Faculty of Economic Sciences, Communication and IT
Computer Science

Per Hurtig

Transport-Layer Performance for Applications and Technologies of the Future Internet

DISSERTATION
Karlstad University Studies
2011:65

Per Hurtig

Transport-Layer Performance for Applications and Technologies of the Future Internet

Karlstad University Studies
2011:65

Per Hurtig. *Transport-Layer Performance for Applications and Technologies of the Future Internet.*

DISSERTATION

Karlstad University Studies 2011:65
ISSN 1403-8099
ISBN 978-91-7063-404-8

Distribution:
Karlstad University
Faculty of Economic Sciences, Communication and IT
Computer Science
SE-651 88 Karlstad
Sweden
+64 54 700 10 00

www.kau.se

Print: Universitetstryckeriet, Karlstad 2011

”Har man tagit fan i båten
får man ro honom i land.”

Transport-Layer Performance for Applications and Technologies of the Future Internet

PER HURTIG
Department of Computer Science, Karlstad University

Abstract

To provide Internet applications with good performance, the transport protocol TCP is designed to optimize the throughput of data transfers. Today, however, more and more applications rely on low latency rather than throughput. Such applications can be referred to as data-limited and are not appropriately supported by TCP. Another emerging problem is associated with the use of novel networking techniques that provide infrastructure-less networking. To improve connectivity and performance in such environments, multi-path routing is often used. This form of routing can cause packets to be reordered, which in turn hurts TCP performance.

To address timeliness issues for data-limited traffic, we propose and experimentally evaluate several transport protocol adaptations. For instance, we adapt the loss recovery mechanisms of both TCP and SCTP to perform faster loss detection for data-limited traffic, while preserving the standard behavior for regular traffic. Evaluations show that the proposed mechanisms are able to reduce loss recovery latency with $30 - 50\%$. We also suggest modifications to the TCP state caching mechanisms. The caching mechanisms are used to optimize new TCP connections based on the state of old ones, but do not work properly for data-limited flows. Additionally, we design a SCTP mechanism that reduces overhead by bundling several packets into one packet in a more timely fashion than the bundling normally used in SCTP.

To address the problem of packet reordering we perform several experimental evaluations, using TCP and state of the art reordering mitigation techniques. Although the studied mitigation techniques are quite good in helping TCP to sustain its performance during pure packet reordering events, they do not help when other impairments like packet loss are present.

Keywords: TCP, SCTP, transport protocols, loss recovery, packet reordering, congestion control, performance evaluation.

Acknowledgments

The work presented in this thesis was made possible by many people. First of all, I would like to thank my supervisor Anna Brunström and my co-supervisor Johan Garcia for being excellent mentors and for their guidance, and patience, throughout my studies.

Secondly, I would like to thank my colleagues at the department of Computer Science for being a truly wonderful bunch of people, making the everyday work more fun. A special thanks to the people in my research group, the distributed systems and communications research group (DISCO), and the co-authors of the publications in this thesis.

Also, this work has been funded by several partners, including the European Regional Development Fund (through the projects E-CLIC and C-BIC), the European Union's 7th Framework Programme (through the NEWCOM++ project), the Knowledge Foundation of Sweden, NordForsk and the Swedish Internet fund. Thank you! During these projects I have also had the opportunity to work with a number of co-partners, including the Compare Karlstad Foundation, the Swedish Defence Research Agency (FOI) and the Communication Research Labs Sweden AB (CRL). Thanks for the good co-operation.

Last but not least I would like to thank my family for their support, especially my fiancée Johanna for her love, patience and support. When I get too stressed out, or focused at work, she reminds me of the important things in life. I also want to thank my parents for supporting me in everything I decide to do.

Karlstad, December 2011

Per Hurtig

List of Appended Papers

This thesis is based on the work presented in the following nine papers. References to the papers will be made using the roman numbers associated with the papers.

I Per Hurtig, Johan Garcia, and Anna Brunstrom. Loss Recovery in Short TCP/SCTP Flows. *Technical Report*, 2006:71, Karlstad University Studies, Sweden, December 2006.

II Per Hurtig and Anna Brunstrom. Enhancing SCTP Loss Recovery: An Experimental Evaluation of Early Retransmit. In *Computer Communications*, Elsevier, Vol. 31, Issue 16, pp. 3778-3788, October 2008.

III Per Hurtig and Anna Brunstrom. Improved Loss Detection for Signaling Traffic in SCTP. In *Proceedings of the IEEE International Conference on Communications (ICC 2008)*, Beijing, China, May 2008.

IV Per Hurtig and Anna Brunstrom. SCTP: Designed for Timely Message Delivery? In *Telecommunication Systems*, Springer, No 47, pp. 323-336, May 2010.

V Per Hurtig and Anna Brunstrom. Enhanced Metric Caching for Short TCP Flows. *Under submission*.

VI Per Hurtig, Wolfgang John, and Anna Brunstrom. Recent Trends in TCP Packet-Level Characteristics. In *Proceedings of the 7th International Conference on Networking and Services (ICNS)*, Venice/Mestre, Italy, May 2011.

VII Jonas Karlsson, Per Hurtig, Giovanni di Stasi, Anna Brunstrom and Andreas Kassler. Impact of Multi-path Routing on TCP Performance. *Under submission*.

VIII Per Hurtig and Anna Brunstrom. Packet Reordering in TCP. In *Proceedings of the IEEE GLOBECOM Workshop CCNet*, Houston, Texas, USA, December 2011.

IX Per Hurtig and Anna Brunstrom. Emulation Support for Advanced Packet Reordering Models. In *Proceedings of the IEEE International Conference on Communications (ICC 2010)*, Cape Town, South Africa, May 2010.

Some of the papers have been subjected to minor editorial changes.

Comments on My Participation

I am responsible for most of the written material and for carrying out the experiments in all papers but Paper I, Paper VI and Paper VII. In Paper I, the experimental design was done by the co-authors, while I was responsible for executing the experiments, analyzing the results and writing the report. In Paper VI, Dr. Wolfgang John performed the actual measurements, and wrote the description of them in Section 2. While the analysis presented in the paper was a joint effort among all the authors, I was responsible

for writing all parts but Section 2. In Paper VII, the experiments and the written material that does not address TCP was done by my co-authors. I did most of the experimental analysis and also contributed with the written parts about TCP and packet reordering.

Other Papers

Apart from the papers included in the thesis, I have authored or co-authored the following papers:

- Johan Garcia, Per Hurtig, and Anna Brunstrom. The Effect of Packet Loss on the Response Times of Web Services. In *Proceedings of the 3rd International Conference on Web Information Systems and Technologies (WebIST 2007)*, Barcelona, Spain, March 2007.

- Per Hurtig and Anna Brunstrom. Packet Loss Recovery of Signaling Traffic in SCTP. In *Proceedings of the International Symposium on Performance of Computer and Telecommunication Systems (SPECTS 2007)*, San Diego, USA, July 2007.

- Per Hurtig and Anna Brunstrom. SCTP Retransmission Timer Enhancement for Signaling Traffic. In *Proceedings of the 5th Swedish National Computer Networking Workshop (SNCNW 2008)*, Karlskrona, Sweden, April 2008.

- Johan Garcia, Per Hurtig, and Anna Brunstrom. KauNet: A Versatile and Flexible Emulation System. In *Proceedings of the 5th Swedish National Computer Networking Workshop (SNCNW 2008)*, Karlskrona, Sweden, April 2008.

- Tanguy Pérennou, Amine Bouabdallah, Anna Brunstrom, Johan Garcia and Per Hurtig. IP-level Satellite Link Emulation with KauNet. In *Proceedings of the International Workshop on Satellite and Space Communications (IWSSC 2009)*, Siena-Tuscany, Italy, September 2009.

- Per Hurtig, Tanguy Pérennou, Johan Garcia and Anna Brunstrom. Using Triggers for Emulation of Opportunistic Networking. In *Proceedings of the 2nd International Workshop on Mobile Opportunistic Networking (MobiOpp 2010)*, Pisa, Italy, February 2010.

- Mark Allman, Konstatin Avrachenkov, Urtzi Ayesta, Josh Blanton and Per Hurtig. Early Retransmit for TCP and SCTP. In *Internet RFCs, ISSN 2070-1721*, RFC 5827, April 2010.

- Tanguy Pérennou, Anna Brunstrom, Tomas Hall, Johan Garcia and Per Hurtig. Emulating Opportunistic Networks with KauNet Triggers. In *EURASIP Journal on Wireless Communications and Networking*, January 2011.

- Per Hurtig and Anna Brunstrom. Short Flows and TCP Metric Caching in Linux. In *Proceedings of the 7th Swedish National Computer Networking Workshop (SNCNW 2011)*, Linköping, June 2011.

- Per Hurtig, Andreas Petlund and Michael Welzl. TCP and SCTP RTO Restart. *Internet-Draft, draft-hurtig-tcpm-rtorestart-01*, work in progress, October 2011.

Contents

Abstract i

Acknowledgments iii

List of Appended Papers v

Comments on My Participation v

Other Papers vii

Introductory Summary 1

1 Introduction . . . 3
2 Research Objective . . . 5
3 Research Problems . . . 6
3.1 Timeliness of Data-Limited Traffic . . . 6
3.2 Packet Reordering . . . 11
4 Research Methodology . . . 14
5 Main Contributions . . . 15
6 Summary of Papers . . . 17
7 Conclusions and Future Research . . . 22

Paper I: Loss Recovery in Short TCP/SCTP Flows 31

1 Introduction . . . 33
2 Background . . . 35
2.1 TCP and SCTP . . . 35
2.2 Network Emulation . . . 36
2.3 KauNet: A Dummynet Extension . . . 37
3 Packet Loss and Positional Dependencies . . . 38
4 Experimental Design and Setup . . . 39
4.1 Experimental Overview . . . 39
4.2 TCP Experiments . . . 41
4.3 SCTP Experiments . . . 43

5 TCP Results . . . 44
5.1 Experimental Baseline . . . 44
5.2 Inflight Bandwidth Limiting Enabled . . . 47
5.3 Limited Transmit Disabled . . . 47
5.4 Minimum RTO . . . 49
5.5 Larger Initial Windows Disabled . . . 49
6 SCTP Results . . . 52
6.1 Performance Comparison . . . 54
6.2 Experimental Baseline . . . 55
6.3 Initial RTO . . . 59
6.4 Minimum RTO . . . 62
7 Discussion . . . 64
7.1 TCP . . . 64
7.2 SCTP . . . 69
8 Future Work . . . 71
9 Summary . . . 71
A TCP and SCTP Congestion Control . . . 75
A.1 Slow Start . . . 75
A.2 Congestion Avoidance . . . 76
A.3 Summary . . . 77
B Experiments Conducted . . . 78
B.1 TCP . . . 78
B.2 SCTP . . . 79
C TCP Implementation Details . . . 80
C.1 FreeBSD 6.0 . . . 80
C.2 Linux 2.6.15 . . . 85
D SCTP Implementation Details . . . 86
D.1 KAME . . . 86
D.2 LKSCTP . . . 87

Paper II: Enhancing SCTP Loss Recovery: An Experimental Evaluation of Early Retransmit 89
1 Introduction . . . 91
2 Packet Loss Recovery in SCTP . . . 93
2.1 Standard Loss Recovery . . . 93
2.2 Proposed Enhancements . . . 95
3 Early Retransmit . . . 96
3.1 Early Retransmit Algorithm . . . 96
3.2 Packet-based Early Retransmit Algorithm . . . 99
4 Experimental Setup . . . 100
4.1 Experimental Environment . . . 100
4.2 Experimental Design . . . 101
5 Results . . . 103

5.1 Static Burst Sizes and Static Inter-burst Gaps . . . 103
5.2 Static Burst Sizes and Random Inter-burst Gaps . . . 108
5.3 Random Burst Sizes and Static Inter-burst Gaps . . . 111
6 Conclusions . . . 113

Paper III: Improved Loss Detection for Signaling Traffic in SCTP 119
1 Introduction . . . 121
2 SCTP Loss Recovery . . . 122
3 Management of the Retransmission Timer . . . 123
3.1 Standard Algorithm . . . 124
3.2 Proposed Algorithm . . . 125
4 Experimental Setup . . . 126
5 Results . . . 128
6 Summary . . . 133

Paper IV: SCTP: Designed for Timely Message Delivery? 135
1 Introduction . . . 137
2 Background: SCTP and TCP . . . 138
3 SCTP: Designed for Timely Message Delivery? . . . 139
4 Loss Recovery Enhancements . . . 141
4.1 Early Retransmit . . . 141
4.2 An Illustration of Early Retransmit . . . 142
4.3 Modified RTO Management Algorithm . . . 144
4.4 An Illustration of the Modified RTO Management Algorithm . . . 146
5 Evaluation of Loss Recovery Enhancements . . . 147
6 A Nagle Algorithm for SCTP . . . 150
6.1 The Nagle Algorithm . . . 150
6.2 Adapting Nagle to SCTP . . . 152
6.3 An Illustration of the Modified SCTP Nagle Algorithm . . . 153
6.4 Evaluation of the Modified SCTP Nagle Algorithm . . . 154
7 Composite Evaluation . . . 157
8 Summary . . . 160

Paper V: Enhanced Metric Caching for Short TCP Flows 163
1 Introduction . . . 165
2 TCP Metric Caching . . . 166
3 Selective Metric Caching . . . 168
4 Experimental Evaluation . . . 170
5 Summary . . . 173

Paper VI: Recent Trends in TCP Packet-Level Characteristics 177
1 Introduction . . . 179
2 Data Collection and Processing . . . 180

3 General Results . . . 182
3.1 IP Traffic Characteristics . . . 182
3.2 TCP Characteristics . . . 183
4 TCP OOS Deliveries . . . 185
4.1 Methodology . . . 186
4.2 OOS Overview . . . 186
4.3 OOS Details . . . 189
4.4 Packet Reordering . . . 190
5 Conclusions . . . 192

Paper VII: Impact of Multi-path Routing on TCP Performance 197
1 Introduction . . . 199
2 Multi-path Routing . . . 201
3 TCP and Packet Reordering . . . 202
4 TCP Performance in a Simple WMN Topology . . . 204
4.1 Simulation Setup . . . 205
4.2 Results . . . 205
5 Chaska Metropolitan Network Topology . . . 208
5.1 Simulation Setup . . . 208
5.2 Results . . . 209
6 Related Work . . . 213
7 Conclusions . . . 213

Paper VIII: Packet Reordering in TCP 217
1 Introduction . . . 219
2 Packet Reordering . . . 220
3 Experimental Evaluation . . . 222
3.1 Mitigation Techniques . . . 222
3.2 Setup . . . 222
3.3 Reordering . . . 223
3.4 Traffic Patterns . . . 223
4 Results . . . 224
5 Discussion & Related Work . . . 228
6 Conclusions . . . 229

Paper IX: Emulation Support for Advanced Packet Reordering Models 233
1 Introduction . . . 235
2 Packet Reordering . . . 236
2.1 Causes and Prevalence . . . 236
2.2 Effects of Packet Reordering . . . 237
2.3 Proposed Mitigations . . . 238
3 Evaluating the Impact of Packet Reordering . . . 238
4 KauNet – Deterministic Network Emulation . . . 239

4.1 KauNet Overview 240
4.2 KauNet Reordering Feature 240
5 Advanced Packet Reordering Example 243
6 Summary 246

Introductory Summary

1 Introduction

During its short lifetime, the Internet has evolved from being a very small research network with limited capabilities to a world-wide public network with billions of users and services. One of the key enablers of this enormous growth is the Transmission Control Protocol (TCP) [73], which is used by a majority of all Internet applications to send and receive data. For instance, web browsers, chat clients and file transfer applications all use TCP. Although the Internet has had a tremendous success over the last decades, it has also faced a number of problems. One of Internet's most severe problems was a series of congestion collapses starting in 1986 [47]. During the collapses the capacity of the Internet dropped several orders of magnitude and virtually no traffic could get through.

Today, the problem of congestion has been mitigated largely by the congestion control mechanisms implemented in TCP [4] and over-provisioning of the Internet backbone. Although congestion control and over-provisioning have helped in reducing the risk of future collapses it has also created problems for novel Internet-based applications that require timely transmission of small amounts of data. For instance, a cloud-based application often take the concept of an ordinary application and moves it into the Internet (the cloud). To enable the same user experience as regular desktop applications, cloud-based services require that multiple short data flows of control information (e.g. keyboard input and screen updates) travel fast between service provider and customer [79]. Not only cloud-based services require timely data exchange. Voice over IP (VoIP), online gaming and Machine-to-Machine (M2M) communication are all examples of popular application types that have similar requirements [24, 45, 88]. Common for these application types is the latency requirements and the traffic pattern they generate. The pattern can be referred to as data-limited, as the traffic flows are either short or have a lower transmission rate than the network supports.

Unfortunately, data-limited flows are unnecessarily delayed, as protocols like TCP are designed to optimize the throughput for data-intensive long-lived flows (bulk traffic). Although the problems surrounding this throughput/latency trade-off are not new, cf. [22, 25], it has become more apparent with the increasing use of services based on data-limited flows. The trade-off between high throughput and low latency is inherent in most transport protocols that offer complex data transport services. TCP, for instance, guarantees that transmitted data will be received error-free, arrive in the same order it was sent and that the underlying network capacity is fairly shared with other TCP users. To enable these services, TCP is designed to detect and react to packet loss, out-of-order delivery of data and network congestion. Most of the mechanisms used to guarantee these services are, however, optimized for bulk traffic, which results in suboptimal performance for latency sensitive applications.

To solve the latency problems of TCP, the Stream Control Transmission Protocol (SCTP) [83] was standardized. As compared to TCP, the design goals of SCTP were to offer better robustness to link failures and to enable timely transmission of data [82]. Although the original intention was to use SCTP for telephony signaling traffic in IP [72]

networks, the possible benefits for other types of applications motivated a general standardization of the protocol within the Internet Engineering Task Force (IETF) [5, 44]. However, to support a general deployment the protocol had to adhere to a number of additional requirements, including the ability to fairly share network resources with existing protocols like TCP. To accomplish fair sharing and similar compatibility requirements, most mechanisms that governed the actual transmission of data were inherited from TCP. Unfortunately, the inherited mechanisms were used "as is" and not properly adapted for the low latency requirements that motivated the development of SCTP. Thus, SCTP does not generally provide better timeliness than TCP.

A common problem for TCP and SCTP is their inability to provide timely loss recovery for data-limited flows (cf. [7,41]). Both protocols use two loss recovery mechanisms called fast retransmit and retransmission timeout [4, 83]. Fast retransmit is preferred as it detects loss faster and reduces the sending rate less. The use of fast retransmit is, however, inhibited when small amounts of data are outstanding[1], a common situation for data-limited flows. Hence, data-limited flows have to rely on slow timeout-based loss recovery more often than bulk traffic. To make TCP and SCTP better suited for data-limited transport, we propose and evaluate two loss recovery adaptations: a packet-based early retransmit algorithm and a modified retransmission timeout management algorithm. The early retransmit algorithm enables fast retransmit for data-limited flows and the modified management algorithm provides faster timeouts for this kind of traffic.

Another problem for data-limited transport is the metric caching that TCP implementations employ. Basically, whenever a TCP connection is closed, metrics describing the state of the connection are cached [85]. Then, when a new connection is opened these metrics are used to optimize the connection. The problem for data-limited flows is their inability to fully utilize network resources, which in turn causes them to cache metrics that have not converged to a correct state. To mitigate the caching problem, we propose a selective caching strategy that only considers converged metrics.

We also address a SCTP specific issue related to timely transmission. When several small pieces of data are sent, they are delayed to allow enough data to arrive for SCTP to bundle them into a full-sized packet. Although this reduces transmission overhead it also increases the average transmission time. To strike a balance between low overhead and timely transmission, we suggest a bundling mechanism similar to the Minshall algorithm [63] that is often used in TCP. The Minshall algorithm does only delay data if small-sized packets are already outstanding. To implement the Minshall algorithm, we also suggest a new definition for full-sized packets in SCTP.

Not only novel applications make up the future Internet. Also, new networking technologies are used to increase Internet availability and to improve network performance in environments with scarce resources. For several reasons, many of these techniques are designed to be wireless. For instance, it enables users to be mobile while still having Internet access. Furthermore, wireless technologies can also enable much cheaper deployment, as no infrastructure is needed. Several users can, for instance, help in for-

[1] The term outstanding refers to data that are transmitted by the sender but not yet acknowledged by the receiver.

warding traffic among each other to reach the Internet.

One problem related to mobility and the lack of infrastructure is the absence of fixed network paths with predictable resources. While users that have fixed connections to the Internet only are affected by competing traffic in the network, many effects can have an impact on users of these novel technologies. Basically, it is hard to achieve good performance and connectivity for wireless transmissions that span multiple hops [34]. To tackle this problem, multi-path routing is often used. This kind of routing allows multiple paths to be used in an effort to e.g. load-balance traffic and increase the probability that packets arrive at their destination [40]. When multiple paths are utilized for a single flow, packets might be forwarded in parallel and thus arrive out-of-order. Even though TCP and SCTP can re-sequence incoming traffic to guarantee in-order delivery to the application, none of the protocols are appropriately designed to tackle the amount of reordering that can be created by e.g. multi-path routing [12]. The problem is, at least partly, related to the loss recovery mechanisms. Basically, the arrival of consecutive out-of-order packets is used by TCP and SCTP as a sign of packet loss. Therefore, if several packets are consecutively reordered, the protocols might interpret this as packet loss and thereby unnecessarily invoke mechanisms that lower the send rate. While standard TCP [4, 73] is not designed to be robust to reordering, there are a number of TCP variants that provides such functionality (cf. [13,15,17]). To increase the understanding of how TCP and reordering robust variants of the protocol perform when packets are reordered due to novel networking technologies, we evaluate a number of TCP versions in several environments.

The remainder of this introductory summary is structured as follows. Section 2 defines the objective of the thesis and formulates two research problems to achieve the objective. Section 3 details the research problems and presents related work. The first part of the section introduces the problem of timely transport of data-limited flows and our proposed enhancements. The second part discusses packet reordering and our evaluations. Section 4 describes commonly used research methodologies in computer networking and those employed in this thesis. Section 5 highlights the main contributions of the thesis. Section 6 briefly summarizes each appended paper and their relation to each other. Finally, Section 7 ends the introductory summary by presenting some conclusions from our work and by pointing out possible future research in the area.

2 Research Objective

As discussed in the previous section, the transport-layer faces several challenges due to the evolution and usage of the Internet. To address these challenges we define the overall objective of this thesis as:

> *To improve Internet application performance by analyzing and adapting the transport-layer and its interaction with novel traffic types and technologies.*

To satisfy this objective, we focus on two research problems. The first problem considers timeliness issues found in TCP and SCTP, when data-limited flows are trans-

ported. More specifically:

> *How can TCP and SCTP be adapted to provide better timeliness for data-limited traffic?*

Problems related to packet reordering, which can be imposed by novel networking techniques, are then considered. More precisely:

> *How is the behavior and performance of TCP affected by packet reordering, and is it possible to ameliorate negative effects caused by reordering?*

In the next section, we expand on the above-mentioned research problems. We also describe the relation between our work and previous work regarding these problems.

3 Research Problems

There is a significant amount of research that relates to our work and we describe the most relevant in this section. While most work described here is focused on either TCP or SCTP, the reasoning and the results are often valid for both protocols.

3.1 Timeliness of Data-Limited Traffic

Good network performance has traditionally been synonymous with high throughput. Research and development related to networking have therefore been focused at increasing capacity, and utilization, of networks. Most performance issues are, however, trade-offs. Data-limited flows, for instance, do not contain enough data to fully utilize the underlying network capacity, and are therefore unable to benefit fully from capacity increases. The problem is illustrated in [25], where the relationship between capacity and latency improvements over the last 25 years is shown. The authors show that the enormous increase in bandwidth capacity (more than 100 times) has only improved flow completion times (FCT) slightly, especially for flows containing small amounts of data. Thus, increasing the network capacity does not solve latency problems for data-limited flows. Actually, the throughput-centric design of transport protocols like TCP delay such flows significantly [25].

Timely transport of data is not a new problem (cf. [22, 25]), although it has become more apparent now as more applications depend on low latency. This has spawned a renewed interest in the topic, resulting in a number of proposals on how to reduce latency for time-sensitive traffic flows. For instance, several new transport protocols have been proposed. These include, for example, DCCP [51, 52] and RTP [80]. So far, new protocols have not seen widespread deployment for generic Internet use. Their limited use is most likely due to their inability to get through NATs and middle-boxes. Common for many new protocols that address timeliness issues is also that they trade reliability for lower latency, by not retransmitting lost data. This trade-off is also made by several proposed TCP variants, like TLTCP [67] and TCP-RC [60]. The difference is that both

TLTCP and TCP-RC are based on TCP and must therefore relax TCP's reliability guarantees. It is, however, possible to achieve lower latency and still guarantee reliability. Brosh et al. [19] devise a number of TCP parameter suggestions to better provide low latency. Researchers at Google have also proposed a more aggressive TCP behavior to reduce latency. They suggest that the initial congestion window of TCP, i.e., the amount of data allowed to be sent immediately after connection establishment, is increased from three to ten full-sized segments [23, 26] to reduce the number of round-trip times that short flows experience. In general, reducing the number of round-trip times is considered to be the most effective way to reduce latency for data-limited flows [11, 26, 84]. The Google proposal has, however, met some resistance as it is considered by some researchers to actually increase the overall latency of the Internet (cf. [36]). Other TCP-related proposals include a modified connection establishment phase where TCP is allowed to transmit and process data during the initial handshake [21, 75]. This shortens a typical flow with one round-trip time. Another approach is to modify components at other networking layers to support low latency requirements. J. Gettys has, for instance, argued for reduced buffer sizes in equipment connected to the Internet [35]. This would, according to Gettys, reduce the average latency of applications and enable a more fair resource sharing between time-sensitive applications and applications that try to maximize throughput. It is also possible to perform cross-layer adaptations where TCP, or another protocol, collaborates with entities at other networking layers to achieve low latency. The Rate Control Protocol (RCP) [25] is an example of such a protocol.

In this thesis we only consider transport-layer aspects of this problem, where parts of the protocol is adapted to provide better timeliness. Interestingly, this problem is rather general and it is hard to say that a certain set of mechanisms is responsible for delaying transmission of such flows. Basically, the entire TCP was designed with bandwidth utilization in mind and therefore the problems are scattered all over the protocol. The same goes for SCTP, as it was built largely with TCP mechanisms, although the intention was to design it for timely message delivery.

To improve the timeliness of data-limited transport in TCP and SCTP, we look at four different mechanisms. First, we consider two mechanisms related to the inability of the protocols to quickly recover from packet loss when transmitting data-limited flows. We then address an issue in the interaction between the loss recovery and the connection establishment phase of TCP, which can cause newly opened data-limited flows to perform unnecessarily poor. Finally, we propose a new message bundling approach in SCTP to promote low transmission overhead while at the same time maintaining timeliness. Each of these issues is further discussed below.

Loss Recovery for Data-Limited Flows

As mentioned earlier, TCP and SCTP use two packet loss recovery methods. We start by describing the most basic method, called retransmission timeout (RTO). The RTO mechanism is defined in [70] for TCP and in [83] for SCTP. Basically, the transport protocol requires every piece of transmitted data to be acknowledged by the receiver. If an acknowledgment, for a given piece of data, is not received within a certain time

interval, the sender retransmits the data. The length of the time interval (often simply referred to as RTO) is based on periodic measurements of the round-trip time (RTT) between the sender and the receiver. To not unnecessarily retransmit data that for some reason is delayed, and not lost, the RTO is often calculated to span a long time interval. The long timeout interval can be a serious problem for applications that require timely transmission of data.

To enable faster loss recovery, the fast retransmit mechanism was designed [47]. It is currently standardized in [4] for TCP and in [83] for SCTP. To detect loss, this mechanism relies on duplicate acknowledgments from the receiver. As mentioned earlier, a receiver acknowledges each received piece of data with an acknowledgment. This is only true if the received data was expected. If an unexpected packet arrives, the receiver will send a duplicate acknowledgment. A duplicate acknowledgment actually has two meanings. First, it tells the sender that a packet was received. Second, it tells the sender that an unexpected packet was received. For instance, consider the scenario of ten packets traveling from a sender to a receiver. Further, let us assume that the second packet, for some reason, is lost within the network. Then, packets three to ten will cause the receiver to send duplicate acknowledgments. The problem with the fast retransmit mechanism is that three duplicate acknowledgments are needed. For data-limited flows there might not be enough data to generate three duplicate acknowledgments following a loss. Thus, data-limited flows have classically been forced to slow RTO-based loss recovery. In [8], for example, it is shown that more than half of all retransmissions conducted by a busy web server were performed by the RTO mechanism.

A number of proposals have been made to enable fast retransmit for data-limited flows as well. Many of the proposed solutions try to induce more duplicate acknowledgments, thereby increasing the chance of fast retransmit. Limited Transmit [2] allows a sender to transmit new packets when receiving duplicate acknowledgments. The intention behind this strategy is to generate additional duplicate acknowledgments to trigger fast retransmit. However, new data is not always available at the sender. This is solved in [7], which allows a sender to transmit zero-byte packets to induce further duplicate acknowledgments. Another example can be found in [61], where it is suggested that packets are split. The splitting should ensure that at least four packets are outstanding at all times. In this way, one original packet might induce several duplicate acknowledgments. A cleaner approach is proposed in [71] where the amount of duplicate acknowledgments needed for fast retransmit is reduced from three to one. While this might be a good solution for certain types of networks, it also makes fast retransmit unnecessarily sensitive to packet reordering. We evaluate and propose a modified version of the early retransmit algorithm [1]. Early retransmit is an algorithm where the number of duplicate acknowledgments needed for loss detection is dynamically adapted based on the amount of outstanding data. The modified version makes the algorithm work correctly when small-sized packets are transmitted.

While the early retransmit algorithm would help many data-limited flows in performing faster loss recovery, it cannot help when a single packet is traveling between sender and receiver, or if all outstanding packets are lost. In such situations, the only

solution is to rely on the time consuming RTO algorithm to detect loss. To decrease the time needed for the RTO algorithm to detect loss, two approaches are generally suggested. Either the parameterization of the algorithm calculating the RTO is changed or the actual algorithm is modified in some way. In [3] different parameterizations are evaluated. The authors find that the parameter controlling the minimum bound of the RTO timer (RTO_{min}) affects loss recovery performance the most. Thus, by lowering RTO_{min} retransmissions occur faster. The importance of the minimum bound is also demonstrated in [86] where the bound is altogether removed. By using a zero-bound together with a timer that operates on micro-second granularity, the authors are able to significantly reduce the time needed for RTO. The problem of reducing the minimum bound is, however, that spurious retransmissions can occur. Such retransmissions could hurt the timeliness of the protocol even more. To avoid that the minimum bound becomes too low, [74] proposes an adaptive minimum RTO. There are also several suggestion on how to undo the effects of spurious retransmissions [3, 56, 77]. In [28] a new RTO algorithm is presented. The authors try to fix some well-known problems in the standard algorithm, by specifying a completely new algorithm that relies on more advanced calculations. The evaluation of the new algorithm shows that it is more predictable in its operation, but also requires much more processing capabilities. Kesselman et al. [50] also propose a new algorithm that does not only rely on measured round-trip times, but also on the size of the congestion window.

We suggest a modification that is complimentary to the above-mentioned approaches. Our modification does not provide a completely new algorithm or a change of parameters. Instead, the modification changes how the timer is managed. With the current management algorithm, only a few packets are retransmitted after RTO seconds if they are lost. In fact, most retransmissions will occur after at least RTO+RTT seconds. To reduce latency, we modify the algorithm to ensure that timeouts always happen exactly RTO seconds after the data was sent. In [87], almost the same approach is used. However, the design of their algorithm does not ensure an exact retransmission after RTO seconds, as they only estimate the offset added by the management algorithm. Furthermore, Ludwig et al. [57] also identified the offset imposed by the algorithm, and suggest that it should be removed. However, in a later paper co-authored by Ludwig [28] the authors recommend not to change the management algorithm, as it could result in too aggressive RTO calculations, with unnecessary retransmissions and congestion control invocation as outcome. As we target data-limited traffic, the conditions are different. For instance, congestion control invocation does not severely affect the performance of data-limited flows, as the amount of outstanding data is already small. For more information on the conditions surrounding the use of the algorithm, please see [42].

TCP Metric Caching

When a TCP connection is established a data structure called the TCP Control Block (TCB) is usually created. The structure is used to store information describing the state of the TCP connection, including e.g. congestion state information, the current value of the RTO timer and the amount of outstanding data. The information contained in the

TCB is typically maintained on a per-connection basis. There is, however, a mechanism called TCP Control Block Interdependency (TCBI) [85] that allows multiple connections, on a host, to share TCB information. The motivation behind the interdependency mechanism is straightforward. Assume that a number of server applications, running on the same machine, communicate with the same number of client applications on another machine, using TCP. As these flows share the network path they also share the underlying network resources. Thus, by sharing the state information between the separate connections, TCP can adapt the operation of each flow to achieve better system performance. The sharing of state information among flows has been proven successful [27, 78] and there exist similar approaches to share network information on an end-host [9] and also among several hosts [81].

The TCBI does not only allow information to be shared among concurrent flows, it also enables sharing between consecutive flows. That is, if a TCP connection is established, used and then closed, TCB information will be cached so that future flows to the same destination can use the information to optimize its state. As network state information can be valid for tens of minutes [69], such caching could be useful. Metrics used for the optimization typically include measured round-trip times and congestion state information. The problem for data-limited flows is that they typically do not utilize the available capacity of the network. Therefore, they are often unable to correctly assess the congestion state of the network. This could result in caching of congestion metrics that do not represent the true state of the network, resulting in poor performance for connections reusing these metrics. To circumvent this problem, we suggest a selective caching strategy that does not consider metrics directly related to the congestion state.

SCTP Message Bundling

When transmitting really small amounts of data, in separate packets, the overhead often becomes drastic. For instance, transmitting 1 byte messages in SCTP would yield 4800% of overhead as each data packet contains a 48 byte header. Generally, this problem is known as the small-packet problem and was first discovered in the 1960s [68].

To relieve the network from such overhead J. Nagle specified a both simple and effective algorithm in 1984 [68]. The algorithm can be described as follows: if there are outstanding data, then the transport protocol buffers all user data, until the outstanding data has been acknowledged or until the protocol can send a full-sized packet. Thus, the algorithm delays the transmission of small-sized packets to promote more data to be bundled in a single packet. The algorithm showed to work well in inhibiting excessive transmissions of small packets, and became a requirement for TCP-enabled hosts in 1989 [18].

However, as the algorithm inhibits transmission of packets it also prolongs the average transmission time. Furthermore, the algorithm is known to interact badly with other mechanisms in TCP and SCTP, e.g. the delayed acknowledgment mechanism. Actually, in some situations the algorithm can stall the transmission of a single packet with up to 500 ms [66].

To solve such latency problems, but still inhibit small packets from being sent, a

number of modifications have been proposed (cf. [63, 64]). G. Minshall proposes a simple modification in [63], which probably is the most well known modification. Instead of always delaying small-sized packets, the Minshall algorithm does only delay small-sized packets if there already is a small-sized packet outstanding. This slight modification of the Nagle algorithm has been proven efficient and is, for example, implemented in the Linux TCP implementation [43].

To support a similar algorithm in SCTP, we redefine the notion of a full-sized SCTP packet. Using the new definition and Minshall's algorithm, SCTP is able to provide both bundling and timely transmission.

3.2 Packet Reordering

The reason for packets to be reordered is simply the existence of parallelism in the forwarding path, which allows packets to overtake each other. For instance, if there are parallel paths between different nodes or parallelism within networking hardware, packets could be forwarded in parallel and possibly be reordered. In the general Internet, this type of parallelism can be caused by e.g. router design [54], route fluttering [69] or wireless retransmissions [55].

In this section we present negative effects that follow from packet reordering, when TCP is used as the transport protocol, and also discuss how common packet reordering is. Furthermore, there is a large body of work on how to circumvent problems related to reordering. We give a background to the ideas behind such techniques in this section.

Effects and Prevalence of Packet Reordering

Both reordering and loss cause packets to arrive out-of-order. If the degree of packet reordering is large enough to generate at least three consecutive duplicate acknowledgments, TCP and SCTP will falsely assume packet loss. In such scenarios, not only the loss recovery is triggered unnecessarily but also the congestion control, as the protocols interpret packet loss as a sign of congestion [4, 83]. Reordering can therefore cause unnecessary retransmissions and lowered transmission rates. Reordering of data packets might also conceal real packet loss, increase buffering demands at receivers and cause other performance problems. However, false retransmissions are often regarded as the most significant problem [6]. Furthermore, reordering of TCP acknowledgments can also affect the behavior and performance of TCP significantly. For a more complete list of problems related to reordering and a deeper discussion about the particular problems, please see [12].

A simple example of the effects that reordering has on TCP performance is given in Figure 1. This graph shows the throughput of a single Linux TCP flow, over a 5 Mbps link, when the data is subjected to different levels of reordering. As indicated by the graph, only a small amount of reordering is necessary to severely hamper the performance. More details on these particular results will follow in Paper VIII.

Paxson [69] and Bennett et al. [12] highlighted the existence of reordering, and the problems related to it, in the late 1990's. While they found reordering to be extremely

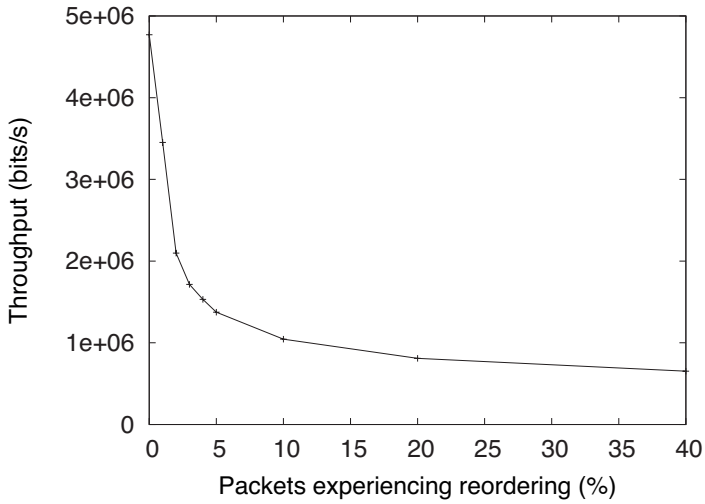

Figure 1: *TCP throughput as a function of increased levels of packet reordering.*

common, e.g. Paxson found 12% and 36% of all connections in two datasets to include reordered segments, later studies have displayed a large variety of reordering prevalence. For instance, Gharai et al. [37] found that 47% of all sessions experienced reordering. Rewaskar et al. [76] found that the amount of segments arriving out-of-order varied between $17 - 51\%$ and that packet reordering caused $0.2 - 14.1\%$ of these arrivals. Mellia et al. [62] reports an average of $5 - 8\%$ of all traffic in an Internet backbone to be out-of-order, where approximately 10.5% of these are due to reordering. We have also performed measurements in the Swedish University Network (SUNET) backbone, and found the amount of connections containing out-of-order arrivals to be approximately 17%, with reordering being responsible for $5 - 22\%$ of these arrivals. The results from the SUNET measurements are presented in Paper VI. Thus, it is hard to determine a general prevalence of reordering within the Internet.

Most likely, reordering is path-dependent as some networking techniques and equipment, only deployed in parts of the Internet, can cause parallel forwarding and thus reordering. Researchers also tend to use different techniques to measure reordering, different definitions of reordering and also measure different types of traffic. This makes it even harder to determine how common reordering is. For example, [10, 12, 58] conduct active measurements using probe traffic, while [62, 69, 76] do passive measurements on collected data. Furthermore, the focus ranges from reordered packets to out-of-order packets. As effects like e.g. packet loss also cause packets to arrive out-of-order, the notion of out-of-order packets is a much broader classification than packet reordering.

Regardless whether reordering is a general Internet problem or not, it can be a problem in parts of the Internet that use parallel forwarding to more efficiently utilize network resources and increase connectivity. Such novel techniques include multi-path

routing [31] used in e.g. wireless mesh networks (WMNs), load-balancing over multiple network interface (e.g. WLAN and 3G) [20, 46, 49] and vertical hand-overs [39]. To use such techniques successfully, it is possible to avoid e.g. unnecessary retransmissions and invocations of congestion control at the transport-layer by taking protective measures.

Avoiding Unnecessary Retransmissions and Congestion Control

To avoid the negative effects that reordered packets have on the loss recovery and congestion control, there are typically three approaches to take. These approaches can be described as reactive, pro-active or mixed. Reactive proposals try to undo bad decisions like congestion control invocation when the protocol has discovered that it has mistaken reordering for packet loss. Pro-active proposals try to inhibit bad decisions by, for example, dynamically increasing the duplicate acknowledgment threshold so that duplicate acknowledgments due to reordering cannot trigger fast retransmit. The mixed approach combines reactive and pro-active mechanisms.

There are few reordering mitigation schemes that are purely reactive. Blanton and Allman [15] have, however, suggested and evaluated such a mechanism. The mechanism they evaluate simply restores the congestion state whenever a retransmission is deemed as unnecessary. Unnecessary retransmissions can happen for other reasons as well. In Section 3.1 the problem of unnecessary retransmissions were discussed in the context of aggressive RTO settings, and a few techniques to detect such retransmissions were referenced [3, 56, 77]. These detection strategies can also be used in the context of reordering, although reordering is more likely to cause unnecessary fast retransmissions. To detect false fast retransmissions it is possible to use TCP timestamps (TS) [48] or duplicate selective acknowledgments (D-SACKs) [16].

Pro-active methods sometimes rely on previously observed reordering events to adapt the duplicate acknowledgment threshold. There are, however, methods that are completely pro-active and do not require any prior knowledge. One such example is TCP with Delayed Congestion Response (TCP-DCR) [13], which dynamically increases the duplicate acknowledgment threshold to avoid unnecessary retransmissions. While the threshold increase enables TCP to better disambiguate between packet loss and packet reordering, it delays the loss recovery. Another pro-active example is TCP for Persistent Reordering (TCP-PR) [17] where duplicate acknowledgments are not used at all for loss recovery, but instead an extra timer is used for fast loss recovery.

Mixed mitigations methods include e.g. Reordering-Robust TCP (RR-TCP) [90] and the Linux TCP implementation. Both RR-TCP and Linux dynamically adjusts the duplicate acknowledgment threshold and retracts unnecessary congestion control decisions, when false retransmits have been detected. RR-TCP, however, performs much more complex calculations to derive a suitable threshold.

In addition to evaluations accompanying a new or modified proposal, some independent evaluations have been made. Leung et al. [55] provide the most complete study, in terms of evaluated proposals. Unfortunately, Leung et al. evaluate the proposals using network simulation and not real implementations, making it hard to compare their find-

ings to ours. Feng et al. [29] have also compared different proposals. However, their evaluation does not consider other network effects (e.g. packet loss) that typically co-exist with reordering. Packet loss is, at least partly, considered in [89] which evaluates several proposals through simulations. The authors find that although most proposals are effective in avoiding poor performance due to reordering, reordering mitigation schemes are not robust to impairments like packet loss. This is consistent with the findings in this thesis.

In this thesis we investigate a number of mitigation techniques belonging to each of the above categories. First, we experiment with the built-in mitigation techniques that are enabled by default in Linux. As mentioned earlier, the approach taken by Linux belongs to the mixed category, consisting of both a reactive and a pro-active mechanism. The reactive part, which tries to undo unnecessary congestion control decisions, depends on the detection of spurious retransmissions using either D-SACKs or TCP TS. The pro-active part uses the maximum observed reordering length to dynamically increase the duplicate acknowledgment threshold. We also evaluate TCP with robustness to non-congestion events (TCP-NCR) [14], which is an evolution of the previously mentioned TCP-DCR. TCP-NCR takes a pro-active approach where the duplicate acknowledgment threshold is increased so that loss recovery is delayed with one round-trip time. The extra time is then used to better discriminate between packet loss and packet reordering. There exist two versions of TCP-NCR, one careful version that sustains half the sending rate during the loss/reordering detection phase and one aggressive version that does not change the transmission rate.

4 Research Methodology

When conducting research in computer networking, two types of research methodologies are often used: analytical and experimental. The analytical approach often involves mathematical techniques to model and solve research problems. The benefit of this approach is the ability to find, and evaluate, theoretical limits of different problems and also to model problems that, for some reason, are impossible to evaluate in real life. For instance, it would be impossible to do real experiments on hardware that does not yet exist and it is certainly infeasible to construct an experimental environment involving millions of network hosts. Simulation is an experimental method closely related to analysis, as it also operates on models of real entities. However, there is a significant difference between analysis and simulation. While analysis tries to predict the outcome of a process, simulations typically execute a software implementation of the entire process to obtain a result. When problems are modeled, or simulated, many details are often left out to speed up and simplify the evaluation. Although a reduction of details is needed to enable large-scale analysis and analysis involving non-existing technologies, taking away too much details can affect the validity of the results. One way to mitigate such validity problems is to complement modeling and simulation with other experimental research methods.

Such methods include emulation and live experiments, and are typically much more

confined to available technology and small-scale experiments. Obvious problems related to live experiments is that results become dependent on the experimental environment and that it is hard to control and repeat experiments. For instance, if some physical component in the environment is malfunctioning, the results might be useless. Also, it is often infeasible to conduct large-scale experiments, as the costs involved in constructing such an environment could be too high. The emulation approach provides a mix of simulation and real experiments, where it is possible to model parts of an experiment while using some real entities. The emulation approach can thus provide an experimenter the freedom of simulating arbitrarily complex network conditions in which real TCP implementations can be evaluated.

Although we use a mix of the above methods for the work described in this thesis, our main approach is network emulation. The main reason for using emulation is the attractiveness of evaluating real implementations under arbitrary and repeatable network conditions. For a majority of the experiments described in the thesis, the network emulator KauNet [32] was used. In addition to functionality provided by most emulators, such as emulation of bandwidth, delay and probability-based packet loss, KauNet is also able to apply emulation effects in a precise and deterministic way. That is, KauNet can be used to emulate network properties like bandwidths, delays and packet loss for specific packets in a traffic flow. By using the deterministic features of KauNet, we have been able to perform repeatable experiments on real protocol implementations, with a high level of control over the emulated scenario. We have also extended KauNet continuously during our work, to add support for e.g. deterministic emulation of packet reordering. An introduction to KauNet and its packet reordering capabilities is provided in Paper IX.

5 Main Contributions

The main contributions of this thesis can be summarized as follows:

1. To support timely transport of data-limited flows, we have evaluated and proposed adaptations of several mechanisms in TCP and SCTP. Specifically:

   - We show that the loss recovery mechanisms in TCP and SCTP become inefficient when the amount of outstanding data is small. This contribution is presented in Paper I.
   - We have proposed and evaluated a modified early retransmit algorithm. The proposed modification enables early retransmit to work correctly for transmission of small packets, and was shown to significantly reduce the time needed for loss recovery. The modified early retransmit algorithm has been adopted by the IETF and is included in the standardized algorithm [1]. This contribution is presented in Paper II and Paper IV.
   - We have also developed, and experimentally evaluated, a modified algorithm for managing the retransmission timer. The algorithm enables quicker loss

detection for connections that are forced to use retransmission timeout for loss recovery. Although this contribution considers only SCTP, an almost identical modification is possible for TCP. A slightly modified version of this algorithm is now being proposed in the IETF standardization process [42]. Paper III and Paper IV present this contribution.

- We have also suggested a definition of full-sized packets in SCTP. Using this definition, SCTP is able to reduce the overhead involved in transmitting several small packets, while at the same time enabling timely data transport. This contribution is presented in Paper IV.
- Finally, we have proposed a selective metric caching strategy for TCP. Most TCP implementations cache state metrics of TCP connections and use them for later connections. For data-limited flows the traditional caching strategy might harm the performance, as such flows often are unable to correctly estimate and use congestion-related metrics. Our selective caching strategy only considers non-congestion related metrics, and is therefore better suited for data-limited flows. This contribution is presented in Paper V.

2. Trends in TCP traffic characteristics have been analyzed in a series of Internet backbone measurements conducted in both 2006 and 2009. We show, for instance, that most TCP connections are data-limited (short) and that well-known TCP features such as selective acknowledgments (SACKs) [59] and Path MTU discovery [65] are commonly used. We also present detailed studies of TCP connections and show that packet reordering and other networking anomalies cause a large fraction of all segments to arrive out-of-order. This contribution is presented in Paper VI.

3. Evaluations of different TCP variants and their ability to provide good performance in environments where packet reordering occurs have also been conducted. More specifically:

   - We show that TCP is not generally able to benefit from the capacity increase provided by per-packet load-balancing in WMNs. The impact of the resulting reordering is proven to be too severe, even for TCP implementations with protection against reordering. However, under ideal conditions, with little competing traffic and many available paths, it was possible for reordering robust TCP implementations to improve their performance slightly using load-balancing. These contributions are presented in Paper VII.
   - We also describe why TCP is vulnerable to packet reordering, and why state of the art mitigation techniques are not able to fully solve this problem. Further, we dismantle the common misbelief that short-lived flows are unable to benefit from reordering mitigations. In addition, we show that packet loss has a dominating performance effect that practically inhibits any gain provided by the mitigation techniques. These contributions are presented in Paper VIII.

- To support experimentation with packet reordering, we have also implemented support for performing deterministic emulation of reordering in the KauNet network emulation system. This contribution is presented in Paper IX.

6 Summary of Papers

This section summarizes the nine papers of the thesis. Each paper is briefly described, addressing its position in the thesis as well as its contributions and limitations.

Paper I – Loss Recovery in Short TCP/SCTP Flows

This report experimentally evaluates several TCP and SCTP implementations in an emulated network environment. The goal of the evaluation is to reveal how performance and behavior is affected when packet loss occurs at different positions in a flow. That is, do transport protocols perform differently depending on when a packet is lost? The results show, among other things, that packet loss recovery becomes a lengthy process whenever a small amount of data is outstanding, a typical situation for data-limited flows. Basically, a sender must have enough outstanding data to generate at least three duplicate acknowledgments. In addition to verifying this loss recovery problem, we identify several implementation mistakes in the transport protocol implementations. While the main focus of this paper is to evaluate TCP and SCTP loss recovery performance we also introduce the different protocols, their congestion control mechanisms and the KauNet network emulator that enabled the position-based packet loss emulation. This report forms the background for the first part of the thesis, as it motivates the need for improving the timeliness of loss recovery.

To study the effects of position-based packet loss for data-limited flows we conducted the evaluation in a controlled and somewhat limited environment. For instance, no complex traffic patterns or scenarios with competing traffic were studied.

Paper II – Enhancing SCTP Loss Recovery: An Experimental Evaluation of Early Retransmit

In this paper, we evaluate the early retransmit algorithm to determine whether it can improve loss recovery performance for data-limited traffic. To support timely loss recovery when the amount of outstanding data is small, early retransmit dynamically lowers the duplicate acknowledgment threshold. We also modify early retransmit to work for traffic flows with small packets. The resulting mechanism is shown to outperform the standard loss recovery mechanisms of SCTP. We find, for instance, that the time needed to recover from packet loss can be reduced with up to 62% if the proposed early retransmit algorithm is used in traffic scenarios where fast retransmit is impossible to use. During the evaluation of early retransmit, we also discover that the retransmission timeout

mechanism sometimes unnecessarily extends the loss recovery time with approximately one round-trip time.

When lowering the duplicate acknowledgment threshold, the transport protocol automatically becomes less robust to packet reordering. In this paper we do not address this issue. We have, however, outlined some possible strategies to avoid problems related to reordering in [1]. The most straightforward strategy is to inhibit further use of early retransmit if the protocol detects that it is retransmitting data unnecessarily. Similarly to Paper I, our experiments were conducted in a controlled environment with simple traffic patterns.

Paper III – Improved Loss Detection for Signaling Traffic in SCTP

In this paper we set out to solve the above-mentioned problem with the retransmission timeout mechanism. We discover that the algorithm managing the retransmission timer unnecessarily delays loss recovery. The problem is not due to an implementation mistake, but rather a consequence of the algorithm design itself. Basically, the management algorithm is designed to continuously extend the timeout interval if the flow is receiving acknowledgments. In practice, this causes most timeouts to occur roughly one round-trip time later than the calculated retransmission timeout. We modify the specification of the algorithm and evaluate the modified algorithm. The results show that significant performance improvements can be achieved using the new algorithm. The time needed to recover from packet loss is reduced with up to 43% in a simple evaluation scenario where the small amount of outstanding data forces SCTP to recover all packet losses using the retransmission timer.

While the modified algorithm is suitable for data-limited flows, it might be overly aggressive for other types of traffic. For instance, when transmitting bulk traffic the extra delay introduced by the original algorithm might work as a safety margin, to inhibit unnecessary retransmission timeouts. As bulk flows have much more data to send, they will typically be able to use fast retransmit for loss recovery. Thus, delaying a retransmission timeout might give fast retransmit enough time to be used instead. Furthermore, bulk flows rely on a large congestion window for good performance. When a timeout occurs the congestion window is reduced far more than if a fast retransmit is conducted. To make a unified proposal that works well for both bulk traffic and data-limited flows we have recently proposed a modified algorithm that, similarly to early retransmit, only is activated when the amount of outstanding data is small [42].

Paper IV – SCTP: Designed for Timely Message Delivery?

In this paper we question if SCTP really was designed with time critical traffic in mind. We evaluate the loss recovery contributions from Paper II and Paper III jointly, using a more realistic setup than before. For example, we evaluate the loss recovery using more complex traffic patterns and also verify that our proposed changes are not unfair to competing traffic. The joint evaluation shows reductions in transfer times with $30-50\%$,

using the proposed algorithms. We also suggest a modified Nagle-like algorithm to allow SCTP to reduce transmission overhead while still being able to transfer data in a timely fashion. In situations where both small and large pieces of data are sent, the Nagle-like algorithm is able to reduce transfer times with up to 70%. Common for both the loss recovery enhancements and the modified Nagle algorithm is that they are better suited for time critical traffic than the original mechanisms, which were designed for TCP bulk traffic.

As compared with the loss recovery evaluations in Paper II and Paper III, this paper includes more realistic and complex scenarios. The evaluation is, however, mostly targeted at signaling environments where the parameterization of the network stack is tuned somewhat different from the general Internet scenario. Also, the evaluations do only consider SCTP.

Paper V – Enhanced Metric Caching for Short TCP Flows

This paper does not solely focus on the loss recovery, but rather on the relationship between the loss recovery and the metric caching that TCP implementations employ. To improve the performance of data-limited flows, we propose and evaluate an enhanced metric caching mechanism for TCP that does not consider the congestion state of short flows. Usually, TCP implementations cache metrics describing the state of a connection. This information is later used by new connections to optimize their startup. Short and data-limited flows are, however, mostly unable to utilize the network enough to have a good estimate on the congestion state, which results in poor performance when the cached metrics are used. In fact, if a single packet is lost in a connection, a subsequent flow to the same destination might be prolonged with several round-trip times. We evaluate the proposed caching strategy and find transmission time reductions with up to 40%, by only caching metrics that are not related to the congestion state.

While the proposal provides an improved caching strategy for short and data-limited flows, it has to be enabled manually. That is, the caching mechanism does not automatically detect when to use the selective strategy or not. The problem of deciding whether to cache congestion metrics or not is hard, as it requires the end-host to infer whether the congestion state of the connection has converged or not.

Paper VI – Recent Trends in TCP Packet-Level Characteristics

In this paper, we try to identify transport-layer trends in the Internet. For instance, how common are data-limited flows, do packet loss really occur and how prevalent are other impairments like packet reordering in the Internet? To answer such questions, we conduct backbone measurements in the Swedish University Network (SUNET). To identify possible trends, we perform a series of measurements at a particular network location in both 2006 and 2009. The results show that most flows really are data-limited, about 1000 bytes or less, and that network impairments are fairly common. For instance, about $16-17\%$ of all connections contained out-of-order arrivals in both 2006 and 2009.

As mentioned earlier, packets are often out-of-order because of loss or reordering. In our measurements, most packets are out-of-order because of packet loss. Actually, the amount of packet reordering seems to have decreased between the measurements. The reason for the decrease could e.g. be due to hardware improvements, where parallelism within routers has been removed. The depicted increase in reordering, due to novel networking technologies, does not impact our results in a significant way.

Due to storage and processing restrictions, each of the 70 measurements analyzed in the paper were limited to span 10 minutes of traffic. While 10 minutes are enough to differentiate short flows from long, and to spot trends in characteristics like packet loss and reordering, it is still a limiting factor of our measurements. Furthermore, for the detailed analysis of TCP characteristics, we only used complete connections to ensure as correct analysis as possible.

Paper VII – Impact of Multi-path Routing on TCP Performance

In this paper we evaluate whether TCP is able to benefit from the capacity increase that per-packet load-balancing can provide, or if the resulting reordering becomes too problematic. As TCP is known to perform badly when packets are reordered, we also evaluate TCP versions that are designed to mitigate the negative effects of reordering. To construct an experimental baseline we use a TCP implementation without protection against reordering. We then compare the results to those of the Linux TCP implementation, which includes reordering robust mechanisms, and the TCP with robustness to non-congestion events (TCP-NCR). The evaluations are conducted in different wireless mesh networks (WMNs), using different multi-hop routing strategies to achieve load-balancing. The results show that it is possible for reordering robust TCP variants to improve the aggregate network performance slightly when load-balancing is conducted on a per-packet basis. However, the benefits are only present in ideal situations where e.g. the number of available paths are more than the number of competing flows. We also discover that the reordering mitigation mechanisms in Linux heavily relies on receiving acknowledgments in-order. If acknowledgments arrive out-of-order, to a certain degree, Linux can not provide good protection against data reordering. However, in general we find that all the evaluated TCP implementations do not perform well when packets are reordered, and that per-flow load-balancing is the best strategy to obtain good TCP performance.

Although we evaluate real TCP implementations in this paper, we perform the experiments in a simulated environment using the ns-2 simulator. The use of ns-2 could have an impact on the validity of the results, as the wireless models in ns-2 sometimes are considered to be overly simplified. However, as the focus of this paper is to study the effects of multi-path routing rather than purely wireless effects the use of ns-2 should not be a problem. Furthermore, the paper lacks experiments using single-path routing. It would have been interesting to compare e.g. per-flow multi-path routing with single-path routing to see whether multi-path offers better performance.

Paper VIII – Packet Reordering in TCP

This paper presents a detailed evaluation of the different packet reordering mitigation techniques we used in the previous paper (Paper VII). For these experiments we do not use a simulated environment, but rather an emulated environment with real end-hosts. Furthermore, we perform more detailed experiments to study effects that seem to impact the performance of reordering mitigation techniques. For instance, when evaluating these algorithms in scenarios where both packet reordering and packet loss occur, the mitigation techniques are of little use. The negative performance effects of packet loss dominate the achievable performance. If only reordering occurs, however, most techniques are able to improve performance. The TCP-NCR algorithms are able to sustain a throughput of approximately $2 - 4$ Mbps over a 5 Mbps link, when subjected to heavy reordering. Linux, on the other hand, is able to almost entirely utilize the available bandwidth in the same scenario. In addition to long-lived bulk flows, we also evaluate short-lived flows in this context. The common belief amongst researchers is typically that reordering mitigations hurts short-lived flows. We show that this is true for some algorithms, like TCP-NCR. However, using Linux's built-in mitigation techniques it is actually possible to slightly enhance the performance.

In contrast to the evaluation of packet reordering mitigations, in Paper VII, we do not consider complex traffic scenarios in this paper. The reason for conducting single-flow experiments is simply that it enabled us to analyze the results in a more detailed manner. For instance, we discovered a design flaw in the TCP-NCR algorithms that made it impossible for TCP to increase the sending rate in situations where reordering events occurred back-to-back in a single flow. To correctly detect problems like this in a more dynamic environment would be very hard, if possible at all. Furthermore, the experiments in the paper do not consider acknowledgment reordering, only data.

Paper IX – Emulation Support for Advanced Packet Reordering Models

This paper introduces the KauNet network emulator that was used for most of the experiments described in this thesis. The paper describes the main features of the emulator, such as the support for pre-defined deterministic network effects, which enables complete repeatability of experiments with a high degree of granularity. Actually, it is possible to apply specific network effects on traffic with a per-packet or per-millisecond granularity. In addition to the general description, this paper also details the design and implementation of the packet reordering capabilities in KauNet. In short, KauNet is able to create reordering in two different ways. The first way is to enable the user to specify exactly which packets that should be reordered, and how much. The second available option is to assign different delays to different packets and allow them to be reordered according to these delays. To demonstrate this functionality, the paper includes some reordering experiments with the TCP implementation in Linux.

The evaluation in this paper does not use any advanced reordering models to evaluate the Linux TCP implementation, it only provides simple experiments that demonstrates

the capabilities of KauNet. Furthermore, we do not address how to construct advanced reordering models in this paper. In general, KauNet does not provide any means to automatically construct advanced emulation models of networking effects, although it is easy to reuse already existing, advanced, models in KauNet emulations.

7 Conclusions and Future Research

During the last decade the Internet has become a unified platform for a wide range of services that previously had separate distribution channels. For instance, Internet now hosts newspapers, many computer games support online interaction, telephony and other communication services are widely deployed and even regular computer applications such as document processors and calendars are now available online. In line with this, Internet access has evolved from being important only for early adopters and technically inclined to being a necessity in most peoples' daily life. Due to these significant changes, the actual data communicated over the Internet and the requirements on its delivery have changed radically. While the concern previously was to download files as quickly as possible, low latency is now becoming more important. Furthermore, as Internet has become such an integrated part of peoples' life a demand for ubiquitous access through e.g. laptops and smart phones have emerged. To support this development, we have proposed and evaluated a number of transport-layer mechanisms that better support low latency data exchange and also evaluated several mechanisms used to provide Internet access using limited infrastructure. To reduce latency we propose a set of mechanisms that tries to perform faster packet loss detection, make the loss detection interact more appropriately with TCP caching mechanisms and also to promote more efficient message bundling in SCTP. Several techniques are used to support more efficient networking in environments with limited infrastructure. As many of these techniques create another problem, i.e., packet reordering, we have evaluated several strategies to overcome reordering related problems for TCP.

The problem of latency is starting to be recognized as a major problem when using Internet services. This is, for example, reflected in the IETF standardization process, which currently address several low latency proposals. However, a lot of work still remains. For instance, a number of proposals have been made to optimize the parts of transport protocols where latency problems most easily are identified, e.g. in the loss recovery. To support low latency applications appropriately more fundamental changes are needed. The problem of fundamental changes, however, is that the incentive to deploy such solutions is not enough at the moment [38]. That is, the Internet currently works to well to make significant changes to it. There are some interesting openings though. For instance, a multi-path TCP (MPTCP) version is currently being standardized within the IETF [30]. The key enabler of MPTCP is the backwards compatibility with TCP, that enables incremental deployment. Although the goal of MPTCP, at the moment, is to provide better throughput, the use of multiple paths enables several ways of reducing latency as well. It would be possible, for example, to detect the path that has the lowest latency and use that path for communication. Another example could be to

redundantly send data over several paths and use the data that arrives first.

The latency/throughput trade-off is even more serious for novel devices such as smart phones and tablets that use a 3G interface to connect to the Internet. While the traffic communicated by such devices consists of even more data-limited flows [33], as compared to laptops and stationary computers, the development for 3G networks has so far mainly been concentrated at increasing the throughput [53]. Some of these issues could possibly be ameliorated using multiple paths, as such devices often are equipped with multiple interfaces (e.g. 3G and WLAN). One way forward could be to utilize one interface for web browsing and other latency sensitive traffic, while using another interface for data intensive traffic like video streaming.

Until now, we have mostly discussed solutions at the transport-layer. The focus on the transport layer is rather logical, as it controls how data is transmitted. However, there is certainly room for improvement in other layers and their interaction. For instance, over-dimensioned buffers have recently gained interest and are considered by some as the single most severe latency problem. Basically, the large amount of buffer space inhibits data-limited flows from achieving low latency if the buffer is filled up by data intensive flows. By reducing buffer space, or deploying better buffer management algorithms, it would be possible to enable a better resource sharing between such traffic types. However, research is needed on how to size these buffers and which queuing disciplines to use.

References

[1] M. Allman, K. Avrachenkov, U. Ayesta, J.Blanton, and P. Hurtig. Early retransmit for TCP and stream control transmission protocol (SCTP). *Internet RFCs, ISSN 2070-1721, RFC 5827*, April 2010.

[2] M. Allman, H. Balakrishnan, and S. Floyd. Enhancing TCP's loss recovery using limited transmit. *Internet RFCs, ISSN 2070-1721, RFC 3042*, January 2001.

[3] M. Allman and V. Paxson. On estimating end-to-end network path properties. In *Proceedings of the ACM SIGCOMM Conference*, Cambridge, USA, August 1999.

[4] M. Allman, V. Paxson, and E. Blanton. TCP congestion control. *Internet RFCs, ISSN 2070-1721, RFC 5681*, September 2009.

[5] H. Alvestrand. A mission statement for the IETF. *Internet RFCs, ISSN 2070-1721, RFC 3935*, October 2004.

[6] C. M. Arthur, A. Lehane, and D. Harle. Keeping order: Determining the effect of TCP packet reordering. In *Proceedings of the 3rd International Conference on Networking and Services (ICNS)*, Athens, Greece, June 2007.

[7] H. Balakrishnan. *Challenges to Reliable Data Transport over Heterogeneous Wireless Networks*. PhD thesis, University of California at Berkely, Berkely, USA, August 1998.

[8] H. Balakrishnan, V. Padmanabhan, S. Seshan, M. Stemm, and R. Katz. TCP behavior of a busy web server: Analysis and improvements. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, San Fransisco, USA, March 1998.

[9] H. Balakrishnan and S. Seshan. The congestion manager. *Internet RFCs, ISSN 2070-1721, RFC 3124*, June 2001.

[10] J. Bellardo and S. Savage. Measuring packet reordering. In *Proceedings of the 2nd ACM SIGCOMM Internet Measurment Workshop (IMW)*, Marseille, France, November 2002.

[11] M. Belshe. More bandwidth doesn't matter (much), May 2010. `http://www.belshe.com/2010/05/24/` [accessed 2011.12.19].

[12] J. C. R. Bennett, C. Partridge, and N. Shectman. Packet reordering is not pathological network behavior. *IEEE/ACM Transactions on Networking*, 7(6), December 1999.

[13] S. Bhandarkar and A. L. N. Reddy. TCP-DCR: Making TCP robust to non-congestion events. Technical Report TAMU-ECE-2003-04, Department of Electrical Engineering, Texas A & M University, July 2003.

[14] S. Bhandarkar, A. L. N. Reddy, M. Allman, and E. Blanton. Improving the robustness of TCP to non-congestion events. *Internet RFCs, ISSN 2070-1721, RFC 4653*, August 2006.

[15] E. Blanton and M. Allman. On making TCP more robust to packet reordering. *ACM SIGCOMM Computer Communication Review*, 32(1), January 2002.

[16] E. Blanton and M. Allman. Using TCP duplicate selective acknowledgement (DSACKs) and stream control transmission protocol (SCTP) duplicate transmission sequence numbers (TSNs) to detect spurious retransmissions. *Internet RFCs, ISSN 2070-1721, RFC 3708*, February 2004.

[17] S. Bohacek, J. P. Hespanha, J. Lee, C. Lim, and K. Obraczka. A new TCP for persistent packet reordering. *IEEE/ACM Transactions on Networking*, 14(2), April 2006.

[18] R. Braden. Requirements for Internet hosts – communication layers. *Internet RFCs, ISSN 2070-1721, RFC 1122*, October 1989.

[19] E. Brosh, S. A. Baset, V. Misra, D. Rubenstein, and H. Schulzrinne. The delay-friendliness of TCP for real-time traffic. *IEEE/ACM Transactions on Networking*, 18(5), October 2010.

[20] K. Chebrolu, B. Raman, and R. R. Rao. A network layer approach to enable TCP over multiple interfaces. *Kluwer Wireless Networks*, 11(5), September 2005.

[21] Y. Cheng, J. Chu, S. Radhakrishnan, and A. Jain. TCP fast open. Internet draft, work in progress, Internet Engineering Task Force, December 2011. draft-cheng-tcpm-fastopen-02.

[22] S. Cheshire. It's the latency, stupid, May 1996. `http://rescomp.stanford.edu/~cheshire/rants/Latency.html` [accessed 2011.12.19].

[23] J. Chu, N. Dukkipati, and M. Mathis. Increasing TCP's initial window. Expired internet draft, Internet Engineering Task Force, July 2010. draft-hkchu-tcpm-initcwnd-01.

[24] M. Claypool. The effect of latency on user performance in real-time strategy games. *Elsevier Computer Networks*, 49(1), September 2005.

[25] N. Dukkipati and N. McKeown. Why flow-completion time is the right metric for congestion control. *ACM SIGCOMM Computer Communication Review*, 36(1), January 2006.

[26] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin. An argument for increasing TCP's initial congestion window. *ACM SIGCOMM Computer Communication Review*, 40(3), June 2010.

[27] L. Eggert, J. Heidemann, and J. Touch. Effects of ensemble-TCP. *ACM SIGCOMM Computer Communication Review*, 30(1), January 2000.

[28] H. Ekström and R. Ludwig. The peak-hopper: A new end-to-end retransmission timer for reliable unicast transport. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, Hong Kong, March 2004.

[29] J. Feng, Z. Ouyang, L. Xu, and B. Ramamurthy. Packet reordering in high-speed networks and its impact on high-speed TCP variants. *Elsevier Computer Communications*, 32(1), January 2009.

[30] A. Ford, C. Raiciu, M. Handley, S. Barré, and J. Iyengar. Architectural guidelines for multipath TCP development. *Internet RFCs, ISSN 2070-1721, RFC 6182*, March 2011.

[31] Z. Fu, B. Greenstein, X. Meng, and S. Lu. Design and implementation of a TCP-friendly transport protocol for ad hoc wireless networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP)*, Paris, France, November 2002.

[32] J. Garcia, E. Conchon, T. Pérennou, and A. Brunstrom. KauNet: improving reproducibility for wireless and mobile research. In *Proceedings of the 1st International Workshop on System Evaluation for Mobile Platforms*, San Juan, Puerto Rico, June 2007.

[33] A. Gember and A. Anand. A comparative study of handheld and non-handheld traffic in campus Wi-Fi networks. In *Proceedings of the 12th Passive and Active Measurement Conference (PAM)*, Atlanta, USA, March 2011.

[34] M. Gerla, K. Tang, and R. Bagrodia. TCP performance in wireless multi-hop networks. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMSCA)*, New Orleans, USA, February 1999.

[35] J. Gettys. Bufferbloat: Dark buffers in the Internet. *IEEE Internet Computing*, 15(3), May 2011.

[36] J. Gettys. IW10 considered harmful. Internet draft, work in progress, Internet Engineering Task Force, August 2011. draft-gettys-iw10-considered-harmful-00.

[37] L. Gharai, C. Perkins, and T. Lehman. Packet reordering, high speed networks and transport protocol performance. In *Proceedings of the 13th International Conference on Computer Communications and Networks (ICCCN)*, Chicago, USA, October 2004.

[38] M. Handley. Why the Internet only just works. *BT Technology Journal*, 24(3), July 2006.

[39] W. Hansmann and M. Frank. On things to happen during a TCP handover. In *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN)*, Bonn, Germany, October 2003.

[40] J. He and J. Rexford. Toward Internet-wide multipath routing. *IEEE Network*, 22(2), March 2008.

[41] P. Hurtig. Fast retransmit inhibitions for TCP. Master's thesis, Karlstad University, February 2006.

[42] P. Hurtig, A. Petlund, and M. Welzl. TCP and SCTP RTO restart. Internet draft, work in progress, Internet Engineering Task Force, October 2011. draft-hurtig-tcpm-rtorestart-01.

[43] Linux Kernel Organization Inc. The Linux kernel archives, 1997. `http://www.kernel.org` [accessed 2011.12.19].

[44] Internet Society (ISOC). Internet engineering task force (IETF), 1986. `http://www.ietf.org`, [accessed 2011.12.19].

[45] ITU-T. ITU-T recommendation G.114. Technical report, International Telecommunication Union, May 2003.

[46] 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), October 2006.

[47] V. Jacobson. Congestion avoidance and control. In *Proceedings of the ACM SIGCOMM Conference*, Stanford, USA, August 1988.

[48] V. Jacobson, R. Braden, and D. Borman. TCP extensions for high performance. *Internet RFCs, ISSN 2070-1721, RFC 1323*, May 1992.

[49] D. Kaspar, K. Evensen, A.F. Hansen, P. Engelstad, P. Halvorsen, and C. Griwodz. An analysis of the heterogeneity and IP packet reordering over multiple wireless networks. In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)*, Sousse, Tunisia, July 2009.

[50] A. Kesselman and Y. Mansour. Optimizing TCP retransmission timeout. In *Proceedings of the 4th International Conference on Networking (ICN)*, Reunion Island, France, April 2005.

[51] E. Kohler, M. Handley, and S. Floyd. Datagram congestion control protocol (DCCP). *Internet RFCs, ISSN 2070-1721, RFC 4340*, March 2006.

[52] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: congestion control without reliability. *ACM SIGCOMM Computer Communication Review*, 36(4), August 2006.

[53] M Laner, P. Svoboda, and E. Hasenleithner. Dissecting 3G uplink delay by measuring in an operational HSPA network. In *Proceedings of the 12th Passive and Active Measurement Conference (PAM)*, Atlanta, USA, March 2011.

[54] M. Laor and L. Gendel. The effect of packet reordering in a backbone link on application throughput. *IEEE Network*, 16(2), September/October 2002.

[55] K-C. Leung, V. O. K. Li, and D. Yang. An overview of packet reordering in transmission control protocol (TCP): Problems, solutions, and challenges. *IEEE Transactions on Parallel and Distributed Systems*, 18(4), April 2007.

[56] R. Ludwig and M. Meyer. The Eifel detection algorithm for TCP. *Internet RFCs, ISSN 2070-1721, RFC 3522*, April 2003.

[57] R. Ludwig and K. Sklower. The Eifel retransmission timer. *ACM SIGCOMM Computer Communication Review*, 30(3), July 2000.

[58] X. Luo and R. K. C. Chang. Novel approaches to end-to-end packet reordering measurement. In *Proceedings of the 5th ACM SIGCOMM Internet Measurement Conference (IMC)*, Berkeley, USA, October 2005.

[59] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options. *Internet RFCs, ISSN 2070-1721, RFC 2018*, October 1996.

[60] D. McCreary, K. Li, S. A. Watterson, and D. K. Loewnthal. TCP-RC: A receiver-centered TCP protocol for delay-sensitive applications. In *Proceedings of the 12th Annual Multimedia Computing and Networking (MMCN)*, San Jose, California, USA, January 2005.

[61] M. Mellia, M. Meo, and C. Casetti. TCP smart framing: a segmentation algorithm to reduce TCP latency. *IEEE/ACM Transactions on Networking*, 13(2), April 2005.

[62] M. Mellia, M. Meo, L. Muscariello, and D. Rossi. Passive analysis of TCP anomalies. *Elsevier Computer Networks*, 52(14), October 2008.

[63] G. Minshall. A suggested modification to Nagle's algorithm. Expired internet draft, Internet Engineering Task Force, June 1999. draft-minshall-nagle-01.

[64] G. Minshall, Y. Saito, J.C. Mogul, and B. Verghese. Application performance pitfalls and TCP's Nagle algorithm. *ACM SIGMETRICS Performance Evaluation Review*, 27(4), 2000.

[65] J. Mogul and S. Deering. Path MTU discovery. *Internet RFCs, ISSN 2070-1721, RFC 1191*, November 1990.

[66] J. C. Mogul and G. Minshall. Rethinking the TCP Nagle algorithm. *ACM SIGCOMM Computer Communication Review*, 31(1), January 2001.

[67] B. Mukherjee and T. Brecht. Time-lined TCP for the TCP-friendly delivery of streaming media. In *Proceedings of the 8th International Conference on Network Protocols (ICNP)*, Osaka, Japan, November 2000.

[68] J. Nagle. Congestion control in IP/TCP internetworks. *Internet RFCs, ISSN 2070-1721, RFC 896*, January 1984.

[69] V. Paxson. End-to-end Internet packet dynamics. *IEEE/ACM Transactions on Networking*, 7(3), June 1999.

[70] V. Paxson, M. Allman, J. Chu, and M. Sargent. Computing TCP's retransmission timer. *Internet RFCs, ISSN 2070-1721, RFC 6298*, June 2011.

[71] A. Petlund, K. Evensen, C. Griwodz, and P. Halvorsen. TCP mechanisms for improving the user experience for time-dependent thin-stream applications. In *Proceedings of the the 33rd Annual IEEE Conference on Local Computer Networks (LCN)*, Montreal, Canada, October 2008.

[72] J. Postel. Internet protocol. *Internet RFCs, ISSN 2070-1721, RFC 791*, September 1981.

[73] J. Postel. Transmission control protocol. *Internet RFCs, ISSN 2070-1721, RFC 793*, September 1981.

[74] I Psaras and V. Tsaoussidis. On the properties of an adaptive TCP minimum RTO. *Elsevier Computer Communications*, 32(5), March 2009.

[75] S Radhakrishnan, Y Cheng, J Chu, A Jain, and B Raghavan. TCP fast open. In *Proceedings of the 7th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, Tokyo, Japan, December 2011.

[76] S. Rewaskar, J. Kaur, and D. F. Smith. A passive state-machine approach for accurate analysis of TCP out-of-sequence segments. *ACM SIGCOMM Computer Communication Review*, 36(3), July 2006.

[77] P. Sarolahti and M. Kojo. Forward RTO-recovery (F-RTO): An algorithm for detecting spurious retransmission timeouts with TCP and the stream control transmission protocol (SCTP). *Internet RFCs, ISSN 2070-1721, RFC 4138*, August 2005.

[78] M. Savoric, H. Karl, and A. Wolisz. The TCP control block interdependence in fixed networks – new performance results. *Elsevier Computer Communications*, 26(4), March 2003.

[79] F. Schneider, S. Agarwal, T. Alpcan, and A. Feldmann. The new web: characterizing AJAX traffic. In *Proceedings of the 9th International Conference on Passive and Active Network Measurement (PAM)*, Cleveland, USA, April 2008.

[80] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobsson. RTP: A transport protocol for real-time applications. *Internet RFCs, ISSN 2070-1721, RFC 3550*, July 2003.

[81] S. Seshan, M. Stemm, and H. Katz. SPAND: Shared passive network performance discovery. In *Proceedings of the 1st USENIX Symposium on Internet Technologies and Systems (USITS)*, Monterey, California, USA, December 1997.

[82] T. Seth, A. Broscius, C. Huitema, and H.P. Lin. Performance requirements for signaling in internet telephony. Expired internet draft, Internet Engineering Task Force, November 1998. draft-seth-sigtran-req-00.

[83] R. Stewart. Stream control transmission protocol. *Internet RFCs, ISSN 2070-1721, RFC 4960*, September 2007.

[84] P. Sun, M. Yu, M. J. Freedman, and J. Rexford. Identifying performance bottlenecks in CDNs through TCP-level monitoring. In *Proceedings of the 1st ACM SIGCOMM Workshop on Measurements up the Stack (W-MUST)*, Toronto, Canada, August 2011.

[85] J. Touch. TCP control block interdependence. *Internet RFCs, ISSN 2070-1721, RFC 2140*, April 1997.

[86] 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*, Barcelona, Spain, 2009.

[87] Z. Wen and K. L. Yeung. On RTO timer implementation in TCP. In *Proceedings of the 10th International Conference on Intelligent Systems Design and Applications (ISDA)*, Cairo, Egypt, December 2010.

[88] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K.D. Johnson. M2M: From mobile to embedded Internet. *IEEE Communications Magazine*, 49(4), April 2011.

[89] D. Yang, K-C. Leung, and V.O.K. Li. Simulation-based comparisons of solutions for TCP packet reordering in wireless networks. In *Proceedings of the IEEE Wireless Communications & Networking Conference (WCNC)*, Hong Kong, March 2007.

[90] M. Zhang, B. Karp, S. Floyd, and L. Peterson. RR-TCP: A reordering-robust TCP with DSACK. In *Proceedings of the 11th IEEE International Conference on Networking Protocols (ICNP)*, Atlanta, USA, November 2003.

Transport-Layer Performance for Applications and Technologies of the Future Internet

To provide Internet applications with good performance, the transport protocol TCP is designed to optimize the throughput of data transfers. Today, however, more and more applications rely on low latency rather than throughput. Such applications can be referred to as data-limited and are not appropriately supported by TCP. Another emerging problem is associated with the use of novel networking techniques that provide infrastructure-less networking. To improve connectivity and performance in such environments, multi-path routing is often used. This form of routing can cause packets to be reordered, which in turn hurts TCP performance.

To address timeliness issues for data-limited traffic, we propose and experimentally evaluate several transport protocol adaptations. For instance, we adapt the loss recovery mechanisms of both TCP and SCTP to perform faster loss detection for data-limited traffic, while preserving the standard behavior for regular traffic. Evaluations show that the proposed mechanisms are able to reduce loss recovery latency with 30-50%. We also suggest modifications to the TCP state caching mechanisms. The caching mechanisms are used to optimize new TCP connections based on the state of old ones, but do not work properly for data-limited flows. Additionally, we design a SCTP mechanism that reduces overhead by bundling several packets into one packet in a more timely fashion than the bundling normally used in SCTP.

To address the problem of packet reordering we perform several experimental evaluations, using TCP and state of the art reordering mitigation techniques. Although the studied mitigation techniques are quite good in helping TCP to sustain its performance during pure packet reordering events, they do not help when other impairments like packet loss are present.

Karlstad University Studies
ISSN 1403-8099
ISBN 978-91-7063-404-8