow. TCP Selective Acknowledgment Options. Standards Track RFC 2018, IETF, October 1996. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc2018.txt. 2.2.3, 5.1.1.1

[MN06] Robert Moskowitz and Pekka Nikander. Host Identity Protocol (HIP) Architecture. Informational RFC 4423, IETF, May 2006. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc4423.txt. 3.1.2, 3.2

[MSM97] Matt Mathis, J. Semke, and Jamshid Mahdavi. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *ACM SIGCOMM Computer Communication Review*, 27(3):67–82, July 1997. ISSN 0146-4833. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.25.3452&rep=rep1&type=pdf, doi:10.1145/263932.264023. 7.1.1

[Nag84] John Nagle. Congestion Control in IP/TCP Internetworks. Informational RFC 896, IETF, January 1984. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc896.txt. 7.1.3.2

[NB09] Erik Nordmark and Marcelo Bagnulo. Shim6: Level 3 Multihoming Shim Protocol for IPv6. Standards Track RFC 5533, IETF, June 2009. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc5533.txt. 3.1.2, 3.2

[NEY+08] Preethi Natarajan, Nasif Ekiz, Ertuğrul Yilmaz, Paul D. Amer, and Janardhan R. Iyengar. Non-Renegable Selective Acknowledgments (NR-SACKs) for

SCTP. In *Proceedings of the 16th IEEE International Conference on Network Protocols (ICNP)*, pages 187–196, Orlando, Florida/U.S.A., October 2008. ISBN 978-1-4244-2506-8. URL: http://www.eecis.udel.edu/~amer/PEL/poc/pdf/ICNP2008-natarajanNonRenegableSacks.pdf, doi:10.1109/ICNP.2008.4697037. 5.3.2, 7.1.3.3, 7.3.2

[NUO+07] Kenichi Nagami, Satoshi Uda, Nobuo Ogashiwa, Hiroshi Esaki, Ryuji Wakikawa, and Hiroyuki Ohnishi. Multi-homing for small scale fixed network Using Mobile IP and NEMO. Technical Report 4908, IETF, June 2007. URL: http://www.ietf.org/rfc/rfc4908.txt. 3.1.1

[NZNP11] Sinh Chung Nguyen, Xiaofei Zhang, Thi-Mai-Trang Nguyen, and G. Pujolle. Evaluation of throughput optimization and load sharing of multipath tcp in heterogeneous networks. In *Wireless and Optical Communications Networks (WOCN), 2011 Eighth International Conference on*, pages 1–5, May 2011. doi:10.1109/WOCN.2011.5872966. 7.1.3

[Ond13] Ondrej Caletka. Automatic configuration on Gentoo, 2013. URL: https://gist.github.com/oskar456/7264828. 5.1.2.1

[PA00] Vern Paxson and Mark Allman. Computing TCP's Retransmission Timer. Standards Track RFC 2988, IETF, November 2000. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc2988.txt. 5.1.2.1

[PACS11] Vern Paxson, Mark Allman, H.K. Jerry Chu, and Matt Sargent. Computing TCP's Retransmission Timer. RFC 6298, IETF, June 2011. ISSN 2070-1721. URL: http://www.ietf.org/rfc/rfc6298.txt. 2.2.3, 2.2.3, 4.1.2.1, 5.1.1.1, 5.1.2.1, 5.1.2.2

[PEK11] Christopher Pluntke, Lars Eggert, and Niko Kiukkonen. Saving Mobile Device Energy with Multipath TCP. In *Proceedings of the 6th ACM International Workshop on MobiArch*, pages 1–6, Bethesda, Maryland/U.S.A., June 2011. ISBN 978-1-4503-0740-6. URL: http://eggert.org/papers/2011-mobiarch-mptcp-saving-energy.pdf, doi:10.1145/1999916.1999918. 5.1.2.1

[PFTK98] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP Throughput: A Simple Model and Its Empirical Validation. In *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM 1998, pages 303–314, New York, NY, USA, 1998. ACM. URL: http://doi.acm.org/10.1145/285237.285291, doi:10.1145/285237.285291. 7.1.1

[Pos80a] Jon Postel. DoD standard Internet Protocol. Technical Report 760, IETF, January 1980. Obsoleted by RFC 791, updated by RFC 777. URL: http://www.ietf.org/rfc/rfc760.txt. 2.1

[Pos80b] Jonathan Bruce Postel. User Datagram Protocol. Standards Track RFC 768, IETF, August 1980. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc768.txt. 2.1

[Pos81a] Jonathan Bruce Postel. Internet Protocol. Standards Track RFC 791, IETF, September 1981. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc791.txt. 1, 5.1.2

[Pos81b] Jonathan Bruce Postel. Transmission Control Protocol. Standards Track RFC 793, IETF, September 1981. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc793.txt. 1, 2.1, 2.2, 2.2.1, 2.2.4, 5.1.1.1, 5.1.1.1, 7.2.1

[PR85] Jonathan Bruce Postel and Joyce K. Reynolds. File Transfer Protocol (FTP). Standards Track RFC 959, IETF, October 1985. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc959.txt. 2

[R D12] R Development Core Team. *R: A Language and Environment for Statistical Computing*. Vienna/Austria, July 2012. URL: http://cran.r-project.org/doc/manuals/refman.pdf. 4.1.3.1

[Ram12] Ramin Khalili and Nicolas Gast and Miroslav Popovic and Utkarsh Upadhyay and Jean-Yves Le Boudec. MPTCP is not Pareto-Optimal: Performance Issues and a Possible Solution. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, pages 1–12, Nice/France, December 2012. URL: http://conferences.sigcomm.org/co-next/2012/eproceedings/conext/p1.pdf. 3.2.2.2, 6, 6.2, 6.2.3, 6.2.3

[RBP+11] Costin Raiciu, Sébastien Barré, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. Improving Datacenter Performance and Robustness with Multipath TCP. In *Proceedings of the ACM SIGCOMM*, Toronto/Canada, August 2011. URL: http://inl.info.ucl.ac.be/system/files/mptcp-sigcomm.pdf. 5, 5.1.2.1

[RDB+10] Thomas Reschka, Thomas Dreibholz, Martin Becke, Jobin Pulinthanath, and Erwin Paul Rathgeb. Enhancement of the TCP Module in the OMNeT++/INET Framework. In *Proceedings of the 3rd ACM/ICST International Workshop on OMNeT++*, Torremolinos, Málaga/Spain, March 2010. ISBN 978-963-9799-87-5. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/SCTP/Paper/OMNeT__Workshop2010-TCP.pdf. 4.1.2.1

[RHW11] Costin Raiciu, Mark Handley, and Damon Wischik. Coupled Congestion Control for Multipath Transport Protocols. Informational RFC 6356, IETF, October 2011. ISSN 2070-1721. URL: http://www.ietf.org/rfc/rfc6356.txt. 5.1.1.1, 6.1, 6.1.2.1, 6.2

[RIP13] RIPE CC. RIPE Network Coordination Centre, 2013. URL: http://www.ripe.net/. 5.3.1.2

[RKT02] Dan Rubenstein, Jim Kurose, and Don Towsley. Detecting Shared Congestion of Flows via End-to-End Measurement. *IEEE/ACM Transactions on Networking*, 10(3):381–395, June 2002. ISSN 1063-6692. URL: https://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.131.6332&rep=rep1&type=pdf, doi:10.1109/TNET.2002.1012369. 3.2.2.1

[RPB+10] Costin Raiciu, Christopher Pluntke, Sébastien Barré, Adam Greenhalgh, Damon Wischik, and Mark Handley. Data Center Networking with Multipath TCP. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, pages 1–6, Monterey, California/U.S.A., October 2010. ISBN 978-1-4503-0409-2. URL: http://www.cs.ucl.ac.uk/staff/C.Raiciu/files/mptcp_dc.pdf, doi:10.1145/1868447.1868457. 1.2

[RPB+12] Costin Raiciu, Christoph Paasch, Sébastien Barré, Alan Ford, Michio Honda, Fabien Duchêne, Olivier Bonaventure, and Mark Handley. How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation (NSDI)*, 2012. URL: https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final125.pdf. 4.1.2.2, 5, 7.1, 7.1.2, 7.1.3, 7.1.3.1, 7.1.3.3, 7.3.2, 7.3.3, 7.3.3, 7.3.3.1

[RTR08] Irene Rüngeler, Michael Tüxen, and Erwin Paul Rathgeb. Integration of SCTP in the OMNeT++ Simulation Environment. In *Proceedings of the 1st ACM/ICST International Workshop on OMNeT++*, Marseille, Bouches-du-Rhône/France, March 2008. ISBN 978-963-9799-20-2. URL: http://dl.acm.org/ft_gateway.cfm?id=1416310&type=pdf, doi:10.4108/ICST.SIMUTOOLS2008.3027. 4.1.2.3

[Rün09] Irene Rüngeler. *SCTP – Evaluating, Improving and Extending the Protocol for Broader Deployment*. PhD thesis, University of Duisburg-Essen, Faculty of Economics, Institute for Computer Science and Business Information Systems, December 2009. URL: http://duepublico.uni-duisburg-essen.de/servlets/DerivateServlet/Derivate-23465/DissPDF.pdf. 7.1.1

[RWH09] Costin Raiciu, Damon Wischik, and Mark Handley. Practical Congestion Control for Multipath Transport Protocols. Technical report, University College London, London/United Kingdom, 2009. URL: http://nrg.cs.ucl.ac.uk/mptcp/mptcp-techreport.pdf. 3.2.2.2, 3.3, 5.3.2, 6, 6.2, 6.2.2, 6.2.2

[SBB10] Samar Shailendra, Ratnajit Bhattacharjee, and Sanjay K. Bose. Optimized Flow Division Modeling for Multi-Path Transport. In *Proceedings of the IEEE International Conference on Communication Systems (ICCS)*, pages 1–4, Kolkata/India, November 2010. ISBN 978-1-4244-9072-1. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5686169&isnumber=, doi:10.1109/INDCON.2010.5712713. 7.3.2

[SBJR13] Irfan Simsek, Martin Becke, Yves.I. Jerschow, and Erwin. P. Rathgeb. A clean-slate security vision for future networks: Simultaneously ensuring information security and establishing smart in-network services using the example of blind packet forwarding. In *Network of the Future (NOF), 2013 Fourth International Conference on the*, pages 1–5, Oct 2013. doi:10.1109/NOF.2013.6724501. 8.1.2

[SBL+13] Golam Sarwar, Roksana Boreli, Emmanuel Lochin, Ahlem Mifdaoui, and Guillaume Smith. Mitigating receiver's buffer blocking by delay aware packet

scheduling in multipath data transfer. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 1119–1124, March 2013. doi:10.1109/WAINA.2013.80. 7.1.3.1, 7.3.2

[Seg12] Robin Seggelmann. *SCTP: strategies to secure end-to-end communication*. PhD thesis, University of Duisburg-Essen, Faculty of Economics, Institute for Computer Science and Business Information Systems, Duisburg, Essen, 2012. Duisburg, Essen, Univ., Diss., 2012. URL: http://duepublico.uni-duisburg-essen.de/servlets/DocumentServlet?id=29603. 2.2, 4.1, 4.1.2.1

[SF13] Michael Scharf and Alan Ford. Multipath TCP (MPTCP) Application Interface Considerations. Informational RFC 6897, IETF, March 2013. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc6897.txt. 5.1.1.1

[SFR03] W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff. *Unix Network Programming*. Addison-Wesley Professional, 2003. ISBN 0-131-41155-1. 2.3.3, 2

[SGTG+12] Amanpreet Singh, Carmelita Goerg, Andreas Timm-Giel, Michael Scharf, and Thomas-Rolf Banniza. Performance comparison of scheduling algorithms for multipath transfer. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 2653–2658, Dec 2012. doi:10.1109/GLOCOM.2012.6503517. 5, 7, 7.1.1, 7.1.3, 7.1.3.1, 7.2.3, 7.3.2, 7.4, 7.5.2.1

[Sim94] William Allen Simpson. The Point-to-Point Protocol (PPP). Standards Track RFC 1661, IETF, July 1994. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc1661.txt. 3.1.2

[SLM+96] Keith Sklower, Brian Lloyd, Glenn McGregor, Dave Carr, and Tom Coradetti. The PPP Multilink Protocol (MP). Technical Report 1990, IETF, August 1996. URL: http://www.ietf.org/rfc/rfc1990.txt. 3.1.2

[SM11] M. Subramaniam and D. Manjula. Effective Transport using Concurrent Multipath Transfer by Redundant Transmission during Path Failure. *European Journal of Scientific Research*, 58(2):247–256, August 2011. ISSN 1450-216X. 7

[Ste97] W. Richard Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. Standards Track RFC 2001, IETF, January 1997. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc2001.txt. 5.1.1.1

[Ste07] Randall R. Stewart. Stream Control Transmission Protocol. Standards Track RFC 4960, IETF, September 2007. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc4960.txt. 1, 2.1, 2.2, 2.2.1, 2.2.2, 2.2.2, 2.2.2, 2.2.4, 2.2.4.1, 3.1, 3.2, 3.2.3.2, 5, 5.1.1.1, 5.1.1.1, 5.1.1.3, 5.1.2.2, 5.2.1, 7.2.1, 7.3.1, 8.1.2

[STR10] Robin Seggelmann, Michael Tüxen, and Erwin Paul Rathgeb. Stream Scheduling Considerations for SCTP. In *Proceedings of the 18th IEEE*

International Conference on Software, Telecommunications and Computer Networks (SoftCOM), September 2010. ISBN 978-953-290-004-0. URL: http://ieeexplore.ieee.org/iel5/5611454/5623609/05623661.pdf?arnumber=5623661&isnumber=. 7.2.2.2

[SXM+00] Randall R. Stewart, Qiaobing Xie, Ken Morneault, Chip Sharp, Hanns Jürgen Schwarzbauer, Tom Taylor, Ian Rytina, Malleswar Kalla, Lixia Zhang, and Vern Paxson. Stream Control Transmission Protocol. Standards Track RFC 2960, IETF, October 2000. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc2960.txt. 2.1, 5.1.1.1

[SXT+07] Randall R. Stewart, Qiaobing Xie, Michael Tüxen, Shin Maruyama, and Masahiro Kozuka. Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration. Standards Track RFC 5061, IETF, September 2007. ISSN 2070-1721. URL: https://www.ietf.org/rfc/rfc5061.txt. 2.2.1, 5

[Tan96] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, Upper Saddle River, New Jersey/U.S.A., 1996. ISBN 0-13-349945-6. 2

[TH04] Aristotelis Tsirigos and Zygmunt J Haas. Analysis of multipath routing-part i: The effect on the packet delivery ratio. *Wireless Communications, IEEE Transactions on*, 3(1):138–146, 2004. 3.1.2

[TRR08] Michael Tüxen, Irene Rüngeler, and Erwin Paul Rathgeb. Interface Connecting the INET Simulation Framework with the Real World. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools)*, pages 1–6, Marseille, Bouches-du-Rhône/France, March 2008. ISBN 978-963-9799-20-2. URL: http://dl.acm.org/ft_gateway.cfm?id=1416267&type=pdf, doi:10.1145/1416222.1416267. 4.1

[Var05] András Varga. *OMNeT++ Discrete Event Simulation System User Manual – Version 3.2*. Technical University of Budapest/Hungary, March 2005. URL: http://www2.ensc.sfu.ca/~ljilja/ENSC835/Spring08/News/Presentations/OMNeT++/usman.pdf. 4.1.1

[Var11] András Varga. *INET Framework for OMNeT++*, 2011. URL: http://inet.omnetpp.org/. 4.1, 4.1.2

[Var12] András Varga. *OMNeT++ Discrete Event Simulation System User Manual – Version 4.2*, December 2012. URL: http://www.omnetpp.org/doc/omnetpp/Manual.pdf. 4.1

[VBOMT13a] Thomas Volkert, Martin Becke, Manuel Osdoba, and Andreas Mitschele-Thiel. Multipath Video Streaming based on Hierarchical Routing Management. In *Proceedings of the 3nd International Workshop on Protocols and Applications with Multi-Homing Support (PAMS)*, pages 1107–1112, Barcelona, Catalonia/Spain, March 2013. ISBN 978-0-7695-4952-1. URL: http://www.homer-conferencing.com/public/MP_HRM.pdf, doi:10.1109/WAINA.2013.161. 5, 7.1.3

[VBOMT13b] Thomas Volkert, Martin Becke, Manuel Osdoba, and Andreas Mitschele-Thiel. Multipath video streaming based on hierarchical routing management. In *Proceedings of 27th IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 1107 – 1112*, Barcelona, Spain, 03 2013. 8.1.2

[VMT12] Thomas Volkert and Andreas Mitschele-Thiel. Hierarchical routing management for improving multimedia transmissions and qoe. In *Proceedings of 13th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, San Francisco, California, USA, 06 2012. 8.1.2

[Wel05] Michael Welzl. *Network Congestion Control: Managing Internet Traffic*. John Wiley & Sons, Chichester, West Sussex/United Kingdom, 2005. ISBN 978-0-470-02528-4. 2.2.4, 2.2.4.1, 3.2.2, 6.3.1.1

[Wel13] Michael Welzl. Potential Applications of Shared Bottleneck Detection. In *Proceedings of the 1st International NorNet Users Workshop (NNUW-1)*, Fornebu, Akershus/Norway, September 2013. URL: https://simula.no/publications/Simula.simula.2132/simula_pdf_file. 3.2.2.1

[WHB08] Damon Wischik, Mark Handley, and Marcelo Bagnulo Braun. The Resource Pooling Principle. *ACM SIGCOMM Computer Communication Review*, 38(5):47–52, October 2008. ISSN 0146-4833. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.161.8858&rep=rep1&type=pdf, doi:10.1145/1452335.1452342. 1.1, 1.2, 3.2, 3.1, 3.2.2.2, 6.1.2.1, 6.1, A.4

[WRGH11] Damon Wischik, Costin Raiciu, Adam Greenhalgh, and Mark Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pages 8–8, Berkeley, CA, USA, 2011. USENIX Association. URL: http://dl.acm.org/citation.cfm?id=1972457.1972468. 6.1.3.1

[WS95] Gary R. Wright and W. Richard Stevens. *TCP/IP Illustrated, Volume 2: The Implementation*. Addison-Wesley Professional, 1995. ISBN 978-0201633542. 5.1.1.1

[YA13] Fan Yang and P. Amer. Non-renegable selective acknowledgments (nr-sacks) for mptcp. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 1113–1118, March 2013. doi:10.1109/WAINA.2013.59. 7.1.3.3, 7.3.2

[YAE13] Fan Yang, P. Amer, and N. Ekiz. A scheduler for multipath tcp. In *Computer Communications and Networks (ICCCN), 2013 22nd International Conference on*, pages 1–7, July 2013. doi:10.1109/ICCCN.2013.6614091. 7.2.1

[YWY08a] Muhammad Murtaza Yousaf, Michael Welzl, and Bülent Yener. Accurate Shared Bottleneck Detection Based On SVD and Outliers Detection. Technical Report NSG-DPS-UIBK-01, University of Innsbruck, Institute of Computer Science, Innsbruck, Tirol/Austria, August 2008. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.3147&rep=rep1&type=pdf. 3.2.2.1

[YWY08b] Muhammad Murtaza Yousaf, Michael Welzl, and Bülent Yener. On the Accurate Identification of Network Paths having a Common Bottleneck. In *Proceedings of the ACM SIGCOMM*, Seattle, Washington/U.S.A., August 2008. Poster Presentation. URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.1874&rep=rep1&type=pdf. 3.2.2.1

[ZDB+10] Xing Zhou, Thomas Dreibholz, Martin Becke, Jobin Pulinthanath, Erwin Paul Rathgeb, and Wencai Du. The Software Modeling and Implementation of Reliable Server Pooling and RSPLIB. In *Proceedings of the 8th ACIS Conference on Software Engineering Research, Management and Applications (SERA)*, pages 129–136, Montréal, Québec/Canada, May 2010. ISBN 978-0-7695-4075-7. URL: https://www.wiwi.uni-due.de/fileadmin/fileupload/I-TDR/ReliableServer/Publications/SERA2010.pdf, doi:10.1109/SERA.2010.26. 4.1.2.4

[ZLK+04] Ming Zhang, Junwen Lai, Arvind Krishnamurthy, Larry Peterson, and Randolph Wang. A Transport Layer Approach for Improving End-to-end Performance and Robustness Using Redundant Paths. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, pages 8–8, Berkeley, CA, USA, 2004. USENIX Association. URL: http://dl.acm.org/citation.cfm?id=1247415.1247423. 3.2

[ZMD+09] Tanja Zseby, Maurizio Molina, Nick Duffield, Saverio Niccolini, and Frederic Raspall. Sampling and Filtering Techniques for IP Packet Selection. Technical Report 5475, IETF, March 2009. URL: http://www.ietf.org/rfc/rfc5475.txt. 2.3.1

Curriculum Vitae

Der Lebenslauf ist in der Online-Version aus Gründen des Datenschutzes nicht enthalten.