Improving the Efficiency of Multipath Transport Protocols

Ming Li

Aalto University

DOCTORAL DISSERTATIONS

Aalto University publication series
DOCTORAL DISSERTATIONS 168/2014

Improving the Efficiency of Multipath Transport Protocols

Ming Li

Doctoral dissertation for the degree of Doctor of Science in Technology to be presented with due permission of the School of Science for public examination and debate in Lecture Hall AS1 at the Aalto University School of Science (Espoo, Finland) on the 21st of November 2014 at 12 noon.

Aalto University
School of Science
Department of Computer Science and Engineering
Data Communications Software

Supervising professor
Professor Antti Ylä-Jääski

Thesis advisors
Dr. Andrey Lukyanenko
Dr. Matti Siekkinen

Preliminary examiners
Professor Xiaoming Fu, Georg-August-University of Goettingen, Germany
Professor Petri Mähönen, RWTH Aachen University, Germany

Opponent
Associate Professor Stefano Secci, University Pierre and Marie Curie, France

Aalto University publication series
DOCTORAL DISSERTATIONS 168/2014

© Ming Li

ISBN 978-952-60-5925-9 (printed)
ISBN 978-952-60-5926-6 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)
http://urn.fi/URN:ISBN:978-952-60-5926-6

Unigrafia Oy
Helsinki 2014

Finland

NORDIC ECOLABEL
441 697
Printed matter

Aalto University

Abstract

Aalto University, P.O. Box 11000, FI-00076 Aalto www.aalto.fi

Author
Ming Li

Name of the doctoral dissertation Improving the Efficiency of Multipath Transport Protocols	
Publisher School of Science	
Unit Department of Computer Science and Engineering	
Series Aalto University publication series DOCTORAL DISSERTATIONS 168/2014	
Field of research Networking	
Manuscript submitted 8 August 2014	**Date of the defence** 21 November 2014
Permission to publish granted (date) 24 October 2014	**Language** English
☐ **Monograph**	☒ **Article dissertation (summary + original articles)**

Abstract

Smart devices equipped with multiple network interfaces are becoming commonplace. However, even when multiple interfaces are successfully connected to the Internet, traditional TCP/IP typically continues to use only a single default interface for data transmission. It has become one of the unsolved issues in today's Internet to improve performance and robustness by aggregating multiple network paths simultaneously.

This thesis investigates how to improve the efficiency of multipath transport protocols. In multipath transmission, for instance, one of the key challenges is to effectively multiplex paths with mismatched characteristics in terms of bandwidths, delays, and loss rates. The first solution is to utilize the redundancy of systematic packet coding so as to compensate for missing packets with a light overhead. This redundancy is updated in a timely fashion according to the dynamic path characteristics, and a pre-blocking warning mechanism is triggered in the case of proactive redundancy underestimation. In the second solution, the timeout impact on aggregate goodput and receive buffers is analyzed. Moreover, the minimum timeout constraint is removed at the sender, while the delayed ACK function is reserved at the receiver. In data center networks, the investigation focuses on the sharing of network resources among multiple transport flows in the many-to-one traffic pattern. The solution mitigates the Incast collapse problem in both single and multi-homed topologies by using an equally weighted congestion control algorithm.

Keywords TCP, multipath transmission, packet coding, delayed ACK, data center

ISBN (printed) 978-952-60-5925-9	**ISBN (pdf)** 978-952-60-5926-6	
ISSN-L 1799-4934	**ISSN (printed)** 1799-4934	**ISSN (pdf)** 1799-4942
Location of publisher Helsinki	**Location of printing** Helsinki	**Year** 2014
Pages 143	**urn** http://urn.fi/URN:ISBN:978-952-60-5926-6	

Preface

I owe my deepest gratitude to my supervisor, Prof. Antti Ylä-Jääski, for letting me freely explore the paths of science under his guidance. He was always there to provide support for every step of the way towards the completion of this thesis.

I would like to thank my instructor, Dr. Matti Siekkinen, for his sound words of wisdom and encouragement when I was struggling with self-doubt. I would also like to express my gratitude to my instructor, Dr. Andrey Lukyanenko, for guiding my research work. Andrey Lukyanenko was always there to listen to me and to give me advice. His patience and support helped me to overcome many challenges and finish this thesis. I am grateful to him for the long discussions that helped me sort out my work. I am also grateful to Prof. Sasu Tarkoma for sharing his expertise, and for his help during my whole research and writing of this thesis. In addition, I would like to thank Prof. Yong Cui for co-authoring the publications included in this thesis and for hosting my visit to Tsinghua University.

Thanks to Prof. Xiaoming Fu and Prof. Petri Mähönen for their rigorous pre-examination. Their comments helped me improve the quality of this thesis. Thanks to Mr. Edward Bonney for his help in polishing the language of this thesis.

My thanks go to Ramya Sri Kalyanaraman for giving me her comments for my first presentation in public. A (special) thank you to Xiaolai You, now my wife, for her company, support and encouragement. She has been of enormous help to me in my research and always ready to stand by my side in facing life's problems.

I owe thanks to the former and current staff of the Data Communications Software group for their help during my doctoral study, especially to Miika Komu, Pin Nie, Zhihua Jin, Zhonghong Ou, Yu Xiao, Prof. Tuomas

Aura, Dmitrij Lagutin and Mohammad Hoque.

I am also grateful to the secretaries and system administrators at our department for creating a great working environment. I would especially like to thank Soili Adolfsson and Kristiina Hallaselkä for their friendly support in taking care of all the important day-to-day matters during my work.

I appreciate the financial support from the Future Internet Graduate School (FIGS), China Scholarship Council, and the project of Cooperative Heterogeneous Mobile and Mesh Networks.

Thanks to all the teachers who cultivated my interest in learning and encouraged me to create. Thanks to all my friends. Without them life would have been much more difficult during the dark times in my life. Lastly, and most importantly, I wish to thank my parents for their endless love. To them I dedicate this thesis.

Helsinki, October 24, 2014,

Ming Li

Contents

List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.

I Ming Li, Andrey Lukyanenko, and Yong Cui. Network coding based multipath TCP. In *Proceedings of the 15th IEEE Global Internet Symposium (INFOCOM '12 Workshop)*, Orlando, FL USA, pp:25-30, March 2012.

II Ming Li, Andrey Lukyanenko, Sasu Tarkoma, Yong Cui and Antti Ylä-Jääski. Tolerating Path Heterogeneity in Multipath TCP with Bounded Receive Buffers. *Computer Networks*, ISSN 1389-1286, Volume 64, pp:1-14, February 2014.

III Ming Li, Andrey Lukyanenko, Sasu Tarkoma and Antti Ylä-Jääski. The Delayed ACK Evolution in MPTCP. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM '13)*, Atlanta, GA USA, pp:2304-2310, December 2013.

IV Ming Li, Andrey Lukyanenko, Sasu Tarkoma and Antti Ylä-Jääski. Efficient New Delayed ACK for TCP: Old Problem, New Insight. In *Proceedings of the ACM Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems (MSWiM '13)*, Barcelona, Spain, pp:355-364, November 2013.

V Ming Li, Andrey Lukyanenko, Sasu Tarkoma and Antti Ylä-Jääski. MPTCP Incast in Data Center Networks. *China Communications fea-*

ture topic on Cloud Computing, ISSN 1673-5447, Volume 11, No.4, pp:25-37, April 2014.

Author's Contribution

Publication I: "Network coding based multipath TCP"

The author of this thesis is the primary author of this publication. He proposed the original idea of integrating packet coding into MPTCP, designed the scheduling algorithm, and did the simulation for evaluation. The second and third authors participated the joint discussions on the content and structure of the publication.

Publication II: "Tolerating Path Heterogeneity in Multipath TCP with Bounded Receive Buffers"

The author of this thesis is the primary author of this publication. He proposed the original idea for the problem settings, designed the scheduling algorithms as well as the packet coding method, and did the simulation for evaluation. The second author did part of the scheduling algorithm design. The third and fourth authors collected the essential references. The fifth author scrutinized the publication in general.

Publication III: "The Delayed ACK Evolution in MPTCP"

The author of this thesis is the primary author of this publication. He proposed the original idea for the problem settings, designed the scheduling algorithm as well as the packet coding method, and did the simulation for evaluation. The second and third authors participated the joint discussions on the content and structure of the publication. The fourth author scrutinized the publication in general.

Publication IV: "Efficient New Delayed ACK for TCP: Old Problem, New Insight"

The author of this thesis is the primary author of this publication. He proposed the original idea for the problem settings, designed the scheduling algorithm as well as the packet coding method, and did the simulation for evaluation. The second and third authors participated the joint discussions on the content and structure of the publication. The fourth author scrutinized the publication in general.

Publication V: "MPTCP Incast in Data Center Networks"

The author of this thesis is the primary author of this publication. He proposed the original idea for the problem settings, designed the congestion control algorithm, and did the simulation for evaluation. The second and third authors participated the joint discussions on the content and structure of the publication. The fourth author scrutinized the publication in general.

List of Abbreviations

ACKs	Acknowledgements
AP	Access Points
APIs	Application Programming Interfaces
CCC	Coupled Congestion Control
cmpTCP	Concurrent Multipath TCP
CMT-SCTP	Concurrent Multipath Transfer SCTP
DA	Delayed ACK Mechanism
DACK	Delayed ACK Timer
DCN	Data Center Network
DCTCP	Data Center TCP
DSS	Data Sequence Signal
DWC	Dynamic Window Coupling
ECN	Explicit Congestion Notification
EDPF	Earliest Delivery Path First
EW-MPTCP	Equally-weighted MPTCP
FEC	Forward Error Coding
FIFO	First In, First Out
FTP	File Transfer Protocol
HCF	Hashed Credits Fair
HIP	Host Identity Protocol

HLB	Head-of-line Blocking
HoL	Head-of-line
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
ISDN	Integrated Services for Digital Network
ISP	Internet Service Provider
LAN	Local Area Network
LQB	Link Quality Balancing
MAC	Media Access Control
mHIP	Multipath HIP
MPTCP	Multipath TCP
MSS	Maximum Segment Size
MTU	Maximum transmission Unit
NDA	New Delayed ACK Mechanism
OLIA	Opportunistic Linked Increase Algorithm
OWTT	One-Way-Trip Time
QoS	Quality of Service
RP	Resource Pooling
RTO	Retransmission Timeout
RTO_{min}	Minimum RTO
RTP	Realtime Transport Protocol
RTT	Round Trip Time
SC-MPTCP	Systematic Coding Multipath TCP
SCTP	Stream Control Transmission Protocol
TOR	Top-of-rack Switch
WWAN	Wide-area Wireless Access Network
WWW	World Wide Web

List of Tables

List of Figures

1. Introduction and Motivation

One of the current unresolved issues in the Internet is the unavailability of simultaneous utilization of multiple paths for data transmission. The Internet consists of interconnected computer networks linked by various mediums, such as wired lines, fiber-optic cables, and wireless links. Although it was originally designed to provide multiple paths between the endpoints for the sake of resilience, hosts had a single interface, and only routers were equipped with several physical interfaces in the early days of the Internet. However, the Internet has since then evolved greatly; for example, most hosts have more than one interface, and the proliferation of mobile devices equipped with 3G, 4G, and WiFi brings with it a growing number of multi-homed hosts onto the Internet. End-users may benefit from increased redundancy and performance from using such multi-homed hosts. Unfortunately, in practice this is not always the case because TCP, which binds each connection to a single interface, carries the majority of Internet traffic. This implies that TCP by itself is incapable of utilizing the extra resources over multiple interfaces.

The idea of enabling multipath transmission, i.e. concurrently transmitting data over multiple paths between two end hosts on the Internet is not new. To the best of my knowledge, it was originally considered by Dr. Maxemchuck in his Ph.D. dissertation [88] in 1975, right after the birth of the Internet. The principal idea has been reinvented a dozen of times since then. An important issue is whether a connection that is split across multiple interfaces can indeed provide the bandwidth aggregation. If data is split across different paths, the packets within the same TCP connection may arrive at the receiver out of order. The receiver will falsely indicate packet losses to the sender, which can lead to unnecessary retransmissions and a substantial reduction of sending rate caused by congestion control.

Earlier research has tried to solve the problem from various viewpoints. In the beginning, most of the focus was on utilizing multiple TCP flows to aggregate bandwidth with an additional mechanism to handle the reordering issue at the receiver end. It was expected that the bandwidth aggregation could henceforth potentially multiply the experienced throughput by the number of available paths. If efficient bandwidth aggregation could be achieved in this way, a multi-homed device could obtain a significantly better performance, especially when the bandwidth across the different paths had similar Quality of Service (QoS) characteristics. However, these approaches did not perform well in the context of heterogeneous paths, where the packet reordering can negatively affect performance and strain the buffer requirements. To fix the reordering problem, various intelligent scheduling algorithms were proposed to address path dissimilarity in terms of capacity, delay, packet losses and queuing behavior.

Gradually, it became apparent that simply utilizing multiple TCP flows resulted in an unfair share of the bandwidth at the same bottleneck; for example, n TCP flows get approximately n times throughput as a single TCP flow does when they go through the same bottleneck. Therefore a solution to this would be for the bandwidth, shared by multiple TCP flows within the same connection, to be similar to a single TCP flow at the shared bottleneck link. Using congestion control to tune the available bandwidth on each path is a powerful scheme with which to achieve TCP-friendliness. A considerable amount of joint congestion control algorithms were proposed to improve the fairness of multiple subflows within the same connection. In addition to TCP-friendliness, the load balancing feature, which moves traffic from more congested paths to less congested ones, was also explored.

Although many attempts were made to provision multipath capability, none of them made it into the mainstream approach as they all required non-trivial changes to the Internet infrastructure. As a result there was insufficient motivation to merit such changes to the Internet. The breakthrough came from Multipath TCP (MPTCP) [44], which draws the experience gathered in previous work, and goes further in solving issues of compatibility, fairness and load balancing.

In addition to being deployed on the Internet, MPTCP has also only recently found use in data centers for carrying internal traffic. Data centers provide a cost-effective way for computing and storage. Indeed, a data center may host thousands of servers interconnected through many switches and routers. Recent growth in cloud applications from well-known companies, such as Google, Microsoft, and Amazon, has resulted in the construction of data centers of unprecedented size. Data centers are constantly evolving in order to process large amounts of traffic with guaranteed bandwidths and short delays. With data center traffic becoming less client-server and more server-server centric, new topologies are emerging, for example, where the servers are multi-homed to provide more redundant paths between communication peers. In these topologies, TCP falls short because it fails to use multiple interfaces at the same time. Utilizing MPTCP to replace TCP is a promising way to offer bandwidth aggregation and traffic load balancing.

However, MPTCP faces the same challenges as TCP in a certain communication pattern produced by some applications, such as search engines and Map-reduce [30] applications. These applications generate a large amount of data in a many-to-one traffic pattern, which may cause Incast effects as many servers reply to a single inquiry and sent data simultaneously to the server that initiated the inquiry. The consequence of the Incast effect is a significantly degraded performance because when multiple replies compete for the same bottleneck, heavy congestion will cause unacceptable delays to the inquiry, undermining the advantage gained by using multipath transmission. This poses a new challenge to the existing congestion control mechanism of MPTCP. It either needs to be revised specifically for a data center context or other mechanisms need to be explored to avoid timeout by intricate competition among the multiple flows within the same connection.

1.1 Research Question, Scope and Methodology

Previous work in the field has contributed a lot to the evolution of multipath transmission. However, new challenges are posed by the revolutionary development of the Internet and data center networks (DCNs). Hence, in this thesis, the following questions are asked: what are the challenges of multipath transmission and what would be potential solutions for the existing and upcoming challenges?

The magnitude of these questions is very large. Therefore, the research scope is limited to a network with heterogeneous paths (in terms of RTT, packet loss, and available bandwidth) and to data center networking.

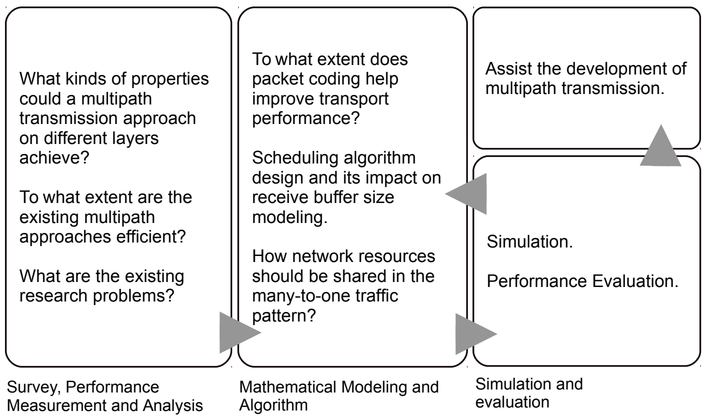

What kinds of properties could a multipath transmission approach on different layers achieve?
To what extent are the existing multipath approaches efficient?
What are the existing research problems?
To what extent does packet coding help improve transport performance?
Scheduling algorithm design and its impact on receive buffer size modeling.
How network resources should be shared in the many-to-one traffic pattern?
Assist the development of multipath transmission.
Simulation.
Performance Evaluation.
Survey, Performance Measurement and Analysis
Mathematical Modeling and Algorithm
Simulation and evaluation

Figure 1.1. Research methodology used in this thesis.

Fig. 1.1 gives an overview of the methodology used in this thesis. As shown in the figure, the research process starts from survey, measurement, and analysis in order to figure out the existing research problems. After that, key algorithms and mathematical models are designed to improve the transport performance. Specifically, we look into the following three questions:

- Question 1: what is the benefit of combining packet coding with MPTCP?
- Question 2: how the Delayed ACK mechanism should evolve in order to mitigate the impact of TCP timeout on overall performance?
- Question 3: how to adapt the congestion control algorithm to deal with the Incast effect in data center networks?

These questions are respectively discussed in detail as follows.

1) MPTCP [44] uses a connection level receive buffer as an aggregate buffer to store out-of-order data. In the context of multipath transmission, the aggregate buffer has to be large enough to accommodate all received packets until the lost or delayed packets arrive. When the paths have a very different path quality in terms of bandwidth, Round Trip Time (RTT), and packet loss ratio, the required aggregate buffer becomes too large to be possible in practice, especially when one path enters Retransmission Timeout (RTO) and the other paths send data at their full speed. If the receive buffer is bound, MPTCP may suffer from the degradation of goodput. The goodput can even be worse than that

of one regular TCP flow, undermining the advantage gained by using multipath transmission. Many error-correction coding approaches have been proposed to recover from packet losses or unexpected packet delays. The existing approaches, however, have not gone deeply enough into the impact of the coding design on computational overhead and required buffer size. For example, most of them use non-systematic coding, in which the output does not contain the input symbols. We seek to understand and reveal the impact of various coding algorithms on the MPTCP performance and also to model the receive buffer size accordingly.

2) The Delayed ACK (DA) [18] is an option of TCP that allows the receiver to delay sending an ACK for every other packet until the Delayed ACK timer is reached. At the sender, the timeout timer should be no less than the Minimum RTO (RTO_{min}) to avoid spurious timeouts. In the gradual changeover from TCP to MPTCP as the dominant protocol in the network, the Delayed ACK is an inherited feature and this can lead to significant performance degradation. For example, when a subflow enters a timeout, the receiver has to buffer data from all the other subflows until the missing packet is received. The out-of-order data may overrun the receive buffer to cause flow control at the sender, which seriously impacts the overall performance. The research goal is to investigate whether it is possible to keep the delayed ACK function while eliminating the negative effect of RTO_{min}.

3) In recent years, MPTCP has been proposed as a replacement for TCP in multi-homed DCNs because it can efficiently offer improved throughput and better fairness [109, 112]. However, it was found that MPTCP has a problem in terms of Incast collapse, where the receiver suffers drastic goodput drop when it simultaneously requests data over multiple servers. Therefore, we conducted an investigation into how network resources should be shared among MPTCP connections in the many-to-one communication pattern, and to what extent the new congestion control algorithm could mitigate the Incast effect.

In the methodology, an experimental approach through simulations was used to study, validate and improve the solutions. The refining process iterates until the desired performance results are obtained. Recent research on network simulators suggests that NS-3 [1] is one of the fastest and most efficient for large scale networks. NS-3 supports all the principal protocols needed and is more efficient than real implementation in terms of making protocol extensions and applying new algorithms. However, using a MPTCP NS-3 simulator poses a challenge in applying the mathematical models and algorithms to real devices and networks other than the simulated ones that have been used. This is because the protocol complexity, parameter settings and traffic pattern may be different between the simulation and the real world. For example, it is relatively easier in the NS-3 simulator than in the Linux kernel to integrate packet coding capability into MPTCP. Moreover, unlike in Linux kernel, the TCP inside NS-3 cannot automatically adjust the buffer size based on requirement. In practice, the limitations of simulator are unavoidable but can be alleviated by using the most common parameter settings that are used in real implementation, comparing the results with those obtained on real networks, and also validating the simulation results with theory.

The reader should note that the focus of this thesis leaves out the following important aspects of multipath transmission.

a) Multipath routing [39, 68, 84] for enhancing resilient end-to-end communication and optimizing routing is beyond the scope of this work. The focus is on how to provide multipath transmission capability to endpoints with an assumption that the IP routing protocol is able to provide multiple routes between the communicating peers.

b) Although packet coding techniques have been used widely in this work, the computational overhead caused by packet coding is not addressed because there has been a wealth of work that utilizes dedicated hardware or multi-core GPUs to accelerate the packet (de)coding speed. The reader should refer to [26, 125, 126] for more information about the packet (de)coding algorithms and computational efficiency.

c) Multihoming support for partial path failure [94, 96] is beyond the scope of this work. Generally speaking, multihoming is the first step towards multipath transmission because it offers the endpoints the capability of "knowing" multiple interfaces and is able to activate a certain interface to send data. However, this thesis focuses on simultaneously utilizing not just one, but more than one interface at the same time.

1.2 Contributions

This thesis is a summary of five publications, where Publication I and Publication II address the first research question, Publication III and Publication IV answer the second research question, and Publication V presents a solution to the third research question. The contributions of these publications are briefly described below. More detailed discussion can be found in Chapter 3.

Publication I proposes integrating packet coding techniques into the MPTCP in order to mitigate the reordering issue at the receiver. At the core of the scheme are the packet coding and scheduling algorithms, which utilize both the regular and packet coding subflows. The regular subflows deliver original packets, while the packet coding subflows deliver linear combinations of the original packets.

Publication II presents how linear systematic packet coding design improves the performance of MPTCP on heterogeneous paths when the receive buffer is bounded. The scheduling algorithms are used to provision proactive and reactive redundancy to counter against expensive retransmissions. A mathematical model of the receive buffer size is made accordingly.

Publication III provides a new Delayed ACK mechanism for MPTCP to mitigate the reordering issue during timeout at the receiver. The goal is to remove the RTO_{min} constraint at the sender while reserving the delayed ACK function at the receiver. In order to eliminate the aggressiveness of timeout after removing RTO_{min}, a packet coding algorithm is designed to encode the timeout retransmitted packets to make the potential spurious retransmissions useful.

Publication IV extends the new Delayed ACK mechanism to general TCP to improve the bandwidth for each flow, and aggregate bandwidth for all flows.

Publication V discusses how the network resources should be shared in the many-to-one communication pattern in data center networks, and proposes a new congestion control algorithm for MPTCP. The goal is to mitigate Incast collapse by allowing multiple MPTCP subflows to compete fairly with a single-TCP flow at the shared bottlenecks.

1.3 Structure of the Thesis

In Chapter 2, the essential background is reviewed. After that, the main contributions of this thesis are discussed in Chapter 3. The original papers are presented after a conclusion in Chapter 4.

2. Background: Network Transport

This chapter presents the background of multipath transmission that is essential for understanding the central arguments of this work. I start with an overview of TCP/IP networks and DCNs in Section 2.1. In Section 2.2, the existing approaches of multipath transmission are classified into five categories, and the design criteria of each category is analysed. Section 2.3 presents the architecture of MPTCP. What follows after that in Section 2.4 is a discussion of packet coding in network transportation.

2.1 TCP/IP Networks

The Internet is a global system of interconnected computer networks that uses the standard Internet protocol suite TCP/IP [150] to serve several billion users worldwide. It carries an extensive range of information resources and services such as the World Wide Web (WWW) [16], email, and peer-to-peer applications [11]. The origin of the Internet reaches back to research commissioned by the United States government in the 1960s in order to build robust and fault tolerant communication via computer networks. In the early days of the Internet, around the 1970s, efforts were undertaken to ensure the reliability of communications. Neither mobile devices nor computers with multiple network interfaces were an immediate design priority, and only the routers were equipped with several physical network interfaces. After that, TCP/IP has been used by the vast majority of applications to transport their data reliably across the Internet.

Although there may be more than one path between two endpoints, the Internet, for example, may potentially contain thousands of paths between two endpoints. TCP/IP, however, always uses a single "best" path[1].

[1]A single path is selected according to a certain routing metric.

Once the TCP connection has been established, it is bound to the IP addresses of two communicating hosts. If one of these addresses changes, for whatever reason, the connection would fail. In fact, a TCP connection cannot even be load-balanced across more than one path if the paths have different QoS characteristics because this may result in packet reordering, and TCP misinterprets this reordering as a signal of congestion and slows down.

In the past, end-users used to be connected to one Internet Service Provider (ISP) only. The Internet is changing, though. For example, multihoming tends to be present everywhere: mobile devices are now equipped with 3G/LTE and WiFi interfaces, multi-homed topologies have been proposed in data centers to provide many redundant paths, and multihoming has become commonplace for the server farms. These multi-interface devices require multipath capability to improve end-to-end communication performance and resilience. Unfortunately, in practice this is a big challenge because more than 95% of Internet traffic is driven by TCP [80], which binds each connection to a single interface even though endpoints have nowadays evolved to have multiple network interfaces. There is a mismatch between single-path transport and the multitude of available network paths. This stands in contrast to the trend that more and more endpoints are becoming multi-homed and there is a demand to take advantage of multiple access interfaces. As a result, the TCP/IP's single-path design is becoming evermore ill-suited to take advantage of multipath transmission.

During the last decade, data centers have risen to dominate the computing landscape on the Internet. They provide a cost efficient solution for provisioning a wide range of computing resources in diverse environments such as business, science, and mobile computing. Data centers in big companies such as Google, Microsoft, and Amazon, have resulted in the construction of data centers of unprecedented size. Multipath transmission is also a challenge for DCNs. For example, in a DCN, many paths are available between two servers, and a routing protocol may pick one according to a certain metric for a particular TCP connection. That can cause collisions where multiple flows go through the same link, affecting throughput to such an extent that the average throughput is halved. In order to offer large bandwidths and short delays, new data center topologies, such as multi-homed topologies, have been proposed to take advantage of multipath transmission.

2.2 Multipath Transmission

The earliest reference to concurrent multipath transmission, referred to as dispersity routing, was in Dr. Maxemchuck's Ph.D. dissertation [88] in 1975. After that a significant amount of research on multipath transmission has been published. In 2006, Shakkottai et al. [122] considered the scenario where users split their traffic amongst all the available IEEE 802.11 access points (APs). In a non-cooperative pricing game, they showed that multihoming outperforms unihoming, both in terms of throughput as well as profit to the ISPs.

As a first step towards multipath transmission, some early approaches have been proposed for performing vertical handoffs from one network to another as a mobile user migrates across the coverage areas [134]. When the coverage areas of two networks overlap, end users expect to have provided access through the higher data rate connection. In this thesis, a similar scenario where a host has multiple active network interfaces is considered. For example, a host is provided access through all of its interfaces instead of only one interface.

In this regard, striping is the principal technique used for aggregating resources through multiple interfaces to obtain higher performance [140]. The striping itself is not specific to a certain protocol layer, but is a common technique which could be utilized by all layers. Many striping approaches were proposed for multipath transmission on various protocol layers. In the remainder of this section, related work is classified according to which of the layers of the protocol stack the proposed approaches perform at: link layer, network layer, transport layer, application layer and session layer.

2.2.1 Link Layer

At the link layer, multipath transmission is typically called bonding or bundling because multiple physical channels are bundled into a single logical channel. The primary goal of link layer bundling is to coordinate multiple independent links between a fixed pair of systems, providing a virtual link with a larger bandwidth than any one individual link.

Multilink PPP (MP) [132], designed for Integrated Services for Digital Network (ISDN), aggregates links using the PPP protocol [129]. MP did not specify any scheduling strategy but suggested that data fragments could be distributed proportional to the links' transmission rates. In order to detect fragment loss and disorder, MP used a 4-byte sequencing header for synchronization and detecting lost fragments at the receiver. Therefore, a reorder buffer was required at the receiver to accommodate the out-of-order fragments. Adiseshu et al. [5] added a strIPe layer, a virtual IP interface below the IP layer and above the data link layer, to aggregate multiple data links. The strIPe layer implemented the sender side striping algorithm and the receiver side resequencing algorithm. They also proposed a family of efficient channel striping algorithms that solved both the variable packet size and the First In, First Out (FIFO) delivery problems. Another technique, Link Aggregation [2], allows multiple links to be aggregated together to form a link aggregation group so that a MAC (Media Access Control) client could treat the link aggregation group as if it were a single link. This mechanism allows load balancing of available links in bridged Local Area Network (LAN) environments, along with improved resilience in the face of the failure of individual links. In addition, the Frame Distributor is responsible for maintaining any frame ordering constraints. Therefore, there is no requirement for the Frame Collector to perform any reordering of frames received from multiple links.

An example of wireless channel bonding was discussed by Snoeren et al. in [133], in which they proposed LQB (Link Quality Balancing), a scheme that bundles multiple channels of the same Wide-area Wireless Access Network (WWAN) technology. In order to equalize the transmission time of different links, LQB adapts the fragment size to the current effective throughput of each link. A link layer receive buffer is also required to reorder fragments. Another example of wireless communication is Fat-VAP [70], aggregating the bandwidth available at accessible APs and also balancing their loads.

The main advantage of link-layer bonding is that the signalling rate of the channel is relatively stable and can be used to mitigate reordering. However, link layer approaches only work on a point-to-point link that connects two devices. Therefore, they are not applicable in a general scenario of end-to-end communications.

2.2.2 IP Layer

The IP layer, originally proposed to handle global addressing and routing, seems a natural candidate to host the multipath capability for enhancing end-to-end communication. In theory, each packet of a TCP flow could be sent over a different path, and the IP protocol would ensure that all packets reach their destination. But, in practice, when packets inside one connection take two or more paths through a network, they could experience different propagation delays and arrive out of order. The TCP receiver sends duplicate acknowledgements to the sender so that the TCP sender's fast retransmission (or even timeout retransmission) mechanism will often mistake packet reordering for packet loss and reduce its offered load to an unacceptably lower level. Kim et al. [75, 76] showed that the TCP suffers significant performance degradation due to frequent packet reordering, especially in a wireless environment. Therefore, the use of multiple paths with varying characteristics poses challenges in the form of reordering for all IP layer approaches.

A frequently used IP layer approach for aggregating bandwidth of multiple IP paths is to use tunnelling mechanisms for transparently redirecting packets from the server or a proxy to all IP addresses at the client. For example, Phatak et al. [102, 103] proposed using IP-in-IP encapsulation (also used for tunnelling in the mobile IP standard [99]) to distribute IP packets across multiple network interfaces. In order to avoid fast retransmission, they used an out-of-order sending scheduler, which distributed packets proportionally to the paths' effective rate. Chebrolu et al. [21, 22] presented a network layer architecture to aggregate bandwidth on multiple paths for real-time applications. They assumed an infrastructure proxy. The proxy (like the Home-Agent in Mobile IP [100]) was aware of the multiple interfaces of the client, and tunnelled the captured packets to the client using IP-in-IP encapsulation. In order to minimize the reordering effect, they used a scheduling algorithm, Earliest Delivery Path First (EDPF), which estimated the delivery time of the packets on each path, and scheduled each packet onto the path that delivered it the earliest. Kim et al. [75, 76] introduced PRISM, a proxy based inverse multiplexer that enabled TCP to efficiently utilize the community members' WWAN connections. PRISM striped each TCP flow over multiple paths. It manipulated the transport-layer acknowledgements (ACKs) so as to appear in order to the sender so that PRISM was able to mask a variety of adverse effects specific to each link via an intelligent ACK-control mechanism. Most IP layer approaches are transparent to upper layers, but PRISM [75, 76] is an IP layer approach that explores the transport-layer feedback mechanism. Thus, PRISM could be treated as a cross-layer approach.

Host Identity Protocol (HIP) [89] and shim6 [95] have been proposed and implemented to provide multihoming support for failover with the possibility of flow-based load balancing. Both HIP and shim6 introduced an additional addressing layer to allow changing IP addresses on network interfaces, while keeping constant transport-layer identifiers. Those protocols enabled IP packet flows to dynamically change paths in the presence of link failure. However, they do not support simultaneous multipath transmission without additional extensions. SIMA [104] was an extension of HIP to use multihoming for assigning separate TCP connections independently to different paths. Gurtov et al. in [49] designed and implemented Multipath HIP (mHIP), a multipath scheduler based on HIP, to distribute the incoming traffic among multiple available paths. Utilizing a Fastest Path First scheduling algorithm (a variation of the EDPF algorithm), they striped packets within a TCP connection to multiple paths with minimal reordering. Unlike other IP layer approaches, mHIP applied a congestion avoidance algorithm to redirect traffic from the more congested paths to the less congested paths. Polishchuk et al. [105] proposed a TCP-friendly congestion control algorithm for mHIP to prevent the stealing of bandwidth from legacy TCP flows at the shared bottlenecks. The concept of joint congestion control algorithm adopted in [49] and [105] is also used by some transport layer approaches (they will be discussed later). However, the reordering and congestion avoidance algorithms used on the IP layer (or between IP and TCP like HIP [89]) have insufficient information (e.g., congestion status and RTT) and need to repeatedly design additional mechanisms which have already been on the transport layer.

As discussed earlier, reordering at the receiver is the main challenge for all IP layer approaches. These approaches use various scheduling algorithms to minimize the reordering effect. We summarize the scheduling algorithms used in the IP layer approaches and find three primary scheduling strategies. The first is the fastest path first scheduling strategy. This strategy was commonly used in various approaches such as [21, 22] and [49]. The second one is to schedule packets proportionally to the paths' rate, such as [102, 103]. This scheduling strategy is also used in the link layer, for example, MP [132]. The third one is to mask the reordering effect by using the fake TCP ACKs [75, 76].

2.2.3 Transport Layer

Compared with IP layer approaches, transport layer approaches can enjoy maximum benefit because congestion control can be used as a mechanism for resource allocation in a network. At this layer, end-systems can easily obtain information about each path: capability, latency, loss rate and congestion state. This information can then be used to react to congestion in the network by moving traffic away from congested paths.

Numerous attempts have been made to tune existing transport protocols, Stream Control Transmission Protocol (SCTP) [135] and TCP, for multipath capability. SCTP was designed with multihoming in mind although it used only one primary path at the same time and switched to another path for retransmission of lost data or as a backup in the case of primary path failure. Several SCTP extensions, such as Concurrent Multipath Transfer SCTP (CMT-SCTP) [10, 66, 67, 85], cmpSCTP [83], WiMP-SCTP [63] and LS-SCTP [3], enabled SCTP to simultaneously transmit data over multiple paths. CMT-SCTP [10, 66, 67, 85] has a separate buffer to guarantee path independence. This design preserves the TCP-friendliness of each flow under the assumption that no bottleneck is shared by multiple paths. CMT-SCTP proposed a few algorithms which augmented and/or modified the current SCTP to counter the negative side-effects of reordering at the receiver. Moreover, CMT-SCTP offers a few retransmission policies for lost packets. cmpSCTP [83] distributes data over the available paths through real-time path monitoring. It has a separate congestion control for each path so as to ensure fair integration with other traffic in the network. WiMP-SCTP [63] offers two transmission modes: “Data-striping Mode” and “Data-duplicating Mode” for a different wireless link status. When the link status is good, the “Data-striping Mode” is selected to aggregate bandwidth. On the other hand, when the network status is bad, the “Data-duplicating Mode” is selected to increase destination reachability. LS-SCTP [3] proposed separating the flow control and congestion control. The congestion control is performed per path, whereas the flow control is performed per connection. It allows the sender to schedule data on certain paths according to bandwidth estimation.

There has been a considerable amount of academic work on multipath SCTP based on CMT-SCTP. For example, Dreibholz and Adhari et al. [4, 37, 38] examined the challenges of CMT-SCTP over asymmetric paths. They identified the issue of sender and receiver queue blocking as a problem that could lead to poor performance. In order to improve the performance, Dreibholz [37] and Adhari et al. [4] proposed a few mechanisms accordingly, such as buffer splitting, chunk rescheduling, and smart fast retransmission. And Dreibholz et al. [38] also proposed using the multistreaming feature of SCTP. That particular feature requires a scheduler on both sides to decide the order in which packets on different paths are sent and delivered to the application. Budzisz et al. [20] proposed an mSCTP-CMT protocol to investigate the applicability of CMT-SCTP for distributing data between two paths under the same association (connection) during the handover transition process. They emphasized the consequence of a sender-introduced reordering and its effect on congestion control. The authors found that in CMT-SCTP, the receive buffer may be filled with out-of-order data caused by complete or short-term failures during handover. They solved this problem by allowing no further data transmission on the path which experiences a single TCP timeout.

In contrast to SCTP, which was designed with multihoming support, TCP is unaware of multiple interfaces and allows only a single IP address per endpoint. However, TCP has dominated Internet traffic, which has sparked a lot of interest in enabling TCP to support simultaneous multipath transmission.

pTCP [61, 62] functions as a wrapper around a modified version of TCP. It opens multiple TCP flows, one for each interface in use. pTCP performs intelligent striping of data across the micro-flows (TCP flows), and does data reallocation to handle variations in the presence of heterogeneous path characteristics. R-MTP [86] is a rate-based reliable transport protocol. It relies on explicit bandwidth probing to estimate bandwidth in order to adjust the rate on the available paths accordingly. For example, it measures packet inter-arrival times and jitter to sense bandwidth scarcity. The probing period must occur on a fine time-scale to reflect the fluctuation of the available bandwidth. R^{2}CP [60, 77] is a receiver-centric transport protocol with a simple sender design. It allows a mobile host to have control over the reliable delivery of data from the sender over multiple paths. R^{2}CP schedules transmissions based on the congestion window and the round-trip-time of each connection to avoid out-of-order arrival at the receiver. Similar to the mSCTP-CMT protocol, R^{2}CP also supports handover. Lee et al. [81] investigated schemes to address the reordering issue in multipath transmission by tuning a few TCP parameters (e.g., increasing the fast retransmit threshold, enabling delayed ACKs, and making use of flow-aware routers). Chen et al. [23] proposed transmitting multiple copies of the same packet on different paths to combat extremely high packet loss rates. But the performance degrades sharply as the loss rate increases beyond 20%. In cTCP, Y. Dong et al. [33] proposed using a single congestion window to control the global throughput and a single sending buffer to be shared among all paths. Its scheduling algorithm is a Credit-Weighted Round-Robin which allowed a fair data distribution among the available paths. Sarkar [120] proposed Concurrent Multipath TCP (cmpTCP), which concurrently splits packets over all the paths from a shared sending buffer, with each path only maintaining a virtual retransmission buffer. Sarkar also developed a Markov model in cmpTCP for estimating the data transport rate on each path and then computed file transmission time by adding estimated data transport rates on all paths.

In the case of M/TCP, Rojviboonchai and Hitoshi [116] proposed implementing an alternative TCP option called the multi-route option to distinguish the routes of the connection. They used One-Way-Trip Time (OWTT) [118] at the sender to estimate the delay time of the forward path and reverse path separately in order to calculate RTO timers. By using the multi-route option and OWTT measurement, the endpoint can deal with the congestion control of each path with high accuracy. Rojviboonchai et al. [117] proposed R-M/TCP, which is a Rate-based M/TCP that performed congestion control in a rate-based and loss avoidance manner. Specifically, R-M/TCP estimates the queue length at the bottleneck link. If the queue length grows beyond a predefined threshold, the sender will recalculate a new congestion window to achieve a fair share at the bottleneck. Tsao and Sivakumar [141] investigated whether multiple interfaces having highly disparate bandwidths were worthy of being aggregated. They proposed three mechanisms for TCP acceleration. The principal idea was to receive TCP data segments over a comparably high-speed WiFi and return ACKs over a low-speed 3G link.

In accordance with the discussion above, it seems straightforward that multipath transmission is just for utilizing multiple TCP (or SCTP) flows with intelligent scheduling algorithms to avoid reordering at the receiver, even in the presence of heterogeneous paths. However, it was found that simply concurrently utilizing multiple TCP flows at a bottleneck link would result in a fairness issue, i.e. an unfair share of the bandwidth at a bottleneck link. For example, NewReno [55] is the most common TCP congestion control variant as it yields an equal share of the congested link. This equal share outcome of NewReno will result in an unfair share of the bandwidth if more than one TCP flow is active for a single MPTCP connection at the bottleneck link. Zhang et al. [151] proposed mTCP as a solution for the problems of fairness by avoiding establishing subflows across the same bottleneck. In CMT-SCTP, Iyengar et al. [66] assumed that the bottleneck queues on the end-to-end paths are independent. The strong assumptions made by Zhang et al. [151] and Iyengar et al [66] for mTCP and CMT-SCTP are not feasible in practice, however.

An important transport layer approach designed with the fairness property in mind is MPTCP [44], which is a major extension to TCP and allows a pair of hosts to use several paths to exchange the segments that carry the data from a single connection. Congestion control is used to guarantee fair resource allocation on multiple paths. Various congestion control algorithms based on the idea of Resource Pooling (RP) [148] have been proposed for MPTCP such as the Fully Coupled Congestion Control [113], (semi-) Coupled Congestion Control (CCC) [110, 149], Dynamic Window Coupling (DWC) [54], and Opportunistic Linked Increase Algorithm (OLIA) [74]. All these algorithms do not modify the Slow Start, Fast Retransmission and Fast Recovery phase of TCP, but only the congestion avoidance phase. The difference between these congestion control variants is discussed in [130]. The CCC has been adopted by the Internet Engineering Task Force (IETF) as the standard congestion control algorithm for MPTCP. The challenge of achieving fairness for legacy TCP flows is not specific to MPTCP, but it is also a challenge for SCTP and all other multipath transmission approaches. For example, congestion control algorithms for CMT-SCTP have been studied in CMT/RP-SCTP [36] and [34, 35]. The transport layer is not the only protocol position to solve the fairness issue; the IP layer approaches also exist [49, 105]. Unlike the IP layer approaches, which need to repeatedly add congestion detection/control mechanisms, TCP layer approaches are able to use these mechanisms directly.

Although SCTP shares the same issues with MPTCP in terms of fairness, reordering, and retransmission policies, moving legacy applications from TCP to SCTP involves a number of challenges such as making SCTP work through NATs, the need to modify applications, and the lack of an easy way to negotiate SCTP versus TCP between a client and a server. None of the issues are insurmountable, but together they make adoption of SCTP as a TCP alternative a challenge. The main difference between SCTP and MPTCP from a compatibility viewpoint lies in the interface they provide to the applications. Specifically, unlike SCTP which modifies the interfaces of legacy TCP to applications, MPTCP presents a single TCP interface to the application layer. This seemingly minor difference makes MPTCP compatible with all legacy applications. As such, the implementation of multipath in TCP, which dominates Internet traffic, is a much more attractive deployment strategy.

2.2.4 Application Layer

Provisioning multipath capability at the application layer has received a lot of attention because a solution that is almost independent of the underlying access technologies and network-layer routing is promising. It is a common practice that an application establishes multiple transport connections, binds them to different IP addresses, and distributes the data in proportion to the available path capacity over these connections.

In the early stage (1980s-2000s) of the research on application layer multipath transmission, the focus was on bandwidth aggregation using multiple TCP connections over the same physical path. For instance, File Transfer Protocol (FTP) [106] is able to establish several connections when a file transfer is initiated. The data to be transferred is divided into fixed-size segments and sent over a connection that is idle and able to transfer data. Allman et al. [8] developed a new application called XFTP that used multiple TCP connections for transferring data over long-fat links, like satellite links. GridFTP [69] is another extension of the FTP protocol implemented for bulk data transfer, where parallel TCP connections are created to increase the throughput in a bottleneck link. Hacker et al. [50] examined the effects of using parallel TCP flows to improve end-to-end network performance while avoiding congestion. They found that in the absence of congestion, the use of parallel TCP connections was equivalent to using a large Maximum Segment Size (MSS) on a single connection with the added benefit of reducing the negative effects of random packet loss. The above approaches aim at increasing application throughput by using multiple TCP connections through the same physical path. However, if they are used for striping data over different physical paths, the reordering issue at the receiver would render them inefficient because they have failed to consider the reordering issue caused by heterogeneous paths.

In the 2000s, researchers started to seek solutions to provide bandwidth aggregation over different physical paths. A simple approach to achieve this goal is to directly add support for multiple interfaces to a given application by opening multiple TCP sockets, one each for every active interface, and performing striping of data across different sockets. Golubchik et al. in [46] investigated the potential benefits of an application layer multipath streaming approach between a set of senders and a receiver. They found that multipath streaming exhibited better loss characteristics than single path streaming. The results were encouraging and could be used in guiding the design of multipath streaming systems. Given that popular files are often replicated on multiple servers, it becomes natural for clients to connect in parallel to several mirror servers to retrieve a file. Rodriguez and Biersack in [115] described a parallel access scheme that allows users to fetch different portions of a file from multiple servers at the same time and reassemble the file locally. Tullimas et al. [142] proposed MultiTCP, a receiver-driven TCP-based system, for multimedia streaming aiming at providing resilience against short term insufficient bandwidth by using multiple TCP connections for the same application. MultiTCP was a "smart" application that allowed the application to control the desired sending rate during congestion periods. Wang et al. [146, 147] proposed Dynamic MPath-Streaming (DMP), a scheme for live streaming over multiple wired TCP connections. No TCP modifications are required, but multiple TCP senders fetch packets with video data from a server that queued and simultaneously pushed them over multiple paths. Packets are assigned additional sequence numbers to ensure correct reassembly at the receiver. DMP is also a "smart" application layer approach because it observes the TCP send buffer scheduling data in proportion to the data rate of available paths. MRTP [87] is a multipath transmission extension to the Realtime Transport Protocol (RTP) [121] for real-time applications. MRTP is complementary to SCTP in supporting multimedia services by exploring multipath transport in ad hoc networks, where link bandwidth may fluctuate and paths are unreliable. An underlying multipath routing topology (multiple interfaces are not necessarily required) is assumed.

In addition to multipath approaches for specific applications, HTTP [42] with multipath capability is currently one of the most common protocols for streaming video on the Internet through multiple paths. Evensen et al. [40, 41] and Kaspar et al. [71, 72] described an HTTP-based method for downloading multimedia content simultaneously over multiple network interfaces.

Application layer approaches split a single file or byte stream into logical segments which are concurrently transmitted over different paths. These kind of approaches seem to be simple in the sense that the applications are in full control of the striping decisions with no requirement for any protocol changes at lower layers so that clients and servers can find an optimal way to collaborate. However, the complexity and overhead at the application layer are considerable. For example, an application-level sequence number has to be included in each of the application defined headers, and the application has to explicitly ensure that the application layer data units, which carry unique application-level sequence numbers, do not get fragmented during transmission. Moreover, a dedicated resequencing mechanism is required to reassemble the data at the receiver. In practice, different paths may have diverse characteristics, and the striping ratio may not exactly match the data rate ratio of different paths. A large receive buffer (on the application level) is required to accommodate the out-of-order data. Moreover, in order to split intelligently, the application has to redundantly implement a bandwidth estimation mechanism in spite of the same mechanism already having been performed by TCP through its congestion control mechanism.

2.2.5 Session Layer

Session-layer striping is an approach that tries to open multiple TCP flows without any changes to existing applications by providing specialized middleware or virtual sockets between the application layer and transport layer. Sivakumar et al. in [131] proposed PSockets (Parallel Sockets), a library that transparently partitions upper layer data into multiple transport streams through the same physical path. PSockets has the same Application Programming Interfaces (APIs) as that of a regular socket. The principal idea is to split data across several open sockets with no manual tuning. Qureshi et al. in [108] presented Tavarua, a middleware for providing network striping capability to applications with high demands on uplink throughput. ATLB [52, 53] consists of a distributed data transfer method and a path-failure detection/recovery method. ATLB calculates the data arrival time for each path, considering the time that data segments spend in the TCP queue at a sender and the time needed for data segments to pass through the network. ATLB enables in-order data delivery to a receiver to mitigate the reordering effect at the receiver. Like other session layer approaches, PATTHEL [13] also provides APIs for the application developers to be able to use it. The difference lies in the fact that first of all PATTHEL incorporates a separate data channel and a control channel, where the control channel is for managing the connection.

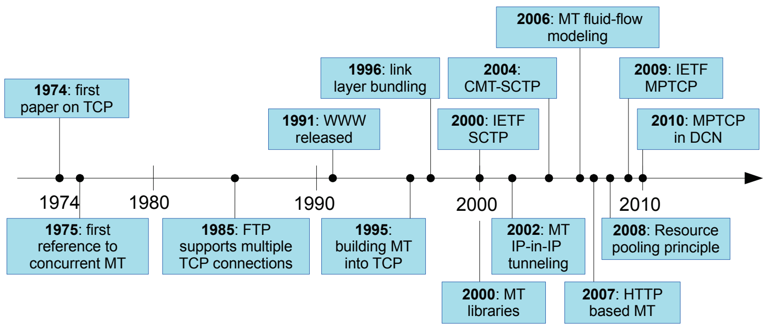

2006: MT fluid-flow modeling
1996: link layer bundling
2004: CMT-SCTP
2009: IETF MPTCP
1974: first paper on TCP
1991: WWW released
2000: IETF SCTP
2010: MPTCP in DCN
1974
1980
1990
2000
2010
1975: first reference to concurrent MT
1985: FTP supports multiple TCP connections
1995: building MT into TCP
2002: MT IP-in-IP tunneling
2008: Resource pooling principle
2000: MT libraries
2007: HTTP based MT

Figure 2.1. Milestones in the evolution of multipath transmission. MT: Multipath transmission

Second, PATTHEL is a cross-layer protocol because it adds an entrance to the routing table in order to deliver data over a certain channel.

Session-layer striping is very similar to application-layer striping. It also faces the same challenges that application layer striping does, for example, the reordering issue. But the good point is that although it still requires client and server-side modifications, it partly solves the compatibility issue faced by the application layer approaches.

2.2.6 Summary

The timeline of the important approaches for provisioning multipath transmission capability presented in this chapter are summarized in Fig. 2.1. The first paper on TCP was published in 1974. In the following year, multi-homed TCP [88] was proposed to concurrently transmit data over multiple paths. From that point onward, various forms of multipath transmission have been proposed. The idea of building multipath capability into TCP was, to the best of my knowledge, first suggested by Huitema [65] as an Internet draft in IETF in 1995. In 2006, Key et al. [73] used fluid-flow modeling to demonstrate that multipath transport can provide not only robustness but also balanced congestion in a stable manner. The latter is achieved with the right coupled congestion control algorithms. Their research provides sufficient incentives to enable multipath capability on the transport protocol. In 2008, Wischik et al. [148] investigated the resource pooling principle, which makes a collection of resources behave like a single pooled resource. This principle is a significant step towards a practical multipath-aware end system. From 2009, IETF started to define and standardize MPTCP, which utilized the CCC algorithm based on the partial resource pooling principle.

In the rest of this section, I first summarize the challenges of all the presented multipath transmission approaches and then evaluate the approaches from the perspective of the challenges. The first challenge the multipath transmission approaches must face is **compatibility**. A solution must require as few changes to the existing infrastructure as possible. Otherwise, it cannot be widely accepted in practice. The second challenge associated with all the multipath approaches that have been presented (excluding those running on the same physical path) is that of the **packet reordering** at the receiver. Given the dynamic nature of network paths, some amount of reordering is inevitable even on the same physical path. However, when packets are delivered over different paths, the packets may arrive at the receiver frequently out of order. Many repeated ACKs sent by the receiver to the sender will slow down the sending rate because of the misinterpretation of out-of-order packets. The packet reordering at the receiver may also cause the head-of-line blocking (HLB) problem (In Publication III, HoL blocking is used to refer to the same problem). When traffic between two hosts follows different paths, each of them may have different QoS characteristics and queuing behaviors. Therefore, packets may probably arrive at the peer out of order. These out-of-order packets must be accommodated at a buffer and reordered before being forwarded to upper layers. Large amount of out-of-order data may overflow the reordering buffer, causing the so-called HLB and leading to a significant degradation in the overall performance. In order to counter against HLB, Barré et al. [14] proposed disabling the under-performing paths when the out-of-order data reaches the maximum allowed resequencing buffer. However, it is always too late to disable the under-performing paths because the HLB may have happened already. The HLB problem is not used to evaluate the existing approaches because HLB is a common problem for all presented approaches no matter which layers they are located on. The third challenge is **fairness and load balancing**. In the early stage of multipath transmission research, most approaches emphasized bandwidth aggregation with scheduling and reordering algorithms. Few of them considered the fairness and load balancing issues. Today, these issues have become challenges along with the revolutionary development of multipath transmission.

It has been noted that currently there is no single multipath approach that is able to cover these challenges in a satisfactory manner. I now proceed to evaluate the multipath transmission approaches from the first three of the challenging perspectives mentioned above.

- Compatibility: the compatibility of an approach is evaluated by employing two metrics. The first is whether the approach is compatible with the legacy applications. Obviously, application layer approaches have a serious compatibility issue because the multipath transmission property needs to be implemented for specific applications. The session layer approaches seem the best because they keep the same APIs as a regular socket (e.g, PSockets [131]), thus requiring no changes to both the legacy protocols and applications at all. The second most compatible is MPTCP which keeps the same interface to the upper layers so that the multipath transmission capability is enabled for all applications transparently. Furthermore, the IP layer and link layer approaches are also compatible with the legacy applications because they do not change the interfaces between applications and transport protocols. The second evaluation metric is the sequence number design on the transport layer because it affects whether the "new" packets are able to traverse the legacy middle-boxes. Striping sequence numbers across two paths leaves gaps in the sequence space seen on any individual path. Some middle-boxes may block the packets carrying discontinuous sequence numbers [59, 111]. Session layer and application layer approaches obviously have no control over the choice of sequence numbers used on the transport layer. Therefore, they are not compatible with legacy middle-boxes. Those approaches are not the only ones that use a single sequence number space, and there are many other approaches that also adopt the same design, for example some of the transport layer approaches such as CMT-SCTP [66, 67], R-MTP [86] and R-M/TCP [117]. A viable solution is to use double sequence number spaces: per subflow sequence number is used to detect losses and drive retransmissions, while the connection-level sequence number is used to allow reordering at the receiver. Only few transport layer approaches use the double sequence number space design such as LS-SCTP [3], cmpSCTP [83] and MPTCP [44]. In summary, transport layer approaches (which employ the double sequence number space design) are the best approaches compatible with both applications and the network infrastructure.

- Reordering: when packets travel through different paths which may have mismatched characteristics, they may arrive at the destination out of order. All the presented approaches deal, to some extent, with the reordering issue on the layer which they are located at. If they ignore or have no control over the reordering mechanism on the transport layer, their approach may suffer from performance degradation because of the misinterpretation of out-of-order packets. For example, the approaches beneath the transport layer just simply strip data on multiple TCP subflows. TCP interprets out-of-order packets as lost and might, therefore, unnecessarily retransmit data and reduce its transmission rate. The double sequence number space design is a viable solution for solving this problem because the sequence numbers carried in the TCP headers are separate on each path so that the interpretation of out-of-order packets and ACKs remain the same as before.

Note that although the IP-in-IP encapsulation [102, 103] is also a kind of double sequence number design, the TCP process is unaware of the fact that the packets are tunnelled through another interface. Therefore, the receiver sees all packets coming from one path. Without careful consideration of the ACK mechanism, the TCP process would still misinterpret the out-of-order packets. Unlike the transport layer and IP layer approaches, the application layer approaches need no additional mechanism to solve the problem of misinterpretation of out-of-order packets because the applications use multiple standalone TCP flows to deliver data and use an application level sequence number space to resequence the packets coming from different paths. In summary, the transport layer and application layer are in the best position to solve the reordering issue.

- Fairness: fairness could also be seen as a compatible property with legacy TCP flows. The formal fairness requirement on multipath transmission was unclear in the beginning. For example, the early research on multipath transmission focused on bandwidth aggregation by taking advantage of the resources through multiple interfaces. The target in the research community matched the potential expectation of end users because an end user can benefit from the aggregated bandwidth if he or she has paid for both accesses. Thus, the fairness emphasized the fairness of each individual subflow; for example, each subflow gets as much bandwidth as a standalone TCP flow does. In recent years, the research focus was on the fairness of the multiple subflows as a whole at shared bottlenecks. The principle is that a multipath transmission should behave as a single TCP flow at shared bottlenecks [130]. Congestion control is used as a powerful tool to achieve aggressiveness control. For this reason, the transport layer is still in the best position because otherwise an approach is needed to implement the congestion control mechanism that has already been implemented on the transport layer (e.g., [49]).

Generally, transport-layer approaches best fit the requirements of today's multipath transmission in terms of compatibility, reordering, and fairness. However, transport layer approaches require significant changes to both endpoints and are, thus, disadvantaged in terms of getting broadly accepted and more quickly deployed in the near future. But approaches like MPTCP may prove themselves in the long term after successful standardization. Indeed, Bonaventure [17] found that in 2013, Apple Inc. enabled MPTCP for a specific application "SIRI" through which they control the server-side.

In the next section, the IETF MPTCP is discussed as a typical example of the multipath transmission protocol.

2.3 IETF MPTCP

The multipath transmission capability has received a lot of attention in the research community ever since the beginning of the Internet. Unfortunately, no multipath protocol has been implemented and adopted widely on the Internet because of compatibility, reordering, and fairness issues. In this section, an extensive overview of MPTCP is presented because our research on multipath transmission is mainly based on MPTCP.

During the past few years, MPTCP has remained an active field of interest for academia, with the IETF also currently showing considerable interest in the protocol. MPTCP aims to offer higher aggregate bandwidth and robustness by pooling multiple paths within one transport connection. For the design of MPTCP, the following set of goals have been identified for its acceptable operation [43]:

- Improved Throughput: MPTCP performs at least as well as a single TCP flow running on the best path.

- TCP-fairness: MPTCP is TCP friendly. For instance, it behaves as a single TCP flow at a shared bottleneck.

- Balance Congestion: MPTCP utilizes the least congested path the most.

- Middle-boxes: MPTCP can traverse various types of middle-boxes, such as NATs and firewalls, because each subflow is equivalent to a normal TCP connection.

- Compatibility: application developers do not need to change their applications to take advantage of MPTCP, which presents a signle TCP interface to upper layers.

The first goal is the prime performance incentive for developing MPTCP, the second goal guarantees bottleneck fairness, while the third goal adheres to the principle of RP. To achieve these first three goals, it is important that intelligent scheduling and congestion control algorithms are developed. The last two goals are very important from an implementation viewpoint because they allow MPTCP to be implemented in practice. Through extensive measurement, Hesmans et al. [56] and Honda et al. [59] found that MPTCP could traverse most of the middle-boxes. The MPTCP client could also fallback to regular TCP to preserve connectivity in the presence of certain middle-boxes or if the server is unaware of MPTCP [59, 111]. Moreover, to allow clients to benefit from MPTCP in its early deployment (e.g., servers have not upgraded to support MPTCP), Detal et al. [31] proposed a protocol converter, MIMBox, to translate MPTCP to TCP and vice versa.

An important difference between MPTCP and TCP is the congestion control algorithm. The standard TCP congestion control [9] yields an equal share of the congested link. It increases the congestion window upon reception of acknowledgements and decreases the slow-start threshold upon detection of losses. As mentioned in Section 2.2.3, MPTCP has adopted the CCC as its congestion control algorithm to update its congestion window according to the following principles:

- For each non-duplicate ACK on subflow i, increase the congestion window of the subflow i by min($\alpha/cwnd_{tot}$, $1/cwnd_i$)

- Upon detection of a loss on subflow i, decrease the subflow congestion window by $cwnd_i/2$

where $cwnd_{tot}$ is the total congestion window of all the subflows and α is a parameter which regulates the aggressiveness of the MPTCP connection to make it fair to TCP flows.

In Section 2.2.6, two sequence number designs adopted by multipath transmission approaches are summarized: single sequence number design and double sequence number design. MPTCP implements double sequence number spaces as follows. In MPTCP, each subflow is equivalent to a normal TCP flow with its legacy 32-bits sequence numbering space. This design allows MPTCP to be compatible with certain middle-boxes. For example, continuous TCP sequence numbers allow MPTCP packets to traverse sequence-number-checking firewalls, which check whether the sequence number is out of the valid range. In addition to the 32-bits sequence number space, MPTCP maintains a 64-bits connection-level sequence number space to reassemble the packets coming from different subflows. A Data Sequence Signal (DSS) option [44] specifies a full mapping from the connection-level sequence number to the subflow sequence number.

2.3.1 Reordering

As in the summary in Section 2.2.6, every multipath transmission approach needs to solve the reordering issue, especially when the data is delivered over heterogeneous paths. Although MPTCP improves the overall throughput by pooling the resources on multiple paths, the reordering issue is still unresolved. The state of the art of the reordering problem in MPTCP is examined as follows.

Barré et al. [14] evaluated the impact of heterogeneous paths on the receive buffer and aggregated throughput. The experiment result shows that losses on one subflow have a limited impact on the performance of the other subflows. However, this observation is based on an assumption that the receive buffer is big enough to accommodate all the out-of-order data. Arzani et al. [12] found that the send buffer size also has significant impact on MPTCP's performance. For example, MPTCP provides higher performance gains with a larger send buffer. Chen et al. [24] explored the performance of MPTCP over wireless networks. In order to avoid performance degradation, they set the receive buffer up to 8 MB, which is not feasible in practice for many devices, especially for mobile devices.

Zhou et al. [154] conducted extensive experiments to examine the goodput performance of MPTCP with bounded receive buffers and found that if the paths had similar end-to-end delays, the MPTCP goodput was near optimal, whereas if there was a large variation of the end-to-end delays, the goodput would be significantly degraded. For a wireless environment, they proposed dynamically adjusting the congestion window for each TCP subflow so as to mitigate the variation of end-to-end path delay, maintaining similar end-to-end delays over multiple paths. For wired environment with stable end-to-end delay, in order to reduce the out-of-order packets, they proposed using a delay-aware scheduling algorithm to predict the receiving sequence. The drawback of [154] is that they ignored the impact of packet losses on the reordering issue. Nguyen et al. [91, 92] evaluated the performance of MPTCP in terms of load sharing and throughput optimization with and without the CCC option respectively. The results showed that the context of mismatched path characteristics had a great impact on the performance of MPTCP with bounded receive buffers. Raiciu et al. [111] proposed mechanisms of opportunistic retransmission and penalizing slow subflows to avoid the reordering problem. For example, if a subflow has caused too many out-of-order packets in the receive buffer, the congestion window of that subflow is reduced by half and its slow-start threshold is set to the current congestion window. But if that subflow has been in the slow-start phase already, the reordering problem may become worse because the penalization mechanism will set its slow-start threshold to a smaller value. Paasch et al. [97] proposed improving the penalization mechanism by adjusting the slow-start threshold only when a subflow is not in its slow-start phase. However, Paasch et al. also identified that the penalization mechanism is far from perfect because a subflow at full sending speed may still overflow the receive buffer while another subflows is in slow-start.

From the previous discussion, some more efficient scheduling algorithms are still needed to mitigate the reordering effect.

2.3.2 Multi-homed Data Center

Traditional DCNs have been built using hierarchical topologies: racks of hosts connect to a top-of-rack switch (TOR); these switches connect to aggregation switches; in turn they are connected to a core switch. Unless traffic is localized to TORs, the higher levels of the topology become serious bottlenecks. In order to address this limitation, new data center topologies are emerging. For example, Greenberg et al. proposed VL2 [47] which uses multiple core switches to provide full bandwidth between any pair of hosts in the network. FatTree uses large quantities of lower speed links between switches. In BCube [48], sources probed congestion on all paths, and then used source routing to avoid congestion. In recent years, a focus has been placed on initiatives to use MPTCP in multi-homed data centers to improve the overall performance. Raiciu et al. [112] were the first proposing a natural evolution of data center transport from TCP to MPTCP in multi-homed data centers. They showed that MPTCP could efficiently and seamlessly use available bandwidth, providing improved throughput and better fairness compared to single path TCP. The same authors in [109] investigated what caused these benefits. They found that using MPTCP allows us to rethink DCNs and approach them with a different mindset as to the relationship between transport protocols, routing and topology. As DCNs are very different from the Internet in many respects, such as bandwidth, topologies, latency and traffic patterns [15], it is not wise to replace TCP with MPTCP immediately without solving the potential challenges first. One of the challenges of great interest is the Incast effect, a behavior of MPTCP that results in the gross underutilization of link capacity in certain many-to-one traffic patterns.

The Incast effect is explained as follows. DCNs support a myriad of services and applications. Some applications, such as search engines (e.g., Google and Bing) and Map-reduce, generate large amounts of intricate traffic as many replies to a single inquiry are sent simultaneously to the node that initiated the inquiry. A new inquiry is not issued until all replies have been received. Although each individual reply has only hundreds of kilobytes of data for transfer, it is commonly constrained by a completion deadline. When the replies traverse a shared bottleneck in a many-to-one fashion, the perceived application-level throughput at the initiator collapses because some of the replies may be much delayed due to timeout caused by heavy congestion. From the protocol viewpoint, it is the TCP congestion control mechanism that causes unacceptable delays in the case of RTO. Therefore, the legacy congestion control and retransmission paradigm need to be revised.

The Incast effect was first termed by Nagle et al. in [90]. There exist many approaches in trying to solve it. Phanishayee et al. [101] observed that the Incast effect is caused by the overflow of the bottleneck buffer. They offered a few approaches including using different TCP variants for better packet loss recovery and using short timeouts. Vasudevan et al. [143, 144, 145] believed that the TCP RTO_{min} does more harm than good in data center networks. They proposed reducing RTO_{min} from its current value of 200 ms to values of 1 ms or less (e.g., 200 µs) so that there would be a shorter idle in each timeout. However, as the authors pointed out, most systems lacked the high-resolution timers required for such fine-grained RTO values. Psaras and Tsaoussidis [107] demonstrated that the conservative RTO_{min} setting might cause severe TCP performance degradation. They proposed a mechanism named Adaptive MINRTO to identify the packets whose ACKs were (possibly) going to be delayed, and applied extended RTO_{min} to those packets only.

Alizadeh et al. [7] proposed Data Center TCP (DCTCP), a modified TCP, which aggressively maintained a low switch queue length so as to keep enough of a buffer to deal with sudden bursts of traffic. In DCTCP, Explicit Congestion Notification (ECN) [114] is the key tool to indirectly control the occupied router buffer size. DCTCP tends to offer lower throughput because it uses a very small buffer space. In order to improve the throughput of DCTCP, Das and Sivalingam [29] proposed TDCTCP by modifying the DCTCP congestion control algorithm and the delayed ACK timeout calculation algorithm. TDCTCP provides more stable throughput than DCTCP although the packet delay is higher than DCTCP.

Haitao et al. [51] proposed ICTCP, a receiver side congestion control. In ICTCP, the receive window sizes of the connections are dynamically adjusted to alleviate the congestion based on the ratio of incoming throughput to expected throughput. Zhang and Ansari [153] proposed a control congestion algorithm, FQCN (Fair Quantized Congestion Notification), in order to improve the fairness of multiple flows sharing one bottleneck link and to facilitate quick convergence of the queue length at the bottleneck link to the equilibrium queue length. Zhang et al. [152] proposed shrinking the Maximum Transmission Unit (MTU) to mitigate the Incast throughput collapse. They found that although decreasing MTU size was not an approach that could fix the problem, it was able to delay the onset of Incast. Tam et al. [139] investigated three root causes for Incast collapse and proposed three approaches for the beginning, the middle and the end of a TCP connection respectively. The three approaches are: admission control of TCP flows, timestamp based retransmission, and reiterated FIN packets for tail loss. Kulkarni and Agrawal [79] proposed a spontaneous proactive retransmission strategy. In order to prevent timeout, they allowed TCP senders to retransmit unacknowledged packets periodically without waiting for signals from the receiver.

Some researchers understood the Incast problem in a different way. For example, Shpiner et al. [128] presented the significant unfairness problem of TCP flows in DCNs and introduced a switch-level hash-based algorithm, Hashed Credits Fair (HCF), to address the Incast effect. Devkota and Reddy [32] suggested modifying the switch to regulate flow rates, so that the severe drop that causes Incast throughput collapse is less likely to happen.

Incast collapse is not specific to MPTCP, but is inherited from TCP. No existing approach, however, is entirely satisfactory to avoid the Incast effect in MPTCP without incurring additional major extensions. Therefore, it motivates us to address the Incast issue of MPTCP by manipulating the TCP extensions for multipath operation because MPTCP has to make the necessary extensions anyway. Specifically, Publication V investigates how to share network resources among different MPTCP flows by performing an additional congestion control based on the CCC mechanism.

2.4 Packet Coding

Packet coding has emerged as an important approach in the operation of communication networks. The major benefit of packet coding stems from the ability of its redundancy to compensate for missing packets. This makes data transmission over lossy networks robust and efficient. In recent years, there has been a resurgence of interest in the use of packet coding in wireless, content distribution, and multicast networks. Packet coding is a technique frequently used in this thesis, such as Publication I, Publication II, Publication III, and Publication IV. Therefore, packet coding is discussed in a separate section to emphasize its importance in this thesis. Some of the related work in the discussions that follows may not be specific to multipath transmission, but the coding and scheduling algorithms can be borrowed for the purpose of multipath transmission. In this thesis, packet coding refers to any linear combinations of packets in a finite field. There are a few coding methods which have been widely used, for example, network coding, forward error correction coding, and fountain coding. The existing work, therefore, is classified into these three categories.

Starting with the initial work of [6, 78], network coding techniques have seen a rapid growth in the theory and their potential applications. These developments have been summarized in [58]. However, to a large extent, network coding theory has not yet been implemented in practical systems. Chou et al. in [25] introduced the idea of embedding the coefficients used in the linear combination in the packet header, and also that of the notion of generations (blocks). Ho in his dissertation [57] made a significant step towards a robust implementation of network coding. Some efforts in [45, 64, 138, 155] have been made to combine TCP and network coding. Huang et al. [64] showed that network coding could compensate for the lost packets so as to improve the throughput in wireless mesh networks. Zhuoqun et al. [155] proposed integrating network coding in wireless mesh networks to minimize reordering and timeout at the receivers. Fragouli and Sundararajan et al. [45, 137, 138] proposed a mechanism that incorporated network coding into a special TCP implementation (TCP-Vegas [19]) with minor changes to the protocol stack. In their mechanism, whenever the source was allowed to transmit, it sent a random linear combination of all packets in the congestion window. The receiver acknowledged the degrees of freedom (i.e. the number of linear independent packets) and not the original packets. Note that the network coding used in [45, 137, 138] was implemented in an end-to-end manner.

Forward Error Coding (FEC) is a fault tolerance technique introduced to improve network throughput and performance. LT-TCP [136] is a transport protocol designed to be robust in environments with high loss rates and bursty losses. It uses adaptive segmentation, loss estimation and FEC to improve goodput by avoiding expensive timeouts. LT-TCP was designed to operate over a single path and cannot leverage additional capacity on multiple paths. Li et al. [82] proposed applying FEC to counter packet losses. They proposed an algorithm to schedule packets on paths such that the average number of lost packets was minimized while the FEC encoding remained fixed irrespective of the network conditions. Nguyen and Zakhor [93] proposed a distributed video streaming application from multiple senders to a single receiver for bandwidth aggregation. Their idea was to apply FEC in the rate allocation algorithm to minimize the probability of packet loss in bursty loss environment. Sharma et al. [123, 124] proposed MPLOT (Multi-Path LOss-Tolerant protocol) to utilize available bandwidth on multiple heterogeneous, highly lossy paths. MPLOT estimated path parameters (loss, capacity, and RTT) continuously to provide adaptive FEC coding. In particular, MPLOT performs latency-aware packet mapping, a similar scheduling strategy to that used in [49, 100, 154]. Specifically, it maps packets that are not required immediately to paths with long delays, while mapping the more immediately useful packets to paths with short RTTs.

Fountain codes [127] are a low-cost approach to the problem of reliable communication over a packet erasure channel. Cui et al. [27, 28] proposed applying the rateless coding to multipath scheduling to mitigate the impact of path heterogeneity.

Coding across packets is a powerful technique to improve network throughput by either recovering missing packets in lossy networks or by achieving the upper bound in certain multicast topologies. In this thesis, packet coding is used as a key element to address the reordering issue of transport protocols. Packet coding is an elegant design choice because the notion of an ordered sequence of TCP segments is missing (in the space of a generation/block) and the packet delays/losses are essentially masked from the receiver. However, it is not our goal to demonstrate that packet coding is a better choice than any other options. Rather, we explore the design space in which packet coding can integrate into transport protocols, and to what extent packet coding could improve their efficiency. In addition to the bright side of packet coding, its overhead in networks and hosts is also considered. For instance, every successful packet reception brings a unit of new information. Therefore, the receiver must collect enough packets before the original packets can be recovered, resulting in an additional decoding delay to end-to-end communication. Sanghavi [119] addressed the question of how many original packets could be revealed before the whole generation was decoded. However, the decoding delay depends on not just the number of recovered packets, but also the order in which they are decoded. In addition to the decoding delay, computational complexity is another concern preventing packet coding from being used widely. Publication II looks into ways of reducing the computational overhead. However, as mentioned previously in the research scope (in Chapter 1), computational overhead incurred by packet coding is not the subject of discussion in this thesis.

3. Summary of Results

In this chapter the main contributions of this thesis are discussed. The contributions consist of a packet coding based MPTCP with bounded receive buffers, the evolution of the Delayed ACK mechanism from TCP to MPTCP, and a new congestion control algorithm for MPTCP in DCNs. The future work is discussed at the end of this chapter.

3.1 Packet Coding Meets MPTCP

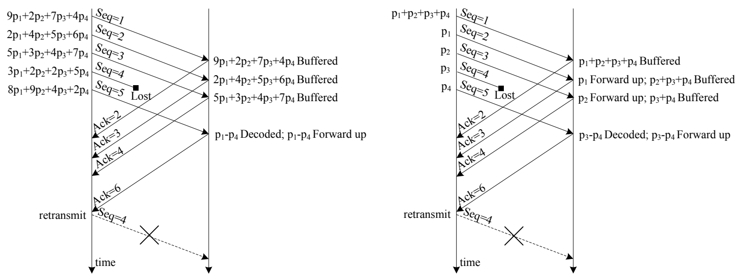
When packets are sent over heterogeneous paths, they may arrive at the receiver out of order. If the resequencing buffer is bounded, the out-of-order data will overrun the buffer and cause HLB. Increasing the size of the resequencing buffer can mitigate the HLB effect, but an infinite buffer is not feasible in practice. As discussed in Section 2.2.6, reordering at the receiver has been a challenge for all multipath approaches. To solve this challenge, it is essential to design an efficient packet scheduling algorithm. This has motivated us to develop a packet coding based scheduling algorithm for MPTCP. The goal is to understand to what extent the packet coding techniques could help mitigate the reordering issue for MPTCP when using bounded receive buffers on heterogeneous paths, and whether additional mechanisms are required to compensate for the insufficiency of packet coding.

Section 2.4 discusses the existing work of integrating packet coding into network transportation. Packet coding is well known for its redundancy property in communication networks. However, it is also known for its negative effects, for example, computational overhead. A principle to reduce the computation overhead is to encode fewer packets. In coding theory, a non-systematic coding is any error-correcting code in which output does not contain the input symbols. In contrast, in a systematic coding, the input data is embedded in the encoded output. Compared with non-systematic coding, systematic coding incurs less computation overhead because some "coded packets" are original packets requiring no encoding and decoding operations. Most existing coding approaches for multipath transmission use, to the best of my knowledge, non-systematic coding. Instead, Publication I and Publication II utilize systematic coding. The packet coding design is described as follows. The data stream is divided into generations. In a generation (Let θ denote the generation size), each packet is interpreted as a symbol over the field $GF(2^s)$, where s indicates the packet size (assuming each packet has the same size). The coded packets are generated by combining the packets from the same generation using random coefficients over the field $GF(2^s)$. Let A_k denote the linear independent coefficient matrix and I_θ denote an identity matrix. The generator matrix M for encoding a generation is

$$M = \Big[A_k | I_\theta\Big], \tag{3.1}$$

where k is the number of redundant packets. If θ is fixed, the larger the k is, the more computational overhead will be incurred.

The packet coding design introduced above is the common part used in Publication I and Publication II. Now a different part is discussed. The core of the packet coding design in Publication I is that packet coding is integrated to some but not all subflows. For example, choosing the subflows having the best performance to carry coding packets is always a good choice in terms of providing enough compensation, even in the worst cases. Specifically, regular subflows and packet coding subflows are used at the same time: the regular subflows deliver original packets, while the packet coding subflows deliver the linear combinations of the original packets. The goal is to mitigate the reordering impact on the receive buffer by taking advantage of redundant coding packets to compensate for the lost packets. However, the computational overhead is not minimized because the compensation is over-provided in most cases. The challenge is how to choose the optimal coding subflows, a process which incurs minimum coding effort while satisfying the compensation requirement. Publication II proposes Systematic Coding Multipath TCP (SC-MPTCP) to solve this challenge. For example, only redundant packets are encoded and each subflow is allowed to deliver them. Moreover, the sender updates the redundancy (i.e. k) according to the estimated retransmission ratio on all subflows. Therefore, the computational overhead further re-



(a) MPTCP using non-systematic coding   (b) SC-MPTCP sending redundancy at first

Figure 3.1. Different transmission procedures

duces because the number of coding packets sent as redundancy is minor compared with that of the original packets in normal cases.

Packet coding is also known for its negative effect on the decoding buffer which further affects the decoding delay. For example, every coding packet brings in new information to the receiver, even though it does not reveal an original packet immediately. Thus, the receiver needs a buffer to accommodate the coding packets from the same generation before being able to decode them. Fig. 3.1 shows an example of a few simplified transmission procedures of MPTCP and SC-MPTCP to explain our choice on coding design. It is assumed that the sending window size is 5 and a sender transmits five packets one of which is lost. Fig. 3.1a shows the transmission procedure of a non-systematic coding approach, where all the packets from the same generation are buffered until enough coded packets arrive. The generation size determines the decoding buffer, with a larger generation, for example, requiring more buffer to accommodate the coding packets from the same generation. Although Publication I uses a systematic coding design, it does not set the sending priority between the original packets and coding packets. Therefore, the advantage of systematic coding is not fully utilized.

Fig. 3.1b presents the transmission procedure of the systematic coding approach used in Publication II. In the process of sending a generation, the redundant packets are sent initially and then the original packets are sent in subsequent transmissions. The goal is to allow the coding packets to arrive at the receiver before the original packets. This scheduling strategy could release the in-order arriving original packets immediately and only buffer the redundancy. For example, as shown in Fig. 3.1b, when p_1 arrives and is forwarded (up) to the upper layer, the coded packet $(p_1+p_2+p_3+p_4)$ could be recoded to $(p_2+p_3+p_4)$ by removing p_1 arithmeti-

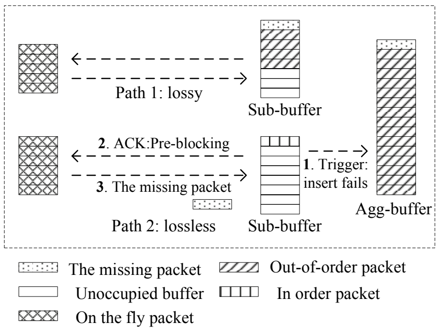


Figure 3.2. Pre-blocking warning trigger illustration

cally so that the buffer space of p_1 could be released immediately. Therefore, the required decoding space (as well as decoding delay) is greatly reduced.

Packet coding is usually used for proactive redundancy, which is timely updated according to dynamic path characteristics. However, the proactive redundancy may be inevitably underestimated in practice anyway. In such a case, reactive redundancy is needed to compensate for the missing packets as quickly as possible. Publication II introduces a pre-blocking warning mechanism to retrieve missing packets without waiting for expensive retransmissions and also explores an appropriate condition to trigger the pre-blocking warning mechanism. As shown in Fig. 3.2, two subflows deliver data to the receiver. The receiver has one buffer for each subflow and one shared aggregate buffer. The subflow in-order arriving packets are inserted into the aggregate buffer for connection-level reordering. If a subflow in-order arriving packet fails to be inserted into the aggregate buffer, it is implied that the current subflow will be blocked with a high probability. This phenomenon is used to trigger the pre-blocking warning mechanism. Wherever it is triggered, an ACK carrying the required number of packets for a certain generation is sent to the sender. The sender would send the required packets immediately on the same subflow in response.

The proposal is evaluated using a network simulator NS-3 (NS-3.10 and NS-3.11 are used in Publication I and Publication II respectively) on a N-path topology with and without the CCC option. Let MTCP denote the MPTCP without the CCC option. Fig. 3.3 shows the minimum required receive buffer to achieve the maximum aggregate throughput. In the simulation, the packet loss ratio and RTT of one subflow are set to 0.1% and

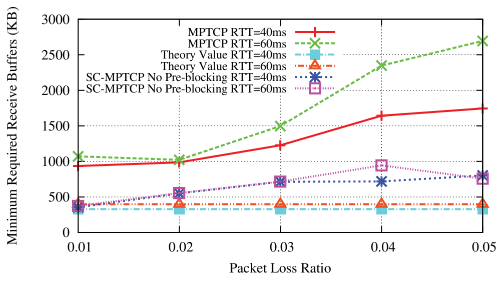


Figure 3.3. Minimum required buffer to approach the maximum aggregate throughput

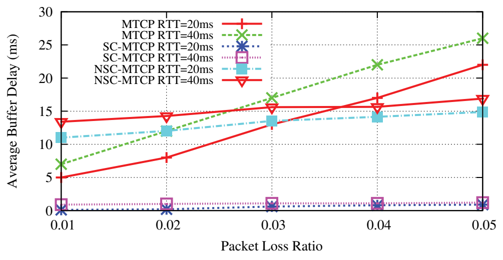


Figure 3.4. Average Buffer Delay

20 ms respectively. The path heterogeneity is increased by setting the packet loss ratio and RTT of the other subflow from 1% to 5% and from 40 ms to 60 ms respectively. It is observed that the required buffer increases when either the packet loss ratio or the RTT of the other subflow grows. We deduce that the larger the path heterogeneity is, the more receive buffer is needed for reordering. In Fig. 3.3, the curves of SC-MPTCP with the pre-blocking warning mechanism enabled are not plotted. Instead, the receive buffer is set according to

$$Buf = 2 \cdot \sum_{i=1}^{N} BW_i \cdot RTT_{max}, \tag{3.2}$$

which is the default receive buffer setting in MPTCP. It is found that SC-MPTCP could always approach the same performance as MPTCP with unlimited receive buffers. This result implies that SC-MPTCP could achieve the desired performance with a normal receive buffer setting.

Fig. 3.4 shows the average buffer delay under the impact of packet loss and RTT. The packet loss ratio and RTT of one subflow are set to 0.1% and 20 ms respectively. The packet loss ratio of the other subflow changes from 1% to 5% using different RTTs. The result is that the average buffer delay of MTCP increases when the path heterogeneity grows. The non-systematic coding MTCP (NSC-MTCP) also introduces non-trivial decod-

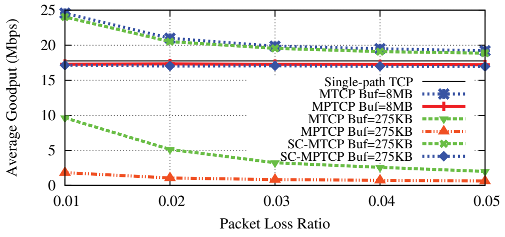


Figure 3.5. Average goodput with various path heterogeneity

ing delays. Compared with NSC-MTCP, SC-MTCP reduces the average buffer delay by up to 90%.

One of the goals of multipath transmission is to aggregate bandwidth. However, due to path heterogeneity and bounded receive buffers, the performance may degrade significantly. Fig. 3.5 shows the aggregate goodput of different protocols. It is observed that SC-MPTCP and SC-MTCP with bounded receive buffers could approach almost the same average goodput as MPTCP and MTCP with large receive buffers, even if the path heterogeneity is huge. Under the same path conditions, the performance of MPTCP and MTCP is poor, even worse than that of a single-path TCP flow. Furthermore, MPTCP degrades more significantly than MTCP. The reason is that due to the resource pooling feature of MPTCP [148], the live subflow moves its traffic away to another subflow (even though it is blocked). This problem does not come from SC-MPTCP but is inherited from MPTCP.

3.2 New Delayed ACK Scheme for MPTCP

Publication III uses a different solution to previous work in order to solve the reordering issue in MPTCP. Specifically, Publication III investigates the impact of TCP timeout on the aggregate goodput and the aggregate buffer. The major contribution is a new Delayed ACK mechanism (NDA) removing the RTO_{min} constraint at the sender while reserving the delayed ACK function at the receiver. The solution requires only minor modification to the legacy Delayed ACK mechanism [18] with no extra traffic overhead. The solution also takes advantage of packet coding techniques but incurs negligible computational overhead.

The DA mechanism is an option of TCP that allows the receiver to delay sending an ACK for every other packet within a window given by the DA

Table 3.1. Path parameters in different scenarios

Scenario	RTT	Bandwidth	Loss ratio	File
Datacenter	200 µs-1 ms	1 Gbps	0.5%	800 MB
LAN	1 ms-10 ms	100 Mbps	0.5%	500 MB
Internet	10 ms-100 ms	100 Mbps	0.5%	300 MB

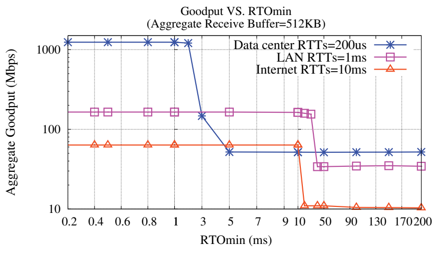


Figure 3.6. The impact of RTO_{min} on aggregate goodput

timer. When DA is enabled at the receiver, the RTO timer at the sender is set to be no less than RTO_{min} in order to avoid spurious timeouts. The TCP RTO timer is modeled by RFC 6298 [98] as

$$\mathrm{RTO} = \max\left(\mathrm{SRTT} + 4 * \mathrm{RTTVAR}, \mathrm{RTO_{min}}\right), \tag{3.3}$$

where SRTT holds the smoothed RTT, and RTTVAR the RTT variation. Currently, there exists no consistent setting of RTO_{min} in practice. A typical value for RTO_{min} is 200 ms, which is one or two orders of magnitude larger than the normal RTT in high speed networks. RTO_{min} can lead to significant performance degradation in the presence of timeouts, especially in multipath transmission where one path enters timeout while the other paths send data at their full speed. Fig. 3.6 shows the aggregate goodput of MPTCP as a function of RTO_{min} in three typical network settings. The aggregate receive buffer is set to 512 KB and the other parameters are set according to Table 3.1. As illustrated by Fig. 3.6, MPTCP achieves the desired aggregate goodput when RTO_{min} is small, whereas upon RTO_{min} reaching a certain value, the aggregate goodput drops sharply. Therefore, removing the RTO_{min} constraint (i.e., RTO_{min} = 0 ms) is effective for all network settings.

To better understand how RTO_{min} affects MPTCP performance, we now review a process of a subflow timeout retransmission using the DA mechanism. RTO_{min} is assumed to be larger than the estimated smoothed RTT. Otherwise, RTO_{min} adds no extra waiting time in the case of timeouts. As shown in Fig. 3.7, the packet p_1 is delivered on a subflow and is lost without subsequent packets coming. In accordance with the rules of TCP

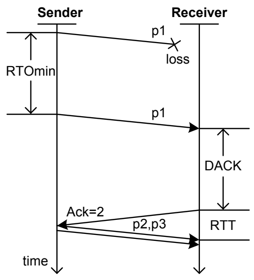


Figure 3.7. TCP timeout on a subflow using the legacy Delayed ACK mechanism

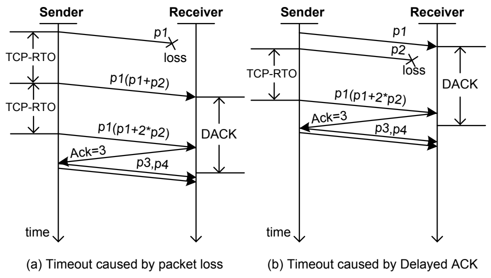


Figure 3.8. TCP timeout using the new Delayed ACK mechanism

RTO, p_1 will be retransmitted after RTO_{min}. In accordance with the rules of the delayed ACK function, when p_1 reaches the receiver, the receiver will delay sending an ACK until the Delayed ACK timer (DACK) expires so that the packets p_2 and p_3 are also delayed in arriving at the receiver. Note that the timeout happens in the context of multipath transmission. When a subflow has a timeout, the receiver has to buffer the out-of-order data from all the subflows until the missing packet (p_1) is received. The out-of-order data may overrun the receive buffer to cause HLB, seriously impacting the overall performance.

Can we just disable the DA mechanism as it harms the performance of MPTCP during timeouts? The answer is no because the DA mechanism has been already used widely on the Internet. It is not possible to switch it off while just ignoring its wide deployment and benefits (e.g., reduced protocol overhead and ACK traffic). In order to solve the dilemma, a new Delayed ACK mechanism, NDA, is proposed to remove the RTO_{min} constraint at the sender while reserving the delayed ACK function at the receiver.

The idea is to retransmit a coded packet instead of an original one during a timeout on any subflow. We first illustrate the principal idea in Fig. 3.8 and then discuss the coding design in detail. When a subflow

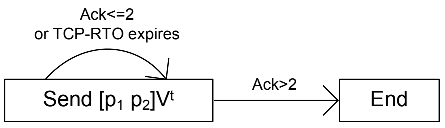


Figure 3.9. Processing flow of RTO using NDA

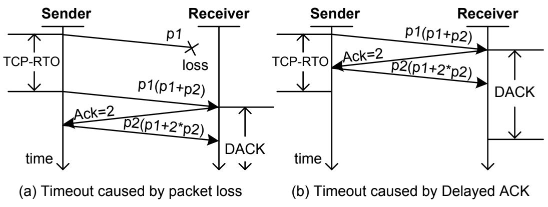


Figure 3.10. Optimized New Delayed ACK mechanism

enters timeout, instead of waiting for RTO_{min}, the sender does the retransmission after TCP-RTO that is set according to

$$\text{TCP-RTO} = \text{SRTT} + 4 * \text{RTTVAR}. \tag{3.4}$$

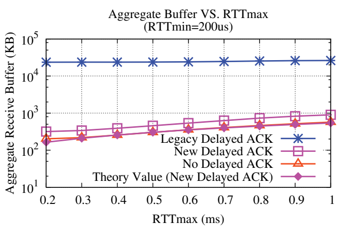
Note that in order to avoid spurious retransmission, the sender retransmits a coded packet (p_1+p_2) instead of p_1, a linear combination of p_1 and p_2. p_2 is the next unsent packet or the next unacked packet on the same subflow.

The coding design now is discussed in detail as follows. On each occasion the encoding/decoding process involves two original packets. Let p_1 and p_2 denote them. Each packet is interpreted as a symbol over the field $GF(2^s)$, where s indicates the packet size. The processing flow of the timeout retransmission is described in Fig. 3.9 where if the sender receives an $ACK{\leq}2$ or does not receive any ACK at all before the TCP-RTO timer expires, a new coded packet would be sent out. Otherwise, the process would end. Each time a new packet is generated, a different predefined coefficient vector V is used.

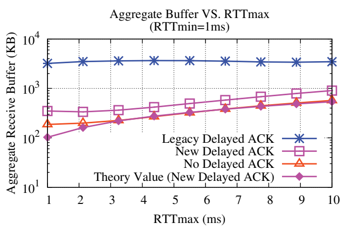
Comparing with the legacy DA mechanism, our solution could greatly reduce the idle time during timeout. However, there still exists optimization space. For example, the receiver could acknowledge every coded packet immediately so that the second coded packet could be sent out after one RTT instead of a successive timeout. The optimized process is illustrated in Fig. 3.10 where the receiver sends an ACK immediately if it receives a timeout retransmitted packet. Therefore, a timeout retransmitted packet could arrive at the receiver in TCP-RTO+RTT. The required receive buffer of MPTCP using NDA is modelled as follows:

$$Buf = \sum_{i=1}^{N} BW_i \cdot \Big[\textit{TCP-RTO} + (d\textit{+1})\textit{RTT}\Big]_{max}, \tag{3.5}$$

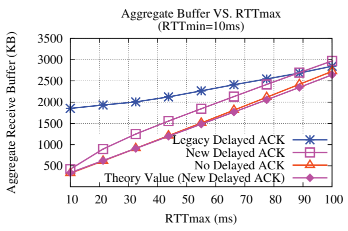
where d indicates the number of RTTs to complete the fast retransmissions before starting the RTO timer.



(a) Data center network



(b) Local network



(c) Internet

Figure 3.11. Aggregate receive buffer size required to handle the out-of-order data

An NS-3 (NS-3.11) network simulator is used to evaluate the solution. In the following, we want to demonstrate that MPTCP using the NDA mechanism requires a much smaller aggregate buffer than MPTCP using the DA mechanism. Fig. 3.11 presents the required aggregate receive buffer in three typical network settings. The RTT of one path is fixed to 200 µs, 1 ms, and 10 ms in three network settings respectively and the RTT of the other path changes. The theory value is calculated according to Eq. (3.5) where d is set to 0. The result shows that the MPTCP using the DA mechanism requires much more aggregate receive buffer than MPTCP using NDA, even one or two orders of magnitude more in local area and data center networks. It implies that NDA is more efficient in high speed networks. Moreover, the MPTCP using NDA always requires only a bit more aggregate buffer than MPTCP without the DA mechanism at all. This is because NDA introduces one or a few more RTTs than no DA mechanism. The result calculated in theory according to Eq. (3.5) closely matches the result of MPTCP without Delayed ACK, which demonstrates

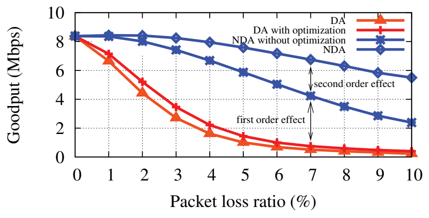


Figure 3.12. Goodput as the function of packet loss in wireless lossy networks

the correctness of the mathematical model on the aggregate receive buffer in MPTCP.

Publication IV demonstrates that the same NDA mechanism also works for legacy TCP in wireless lossy environments. Fig. 3.12 plots the goodput of a single TCP flow as the function of loss ratio using four different Delayed ACK strategies, where DA with optimization means that in the DA mechanism the receiver is allowed to send ACK immediately if it receives a retransmitted packet. It is shown that the TCP flow using the DA mechanism has the worst performance. Although the optimized DA could improve the goodput, the improvement is limited. This result implies that the transmission idle during each timeout is the main reason for TCP performance degradation. Among the Delayed ACK strategies, NDA allows TCP to obtain the best performance. The gain comes from two-order effects. One comes from removing the RTO_{min} constraint because a smaller RTO timer makes the TCP react quickly to timeouts, resulting in a small timeout idle. The other effect comes from eliminating consecutive RTO by allowing the receiver to acknowledge each timeout retransmission, which further reduces the timeout idle.

3.3 Adapting MPTCP for Data Centers

As discussed in Section 2.3.2, multi-homed topologies have been proposed to offer bandwidth aggregation in data centers. MPTCP is a natural evolution of TCP so as to leverage path diversity to improve performance and provide robust data transfers. However, in data center networks, MPTCP faces the Incast congestion challenge as TCP does in the many-to-one communication pattern. Publication V investigates how network resources should be shared among multiple MPTCP flows in the many-to-one traffic pattern. The main contribution is a new congestion control algorithm for MPTCP, Equally-weighted MPTCP (EW-MPTCP), to be used

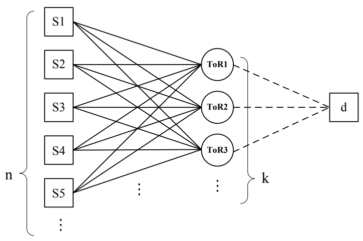


Figure 3.13. k-homed FatTree topology

in multi-homed data center networks. EW-MPTCP focuses on avoiding packet losses before Incast congestion.

In the Incast communication pattern, the receiver does not issue the next request until all the responses return within a limited time. This communication pattern implies that all the responses should be treated as a whole when they go through a shared bottleneck. Otherwise, if n MPTCP connections compete at a shared bottleneck, all the connections together are approximately n times as aggressive as one MPTCP connection. The radical aggressiveness may cause the bottleneck buffer to overflow with the responses for the same request, which will significantly enlarge the request completion time. EW-MPTCP utilizes a congestion avoidance approach to improve the aggressiveness of the MPTCP connections as a whole. The goal is to allow MPTCP connections for the same request to behave as aggressively as an individual TCP connection when they go through a shared bottleneck.

In our study, it is observed that the Incast collapse usually happens on the access links between the servers and ToRs (Top-of-Rack servers). Thus, we extract the access networks to build a k-homed FatTree topology shown in Fig. 3.13 where n senders transmit their data portion through k different paths to a single receiver. The weight of a standard TCP connection is defined as 1. Let $W_{i,j}$ ($1 \le i \le n$, $1 \le j \le k$) denote the weight of an MPTCP subflow $f_{i,j}$, where $f_{i,j}$ is the j^{th} subflow of the i^{th} MPTCP connection. The following equilibrium should be satisfied

$$\sum_{i=1}^{n}\sum_{j=1}^{k} W_{i,j} = 1. \tag{3.6}$$

Note that among different MPTCP connections, we argue that the aggressiveness of each individual MPTCP connection should be equally weighted because first the MPTCP connections are established between different communication peers (e.g., different senders). Each connection should be treated equally. Second, equally weighting MPTCP connections is an appealingly simple mechanism in that it does not require any sort of explicit shared bottleneck detection and could scale well with different numbers of ToRs. Therefore, for each fixed i, the following equilibrium should be satisfied

$$\sum_{j=1}^{k} W_{i,j} = 1/n. \tag{3.7}$$

The physical meaning of Eq. (3.7) is that each MPTCP connection acquires aggressiveness proportionally to $1/n$. Note that MPTCP utilizes the CCC algorithm to make itself no more aggressive than single-TCP. Therefore, at the shared bottleneck, multiple MPTCP connections receive the same throughput as one TCP flow.

We now introduce the question of how to fit the equally-weighted concept into the congestion control algorithm. The equally-weighted algorithm is performed upon the CCC algorithm (see Section 2.3) on each subflow of an MPTCP connection. Specifically, each subflow needs to perform an additional congestion control by weighting the congestion window in reverse proportion to the number of responses. The expected result is that the subflows as a whole could fairly compete with a single-TCP flow at the shared bottleneck. Unlike the CCC algorithm which only performs in the congestion-avoidance phase, the equally-weighted algorithm also performs in the slow-start phase because in the slow-start phase the congestion window tends to grow exponentially for a short period, during which the Incast congestion may have already happened. The EW-MPTCP algorithm (combined with the CCC algorithm) is described as follows:

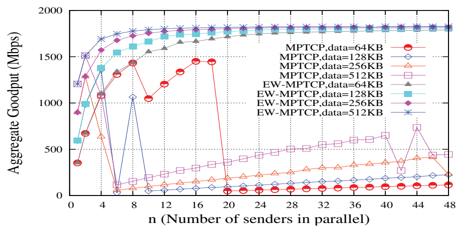
- During the addictive increase phase, for each ACK on subflow s, increase window by $\min\left(\alpha/cwnd_{tot}, 1/cwnd_s\right)/n$.

- During the addictive increase phase, for each loss on subflow s, decrease window by $cwnd_s/2$.

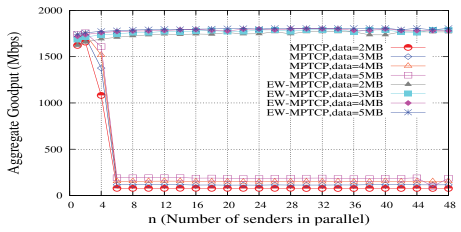
- During the slow start phase, set the initial SSThresh to ssh_0/n.

Here, ssh_0 denotes the default SSThresh, which is set to 65536 in various TCP implementations. We assume the sender knows the value of n either in advance under the control of one entity or in an initialization process at the beginning of the connection.

In EW-MPTCP, the congestion control behavior of the subflows is not only affected by the equally weighted congestion control but also the coupled congestion control inherited from MPTCP. Therefore, EW-MPTCP also keeps the resource pooling feature of MPTCP.



(a) fixed data volume per server



(b) fixed data volume per request

Figure 3.14. Goodput of EW-MPTCP and MPTCP

EW-MPTCP is evaluated by a network simulator NS-3 (NS-3.17). In the simulation, two typical scenarios are considered: fixed data volume per server and fixed data volume per request. In the first scenario, each sender generates the same amount of data traffic to the receiver with the number of senders increasing. In the second scenario, the total data volume of all senders is fixed regardless of the number of senders. The comparison between the goodput of MPTCP and EW-MPTCP is shown in Fig. 3.14. The result shows that MPTCP has the same goodput as EW-MPTCP only when the number of senders is small and starts to suffer when the number of senders increases in both scenarios. Under the same setup, EW-MPTCP achieves a smooth and increasing goodput when the number of senders grows.

In the simulation shown in Fig. 3.14, the number of senders is set to, for instance, fewer than 49. This constraint is relaxed to study to what extent EW-MPTCP could handle the Incast congestion and what its limitations are. Fig. 3.15 presents the simulation result where the total volume of data per request is fixed. In the simulation, three different sizes of MTU

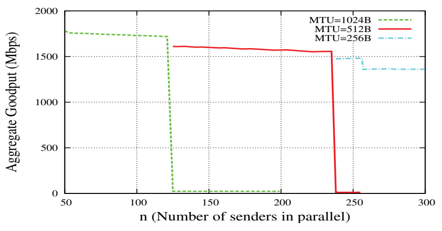


Figure 3.15. Goodput of EW-MPTCP with more senders in the case where the data traffic volume per request is fixed

are used while increasing the number of senders from 50 to 300. It is shown that EW-MPTCP would also collapse at some point and a small MTU could mitigate the collapse. Note that Incast congestion in data centers is not originally from MPTCP but is inherited from TCP. Publication V has also demonstrated that the algorithm not only works for MPTCP on multi-homed data center topologies but also works for TCP on single-homed data center topologies.

3.4 Open Questions

In this section, we discuss future work that can be based on the results of this thesis. First, multipath transmission research in this thesis has mainly focused on MPTCP. The solutions could be applied to other protocols, like CMT-SCTP, as well and vice versa. Therefore, more efforts are needed to evaluate the performance of MPTCP and CMT-SCTP by utilizing each other's mathematical models and key algorithms.

Second, MPTCP takes the advantage of multiple available paths to support load balancing, moving traffic from more congested paths to less congested ones. From the mobility viewpoint, it might be the ideal protocol to support mobility if it can pro-actively detect a connectivity failure on one path and switch to another path. Therefore, it is worth investigating how MPTCP behaves in handover and what changes are required to MPTCP for desired performance (in terms of delays and bandwidth variation) during the handover period. We believe that the packet coding scheme presented in Publication I and Publication II can be used for improving the handover performance.

Third, the solution in Publication V shows an efficient congestion control algorithm for MPTCP as well as TCP in the DCNs to avoid the Incast effect in the many-to-one communication pattern. More efficient algorithms are required to scale to large DCNs with thousands of servers replying to the inquiry. In accordance with the development of multipath transmission, compatibility is one of the most important factors affecting whether a solution could be accepted widely or not. Unlike the Internet, which requires new protocols to be backward compatible with it, a DCN is a relatively close network managed by a single administration so that new networking techniques can be implemented without much concern about interoperability. Therefore, in the DCN environment we suggest "reusing" the existing TCP Incast solutions which require one administration.

4. Conclusion

Communication devices equipped with multiple network interfaces are now increasingly emerging. For example, smart phones and laptops are often shipped with the built-in network adapters of different wireless technologies. Concurrently using multiple interfaces to transmit data has attracted much attention from the industry and academia. Considerable efforts have been put into developing multipath transmission capability at different protocol layers. Despite these efforts, however, research achievements have not yet addressed all the challenges posed by the new multi-homed devices and networks.

This thesis focuses on understanding and analyzing various multipath transmission approaches. Based on the existing work, a practical multipath transmission solution should possess a few key features. These include, for example, a) application and network infrastructure compatibility, b) a reordering buffer and a mechanism to handle the out-of-order data, c) a coupled congestion control algorithm to keep MPTCP TCP-friendly, and d) load balancing to move traffic from more congested paths to less congested ones. Our research focuses on the reordering issue caused by mismatched path characteristics. To solve this issue, various methods were leveraged such as mathematical buffer models, packet coding based packet scheduling algorithms, and new Delayed ACK mechanisms. Specifically, two different solutions have been proposed under the same practical assumption that the reordering buffer is bounded. The first solution proposed the adoption of coded packets as proactive redundancy to counter against expensive retransmissions. The redundancy is continuously updated according to the estimated path characteristics. In order to avoid the proactive redundancy being underestimated, the pre-blocking warning mechanism retrieves the reactive redundancy without waiting for normal retransmissions. In the other solution, the reordering issue is mitigated by removing the RTO_{min} constraint from the RTO timer calculation while reserving the delayed ACK function. In order to eliminate the aggressiveness of the timeout timer after being decreased, the timeout retransmitted packets are encoded to benefit from potential spurious retransmissions.

In addition, the MPTCP Incast effect has also been investigated in multihomed DCNs. In the Incast communication pattern, a client sends a request to many servers and the client does not issue new requests until all responses from the current request have been returned. We argue that all responses from the servers to the client should be treated as a whole so that the response traffic would behave in a TCP friendly fashion at shared bottlenecks without incurring heavy congestion. To achieve this goal, an equally-weighted congestion control algorithm is proposed for MPTCP. Specifically, an MPTCP connection allows each of its subflows to perform an additional congestion control operation by weighting the congestion window in reverse proportion to the number of servers.

Currently, multipath transmission can potentially be used in almost all networks because networks are designed with multipath capability. The largest example of a multipath network is the Internet itself. Multipath transmission could provide both the users and the network operators with an opportunity to efficiently and flexibly utilize their respective resources. The vision of multipath transmission in the near future is that, on one hand, multiple interfaces for various techniques keep the users connected while, on the other hand, multiple interfaces are used in parallel to improve the data transmission performance or manage the network resources appropriately. The solutions presented in this thesis could provide great insight into the development of multipath transmission in the future.

Bibliography

[1] Network simulator (NS-3). http://www.nsnam.org [Available Online].

[2] IEEE Standard for Local and Metropolitan Area Networks - Link Aggregation. *IEEE Std 802.1AX-2008*, pages c1–145, 2008.

[3] A. Abd El Al, T. Saadawi, and M. Lee. LS-SCTP: a bandwidth aggregation technique for stream control transmission protocol. *Computer Communications*, 27(10):1012–1024, 2004.

[4] H. Adhari, T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tüxen. Evaluation of concurrent multipath transfer over dissimilar paths. In *IEEE Workshops of International Conference on Advanced Information Networking and Applications (WAINA '11)*, pages 708–714, 2011.

[5] H. Adiseshu, G. Parulkar, and G. Varghese. A reliable and scalable striping protocol. In *Proceedings of the Applications, technologies, architectures, and protocols for computer communications*, volume 26, pages 131–141. ACM, 1996.

[6] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[7] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. Data center tcp (dctcp). In *Proceedings of the ACM SIGCOMM conference*, volume 40, pages 63–74. ACM, 2010.

[8] M. Allman, H. Kruse, and S. Ostermann. An application-level solution to TCP's satellite inefficiencies. In *Proceedings of the 1st International Workshop on Satellite-based Information Services (WOSBIS '96)*, 1996.

[9] M. Allman, V. Paxson, and E. Blanton. TCP Congestion Control. *IETF RFC 5681*, September 2009.

[10] P. Amer, M. Becke, T. Dreibholz, N. Ekiz, J. Iyengar, P. Natarajan, R. Stewart, and M. Tuexen. Load Sharing for the Stream Control Transmission Protocol (SCTP). *draft-tuexen-tsvwg-sctp-multipath-07 (work in process), Internet Draft, IETF*, March 2013.

[11] S. Androutsellis-Theotokis and D. Spinellis. A Survey of Peer-to-peer Content Distribution Technologies. *ACM Computing Surveys (CSUR)*, 36(4):335–371, Dec. 2004.

[12] B. Arzani, A. Gurney, S. Cheng, R. Guerin, and B. T. Loo. Impact of Path Characteristics and Scheduling Policies on MPTCP Performance. In *Proceedings of the 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 743–748, May 2014.

[13] A. Baldini, L. De Carli, and F. Risso. Increasing performances of TCP data transfers through multiple parallel connections. In *Proceedings of IEEE Symposium on Computers and Communications (ISCC)*, pages 630–636, July 2009.

[14] S. Barré, C. Paasch, and O. Bonaventure. MultiPath TCP: From Theory to Practice. In *Proceedings of the 10th International IFIP TC 6 Conference on Networking - Volume Part I*, NETWORKING '11, pages 444–457. Springer-Verlag, 2011.

[15] T. Benson, A. Akella, and D. A. Maltz. Network traffic characteristics of data centers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 267–280. ACM, 2010.

[16] T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret. The world-wide web. *Communications of the ACM*, 37(8):76–82, 1994.

[17] O. Bonaventure. Apple seems to also believe in Multipath TCP. 2013. `http://perso.uclouvain.be/olivier.bonaventure/blog/html/2013/09/18/mptcp.html` [Available Online].

[18] R. Braden. Requirements for internet hosts-communication layers. *IETF RFC 1122*, 1989.

[19] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications (SIGCOMM '94)*, pages 24–35. ACM, 1994.

[20] L. Budzisz, R. Ferrús, F. Casadevall, and P. Amer. On concurrent multipath transfer in SCTP-based handover scenarios. In *Proceedings of the IEEE International Conference on Communications (ICC '09)*, pages 1–6, 2009.

[21] K. Chebrolu, B. Raman, and R. R. Rao. A network layer approach to enable TCP over multiple interfaces. *Wireless Networks*, 11(5):637–650, 2005.

[22] K. Chebrolu and R. R. Rao. Bandwidth aggregation for real-time applications in heterogeneous wireless networks. *IEEE Transactions on Mobile Computing*, 5(4):388–403, 2006.

[23] J. Chen, K. Xu, and M. Gerla. Multipath tcp in lossy wireless environment. In *Proceedings of IFIP 3rd Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net '04)*, pages 263–270, 2004.

[24] Y.-C. Chen, Y. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley. A Measurement-based Study of Multipath TCP Performance over Wireless Networks. In *Proceedings of the 2013 ACM Conference on Internet Measurement Conference (IMC '13)*, pages 455–468, 2013.

[25] P. A. Chou, Y. Wu, and K. Jain. Practical network coding. In *Proceedings of the Allerton Conference on Communication, Control, and Computing*, 2003.

[26] X. Chu and K. Zhao. Practical random linear network coding on GPUs. In *GPU Solutions to Multi-scale Problems in Science and Engineering*, pages 115–130. Springer, 2013.

[27] Y. Cui, L. Wang, X. Wang, H. Wang, and Y. Wang. FMTCP: A Fountain Code-Based Multipath Transmission Control Protocol. *IEEE/ACM Transactions on Networking (TON)*, January 2014.

[28] Y. Cui, X. Wang, H. Wang, G. Pan, and Y. Wang. FMTCP: A fountain code-based multipath transmission control protocol. In *Proceedings of the 32nd IEEE International Conference on Distributed Computing Systems (ICDCS '12)*, pages 366–375, 2012.

[29] T. Das and K. M. Sivalingam. TCP improvements for data center networks. In *Proceedings of the 5th International Conference on Communication Systems and Networks (COMSNETS '13)*, pages 1–10. IEEE, 2013.

[30] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[31] G. Detal, C. Paasch, and O. Bonaventure. Multipath in the Middle(Box). In *Proceedings of the 2013 Workshop on Hot Topics in Middleboxes and Network Function Virtualization (HotMiddlebox '13)*, pages 1–6, 2013.

[32] P. Devkota and A. N. Reddy. Performance of quantized congestion notification in TCP incast scenarios of data centers. In *Proceedings of the IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS '10)*, pages 235–243. IEEE, 2010.

[33] Y. Dong, N. Pissinou, and J. Wang. Concurrency Handling in TCP. In *Proceedings of the 5th Annual Conference on Communication Networks and Services Research (CNSR '07)*, pages 255–262, May 2007.

[34] T. Dreibholz, H. Adhari, M. Becke, and E. P. Rathgeb. Simulation and experimental evaluation of multipath congestion control strategies. In *Proceedings of the 26th International Conference on IEEE Advanced Information Networking and Applications Workshops (WAINA '12)*, pages 1113–1118, 2012.

[35] T. Dreibholz, M. Becke, H. Adhari, and E. P. 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 '11)*, pages 397–404, 2011.

[36] T. Dreibholz, M. Becke, J. Pulinthanath, and E. P. 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 '10)*, pages 312–319. IEEE, 2010.

[37] T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tuxen. On the use of concurrent multipath transfer over asymmetric paths. In *Proceedings of the*

IEEE Global Telecommunications Conference (GLOBECOM '10), pages 1–6, 2010.

[38] T. Dreibholz, R. Seggelmann, M. Tuexen, and E. 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 '10)*, volume 8, 2010.

[39] F. Ducatelle, G. Di Caro, and L. M. Gambardella. Ant agents for hybrid multipath routing in mobile ad hoc networks. In *Proceedings of the 2nd Annual Conference on Wireless On-demand Network Systems and Services (WONS '05)*, pages 44–53. IEEE, 2005.

[40] K. Evensen, D. Kaspar, C. Griwodz, P. Halvorsen, A. Hansen, and P. Engelstad. Improving the performance of quality-adaptive video streaming over multiple heterogeneous access networks. In *Proceedings of the 2nd annual ACM conference on Multimedia systems*, pages 57–68. ACM, 2011.

[41] K. Evensen, T. Kupka, D. Kaspar, P. Halvorsen, and C. Griwodz. Quality-adaptive scheduling for live streaming over multiple access networks. In *Proceedings of the 20th international workshop on Network and operating systems support for digital audio and video*, pages 21–26. ACM, 2010.

[42] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol–HTTP/1.1, June 1999.

[43] A. Ford, C. Raiciu, M. Handley, S. Barré, and J. Iyengar. Architectural guidelines for multipath TCP development. *IETF RFC 6182*, March 2011.

[44] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. TCP Extensions for Multipath Operation with Multiple Addresses. *IETF RFC 6824*, January 2013.

[45] C. Fragouli, D. Lun, M. Médard, and P. Pakzad. On feedback for network coding. In *Proceedings of the 41st Annual Conference on Information Sciences and Systems (CISS '07)*, pages 248–252. IEEE, 2007.

[46] L. Golubchik, J. C. S. Lui, T. F. Tung, A. L. Chow, W.-J. Lee, G. Franceschinis, and C. Anglano. Multi-path continuous media streaming: What are the benefits? *Performance Evaluation*, 49(1):429–449, 2002.

[47] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: a scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM conference on Data communication*, volume 39, pages 51–62, 2009.

[48] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: a high performance, server-centric network architecture for modular data centers. In *Proceedings of the ACM SIGCOMM conference on Data communication*, volume 39, pages 63–74, 2009.

[49] A. Gurtov and T. Polishchuk. Secure multipath transport for legacy Internet applications. In *Proceedings of the 6th International Conference on Broadband Communications, Networks, and Systems*, pages 1–8, 2009.

[50] T. J. Hacker, B. D. Athey, and B. Noble. The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS '02)*, pages 434–443. IEEE, 2002.

[51] W. Haitao, Z. Feng, C. Guo, and Y. Zhang. ICTCP: Incast Congestion Control for TCP in Data-Center Networks. *IEEE/ACM Transactions on Networking*, 21(2):345–358, 2013.

[52] Y. Hasegawa, I. Yamaguchi, T. Hama, H. Shimonishi, and T. Murase. Improved data distribution for multipath TCP communication. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '05)*, November 2005.

[53] Y. Hasegawa, I. Yamaguchi, T. Hama, H. Shimonishi, and T. Murase. Deployable multipath communication scheme with sufficient performance data distribution method. *Computer Communications*, 30(17):3285–3292, 2007.

[54] S. Hassayoun, J. Iyengar, and D. Ros. Dynamic Window Coupling for multipath congestion control. In *Proceedings of the 19th IEEE International Conference on Network Protocols (ICNP '11)*, pages 341–352, 2011.

[55] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida. The NewReno modification to TCP's fast recovery algorithm. *RFC 6582*, 2012.

[56] B. Hesmans, F. Duchene, C. Paasch, G. Detal, and O. Bonaventure. Are TCP extensions middlebox-proof? In *Proceedings of the 2013 workshop on Hot topics in middleboxes and network function virtualization*, pages 37–42, 2013.

[57] T. Ho. *Networking from a network coding perspective*. PhD thesis, MIT, 2004.

[58] T. Ho and D. Lun. *Network coding: an introduction*. Cambridge University Press, 2008.

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

[60] H.-Y. Hsieh, K.-H. Kim, Y. Zhu, and R. Sivakumar. A Receiver-centric Transport Protocol for Mobile Hosts with Heterogeneous Wireless Interfaces. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom '03)*, pages 1–15, 2003.

[61] H.-Y. Hsieh and R. 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, 2002.

[62] H.-Y. Hsieh and R. Sivakumar. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. *Wireless Networks*, 11(1-2):99–114, 2005.

[63] C.-M. Huang and C.-H. Tsai. WiMP-SCTP: Multi-path transmission using stream control transmission protocol (SCTP) in wireless networks. In *Proceedings of the 21st IEEE International Conference on Advanced Information Networking and Applications Workshops (AINAW '07)*, volume 1, pages 209–214, 2007.

[64] Y. Huang, M. Ghaderi, D. Towsley, and W. Gong. TCP performance in coded wireless mesh networks. In *Proceedings of the 5th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 179–187. IEEE, 2008.

[65] C. Huitema. Multi-homed TCP. *IETF Internet Draft (Expired)*, May 1995.

[66] J. Iyengar, K. Shah, P. Amer, and R. Stewart. Concurrent multipath transfer using SCTP multihoming. In *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS '04)*, 2004.

[67] J. R. Iyengar, P. D. Amer, and R. Stewart. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964, 2006.

[68] R. Jansen, S. Hanemann, and B. Freisleben. Proactive distance-vector multipath routing for wireless ad hoc networks. In *Proceedings of the Communication Systems and Networks (CSN '03)*, 2003.

[69] Jason Lee and Dan Gunter and Brian Tierney and Bill Allcock and Joe Bester and John Bresnahan and Steve Tuecke. Applied techniques for high bandwidth data transfers across wide area networks. In *Proceedings of the International Conference on Computing in High Energy and Nuclear Physics*, 2001.

[70] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi. FatVAP: Aggregating AP Backhaul Capacity to Maximize Throughput. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, volume 8, pages 89–104, 2008.

[71] D. Kaspar, K. Evensen, P. Engelstad, and A. F. Hansen. Using HTTP pipelining to improve progressive download over multiple heterogeneous interfaces. In *Proceedings of the IEEE International Conference on Communications (ICC '10)*, pages 1–5. IEEE, 2010.

[72] D. Kaspar, K. Evensen, P. Engelstad, A. F. Hansen, P. Halvorsen, and C. Griwodz. Enhancing video-on-demand playout over multiple heterogeneous access networks. In *Proceedings of the 7th IEEE Consumer Communications and Networking Conference (CCNC '10)*, pages 1–5. IEEE, 2010.

[73] P. Key, L. Massoulié, and D. Towsley. Combining multipath routing and congestion control for robustness. In *Proceedings of the 40th Annual Conference on Information Sciences and Systems*, pages 345–350. IEEE, 2006.

[74] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. Le Boudec. Non pareto-optimality of mptcp: Performance issues and a possible solution. In *Proceedings of the 8th ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT '12)*, December 2012.

[75] K.-H. Kim and K. G. Shin. Improving TCP performance over wireless networks with collaborative multi-homed mobile hosts. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 107–120. ACM, 2005.

[76] K.-H. Kim and K. G. Shin. PRISM: improving the performance of inverse-multiplexed TCP in wireless networks. *IEEE Transactions on Mobile Computing*, 6(12):1297–1312, 2007.

[77] K.-H. Kim, Y. Zhu, R. Sivakumar, and H.-Y. Hsieh. A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces. *Wireless Networks*, 11(4):363–382, 2005.

[78] R. Koetter and M. Médard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, 2003.

[79] S. Kulkarni and P. Agrawal. A probabilistic approach to address TCP incast in data center networks. In *Proceedings of the 31st International Conference on Distributed Computing Systems Workshops (ICDCSW '11)*, pages 26–33. IEEE, 2011.

[80] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet Inter-domain Traffic. In *Proceedings of the ACM SIGCOMM 2010 Conference*, pages 75–86, 2010.

[81] Y. Lee, I. Park, Y. Choi, and A. M. Routing. Improving TCP performance in multipath packet forwarding networks. *Journal of Communications and Networks*, 4:148–157, 2002.

[82] Y. Li, Y. Zhang, L. L. Qiu, and S. Lam. Smarttunnel: Achieving reliability in the internet. In *Proceedings of the 26th IEEE International Conference on Computer Communications (INFOCOM '07)*, pages 830–838, 2007.

[83] J. Liao, J. Wang, and X. Zhu. cmpSCTP: An extension of SCTP to support concurrent multi-path transfer. In *Proceedings of the IEEE International Conference on Communications (ICC '08)*, pages 5762–5766, 2008.

[84] C. Lim, S. Bohacek, J. P. Hespanha, and K. Obraczka. Hierarchical max-flow routing. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM '05)*, volume 1, pages 6–pp, 2005.

[85] J. Liu, H. Zou, J. Dou, and Y. Gao. Rethinking Retransmission Policy In Concurrent Multipath Transfer. In *Proceedings of the International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIHMSP '08)*, pages 1005–1008. IEEE, 2008.

[86] L. Magalhaes and R. Kravets. Transport level mechanisms for bandwidth aggregation on mobile hosts. In *Proceedings of the 9th IEEE International Conference on Network Protocols*, pages 165–171, 2001.

[87] S. Mao, D. Bushmitch, S. Narayanan, and S. S. Panwar. MRTP: a multiflow real-time transport protocol for ad hoc networks. *IEEE Transactions on Multimedia*, 8(2):356–369, 2006.

[88] N. F. Maxemchuk. *DISPERSITY ROUTING IN STORE-AND-FORWARD NETWORKS*. PhD thesis, University of Pennsylvania, 1975. Available as `http://repository.upenn.edu/dissertations/AAI7524101`.

[89] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host identity protocol. *IETF RFC 5201*, April 2008.

[90] D. Nagle, D. Serenyi, and A. Matthews. The Panasas ActiveScale storage cluster: Delivering scalable high bandwidth storage. In *Proceedings of the ACM/IEEE conference on Supercomputing*, page 53. IEEE Computer Society, 2004.

[91] S. C. Nguyen and T. M. T. Nguyen. Evaluation of multipath TCP load sharing with coupled congestion control option in heterogeneous networks. In *Proceedings of the Global Information Infrastructure Symposium (GIIS '11)*, pages 1–5, 2011.

[92] S. C. Nguyen, X. Zhang, T. M. T. Nguyen, and G. Pujolle. Evaluation of throughput optimization and load sharing of multipath tcp in heterogeneous networks. In *Proceedings of the 8th International Conference on Wireless and Optical Communications Networks (WOCN '11)*, pages 1–5. IEEE, 2011.

[93] T. Nguyen and A. Zakhor. Distributed video streaming with forward error correction. In *Proceedings of the Packet Video Workshop (PV)*, volume 2002, 2002.

[94] P. Nikander, E. T. Henderson, C. Vogt, and J. Arkko. End-host mobility and multihoming with the host identity protocol. *IETF RFC 5206*, April 2008.

[95] E. Nordmark and M. Bagnulo. Shim6: Level 3 multihoming shim protocol for IPv6. *IETF RFC 5533*, June 2009.

[96] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure. Exploring Mobile/WiFi Handover with Multipath TCP. In *Proceedings of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design (CellNet '12)*, pages 31–36, 2012.

[97] C. Paasch, R. Khalili, and O. Bonaventure. On the Benefits of Applying Experimental Design to Improve Multipath TCP. In *Proceedings of the 9th ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '13)*, pages 393–398, 2013.

[98] V. Paxson, M. Allman, J. Chu, and M. Sargent. Computing TCP's retransmission timer. *IETF RFC 6298*, June 2011.

[99] C. Perkins. IP Mobility Support for IPv4, Revised. *IETF RFC 5944*, November 2010.

[100] C. E. Perkins. Mobile IP. *IEEE Communications Magazine*, 35(5):84–99, 1997.

[101] A. Phanishayee, E. Krevat, V. Vasudevan, D. G. Andersen, G. R. Ganger, G. A. Gibson, and S. Seshan. Measurement and Analysis of TCP Throughput Collapse in Cluster-based Storage Systems. In *Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST '08)*, volume 8, pages 1–14, 2008.

[102] D. S. Phatak and T. Goff. A novel mechanism for data streaming across multiple IP links for improving throughput and reliability in mobile environments. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)*, volume 2, pages 773–781, 2002.

[103] D. S. Phatak, T. Goff, and J. Plusquellic. IP-in-IP tunneling to enable the simultaneous use of multiple IP interfaces for network level connection striping. *Computer Networks*, 43(6):787–804, 2003.

[104] S. Pierrel, P. Jokela, and J. Melen. Simultaneous Multi-Access extension to the Host Identity Protocol. *draft-pierrel-hip-sima-00 (work in process), Internet Draft, IETF*, June 2006.

[105] T. Polishchuk and A. Gurtov. Improving TCP-friendliness and Fairness for mHIP. *Inforcommunications Journal*, 3(1), 2011.

[106] J. Postel and J. Reynolds. File transfer protocol. *IETF RFC 959*, October 1985.

[107] I. Psaras and V. Tsaoussidis. On the properties of an adaptive TCP Minimum RTO. *Computer Communications*, 32(5):888–895, 2009.

[108] A. Qureshi, J. Carlisle, and J. Guttag. Tavarua: video streaming with wwan striping. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 327–336. ACM, 2006.

[109] C. Raiciu, S. Barré, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley. Improving datacenter performance and robustness with multipath TCP. In *Proceedings of the ACM SIGCOMM conference*, volume 41, pages 266–277, 2011.

[110] C. Raiciu, M. Handley, and D. Wischik. Coupled congestion control for multipath transport protocols. *IETF RFC 6356*, October 2011.

[111] C. Raiciu, C. Paasch, S. Barré, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. 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 '12)*, volume 12, pages 29–29, 2012.

[112] C. Raiciu, C. Pluntke, S. Barré, A. Greenhalgh, D. Wischik, and M. Handley. Data center networking with multipath TCP. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, page 10, 2010.

[113] C. Raiciu, D. Wischik, and M. Handley. Practical congestion control for multipath transport protocols. *University College of London Technical Report*, 2009.

[114] K. Ramakrishnan, S. Floyd, D. Black, et al. The addition of explicit congestion notification (ECN) to IP, September 2001.

[115] P. Rodriguez and E. W. Biersack. Dynamic parallel access to replicated content in the Internet. *IEEE/ACM Transactions on Networking (TON)*, 10(4):455–465, 2002.

[116] K. Rojviboonchai and A. Hitoshi. An evaluation of multi-path transmission control protocol (M/TCP) with robust acknowledgement schemes. *IEICE transactions on communications*, 87(9):2699–2707, 2004.

[117] K. Rojviboonchai, T. Osuga, and H. Aida. RM/TCP: protocol for reliable multi-path transport over the Internet. In *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA '05)*, volume 1, pages 801–806, 2005.

[118] K. Rojviboonchai, N. Watanabe, and H. Aida. One-way-trip time (OWTT) measurement and retransmission policy for congestion control in M/TCP. In *Proceedings of the Annual Conference of IPSJ*, 2002.

[119] S. Sanghavi. Intermediate performance of rateless codes. In *Proceedings of the Information Theory Workshop (ITW)*, pages 478–482. IEEE, 2007.

[120] D. Sarkar. A Concurrent Multipath TCP and Its Markov Model. In *Proceedings of the IEEE International Conference on Communications (ICC '06)*, pages 615–620, June 2006.

[121] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. *IETF RFC 3550*, July 2003.

[122] S. Shakkottai, E. Altman, and A. Kumar. The Case for Non-Cooperative Multihoming of Users to Access Points in IEEE 802.11 WLANs. In *Proceedings of the 25th IEEE International Conference on Computer Communications (INFOCOM '06)*, 2006.

[123] V. Sharma, S. Kalyanaraman, K. Kar, K. Ramakrishnan, and V. Subramanian. MPLOT: A transport protocol exploiting multipath diversity using erasure codes. In *Proceedings of the 27th IEEE Conference on Computer Communications (INFOCOM '08)*, pages 121–125, 2008.

[124] V. Sharma, K. Kar, K. Ramakrishnan, and S. Kalyanaraman. A transport protocol to exploit multipath diversity in wireless networks. *IEEE/ACM Transactions on Networking (TON)*, 20(4):1024–1039, 2012.

[125] H. Shojania and B. Li. Parallelized progressive network coding with hardware acceleration. In *Fifteenth IEEE International Workshop on Quality of Service*, pages 47–55. IEEE, 2007.

[126] H. Shojania, B. Li, and X. Wang. Nuclei: GPU-Accelerated Many-Core Network Coding. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '09)*, pages 459–467, April 2009.

[127] A. Shokrollahi. Raptor codes. *IEEE Transactions on Information Theory*, 52(6):2551–2567, 2006.

[128] A. Shpiner, I. Keslassy, G. Bracha, E. Dagan, O. Iny, and E. Soha. A switch-based approach to throughput collapse and starvation in data centers. *Computer Networks*, 56(14):3333–3346, 2012.

[129] W. Simpson. The Point-to-Point Protocol (PPP). *IETF RFC 1661*, July 1994.

[130] A. Singh, M. Xiang, A. Konsgen, C. Goerg, and Y. Zaki. Enhancing fairness and congestion control in multipath TCP. In *Proceedings of the 6th Joint IFIP Wireless and Mobile Networking Conference (WMNC '13)*, pages 1–8, 2013.

[131] H. Sivakumar, S. Bailey, and R. L. Grossman. PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks. In *Proceedings of the ACM/IEEE conference on Supercomputing (CDROM '00)*, page 37. IEEE Computer Society, 2000.

[132] K. Sklower, B. Lloyd, G. McGregor, D. Carr, and T. Coradetti. The PPP Multilink Protocol (MP). *IETF RFC 1990*, August 1996.

[133] A. C. Snoeren. Adaptive inverse multiplexing for wide-area wireless networks. In *Proceedings of the Global Telecommunications Conference (GLOBECOM '99)*, volume 3, pages 1665–1672. IEEE, 1999.

[134] M. Stemm and R. H. Katz. Vertical handoffs in wireless overlay networks. *Mobile Networks and applications*, 3(4):335–350, 1998.

[135] R. Stewart. Stream control transmission protocol. *IETF RFC 4960*, September 2007.

[136] V. Subramanian, S. Kalyanaraman, and K. Ramakrishnan. An end-to-end transport protocol for extreme wireless network environments. In *Proceedings of the Military Communications Conference (MILCOM)*, pages 1–7. IEEE, 2006.

[137] J. Sundararajan, D. Shah, M. Medard, S. Jakubczak, M. Mitzenmacher, and J. Barros. Network coding meets tcp: Theory and implementation. *Proceedings of the IEEE*, 99(3):490–512, March 2011.

[138] J. K. Sundararajan, D. Shah, M. Médard, M. Mitzenmacher, and J. Barros. Network coding meets TCP. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '09)*, pages 280–288. IEEE, 2009.

[139] A. S.-W. Tam, K. Xi, Y. Xu, and H. J. Chao. Preventing TCP incast throughput collapse at the initiation, continuation, and termination. In *Proceedings of the 20th IEEE International Workshop on Quality of Service*, page 29. IEEE Press, 2012.

[140] C. B. S. Traw and J. M. Smith. Striping within the network subsystem. *IEEE Network*, 9(4):22–32, 1995.

[141] C.-L. Tsao and R. Sivakumar. On effectively exploiting multiple wireless interfaces in mobile hosts. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pages 337–348. ACM, 2009.

[142] S. Tullimas, T. Nguyen, R. Edgecomb, and S.-c. Cheung. Multimedia streaming using multiple TCP connections. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 4(2):12, 2008.

[143] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, and G. A. Gibson. A (In) Cast of Thousands: Scaling Datacenter TCP to Kiloservers and Gigabits. *Carnegie Mellon University Parallel Data Lab Technical Report CMU-PDL-09-101*, 2009.

[144] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller. Safe and effective fine-grained TCP retransmissions for datacenter communication. In *Proceedings of the ACM SIGCOMM conference on Data communication*, volume 39, pages 303–314. ACM, 2009.

[145] V. Vasudevan, H. Shah, A. Phanishayee, E. Krevat, D. Andersen, G. Ganger, and G. Gibson. Solving TCP incast in cluster storage systems. In *The 7th USENIX Conference on File and Storage Technologies (FAST'09)*, 2009.

[146] B. Wang, W. Wei, Z. Guo, and D. Towsley. Multipath Live Streaming via TCP: Scheme, Performance and Benefits. In *Proceedings of the 3rd ACM International Conference on emerging Networking EXperiments and Technologies (CoNEXT '07)*, pages 11:1–11:12. ACM, 2007.

[147] B. Wang, W. Wei, Z. Guo, and D. Towsley. Multipath live streaming via TCP: scheme, performance and benefits. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP '09)*, 5(3):25, 2009.

[148] D. Wischik, M. Handley, and M. B. Braun. The resource pooling principle. *ACM SIGCOMM Computer Communication Review*, 38(5):47–52, 2008.

[149] D. Wischik, C. Raiciu, A. Greenhalgh, and M. 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)*. USENIX Association, 2011.

[150] G. R. Wright and W. R. Stevens. *TCP/IP Illustrated*. Addison-Wesley Professional, 1995.

[151] M. Zhang, J. Lai, A. Krishnamurthy, L. L. Peterson, and R. Y. Wang. A Transport Layer Approach for Improving End-to-End Performance and Robustness Using Redundant Paths. In *Proceedings of the General Track: USENIX Annual Technical Conference*, pages 99–112, 2004.

[152] P. Zhang, H. Wang, and S. Cheng. Shrinking MTU to mitigate TCP incast throughput collapse in data center networks. In *Proceedings of the 3rd International Conference on Communications and Mobile Computing (CMC '11)*, pages 126–129. IEEE, 2011.

[153] Y. Zhang and N. Ansari. On mitigating TCP incast in data center networks. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '11)*, pages 51–55. IEEE, 2011.

[154] D. Zhou, W. Song, and M. Shi. Goodput improvement for multipath TCP by congestion window adaptation in multi-radio devices. In *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC '13)*, pages 508–514. IEEE, 2013.

[155] X. Zhuoqun, C. Zhigang, Y. Hui, and Z. Ming. An improved MPTCP in coded wireless mesh networks. In *Proceedings of the 2nd IEEE International Conference on Broadband Network & Multimedia Technology*, pages 795–799, 2009.

Aalto-DD 168/2014

ISBN 978-952-60-5925-9 (printed)
ISBN 978-952-60-5926-6 (pdf)
ISSN-L 1799-4934
ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Aalto University
School of Science
Department of Computer Science and Engineering
www.aalto.fi