Towards a Low Latency Internet: Understanding and Solutions

Mohammad Rajiullah

Faculty of Health, Science and Technology

Computer Science

DISSERTATION | Karlstad University Studies | 2015:41

Towards a Low Latency Internet: Understanding and Solutions

Mohammad Rajiullah

DISSERTATION | Karlstad University Studies | 2015:41

Towards a Low Latency Internet: Understanding and Solutions

Mohammad Rajiullah

DISSERTATION

Karlstad University Studies | 2015:41

URI: urn:nbn:se:kau:diva-37487

ISSN 1403-8099

ISBN 978-91-7063-659-2

Distribution:
Karlstad University
Faculty of Health, Science and Technology
Department of Mathematics and Computer Science
SE-651 88 Karlstad, Sweden
+46 54 700 10 00

Print: Universitetstryckeriet, Karlstad 2015

WWW.KAU.SE

Towards a Low Latency Internet: Understanding and Solutions

MOHAMMAD RAJIULLAH
Department of Computer Science, Karlstad University, Sweden

Abstract

Networking research and development have historically focused on increasing network throughput and path resource utilization, which particularly helped bulk applications such as file transfer and video streaming. Recent overprovisioning in the core of the Internet has facilitated the use of interactive applications like interactive web browsing, audio/video conferencing, multiplayer online gaming and financial trading applications. Although the bulk applications rely on transferring data as fast as the network permits, interactive applications consume rather little bandwidth, depending instead on low latency. Recently, there has been an increasing concern in reducing latency in networking research, as the responsiveness of interactive applications directly influences the quality of experience.

To appreciate the significance of latency-sensitive applications for today's Internet, we need to understand their traffic pattern and quantify their prevalence. In this thesis, we quantify the proportion of potentially latency-sensitive traffic and its development over time. Next, we show that the flow start-up mechanism in the Internet is a major source of latency for a growing proportion of traffic, as network links get faster.

The loss recovery mechanism in the transport protocol is another major source of latency. To improve the performance of latency-sensitive applications, we propose and evaluate several modifications in TCP. We also investigate the possibility of prioritization at the transport layer to improve the loss recovery. The idea is to trade reliability for timeliness. We particularly examine the applicability of PR-SCTP with a focus on event logging. In our evaluation, the performance of PR-SCTP is largely influenced by small messages. We analyze the inefficiency in detail and propose several solutions. We particularly implement and evaluate one solution that utilizes the Non-Renegable Selective Acknowledgments (NR-SACKs) mechanism, which has been proposed for standardization in the IETF. According to the results, PR-SCTP with NR-SCAKs significantly improves the application performance in terms of low latency as compared to SCTP and TCP.

Keywords: latency; traffic classification; slow-start; TCP; SCTP; PR-SCTP; NR-SACKs; event logging; performance evaluation

Acknowledgements

This thesis has benefited from the help of many people. First and foremost, I would like to thank my supervisor, Professor Anna Brunstrom, for giving me the opportunity to pursue my doctoral studies and for continuously providing me with intellectual support and encouragement throughout my work. Her insight, insistence on simplicity, detailed and constructive comments, tremendous patience with my writing all together have been of great value in my work. Next, I would like to express my sincere thanks to my co-supervisor, Professor Stefan Lindskog, for his detailed review and excellent advice during the preparation of my thesis.

I would also like to acknowledge all my co-authors of the papers included in this thesis, Reine Lundin and Per Hurtig from my department, Andreas Petlund, Olga Bondarenko, Ahmed Elmokashfi, Carsten Griwodz and Lilian Calvet from Simula Research lab, Norway, Michael Welzl from Oslo University, Norway, and Bob Briscoe from British Telecom, UK. In addition, my warm thanks are due to all my colleagues at the Computer Science Department at Karlstad University, in particular the distributed systems and communications research group, DISCO, for their valuable suggestions regarding my research and help in my daily life.

I am grateful to Compare Business Innovation Center (CBIC) and the European Union 7th framework program (FP7), reducing Internet transport latency (RITE) project, for financial support during my research. In addition, I would like to thank all partners from the RITE project for extensive and constructive discussions during the project meeting and IETF events that have strengthened my basis in networking research.

Thanks to my wife, Farhana, for her love, patience and support during my work. Her selfless encouragement makes me want to excel. Last but not least, I am massively indebted to my family back in Bangladesh for their unending encouragement and love.

Mohammad Rajiullah
Karlstad, September, 2015

List of Appended Papers

The thesis is based on the work presented in the following seven papers. Reference to the papers will be made using the associated Roman numbers.

I. Olga Bondarenko, **Mohammad Rajiullah**, Carsten Griwodz, Lilian Calvet, Anna Brunstrom, Andreas Petlund and Ahmed Elmokashfi, "A Method for Hierarchical Clustering of Internet Traffic and its Use in Detecting Application-Limited Flows," under submission.

II. **Mohammad Rajiullah**, Bob Briscoe, Anna Brunstrom and Andreas Petlund, "What is Top Speed without Acceleration?", to be submitted.

III. **Mohammad Rajiullah**, Per Hurtig, Anna Brunstrom, Andreas Petlund and Michael Welzl, "An Evaluation of Tail Loss Recovery Mechanisms for TCP," In *ACM SIGCOMM Computer Communication review (CCR)*, Volume 45, Number 1, pages 6–11, January, 2015.

IV. **Mohammad Rajiullah**, Anna Brunstrom and Stefan Lindskog, "Priority Based Delivery of PR-SCTP Messages in a Syslog Context," In *Proceedings of the 5th International ICST Conference on Access Networks (AccessNets)*, pages 299–310, Budapest, Hungary, November 3–5, 2010.

V. **Mohammad Rajiullah**, Reine Lundin, Anna Brunstrom and Stefan Lindskog,"Syslog Performance: Data Modeling and Transport," In *Proceedings of the 3rd International Workshop on Security and Communication Networks (IWSCN)*, pages 31–37, Gjøvik, Norway, May 18–20, 2011.

VI. **Mohammad Rajiullah** and Anna Brunstrom,"On the Effectiveness of PR-SCTP in Networks with Competing Traffic," In *Proceedings of the IEEE Symposium on Computers and Communications (ISCC)*, pages 898–905, Corfu, Greece, June 28–July 1, 2011.

VII. **Mohammad Rajiullah**, Reine Lundin, Anna Brunstrom and Stefan Lindskog, "Performance Analysis and Improvement of PR-SCTP for Small Messages," In *Computer Networks, Elsevier*, Volume 57, Issue 18, pages 3967–3986, December 2013.

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

Comments on my Participation

Paper I I collaborated in problem formulation, building the paper framework, data transformation and result analyses. Moreover, I authored the following sections in the paper: abstract, introduction, related work and data set descriptions. I helped in reviewing the other parts.

Paper II I was responsible for various trace processing and analysis in the paper including flow size analysis in Section 3.2.2 and Section 3.2.4, utilization analysis in Section 3.2.3 and Section 3.2.4 and value per byte analysis in Section 3.3.1. I authored a large part of the paper based on the initial framing of the paper done by Bob Briscoe. Other co-authors helped me with useful reviews and discussions.

Paper III I was partly responsible in various ways during the implementation and validation of the RTO restart algorithm in Linux. In the paper, I was responsible for the experiment using realistic tail loss. Furthermore, I have mainly authored the relevant experiment section, related work, abstract and the conclusion of the paper. Per Hurtig ran the controlled tail loss and web page experiments. All the co-authors helped me in updating the paper based on the reviews of the CCR reviewers.

Paper IV I was responsible for carrying out the experimental evaluations and for the written material. The other co-authors helped me in developing the underlying ideas in the paper and gave useful comments in reviewing.

Paper V I was responsible for carrying out all the experiments and for all the written parts. My co-author, Anna Brunstrom, collaborated with me during the development of the ideas in the paper. Anna also provided constructive comments during the review process.

Paper VI I was responsible for all the experiments in the paper. I authored most of the written material except the syslog data modeling section, which was primarily authored by Reine Lundin. The other co-authors helped me during the development of the ideas in the paper and with suggestions in reviewing.

Paper VII I was responsible for all the experiments in the paper and the implementation of the NR-SACK based PR-SCTP optimization in the FreeBSD operating system. I authored most of the paper, except the section on event logging that Stefan Lindskog authored and the section describing the syslog specific application scenario that was written by Reine Lundin. Reine also prepared the primary syslog trace that I employed in the use case specific experiments in the paper. All the co-authors helped me in developing the ideas in the paper. They also contributed during the review process.

Other Publications

- **Mohammad Rajiullah**, Reine Lundin, Anna Brunstrom, and Stefan Lindskog, "Data Modeling and Transport of Syslog Messages," In *Proceedings of the 7th Swedish National Computer Networking Workshop (SNCNW 2011)*, Linköping, Sweden, June, 2011.

- **Mohammad Rajiullah** and Anna Brunstrom, "Perfromance Optimization of PR-SCTP for small messages," In *IEEE Swedish Communication Technologies Workshop (Swe-CTW)*, Stockholm, Sweden, October, 2011.
- **Mohammad Rajiullah** and Anna Brunstrom, "Optimizing PR-SCTP Performance using NR-SACKs," In *Proceedings of the 2nd Baltic Conference on Future Internet Communications (BCFIC 2012)*, Vilnius, Lithuania, April, 2012, IEEE.
- **Mohammad Rajiullah** and Anna Brunstrom, "Performance Improvement of PR-SCTP using Non-Renegable Selective Acknowledgements (NR-SACKs)," In *Proceedings of the 8th Swedish National Computer Networking Workshop (SNCNW 2012)*, Stockholm, Sweden, June 2012.
- **Mohammad Rajiullah** and Anna Brunstrom, "Evaluation and Analysis of NR-SACKs based PR-SCTP," In *SAIL summer school*, Santander, Spain, June, 2012.
- **Mohammad Rajiullah**,"Performance Analysis and Improvement of PR-SCTP in an Event Logging Context", Licentiate thesis, November, 2012.
- Anna Brunstrom, Andreas Petlund and **Mohammad Rajiullah**, "Reducing Internet Transport Latency for Thin Streams and Short flows", In *Future Network and MobileSummit*, Lisbone, Portugal, 2013.
- Andreas Petlund *et al.*"End-system analysis and preliminary development report", deliverable 1.1, Work Package 1, confidential deliverable, Reducing Internet Transport Latency (RITE) project (EU FP7/ICT-317700), 2013.
- Anna Brunstrom *et al.* "Report on Design and Initial Evaluation of End-system, Application-layer and API Mechanisms", deliverable 1.2, Work Package 1, confidential deliverable, Reducing Internet Transport Latency (RITE) project (EU FP7/ICT-317700), 2014.
- David Hayes *et al.* "Report on Prototype Development and Evaluation of End-system, Application layer and API Mechanisms", deliverable 1.3, Work Package 1, confidential deliverable, Reducing Internet Transport Latency (RITE) project (EU FP7/ICT-317700), 2015.
- Bob Briscoe *et al.* "Report on Prototype Development and Evaluation of Network and Interaction Techniques", deliverable 2.3, Work Package 2, confidential deliverable, Reducing Internet Transport Latency (RITE) project (EU FP7/ICT-317700), 2015.

Contents

Introductory Summary

1 Introduction

Applications producing bulk traffic, such as file downloading and video streaming, have dominated the Internet for a long time. This has recently changed. Over-provisioning in the core of the Internet solves many problems that inhibited the use of interactive applications such as audio/video conferencing, multiplayer gaming and stock trading over the Internet. Moreover, web applications are getting increasingly interactive. Unlike bulk applications, such applications are time constrained. According to ITU-T, in applications such as audio conferencing, one-way transmission times exceeding 150–200 ms significantly degrade user experience [1]. For online gaming, the measured tolerable latency is between 100 and 1000 ms [2]. In the case of commercial websites, most users cannot tolerate more than two seconds of page load delay [3]. So, the question is, "is the Internet ready to provide sufficient performance for these applications?"

With a few rare exceptions (e.g. [4,5]), research and development in networking have historically focused on how to increase network throughput and path resource utilization. These efforts have resulted in an average global broadband bandwidth of about 4.6 Mbps[1], with a year-over-year increase of approximately 42% [6].The increased bandwidth particularly helps applications that generate bulk traffic, providing reduced downloading time. One might conclude that the reduction automatically reduces the average end-to-end latency for all applications. However, this is not necessarily true, because the performance of interactive applications does not depend on flow completion time or total download time. Over-buffering and long queues in the network have only increased link utilization and reduced download times for large files, but the average delivery time of each individual network packet has not improved much, which influences the performance of the interactive applications.

As low latency delivery of application data influences responsiveness or "quality of experience (QoE)" of interactive applications, latency has become a critical concern today for service providers [7–12]. For example, Amazon and Shopzilla reported significant revenue increases when decreasing latency. Amazon increased their revenues by 1% for every 100 ms reduction and Shopzilla had a 12% increase when they reduced latency from 6 to 1.2 seconds [12], whereas Microsoft Bing found that introducing 500ms extra delay translated to 1.2% less advertising revenue. Google experimented with injecting a 400-ms delay before they returned their search page. Initial searches declined linearly over time, dropping by 0.76% after six weeks and continuing linearly after that. These results were presented in a joint Microsoft-Google presentation on the value of reducing Web delay [9]. According to [8], an extra delay of one second in page loading time can impact conversions by 7%, page views by 11% and customer satisfaction by 16%. For example, this extra delay may cause a reduction of 2.5 million $ in revenues in a year for a 100,000 $/day e-commerce

[1] As of the second quarter of 2014, from Akamai (http://www.akamai.com) on average broadband speeds, as seen by their servers.

site [8]. Consequently, timeliness in the current Internet has started to receive a great deal of attention lately [13–15].

Latency-sensitive applications typically exchange relatively small amounts of data. For example, online gaming or many cloud based applications require a timely exchange of short data flows of control information (e.g., keyboard input and screen updates) between service provider and customer [16, 17]. Voice over IP (VoIP) [18] and Machine-to-Machine (M2M) communications [19] are other popular applications that have similar requirements. Furthermore, web data transfers triggered by user interactions generate short data flows [20, 21]. These applications generate traffic that are limited in size and rate and are consequently called application-limited flows. Application-limited flows often do not use the entire link capacity. These flows are less concerned about the throughput of the network; instead they care about the latency for either the whole flow or individual packets.

In order to acknowledge the importance of latency-sensitive applications, we need to understand their traffic pattern and quantify their prevalence in the Internet. To this end, using a wide range of transport level features, we classify Internet traffic (CAIDA packet traces) collected from the network core [22] covering the time period, 2008–2012. Results from the classification give us both the traffic pattern and the proportions of application-limited flows and their development over the years. According to our results, Internet traffic can be split into six classes including different families of bulk flows and application-limited flows. The class of short web–like flows contains the majority of the flows. Moreover, classes containing Dynamic Adaptive Streaming over HTTP [23] (DASH)–like flows, game–like flows, and persistent web–like flows have expanded in terms of their share of flows in recent years. These observations will in general help us to make qualified decisions on trading off bandwidth for latency in the case of application-limited flows. Furthermore, as reducing latency in the Internet is a work in progress, we envision that our findings will contribute to the overall understanding of the importance of reducing latency in different sectors of the Internet. Our classification is based on an unsupervised method for hierarchical clustering [24] to classify Internet flows.

Increasing capacity in the network does not necessarily improve the performance of application-limited flows. The start-up mechanism in the Internet transport protocols can contribute to latency. Transport protocols are designed on the principle that each time a new flow starts or re-starts after an idle period the available capacity is unknown. Internet applications running over the Transmission Control Protocol (TCP) [25] therefore use some alternatives of the slow-start algorithm [26] at (re)start. In slow-start, every flow starts with sending a few initial number of packets, waits for the response, then doubles the number of packets in subsequent rounds unless a threshold number of packets have been sent or the response shows losses. In this thesis, we show that a growing proportion of Internet flows are limited by the slow-start limitation not capacity. Our analysis is based on a longitudinal study of CAIDA packet traces. According to our analysis, the majority of the flows on the Internet are small. Comparing the flow sizes between the years also suggests that the

distribution of transfer sizes is not changing very much as capacity grows. The majority of the flows are not large enough to make use of capacities available in the Internet. Besides, most of the time, a new flow arrives at an empty bottleneck, suggesting potential improvement for flow start-up to use the idle capacity.

While there are numerous other sources of latency in the Internet [27], the loss recovery mechanism in TCP is one of the most important components. TCP's loss recovery mechanism can add several round trip times to the transmission time of a TCP segment, which may especially harm short flows [28]. A recent large scale study at Google (*www.google.com*) has found that approximately 10% of all connections have at least one segment loss, which delays the transmission time by five times as compared to flows without any loss [29]. The same source also reported that TCP recovers 77% of these losses through lengthy retransmission timeouts (RTOs), because segments are often lost at the tail of a flow or segment burst with no new data available for transmission preventing any faster loss recovery mechanisms. In this thesis, we evaluate two recently proposed loss recovery mechanisms to reduce tail loss latency, RTO Restart (RTOR) and Tail Loss Probe (TLP), together with a new mechanism that we propose, which applies the idea of RTOR in TLP (TLPR). Our evaluations show that the relative performance of RTOR and TLP is scenario dependent, but TLPR provides the maximum gain in terms of latency reduction in most of the scenarios evaluated.

Furthermore, TCP's in-order reliable data delivery may cause head-of-line (HOL) blocking delays. In this case, when a packet loss happens, all the subsequent packets are stalled at the receiver unless the lost packet is recovered. To primarily overcome the limitations of TCP for telephony signaling transport, the message based Stream Control Transmission Protocol (SCTP) [30] was standardized. SCTP provides advanced features such as multi-homing, multi-streaming, and partial ordering. Both multi-streaming and partial ordering in SCTP can in general avoid HOL blocking delay if the corresponding application can accept out of order data. Moreover, PR-SCTP [31] is a partially reliable extension to SCTP that provides partial reliability on a per message basis. Using PR-SCTP, applications can specify particular reliability requirements for each message, which allows an application to skip (re)transmission of data that are no longer needed or avoid waiting for data that are not received. PR-SCTP thus offers a flexible tradeoff between timeliness and reliability for latency-sensitive applications. Real time multimedia applications have been shown to benefit from PR-SCTP [32, 33] by trading reliability for timeliness when network resources are congested. In addition, several Internet Engineering Task Force (IETF) working groups, such as real-time communication in WEB-browser (rtcweb) [34, 35] and IP Flow Information Export (IPFIX) [36], are also considering PR-SCTP for partial reliability.

In the thesis, we analyze PR-SCTP as a promising transport candidate for event logging. According to our analysis, quite a few factors can limit the performance of PR-SCTP. Most importantly, loss recovery in PR-SCTP can be very inefficient, for example when a large number of messages with different

reliability requirements are lost. This may happen in event logging where messages are typically small and are associated with different priority levels. In the thesis, we propose a couple of mitigations to improve the performance of PR-SCTP. One of the solutions takes advantage of non-renegable selective acknowledgements (NR-SACKs) [37, 38]. The NR-SACKs mechanism has been proposed for standardization in the IETF and is available in the FreeBSD operating system. In the evaluation, we focused on a specific event logging system, syslog [39]. Our evaluations indicate that NR-SACKs based PR-SCTP reduces the average message transfer delay as compared to TCP, SCTP and the existing PR-SCTP. Furthermore, unlike UDP, the proposed method is as reliable as TCP or SCTP for high priority messages.

The rest of the thesis is organized as follows. Research objectives are defined in Section 2. After reviewing the relevant work related to this thesis in Section 3, we describe the research methodology used in the thesis in Section 4. In Section 5, we discuss the main contributions of the research. Section 6 outlines the appended papers. Conclusions and future research are discussed in Section 7. The remainder of the thesis includes the appended papers.

2 Research Objectives

The overall objective of this thesis is

> ***to increase our understanding of the latency issues in the Internet and to propose and evaluate some transport level mechanisms to reduce this latency.***

To carry out our overall research objective, we focus on a number of sub-objectives. The first two of the following sub-objectives are tied to the overall objective of contributing knowledge towards the understanding of the latency issue.

The importance of reducing latency is largely influenced by the widespread presence of latency-sensitive flows in the Internet. Understanding such flows and their development over the past several years may help us make well qualified decisions on the latency-bandwidth trade-offs for these flows. More specifically, our first research objective is

> ***to identify latency-sensitive flows and quantify their ubiquity in the Internet.***

Subsequently, we investigate the flow start-up mechanisms in the Internet that have been a major source of latency as network links have become faster. The research objective is

> ***to investigate why adding more bandwidth does not help to get flows up to speed.***

Furthermore, we concentrate on two more sub-objectives to fulfill the second part of our overall research objective. These objectives are related to optimizations in different transport level mechanisms to support latency-sensitive flows. The first sub-objective is

> ***to enhance the loss recovery mechanisms in TCP to reduce latency for delay sensitive applications.***

Finally, we investigate on how to reduce HOL blocking delays for latency-sensitive applications. The idea is to explore the possibility of prioritization at the transport layer during the loss recovery. The research objective is

> ***to investigate the trade-off between reliability and timeliness for latency-sensitive applications.***

In the last sub-objective, using PR-SCTP, we investigate the benefit of data prioritization at the transport level. Our primary application is event logging with a focus on syslog. We determine a number of factors that influence PR-SCTP's performance. Moreover, we propose and implement an optimization to PR-SCTP. We thoroughly investigate our optimization in PR-SCTP for syslog.

3 Related Work

This section describes the required background and the most relevant previous research related to the work presented in this thesis. Since one of the main objectives of the thesis is to understand the latency issue in the Internet, we start with the definition of latency along with the definition of bandwidth to illustrate the difference. Furthermore, we present some existing work that shows that the bandwidth limitations no longer constrain most latency-sensitive applications.

In order to quantify the presence of latency-sensitive flows in the thesis, we needed to classify overall Internet flows. The classification is relevant to existing traffic classification work. We briefly mention those works here. We also show that a large portion of Internet flows are limited by the flow start-up latency. There is some work that also characterizes the limitation of flow start-up that contributes to latency. We summarize them here as well.

The second part of the thesis investigates the possibility for optimization in transport level mechanisms such as loss recovery to support latency-sensitive flows. We therefore include some of the existing work on loss recovery mechanisms in TCP for latency-sensitive flows.

Finally, we have worked on the possibility of prioritization in transport layer to reduce latency. We particularly investigate the applicability of partial reliability in PR-SCTP. Although partial reliability was standardized as PR-SCTP in 2004 [31], its origins date back to the early 1990s. This section presents some of the related work on partial reliability followed by an outline of PR-SCTP. We mainly study the performance of PR-SCTP with syslog as

the application in the thesis, but there are already many other applications that have been shown to benefit from PR-SCTP. We outline them here as well. Moreover, we include a subsection on event logging to illustrate the benefit of PR-SCTP as its transport together with some related work where partial reliability also has been shown useful for syslog. In addition, in our analysis, we see the influence of message sizes on the PR-SCTP performance. We therefore present some of the existing work that also looked into the importance of the message size from both the application and network perspectives. Finally, we mention the NR-SACK mechanism that we use to optimize the performance of PR-SCTP for small messages.

3.1 Increasing Bandwidth does not Solve the Latency Problem

Bandwidth is a very common term in communication technologies. In the most common definition, it indicates the throughput of a communication path or the amount of data that can be transferred at a given time over a logical or physical communication path. On the other hand, latency is the time required to complete a task that involves a source sending a particular packet(s) and the destination receiving it. For example, every chunk of data produced by end points in a real time interactive video conference or an online game is important. The task is therefore to transmit every chunk. On the other hand, the task can be to complete a whole (often small) transmission, for instance to download java-script codes to start an application.

Latency can correlate with bandwidth for a large transmission, since greater bandwidth means a wider communication path, and more data can be sent in parallel. However, for applications such as web browsing, VoIP or online gaming, larger bandwidth does not necessarily provide lower latency. These applications do not send much data and thus are limited by latency, not by bandwidth. For example, a large bandwidth connection for VoIP does not necessarily solve the problem of choppy or sluggish voice transmission if the latency is too high. Upgrading the connection bandwidth does not help much.

Belshe [40] performed a detailed quantitative study on the impact of bandwidth and latency on the page load times in a browser for various popular web destinations. Web browsing is chosen because web pages consist of short, application-limited flows, which require fetching hundreds of relatively small resources from different sources in the Internet. The effective round trip time (RTT) represents the latency in the study. In the first sub-figure in Figure 1, RTT is held fixed and only the bandwidth is changed from 1 to 10 Mbps. There is a large improvement in page loading time when the bandwidth is doubled from 1 to 2 Mbps. From 3 Mbps and on we see diminishing returns as the improvement becomes very small with the increase of bandwidth. However, we see a completely different trend in the second sub-figure when RTT is decreased and the bandwidth is held constant. For every 20 ms of RTT improvement, we see a linear decrease of page loading times. Similar observations were reported in [41, 42]. Studies show that users while browsing can perceive the lag once a

delay between 100 to 200 ms is added. Once the delay exceeds the threshold of 300 ms, the web page is reported to be sluggish and after 1000 milliseconds of delay, users are likely to move on [43]. Latency, and not only the bandwidth, should therefore be a concern while choosing an ISP.

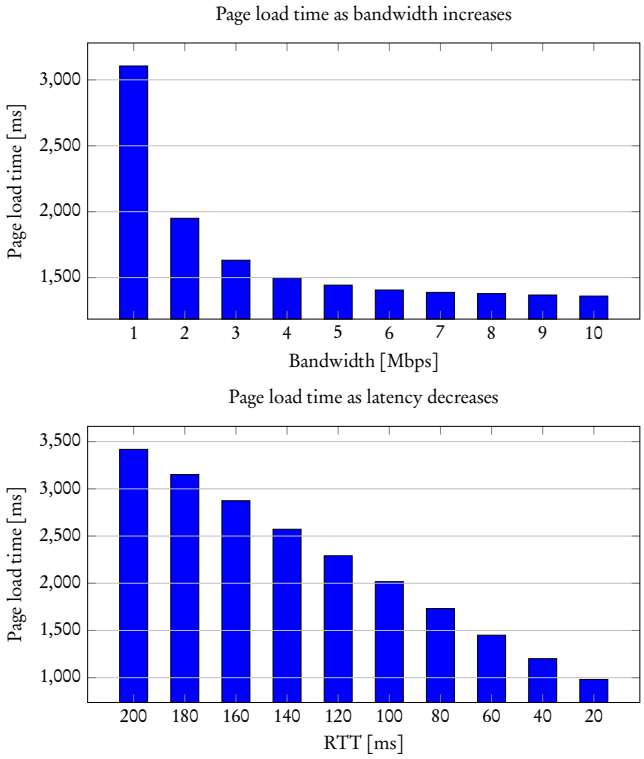

Figure 1: Page load time vs. bandwidth and latency [40].

To better understand the impact of latency in today's communication, one needs to know the possible sources of latency in the Internet. Several components in the communication path separately contribute to latency in the overall time required for data delivery. Apart from the propagation delay constrained by the speed of light, the other sources of latency are [27]:

- Structural delays: Suboptimal placement of network components like servers, caches, proxies and suboptimal routes between source and destination may contribute to the latencies experienced by application users.
- End-to-end interaction delays: Transport protocols such as TCP contribute to latency from initiating a connection (even more for initiating a secure connection), recovering packet losses and aggregating messages.
- Transmission path delays: Delays from medium acquisition, serialization, link error recovery, switching/forwarding delay and queuing delay contribute to the latency that a single packet experiences along the transmission path from the source to the destination.

- Link capacity related delays: Delays from sharing insufficient capacity among flows and from underutilizing excess capacity can contribute to overall latency.
- Intra-end-Host delays: Delays from buffering or head-of-line blocking in the networking stack within the hosts internally contribute to the overall latency.

A detailed survey of the possible sources of latency and how these have been addressed in existing work can be found in [27].

3.2 Traffic Classification

Understanding Internet flows has been a popular research topic for over a decade [44–49]. One frequently investigated aspect is the size distribution of Internet flows. The most well-known finding is that a small percentage of flows contribute the most to the total traffic volume, whereas the rest of the flows are small. This is commonly known as the elephant and mice phenomenon [50]. Other aspects of Internet flows that have been investigated are their duration, where long running flows are called tortoise and short ones dragonfly [45], their rate, where faster flows are called cheetah and slower ones snail [47], and their burstiness, where bursty flows are called porcupine and non-bursty ones are called stingy [47]. Brownlee *et al.* [45] showed that most flows in the Internet are dragonflies, where only a small number of tortoises continue from tens of minutes to days, but they carry a high percentage of total traffic volume (50–60%) on links. Qian *et al.* [48] showed an order of increase in the size of elephant, cheetah and tortoise flows in Internet as compared to the study in [46, 47]. Furthermore, studies in [46–48] also showed how the different aspects of Internet flows are related. The size and the rate of a flow were found to be strongly correlated, whereas the size and duration of a flow were found to be independent of each other [46, 47]. In a recent study [48], however, a less strong correlation between the size and rate of a flow was observed.

One other way to understand Internet flows is to classify traffic, which gives information about application usage, trends, emerging applications, anomaly detection etc. The traditional way of traffic classification based on well-known port numbers is no longer reliable [51–53]. Most of the applications either do not use IANA [54] registered well-known port numbers or use port numbers of another service (e.g., port 80 of HTTP) to hide themselves for reasons such as bypassing firewalls. Due to the proliferation of encrypted traffic and privacy concerns, the effectiveness of payload based traffic classification [51] is also questionable. Consequently, a large number of techniques have emerged in recent years [55–61] that classify traffic based on a combination of observable flow level features. These methods use machine learning and learn from empirical data to automatically assign flows to corresponding classes. At the end, these methods categorize flows into coarse grained classes of bulk flows, VoIP flows etc.

3.3 Initial Sending Rate

Both TCP and SCTP use congestion control [62] to adapt to the available capacity in the network path. Still, initially or after a long idle period, it is difficult to determine an appropriate sending rate for a flow that is not harmful for other flows sharing the same path. TCP's and SCTP's congestion control use a slow-start algorithm [26] to probe for the appropriate sending rate. Using this algorithm, each flow starts with a small congestion window; then, for each acknowledgement received, it increases the congestion window, which roughly doubles for each RTT and leads to an exponential increase of the congestion window. However, slow-start may often not be able to reach the available capacity before the end of a flow (as is the case of flows that do not fit in the initial congestion window), significantly contributing to the overall latency. Several earlier works [63–82] characterized the limitation of slow-start, especially for poor bandwidth utilization, which increases latency. Beside, slow-start's exponential increase of the congestion window can be too aggressive; it can overshoot available capacity, especially for large flows, and cause excessive packet loss [64, 66–69, 71, 72].

3.4 Loss Recovery in TCP

Paper III proposes several modifications to loss recovery in TCP. The related background is however briefly mentioned. In the following, we therefore discuss the standard loss recoveries in TCP in some details.

TCP recovers loss by detecting and retransmitting the lost segment(s). One of the methods TCP uses for loss recovery is called fast retransmit (FR) [26]. When a packet is lost, all the following packets arriving at the receiver out of order trigger duplicate acknowledgments (dupACKs) that indicate the expected packet. However, reordering of packets in the Internet can also cause dupACKs. The TCP sender therefore usually does not infer a loss unless a threshold number of dupACKs, three in the standard, are received.

TCP uses another method called retransmission timeout (RTO) [26] when FR is not possible in scenarios where three dupACKs are not available. This is a typical scenario for application-limited traffic having only a few outstanding segments. In this case, the TCP sender relies on the retransmission timer. It detects a packet loss when the corresponding retransmission timer goes off. This method is called RTO. It detects a loss slower than FR. A detailed description of the RTO mechanism is given in RFC 6298 [83]. Figure 2 shows both FR- and RTO-based loss recoveries in TCP[2]. In this figure, when there are three outstanding segments following a loss, the sender can use FR as soon as it receives three dupACKs. However, in the right hand side scenario in the figure when three dupACKs are not available, the sender uses RTO[3]. In such a scenario, the connection is typically limited by the congestion window, the receiver window or the application.

[2]For the illustration, other enhanced loss recovery mechanisms have not been considered.

[3]The dupACK threshold can be dynamically adjusted up to a limit while using the early retransmit mechanism [84].

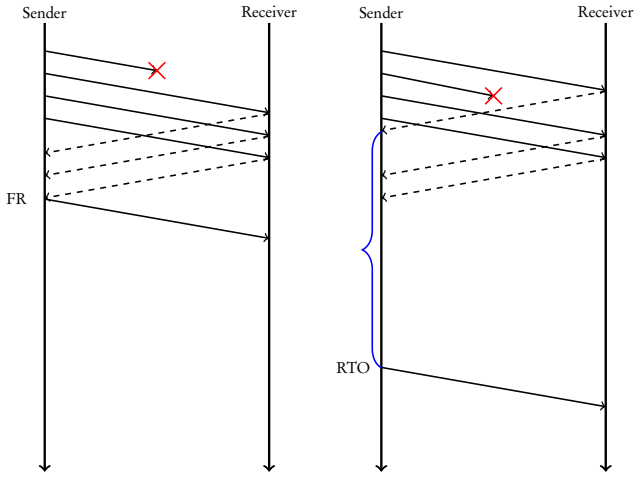

Figure 2: Two loss recovery mechanisms in TCP, FR on the left and RTO on the right.

In an application-limited scenario, applications generally send short flows or multiple bursts of short flows. For example, a typical web application produces short flows. Ramachandran [85] showed that 70% of all objects found in the top 500 sites are too small to invoke FR. Furthermore, [86] showed that RTO was used in about 50% of all loss recoveries by a busy web server. Rewaskar *et al.* [87] analyzed different sets of TCP connections collected from transmission capacities ranging from 155 Mbps to OC–48 (transmission speeds up to 2488.32 Mbit/s) and found that RTO is responsible for 32.8% to 66% of all retransmissions, whereas FR is responsible for only 9.6% to 17%. Moreover, according to [88], 77% of losses at the *Google* server are recovered by RTO. Furthermore, applications that use repeated burst transmissions including web applications using persistent connections, online games, stock trading systems, remote computer operations and sensor networks are typically interactive and generate application-limited flows. Due to their interactive natures, all these applications are time critical. A delayed response to loss events reduces application performance.

The existing solutions for application-limited flows can be categorized into two classes, one class of solutions triggers FR more often [84, 89–91] where the other class [92–99] improve the existing RTO mechanism so that less time is spent to detect losses. None of these existing solutions perform well when tail loss (last segment(s) in a flow is lost) happens. RTOR [100] and TLP [88] are two recent mechanisms that are being standardized in IETF to fight tail loss latency. We mention these mechanisms very briefly in Paper III. We therefore describe them more thoroughly for readers in this section.

3.4.1 RTOR

All mechanisms that improve the original RTO calculation in TCP are complementary to how RTOR works. The RTO management algorithm in RFC 6298 [83] recommends that the retransmission timer is restarted on the reception of an ACK that acknowledges new data while outstanding data are still

available. A retransmission therefore occurs after RTO seconds from when the ACK was received, not RTO seconds after the transmission of the potentially lost segment(s). In most cases, the extra delay is approximately one RTT that is added to the total loss recovery time. This delay can be quite significant in a high RTT scenario such as in a mobile network. In addition, if the receiver uses delayed ACK and the ACK that restarts the timer is a delayed ACK, the total loss recovery time becomes RTO+RTT+delACK, where delACK corresponds to the receiver's delayed ACK setting. The RTO management is illustrated in Figure 3.

There are two RTO scenarios in Figure 3. In both cases, three packets are sent. In the left hand scenario, when the TCP sender sends the first packet, the retransmission timer is started and this packet happens to be lost in the network. The timer will expire after RTO seconds. However, if the last of the three packets is lost (tail loss), as seen in the right hand scenario in the figure, the timer is restarted when the delayed ACK arrives at the sender. Consequently, the third packet will be retransmitted after RTO+ y seconds, not RTO seconds, after its first transmission. Moreover, if the second packet is also lost, acknowledgment delay at the receiver will add further delay to the loss recovery.

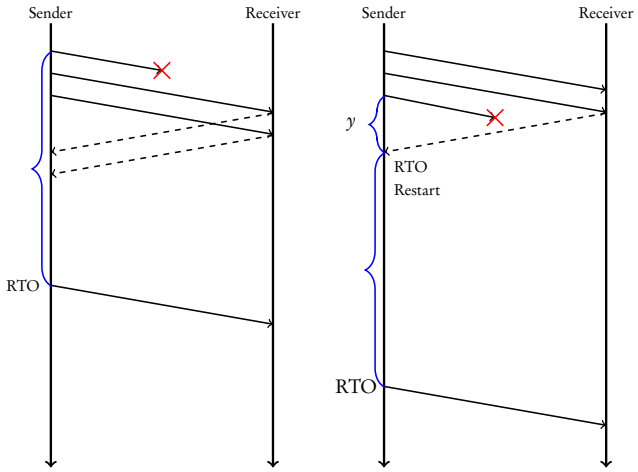

Figure 3: Two cases in RTO management in TCP.

In RTOR, when a retransmission timer is reset, it is reset to "RTO - T_earliest", where T_earliest is the time that has elapsed since the earliest outstanding segment was transmitted, so that retransmission will always occur after exactly RTO seconds. T_earliest also corresponds to the y in Figure 3. Although this approach might make the RTO a little more aggressive, it still follows the standard [83] that requires that no segment should be retransmitted earlier than RTO seconds after its first transmissions. The RTOR algorithm is described as follows. When an ACK is received that acknowledges new data:

1. T_earliest $\leftarrow 0$
2. Then, if the following two conditions hold,
   (a) the number of outstanding segments is less than four and

(b) there are no unsent data ready for transmission, then:

set T_earliest to the time elapsed since the earliest outstanding segment was sent.

3. Restart the retransmission timer so that it will expire after "RTO - T_earliest" seconds (for the current value of RTO).

A solution similar to RTOR is proposed in [99], although the solution is based on an estimate of the implicit RTT. Previously, the work in [97] mentioned the problem with implicit RTT in the RTO and suggested that it should be removed. However, Ludwig [98] mentioned the implicit RTT as a safety margin against spurious retransmission and recommended not to remove it. The performance impact due to spurious retransmissions is however typically low for application-limited flows, since the number of outstanding segments is small. In addition, solutions such as F-RTO [97, 101] are already implemented in operating systems such as Linux to undo the effect of the spurious retransmissions.

3.4.2 TLP

TLP [29, 88] tries to trigger loss recovery by e.g., early retransmit (ER) [84] by inducing additional selective acknowledgments (SACKs) at the receiver, even when the last segment is lost. TLP introduces a separate timer event called probe timeout (PTO). Using TLP, a TCP sender transmits one probe segment after a PTO if acknowledgements are due for the connection but it is otherwise idle because, for instance, the connection is congestion window, receiver window or application-limited. Equation 1 details the calculation of the PTO. The PTO values can be different depending on the number of outstanding segments. In the equation given below, SRTT is the smoothed RTT and WCDelAckT is the worst case delayed ACK timer. When there is only one outstanding segment, PTO is additionally inflated to compensate for the delayed ACK setting at the receiver. WCDelAckT is recommended to be set to 200 ms in [88].

$$PTO = \begin{cases} \max(2 * SRTT, 10ms), & \text{if there are more than one outstanding segments} \\ \max(2 * SRTT, 1.5 * SRTT & \\ +WCDelAckT), & \text{otherwise.} \end{cases} \quad (1)$$

The transmitted probe, also known as a loss probe, can be a new segment, if available and the receive window permits; otherwise the most recently transmitted segment is retransmitted. When there is tail loss, the ACK from the loss probe helps in two scenarios: (i) the actual probe is able to repair a loss of the last segment; (ii) previously lost segments can be recovered either by provoking ER or by forward acknowledgments (FACK) [102]. On the other hand, if the loss probe is sent in an absence of loss, there is no change in the congestion control or loss recovery state of the connection, with the exception of any state

related to TLP. Besides, TLP applies only when a TCP connection is in open state, meaning ACKs are being received in order.

3.5 Partial Reliability

Transport protocols such as TCP or User Datagram Protocol (UDP) are often poor choices for multimedia traffic such as MPEG flows with multiple frames with different QoS requirements, real-time gaming traffic with both critical status update and non critical object location and state information. TCP's flow control, reliability and in-order delivery often induce extra latency, which is inconvenient for continuous media, video or audio, with real time requirements.

UDP is on the other hand unreliable and connectionless, and provides no congestion control. UDP would therefore be an acceptable choice if loss in the underlying network does not go beyond the application's tolerance of loss. However, continuous growth of applications without using congestion control could lead to a congestion collapse in the Internet [103].

With neither TCP nor UDP being appropriate for multimedia traffic, some solutions are based on implementing required transport functions in the application layer on top of UDP, i.e. by using RTP [104]. Nevertheless, implementing transport functions such as round trip time estimation, congestion control and flow control is non trivial. These functions require wide scale testing before deployment. A possible solution for transporting data with different QoS is to relax the reliability service for some data at the transport layer while keeping TCP's proven congestion control and flow control services [105] at the same layer. This is referred to as partial reliability. The idea is to allow the application layer to specify the reliability requirements according to the flexible trade-off between delay and reliability required by several multimedia applications.

The pioneer work of Dempsey [106] showed the feasibility of retransmission based partial reliability for multimedia transmission, for which retransmissions were not thought to be practical. Using an appropriately sized playout buffer at the receiver was found to be a solution to make the retransmissions useful. Dempsey *et al.* [106,107] developed a retransmission based error control service, partially error controlled connection (PECC), that allows an application to specify its loss tolerance, which enables the receiver to occasionally ask the sender for retransmissions that fulfill the minimal reliability requirements while keeping the overall delay low. Conard *et al.* [108] proposed a message based protocol called POCv2 as an extension of the earlier partial order connection (POC) [109, 110] that also allows any application to assign each application message to one of the three reliability classes: reliable, partially reliable and unreliable. The receiver then asks for retransmissions based on these assignments.

Another sender based partially reliable transport protocol called heterogeneous packet flows (HPF) was proposed in [105]. Similar to POCv2, HPF enables an application to specify particular reliability requirements for data

units. However, the decision to retransmit is kept to the sender unlike POCv2 or PECC.

PR-SCTP [31] is a partially reliable extension of SCTP. It can provide partial reliability on a per-message granularity based on application-level specifications. PR-SCTP allows applications to set different reliability services for different messages. *Timed reliability* is one such reliability service where the application can set a lifetime value on every message before forwarding it to the PR-SCTP layer. PR-SCTP simply abandons a message and does not (re)transmit if the associated lifetime value is expired. Instead, it sends a special control chunk called *forward_tsn* that tells the receiver to advance its cumulative ACK point and to no longer expect that particular abandoned message. This might be useful for any time sensitive traffic that becomes useless beyond a certain time. Network resources are better used for useful messages, especially during a congestion period. Two additional partially reliable services based on a limited number of retransmissions and prioritization of user messages have been standardized in IETF [111]. A detailed survey of partially reliable transport protocols is given in [112].

3.5.1 Applications Using PR-SCTP

A vast number of existing work [32, 33, 113–124] showed the usefulness of PR-SCTP for real time multimedia streaming. Low bit rate video conferencing, remote video surveillance, and interactive gaming are some examples of target scenarios. In streaming, using either MPEG or H.246/AVC [125] encoding, source data are encoded into three types of frames: I- frame (intra coded frame), P-frame (predicted frame) and B-frame (bi-directional frame). Multimedia streaming, both audio and video, typically prefers timeliness over strict reliability. However, I-frames contain the most information and are therefore critical for the playback performance at the receiver. The streaming applications can therefore use timed reliability service in PR-SCTP to set longer lifetimes for I-frames, as compared to P- or B-frames, to save unnecessary delays but still ensure good media playback performance by giving reliable delivery for I-frames.

Furthermore, Fitzpatrick *et al.* [126, 127], Chou *et al.* [128] and Huan *et al.* [129] proposed PR-SCTP as the transport for real time traffic, such as VoIP conference or video streaming, primarily to reduce head-of-line blocking delay on a per stream basis by partial reliability. PR-SCTP has also been proposed as the transport for Internet protocol television (IPTV) traffic in [130], and for session initiated protocol (SIP) messages in [131].

The aim of RTCweb [34] is to provide a whole new level of interactivity for web users with real time communications such as gaming, texting, audio/video conference along with regular file transfer using web browsers. In the RTCweb proposal, PR-SCTP is used to provide both reliable and unreliable delivery semantics [35]. For example, in multi user network gaming using RTCweb, partially reliable delivery is sufficient for position and object state information that are only consistent for a short time, whereas reliable delivery is needed for critical state information in the game or for non real time file transfer.

Moreover, IETF standardized IPFIX [36] for collecting Internet flow information from routers, probes and other devices. Network operators can use IPFIX to obtain measurements, and accounting and billing information for users. In the IPFIX proposal, PR-SCTP is mentioned as the mandatory transport for IPFIX messages, while both TCP and UDP are mentioned as optional. Primary reasons are PR-SCTP's per message based partial reliability and multi-streaming that reduce the head-of-line blocking delay [30]. For instance, IPFIX messages from security and billing applications require reliable delivery, but a capacity planning related application can use partially reliable delivery due to its loss tolerance.

3.5.2 Event Logging

Event logging in computing systems is useful for various purposes, for instance trouble shooting when anything goes wrong and monitoring performance and behavior. An event log describes events that notify users of some incidents related to operating systems on servers, clients or other networking devices. Event logs are saved locally and are often transferred to some centralized server. An event log contains information about the severity level of the event that generates it and can therefore be prioritized. Accordingly, a partially reliable transport can be a reasonable choice for transferring event logs over computer networks.

While IPFIX is mainly proposed for network operators, any UNIX-like operating system commonly uses the syslog protocol [132] for event logging. The syslog system facilitates any machine or device in sending event logs over networks to the syslog server. UDP was mentioned as the transport for the initial syslog specification [133], but TCP has later been specified in the more recent specification [134]. While reliable delivery for the critical syslog messages is important, it has some potential drawbacks. Syslog runs inside a priority process in Unix/Linux, so if a syslog sender is blocked for some reason, such as that the receiver is unable to accept any messages, we can have a system wide halt [39]. RFC 5424 [39] suggests implementing the transport system in a way that a sender can discard messages that may otherwise block the sender. In this case, low priority messages are recommended to be discarded in favor of high priority messages. A partially reliable transport is therefore a good candidate for syslog messages. Furthermore, [135] proposed an application based solution, where loss of a high priority syslog message is detected and retransmitted faster than low priority messages, providing timely delivery for high priority messages.

3.6 Performance of Small Packets

In this thesis, we show the influence of small packets on the performance of PR-SCTP. In the following we separately discuss the existing work on the overhead for small messages that may lead to small packets and the implication of packet sizes in the buffering mechanisms at the network.

3.6.1 Overhead for Small Messages

Byte based TCP is inconvenient as a transport for telephony signaling messages. Such messages are small in size and need special application layer marking to be interpreted at the receiver. SCTP was therefore designed as message based to facilitate transporting telephony messages [30]. In a SCTP packet, each of these messages is placed in a separate chunk that has a 16-byte header. Considering a 20-byte IP header and a 12-byte common header for a SCTP packet, the overhead of small messages in SCTP can be substantially high. This is a classical problem that was first observed in the Tymnet network in the 1960s [136]. Dreibholz *et al.* [137] also showed that user message size influences SCTP throughput.

In 1984, John Nagle presented a simple but elegant solution [136] to solve the small packet overhead problem in TCP. In Nagle's algorithm, a TCP sender buffers all user data until a full sized packet is formed or until no acknowledgements are due for the connection. The Nagle algorithm is included in most TCP implementations today but is turned off by default, because this can potentially affect the timeliness of user data. Although Nagle's algorithm was not mentioned in the SCTP specification in RFC 4960 [30], the socket API extensions for SCTP [138] include a socket option, 'SCTP_NODELAY', to turn on/off any Nagle-like algorithm. By turning on the Nagle algorithm, multiple chunks can always be bundled in a single SCTP packet. In addition, if multiple chunks are queued up in the send buffer, for instance due to congestion control, they are always bundled.

3.6.2 Byte Based or Packet Based Buffering

Drop-tail is the simplest and is a traditional buffering mechanism used in Internet routers. During congestion, packets are simply dropped if all buffer space is filled up. However, there are several shortcomings of drop-tail buffering [139]. An advanced queue management (AQM) by the name of Random Early Detection (RED) was proposed in [140]. RED requires a large number of parameter tuning, which leads to several new queue management algorithms such as CoDel [141], PIE [142], and a combination of CoDel with flow fair queuing (FQ_CoDel) [143]. However, AQMs have only occasionally been deployed in the core network [144] and ISP's infrastructure [145]. Drop-tail buffering still exists in access network equipment and in middle boxes such as firewalls [146].

In every buffering mechanism, buffer occupancy can be measured either in terms of bytes or in terms of packets [139]. Unlike packet based buffering, packet sizes are taken into account in byte based buffering. In a byte based one, small packets are more likely to have space in the buffer during congestion, which provides lower loss probability for control packets such as DNS messages that tend to be small. However, recent work in IETF [146] disapproved the preferential treatment of small packets as it might induce DoS vulnerabilities.

3.7 NR-SACKs

In both TCP and SCTP, SACKs allow a receiver to acknowledge out of order data [26,30]. The sender uses SACK information for selectively retransmitting. However, SACK information is only advisory; the sender must keep the selectively acknowledged (SACKed) data until these are cumulatively acknowledged, because the receiver can always discard any SACKed data. Discarding SACKed data before delivering to the receiver application is known as 'reneging' [147].

Unlike TCP, SCTP provides unordered delivery that provides a SCTP receiver the ability to forward out-of-order data to the receiving application. Although these data will be SACKed, they are non-renegable. However, using the standard SACK mechanism, the SCTP receiver cannot provide a distinction between SACKed data that are renegable and SACKed data that are non-renegable.

NR-SACK [37,38] is a recently proposed acknowledgement (ACK) mechanism for SCTP that allows a receiver to explicitly provide the renegibility information of SACKed data. The NR-SACK chunk, as shown in Figure 4, extends the regular SACK chunk in SCTP. Using NR-SACK, the receiver can list non-renegable out of order data chunks in 'NR Gap Ack' Blocks in the figure. The NR-SACK mechanism has been proposed for standardization in IETF along with CMT-SCTP [38].

Type=0x10	Chunk Flags	Chunk Length
Cumulative TSN Ack		
Advertised Receiver Window Credit		
Number of Gap Ack Blocks=N		Number of NR Gap Ack Blocks=M
Number of Duplicate TSNs=X		U N U S E D
Gap Ack Block #1 Start		Gap Ack Block #1 End
…		
Gap Ack Block #N Start		Gap Ack Block #N End
NR Gap Ack Block #1 Start		NR Gap Ack Block #1 End
…		
NR Gap Ack Block #M Start		NR Gap Ack Block #M End
Duplicate TSN 1		
…		
Duplicate TSN X		

Figure 4: An NR-SACK chunk [38].

4 Research Methodology

Theoretical computer science basically studies what can be computed and the associated cost. Experimental computer science on the other hand uses scientific methods for its inquiries. Scientific method is not strictly defined. However, in most cases, scientific method refers to a cycle of observation,

hypothesis building or description, and experimental testing for the verification of prediction.

In the iterative scientific method, a hypothesis is constructed as a tentative answer to the question formulated in the observation stage. This hypothesis is then verified in the experiment stage. The scientific method, as illustrated in Figure 5, was used as the research methodology in this thesis. In this model, the observation stage consists mainly of a literature review. This phase may employ published works to identify the relevant problems in which a researcher is interested. Furthermore, the state of the art or the highest level of development for the relevant problem is pointed out in this stage. Once the observation phase is complete and a problem statement or research question is stated, a hypothesis is formulated. This hypothesis generally delivers a feasible answer to the research questions and allows making predictions based on some assumptions about the system under consideration. The next stage is hypothesis testing. This is called experiment and performance evaluations.

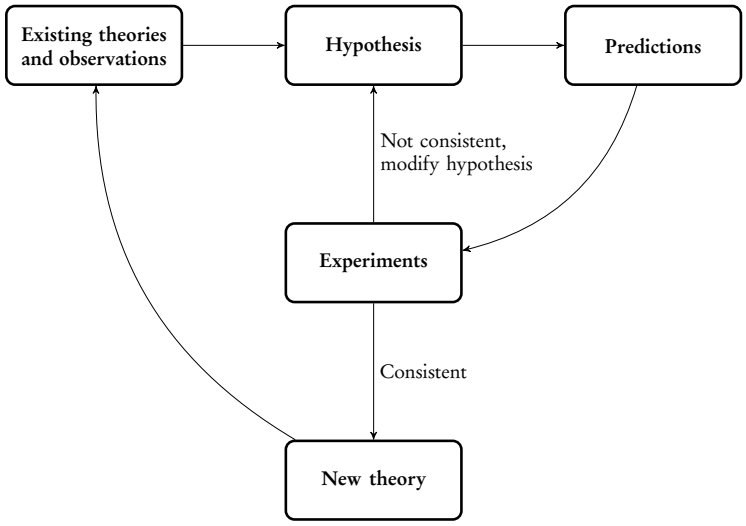

Figure 5: Diagram describing the iterative nature of the scientific method, adapted from [148].

There are several methods that are used in hypothesis testing such as real measurements, simulation, emulation and analytical methods. Measuring real systems represents the lowest possible level of abstraction, but is also the most difficult technique. One of the goals of performance analysis is to characterize the system as certain parameters are varied. Since it is very difficult to change parameters in running systems, real measurements are rarely used. Moreover, the measurements themselves may perturb the system, which is also quite difficult to isolate. Therefore, other kinds of experimental techniques are preferable in many cases.

A simulator is a computer program that models important features of a system under test. Since this is a computer program, varying several parameters is

comparatively easy. However, considerable effort is needed to write and debug a reasonably sized simulation program. Besides, the model used might ignore a critical behavior of the system, improperly handled initial conditions may lead to incorrect conclusions or too much simplification of assumptions may limit the accuracy of the end result. Nevertheless, simulation is quite popular because of its high degree of flexibility and its relative ease of implementation [149].

Since both simulation and real measurements have their drawbacks, a mixture of real and simulated entities called emulation is becoming more and more common. Emulation is quite frequently used in networking research. In this case, real machines with real implementations of networking stacks communicate over an emulated network. Dummynet [150] is one such emulator that can emulate different network scenarios.

Finally, an analytical method can also be used for hypothesis testing. An analytical method mathematically models a system under investigation. As compared to the above mentioned methods, results from analytical models can be less accurate [149]. However, an analytical method can give quick insight into the overall behavior of the system. Besides, an easy to perform analysis can simply be used to coarsely verify the results of other methods.

In this thesis, Papers I and II are based on real measurements made by CAIDA from the core Internet. We made several analyses using these measurements. Paper II also used analytical methods. Furthermore, Papers III-VII used emulation based experiment methods.

5 Main Contributions

We have investigated the traffic pattern of application-limited, latency-sensitive traffic and their share in the Internet over time. In order to classify potentially latency-sensitive traffic, we have clustered traffic collected from the Internet backbone over a time period, 2008–2012. According to our results, Internet traffic can be put into six classes including greedy and application limited traffic. While we have found that most of the application-limited traffic are low bandwidth traffic, a growing share comes from bandwidth intensive application-limited traffic such as video streaming. Overall, our analysis shows that the share of latency-sensitive flows has grown over recent years. Our classification is based on a hierarchical clustering method. All the results and analyses are presented in Paper I.

Furthermore, we have thoroughly studied the limitation of the slow-start algorithm that a flow uses at (re)start. We have shown that a growing proportion of flows in the Internet is limited by slow-start as network links get faster. According to our study, typical flows in the Internet are not large enough to exploit the capacity investment. We have also found that most of the time a new flow arrives at an empty access link, which suggests a great potential for a better flow start-up to use the idle capacity. We have also studied the lack of scalability in existing solutions that are purely end-to-end and the deployment challenges

in solutions that require network support. The whole study is presented in Paper II.

Next, we evaluated two recently proposed loss enhancements in TCP to counter tail losses, RTOR and TLP in addition to our new proposal, TLPR, that combines the logic of RTOR with TLP's timer management. In the evaluation, we have considered three different test scenarios: one with controlled tail loss, one with realistic background traffic producing losses due to congestion, and the last including real web page downloading. All three enhancements outperform the standard TCP. While the relative gain using TLP and RTOR is scenario dependent, TLPR has the largest gain in almost all scenarios evaluated. The results and analyses are presented in Paper III.

Finally, we analyzed the applicability of PR-SCTP for event logging applications with a focus on syslog. Any logging application is latency sensitive due to the need of a timely reporting of critical events. We evaluated the performance of PR-SCTP for syslog in experiments involving different network scenarios. To do this, we analyzed and modeled real syslog traces from an operational system as an input in our evaluations. Our evaluations suggest that PR-SCTP performance is impeded when message sizes are small. We then determined that the key mechanism in the existing PR-SCTP, the *forward_tsn* mechanism, becomes inefficient in the presence of small messages. In addition, small messages increase overhead. Small messages may also lead to small packets. However, when packet based buffering is used in the network, every packet, regardless of its size, has the same risk of being dropped. In such an instance, PR-SCTP encounters a higher loss rate per application byte when network resources are shared among competing flows.

We have proposed two enhancements to mitigate the inefficiency in the *forward_tsn* mechanism. We have implemented and evaluated one of the proposed solutions that takes advantage of the NR-SACK mechanism. NR-SACKs are available in the current FreeBSD operating system. The NR-SACK mechanism has been proposed for standardization in IETF as a part of concurrent multipath transfer (CMT) in SCTP [38]. In our evaluation, NR-SACK based PR-SCTP reduces the average message transfer delay by more than 75% as compared to existing PR-SCTP in some scenarios.

Our further evaluation of NR-SACK based PR-SCTP for syslog shows a significant performance improvement as compared to existing PR-SCTP, TCP and SCTP. NR-SACK based PR-SCTP provides a shorter average message transfer delay. Furthermore, in contrast to UDP, it provides reliable delivery of high priority log messages. All in all, using NR-SACK based PR-SCTP, a syslog application can have a flexible trade-off between timeliness and reliability. All the results and analyses are documented in Papers IV–VII.

6 Summary of Papers

Paper I: A Method for Hierarchical Clustering of Internet Traffic and its Use in Detecting Application-Limited Flows

In this paper, we classify traffic collected from the Internet backbone during the period 2008–2012. Our main aim is to understand the flow level characteristics of potentially latency-sensitive flows and their prevalence and development over recent years in the Internet. By knowing the proportion of latency-sensitive flows as compared to the greedy flows, researchers may make well grounded choices regarding trading off bandwidth for latency by adjusting level of aggression in retransmissions, for example. The classification produces different clusters including traditional greedy flows, DASH–like flows, short web–like flows, interactive flows such as from online gaming and persistent web flows. While the majority of the flows fall into rather short web flows, the proportion of DASH flows and interactive flows has increased in recent years. Moreover, not only do we see an increased number of DASH flows as compared to earlier measurements, but their bandwidth share has also recently increased. Our classification is based on a hierarchical clustering method. The paper also includes all the design rationales involving feature selection and the metric for similarity measure in the classification.

Paper II: What Use is Top Speed without Acceleration?

In this paper, we extensively study the lack of scalability of today's transport dynamics that largely limits the effectiveness of investing in more capacity. We quantify the scaling problem using CAIDA traces collected from an Internet backbone covering the time period 2002–2012. In our analysis, we show that a growing proportion of traffic is limited by slow-start rather than capacity. Overall, based on our analysis, flows large enough to make the most of increased capacity investment are rare in the Internet. Besides, most of the time a new flow arrives at an empty access link, which shows a great potential for improving the flow start-up mechanism to use the idle capacity. We do not propose a new solution or take any side in any existing solutions in the paper. We instead discuss the lack of scalability in the known solutions such as increasing the congestion window or using multiple concurrent flows that are the easiest to deploy. Besides, we mention the critical deployment challenges of existing solutions that require coordination between networks and end-systems.

Paper III: An Evaluation of Tail Loss Recovery Mechanisms for TCP

In this paper, we extensively evaluate two recently proposed tail loss recovery mechanisms, RTOR and TLP for TCP, together with our proposal, TLPR, that combines the idea of RTOR with TLP's timer management. We consider three distinct network scenarios in the evaluation. We use controlled loss in the first scenario whereas we use realistic background traffic to induce congestive loss in the second scenario. In the third scenario, we include real web page downloading in a network with artificially correlated losses. All three mechanisms reduce latency as compared to the standard TCP in the evaluated scenarios. When RTOR is triggered,

it can reduce at least one RTT from the total latency as compared to the standard. Our results show that the performance comparison of TLP and RTOR is scenario dependent, but TLPR provides the best performance in almost all cases.

Paper IV: Priority Based Delivery of PR-SCTP Messages in a Syslog Context

In this paper, we discuss the problem with the existing transport services such as TCP and UDP for syslog. We also describe several features of SCTP in relation to the syslog protocol and suggest PR-SCTP as a transport alternative for syslog. In our emulation based experimental results, PR-SCTP shows better performance than TCP in terms of the average delay for message transfer. Furthermore, PR-SCTP exhibits a lower average packet loss than UDP. In both cases, PR-SCTP exploits priority properties of syslog messages during loss recovery. However, the study reported in this paper is quite restricted. We chose a fixed message size for syslog, which limits observations of PR-SCTP performance. In addition, we only emulate an artificial loss scenario in the network without any competing traffic.

Paper V: Syslog Performance: Data Modeling and Transport

In this paper, we first model syslog data using real syslog traces from an operational network. The model includes several traffic parameters such as message size, interarrival time and fraction of important messages. The model is then used as an input in the performance evaluation of PR-SCTP. In the experiments, real congestion is introduced in the network by running several competing flows. Furthermore, in our previous work, we assumed a message size of 250 bytes as an approximation for syslog messages. In contrast, our study of real syslog traffic exhibits a far lower mean message size. Our evaluations show that PR-SCTP performance is heavily influenced by the syslog data size characteristics.

Paper VI: On the Effectiveness of PR-SCTP in Networks with Competing Traffic

Based on the findings reported in Paper V, a broad evaluation of PR-SCTP for different network scenarios and traffic characteristics is presented in this paper. We find that a number of factors can influence the performance of PR-SCTP. Firstly, small messages increase the overhead, which impacts the message transfer delay. Secondly, small messages can lead to small or less than full sized packets when they are bundled. When packet based buffering is used in the network, these packets may raise the loss rate per application bytes as compared to other competing flows that use full sized packets. Lastly and most importantly, the *forward_tsn* mechanism in PR-SCTP becomes inefficient, particularly when messages with different reliability requirements are lost in bursts. Furthermore, we propose an enhancement of PR-SCTP that requires an extension of the existing *forward_tsn* chunk to mitigate the inefficiency in existing PR-SCTP.

Paper VII: Performance Analysis and Improvement of PR-SCTP for Small Messages

In [151], we propose, implement and initially evaluate an enhancement to improve the *forward_tsn* efficiency in the existing PR-SCTP. The enhancement takes advantage of the NR-SACKs mechanism in the FreeBSD OS. The NR-SACKs mechanism has been proposed for standardization in IETF [38]. An NR-SACK chunk can selectively acknowledge out-of-order, non-renegable data. In this paper, we extensively evaluate and analyze PR-SCTP with NR-SACKs for different network scenarios and traffic characteristics. According to our analysis, using NR-SACKs improves the *forward_tsn* mechanism in PR-SCTP, particularly when application messages are small, have mixed reliability requirements and are bundled due to congestion control. We further investigate syslog traces collected from a bigger operational network than described in Paper V. We then perform a trace based evaluation to compare the performance of NR-SACK based PR-SCTP with existing transport protocols for carrying syslog messages. Our evaluation suggests a significant improvement in terms of average message transfer delay in PR-SCTP using our NR-SACK based optimization as compared to the existing PR-SCTP, TCP, and SCTP.

7 Conclusions and Future Work

Historically, networking researchers have worked on improving throughput and resource utilization, which particularly supports traditional bulk applications. However, today's web and cloud-based applications are interactive and latency-sensitive and, here, bandwidth maximization is no longer a constraining factor. It is therefore the right time for researchers to instead focus on latency. In this thesis, we show the share of latency-sensitive flows among overall Internet flows and their development over time. These flows have not only increased in number; their bandwidth share has also grown significantly in recent years owing to the adoption of HTTP segment streaming for video transmission. From our findings, the ubiquity of flows that are potentially latency-sensitive supports the growing concern of latency in the Internet.

In the thesis, we analytically show that the flow start-up mechanism in the Internet is a major component of latency. At the (re)start, every congestion controlled flow probe for the available bandwidth by linearly increasing the congestion window for every feedback received. Flows that do not entirely fit in the initial congestion window therefore experience increased latency while cautiously getting up to the speed. Over the years, as the network links are getting faster, most of the flows complete before they even discover that they did not need to be cautious. Furthermore, using a scaling model supported by a longitudinal analysis of packet traces collected from an Internet backbone, we show in which dimensions traffic has been scaling over the last decade. We show that most of the flows are limited by the flow start-up mechanism, not

capacity. In the thesis, we discuss why solutions such as increasing the size of the initial window and opening multiple flows are not scalable solutions to the flow start-up problem. We further discuss the limitation of existing alternatives that rely on various approaches such as state sharing at the sender, bandwidth estimation techniques and explicit signaling from the network to obtain information about the network. We furthermore show why none of these approaches is feasible as a general solution. Nevertheless, we believe that solutions that involve explicit signaling from the network without requiring any major change in the network will be required to solve the flow-start problem. We consider that the design of such a solution is a part of our future work.

Furthermore, in the thesis, we propose and evaluate different transport level mechanisms to support latency-sensitive applications. Tail loss events are quite common in the Internet. Application limited flows such as short flows suffer from long loss recovery delays when tail loss occurs. We compare the performance of two recently proposed loss recovery mechanisms, RTOR and TLP along with our new proposal, TLPR, that integrates the logic of RTOR with TLP to improve tail loss latency. In our evaluation, TLPR reduces the loss recovery times the most. However, as all these mechanisms allow TCP to be more aggressive than the standard, spurious retransmissions are more likely to take place. In the future, we aim to thoroughly study the aggressiveness of these loss recovery mechanisms. In addition, our knowledge about the proportion of application limited flows in the Internet will assist us in reaching a deep understanding of the latency/throughput trade-off for, for example, adjusting the level of aggression for retransmissions.

Finally, we investigate the possibility of prioritization at the transport layer to improve the performance of latency sensitive applications with a focus on syslog. To enable prioritization, we propose PR-SCTP as a potential transport protocol. In our evaluation, however, PR-SCTP exhibits a performance penalty when syslog messages are small with heterogeneous reliability requirements and bundled into packets due to congestion control. The existing *forward_tsn* mechanism in PR-SCTP becomes inefficient when these messages, bundled in a packet, are lost. We therefore propose and implement a solution to improve the efficiency in the *forward_tsn* mechanism. The proposed solution utilizes the NR-SACK mechanism of SCTP. In our evaluation, NR-SACK based PR-SCTP improves the application's performance in general. Moreover, NR-SACK based PR-SCTP shows improved performance as compared to the existing transport protocols SCTP, TCP and UDP for syslog traffic delivery.

There is a further possibility for improving PR-SCTP performance if expired messages can be discarded from the send buffer. This can occur when several messages are queued up during a congestion period. In this case, message lifetimes may expire due to long waiting times in the queue. We plan to modify the SCTP send buffer management functions as a part of our future work in such a way that discarding messages from the send buffer is possible. In addition to timed reliability service in [31], as mentioned before, two additional partial reliability services for PR-SCTP (described in [111]) have been standardized in IETF. One of the services, called "priority policy", allows an application

to associate a specific priority to each user message. Using this service, low priority messages are abandoned when a user message is stored in the send buffer and there is not enough space available.

In the future, we aim to modify a real syslog system to use NR-SACK based PR-SCTP. In this thesis, we have considered only a timed reliability based PR-SCTP service. A mapping of reliability requirements for different types of log messages to several PR-SCTP services is also part of future work. This might require defining and standardizing new PR-SCTP services.

References

[1] ITU-T. One-way transmission time. *ITU-T recommendation G.114*, 2003.

[2] M. Claypool and K. Claypool. Latency and player actions in online games. *Communications of the ACM*, 49(11):40–45, November 2006.

[3] F. F. Nah. A study on tolerable waiting time: How long are web users willing to wait? *Behavior and Information Technology*, 23(3):153–163, 2004.

[4] S. Cheshire. It's the Latency Stupid. `http://rescomp.stanford.edu/~cheshire/rants/Latency.html`, May 1996. Accessed: 2015-10-05.

[5] N. Dukkipati and N. McKeown. Why flow-completion time is the right metric for congestion control. *ACM SIGCOMM Computer Communication Review*, 36(1):59–62, January 2006.

[6] Akamai's [state of the internet]. *Akamai*, 8(2), 2014. `http://www.stateoftheinternet.com/downloads/pdfs/2014-state-of-the-internet-connectivity-report-2014-q2.pdf`, Accessed: 2015-10-05.

[7] S. Souders. Velocity and the bottom line. `http://radar.oreilly.com/2009/07/velocity-making-your-site-fast.html`, 2009. Accessed: 2015-10-05.

[8] B. Simic. The performance of web applications: Customers are won or lost in one second, November 2008. Aberdeen Group.

[9] E. Schurman and J. Brutlag. Performance related changes and their user impact. In *Proceedings of Velocity*. San Jose, CA, USA, June 2009.

[10] U. Hoelzle. The google gospel of speed. `http://www.google.com/think/articles/the-google-gospel-of-speed-urs-hoelzle.html`, January 2012. Accessed: 2015-10-05.

[11] T. Hoff. Latency is everywhere and it costs you sales - how to crush it. `http://goo.gl/F3Vvlh`, July 2009. Accessed: 2015-10-05.

[12] P. Dixon. Shopzilla's site redo - you get what you measure. In *Proceedings of Velocity*. San Jose, CA, USA, June 2009.

[13] The bufferbloat projects. `http://www.bufferbloat.net/`. Accessed: 2015-10-05.

[14] RITE project. `http://www.riteproject.eu/`. Accessed: 2015-10-05.

[15] Make the web faster. `https://developers.google.com/speed/`. Accessed: 2015-10-05.

[16] M. Claypool. The effect of latency on user performance in real-time strategy games. *Computer Networks*, 49(1):52–70, September 2005.

[17] F. Schneider, S. Agarwal, T. Alpcan, and A. Feldmann. The new web: characterizing AJAX traffic. In *Proceedings of International Conference on Passive and Active Network Measurement*, pages 31–40. Springer-verlag, Cleveland, OH, USA, April 2008.

[18] ITU-T. E.800: Terms and definitions related to quality of service and network performance including dependability. `https://www.itu.int/rec/T-REC-E.800-199408-S/en`. Accessed: 2015-10-05.

[19] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. D. Johnson. M2M: From mobile to embedded Internet. *IEEE Communications Magazine*, 49(4):36–43, April 2011.

[20] D. Ciullo, M. Mellia, and M. Meo. Two schemes to reduce latency in short lived TCP flows. *IEEE Communications Letters*, 13(10):806–808, October 2009.

[21] S. Ramachandran. Lets make the web faster. `http://code.google.com/speed/articles/web-metrics.html`, May 2010.

[22] Center for Applied Internet Data Analysis (CAIDA). University of California, San Diego Supercomputer. `http://www.caida.org`. Accessed: 2015-10-05.

[23] T. Stockhammer. Dynamic adaptive streaming over HTTP: Standards and design principles. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, pages 133–144. ACM, New York, NY, USA, 2011.

[24] Johnson and S. C. Hierarchical clustering schemes. *Psychometrika, Springer-Verlag*, 32(3):241–254, 1967. ISSN 0033-3123.

[25] J. Postel. Transmission control protocol. Request for Comments 793, Internet Engineering Task Force, September 1981.

[26] M. Allman, V. Paxson, and E. Blanton. TCP congestion control. Request for Comments 5681, Internet Engineering Task Force, September 2009.

[27] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, I. Tsang, S. Gjessing, G. Fairhurst, G. Carsten, and M. Welzl. Reducing internet latency: A survey of techniques and their merits. *To appear in IEEE Communications Surveys & Tutorials*, 2014.

[28] N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin. An argument for increasing TCPs initial congestion window. *ACM SIGCOMM Computer Communication Review*, 40 (3):27–33, July 2010.

[29] T. Flach, N. Dukkipati, A. Terzis, B. Raghavan, N. Cardwell, Y. Cheng, A. Jain, S. Hao, E. Katz-Bassett, and R. Govindan. Reducing web latency: the virtue of gentle aggression. In *Proceedings of ACM SIGCOMM*, pages 159–170. ACM, Hong Kong, China, August 2013.

[30] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream control transmission protocol (SCTP). Request for Comments 4960, Internet Engineering Task Force, September 2007.

[31] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, and P. Conard. Stream control transmission protocol (SCTP) partial reliability extension. Request for Comments 3758, Internet Engineering Task Force, May 2004.

[32] H. Wang, Y. Jin, W. Wang, J. Ma, and D. Zhang. The performance comparison of PRSCTP, TCP and UDP for MPEG-4 multimedia traffic in mobile network. In *Proceedings of International Conference on Communication Technology*, pages 403–406. IEEE, Beijng, China, April 2003.

[33] H. Sanson, A. Neira, L. Loyola, and M. Matsumoto. PR-SCTP for real time H. 264/AVC video streaming. In *Proceedings of International Conference on Advanced Communication Technology*, pages 59–63. IEEE, Gangwon-Do, Korea, February 2010.

[34] H. Alvestrand. Overview: Real time protocols for browser-based applications. Internet draft (work in progress), Internet Engineering Task Force, June 2015. draft-ietf-rtcweb-overview-14.

[35] R. Jesup, S. Loreto, and M. Tuexen. WebRTC data channels. Internet draft (work in progress), Internet Engineering Task Force, January 2015. draft-ietf-rtcweb-data-channel-13.txt.

[36] E. Boschi, L. Mark, J. Quittek, M. Stiemerling, and P. Aitken. IP flow information export (IPFIX) implementation guidelines. Request for Comments 5153, Internet Engineering Task Force, April 2008.

[37] E. Yilmaz, N. Ekiz, P. Natarajan, P. Amer, J. T. Leighton, F. Baker, and R. Stewart. Throughput analysis of non-renegable selective acknowledgments (NR-SACKs) for SCTP. *Computer Communications*, 33(16): 1982–1991, October 2010.

[38] P. Amer, M. Becke, T. Dreibholz, N. Ekiz, J. Iyengar, P. Natarajan, R. Stewart, and M. Tuexen. Load sharing for the stream control transmission protocol (SCTP). Internet draft (work in progress), Internet Engineering Task Force, May 2015. draft-tuexen-tsvwg-sctp-multipath-10.

[39] R. Gerhards. The syslog protocol. Request for Comments 5424, Internet Engineering Task Force, March 2009.

[40] M. Belshe. More bandwidth doesn't matter (much). `https://docs.google.com/a/chromium.org/viewer?a=v&pid=sites&srcid=Y2hyb21pdW0ub3JnfGRldnxneDoxMzcyOWI1N2I4YzI3NzE2`. Accessed: 2015-10-05.

[41] S. Sundaresan, W. D. Donato, N. Feamster, R. Teixeira, S. Crawford, and A. Pescapè. Broadband Internet performance: a view from the gateway. In *Proceedings of ACM SIGCOMM*, pages 134–145. ACM, Toronto, ON, Canada, August 2011.

[42] N. Feamster. Hidden sources of Internet latency. In *Proceedings of Workshop on Reducing Internet Latency*. Internet Society, London, UK, 2013.

[43] I. Grigorik. *High Performance Browser Networking*. O'Reilly Media, 2013. ISBN 978-1-4493-4471-9.

[44] K. Thompson, G. J. Miller, and R. Wilder. Wide-area Internet traffic patterns and characteristics. *IEEE Network*, 11(6):10–23, Nov/Dec 1997.

[45] N. Brownlee and K. C. Claffy. Internet stream size distributions. In *Proceedings of ACM SIGMETRICS*, pages 282–283. ACM, Marina Del Rey, California, USA, June 2002.

[46] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the characteristics and origins of Internet flow rates. In *Proceedings of ACM SIGCOMM*, pages 309–322. ACM, Pittsburgh, PA, USA, 2002.

[47] K. Lan and J. Heidemann. A measurement study of correlations of Internet flow characteristics. *Computer Networks*, 50(1):46–62, January 2006.

[48] F. Qian, A. Gerber, Z. M. Mao, S. Sen, O. Spatscheck, and W. Willinger. TCP revisited: a fresh look at TCP in the wild. In *Proceedings of ACM SIGCOMM Conference on Internet Measurement*, pages 76–89. ACM, Chicago, Illinois, USA, November 2009.

[49] S. Molnar and Z. Moczar. Three-dimensional characterization of Internet flows. In *Proceedings of IEEE International Conference on Communications*, pages 1–6. IEEE, Kyoto, Japan, June 2011.

[50] K. Papagiannaki, N. Taft, S. Bhattacharyya, P. Thiran, K. Salamatian, and C. Diot. A pragmatic definition of elephants in Internet backbone traffic. In *Proceedings of ACM SIGCOMM Workshop on Internet Measurement*, pages 175–176. ACM, Marseille, France, November 2002.

[51] S. Sen, O. Spatscheck, and D. Wang. Accurate, scalable in-network identification of P2P traffic using application signatures. In *Proceedings of International conference on World Wide Web*, pages 512–521. ACM, New York, NY, USA, May 2004.

[52] A. W. Moore and K. Papagiannaki. Toward the accurate identification of network applications. In *Proceedings of International Conference on Passive and Active Network Measurement*, pages 41–54. Springer-verlag, Boston, MA, March/April 2005.

[53] A. Madhukar and C. Williamson. A longitudinal study of P2P traffic classification. In *Proceedings of International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 179–188. IEEE, Monterey, CA, USA, September 2006.

[54] Internet Assigned Numbers Authority (IANA). `http://www.iana.org/assignments/port-numbers`. Accessed: 2015-10-05.

[55] A. McGregor, M. Hall, P. Lorier, and J. Brunskill. Flow clustering using machine learning techniques. In *Proceedings of International Conference on Passive and Active Network Measurement*, pages 205–214. Springer-verlag, Antibes Juan-les-Pins, France, April 2004.

[56] S. Zander, T. Nguyen, and G. Armitage. Automated traffic classification and application identification using machine learning. In *Proceedings of IEEE Conference on Local Computer Networks*, pages 250–257. IEEE, Sydney, Australia, November 2005.

[57] J. Erman, A. Mahanti, M. Arlitt, and C. Williamson. Identifying and discriminating between web and peer-to-peer traffic in the network core. In *Proceedings of International conference on World Wide Web*, pages 883–892. ACM, Banff, AB, CANADA, May 2007.

[58] C. Bacquet, K. Gumus, D. Tizer, A. N. Zincir-Heywood, and M. I. Heywood. A comparison of unsupervised learning techniques for encrypted traffic identification. *Journal of Information Assurance and Security*, 5(1): 464–472, 2010.

[59] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, G. Wei, and L. T. Yang. Internet traffic classification using constrained clustering. *IEEE Transactions on Parallel and Distributed Systems*, 25(11):2932–2943, November 2014.

[60] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. In *Proceedings of ACM SIGMETRICS*, pages 50–60. ACM, Banff, AB, Canada, June 2005.

[61] W. Li and A. W. Moore. A machine learning approach for efficient traffic classification. In *Proceedings of International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 310–317. IEEE, Istanbul, Turkey, October 2007.

[62] V. Jacobson. Congestion avoidance and control. *ACM SIGCOMM Computer Communication Review*, 18(4):314–329, 1988.

[63] S. Floyd, M. Allman, A. Jain, and P. Sarolahti. Quick-start for TCP and IP. Request for Comments 4782, Internet Engineering Task Force, January 2007. (Status: Experimental).

[64] N. Hu and P. Steenkiste. Improving TCP startup performance using active measurements: algorithm and evaluation. In *Proceedings of IEEE International Conference on Network Protocols*, pages 107–118. IEEE, Atlanta, Georgia, USA, November 2003.

[65] D. Liu, M. Allman, K. Fall, and L. Wang. Congestion control without a startup phase. In *Proceedings of International Workshop on Protocols for Future, Large-scale & Diverse Network Transports*, pages 61–66. Los-Angeles, CA, USA, February 2007.

[66] S. Ha and I. Rhee. Hybrid slow start for high-bandwidth and long-distance networks. In *Proceedings of International Workshop on Protocols for Future, Large-scale & Diverse Network Transports*. Manchester, UK, March 2008.

[67] D. Cavendish, K. Kumazoe, M. Tsuru, Y. Oie, and M. Gerla. CapStart: An adaptive TCP slow start for high speed networks. In *Proceedings of International Conference on Evolving Internet*, pages 15–20. IEEE, Cannes/La Bocca, French Riviera, France, August 2009.

[68] L. S. Brakmo and L. L. Peterson. TCP Vegas: End to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8):1465–1480, October 1995.

[69] R. Wang, G. Pau, K. Yamada, M. Sanadidi, and M. Gerla. TCP startup performance in large bandwidth networks. In *Proceedings of IEEE Conference on Computer Communications*, volume 2, pages 796–805. IEEE, Hong Kong, China, March 2004.

[70] S. Floyd. Limited slow-start for TCP with large congestion windows. Request for Comments 3742, Internet Engineering Task Force, March 2004.

[71] D. Leith, R. Shorten, G. McCullagh, J. Heffner, L. Dunn, and F. Baker. Delay-based AIMD congestion control. In *Proceedings of International Workshop on Protocols for Future, Large-scale & Diverse Network Transports*, pages 1–6. Los-Angeles, CA, USA, February 2007.

[72] H. Wang, H. Xin, D. S. Reeves, and K. G. Shin. A simple refinement of slow-start of TCP congestion control. In *Proceedings of IEEE Symposium on Computers and Communications*, pages 98–105. IEEE, Antibes-Juan Les Pins, France, March 2000.

[73] R. Sallantin, C. Baudoin, E. Chaput, F. Arnal, E. Dubois, and A. Beylot. Initial spreading: a fast start-up TCP mechanism. In *Proceedings of IEEE Conference on Local Computer Networks*, pages 492–499. IEEE, Sydney, Australia, October 2013.

[74] M. Allman, S. Floyd, and C. Partridge. Increasing TCP's initial window. Request for Comments 3390, Internet Engineering Task Force, October 2002.

[75] M. Allman, C. Hayes, and S. Ostermann. An evaluation of TCP with larger initial windows. *ACM SIGCOMM Computer Communication Review*, 28(3):41–52, July 1998.

[76] 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: 27–33, July 2010.

[77] R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker. Recursively cautious congestion control. In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*, pages 373–385. USENIX, Philadelphia, PA, USA, June 2014.

[78] V. Konda and J. Kaur. RAPID: Shrinking the congestion-control timescale. In *Proceedings of IEEE Conference on Computer Communications*, pages 1–9. IEEE, Rio de Janeiro, Brazil, April 2009.

[79] P. Sarolahti, M. Allman, and S. Floyd. Determining an appropriate sending rate over an underutilized network path. *Computer Networks*, 51(7):1815–1832, May 2007.

[80] H. Balakrishnan, H. S. Rahul, and S. Seshan. An integrated congestion management architecture for Internet hosts. In *Proceedings of ACM SIGCOMM*, pages 175–187. ACM, Cambridge, MA, USA, September 1999.

[81] R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker. How to improve your network performance by asking your provider for worse service. In *Proceedings of Workshop on Hot Topics in Networks*, pages 1–7. ACM, College Park, MD, USA, November 2013.

[82] R. H. Katz and V. N. Padmanabhan. TCP fast start: A technique for speeding up web transfers. In *Proceedings of IEEE Globecom Internet Mini-Conference*. IEEE, Sydney, Australia, November 1998.

[83] V. Paxson, M. Allman, J. Chu, and M. Sargent. Computing TCP's retransmission timer. Request for Comments 6298, Internet Engineering Task Force, June 2011.

[84] M. Allman, K. Avrachenkov, U. Ayesta, J. Blanton, and P. Hurtig. Early retransmit for TCP and stream control transmission protocol (SCTP). Request for Comments 5827, Internet Engineering Task Force, April 2010.

[85] S. Ramachandran. Web metrics: Size and number of resources. `http://code.google.com/speed/articles/web-metrics.html`, May 2010.

[86] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, M. Stemm, and R. H. Katz. TCP behavior of a busy Internet server: Analysis and improvements. In *Proceedings of IEEE Conference on Computer Communications*, volume 1, pages 252–262. IEEE, San Francisco, CA, USA, March 1998.

[87] S. Rewaskar, J. Kaur, and F. D. Smith. A passive state-machine approach for accurate analysis of TCP out-of-sequence segments. *ACM SIGCOMM Computer Communication Review*, 36(3):51–64, January 2006.

[88] N. Dukkipati, N. Cardwell, Y. Cheng, and M. Mathis. Tail loss probe (TLP): An algorithm for fast recovery of tail losses. Internet draft (work in progress), Internet Engineering Task Force, February 2013. draft-dukkipati-tcpm-tcp-loss-probe-01.

[89] M. Allman, H. Balakrishnan, and S. Floyd. Enhancing TCP's loss recovery using limited transmit. Request for Comments 3042, Internet Engineering Task Force, January 2001.

[90] M. Mellia, M. Meo, and C. Casetti. TCP smart framing: a segmentation algorithm to reduce TCP latency. *IEEE/ACM Transactions on Networking*, 13(2):316–329, April 2005.

[91] 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 IEEE Conference on Local Computer Networks*, pages 176–183. IEEE, Montreal, Canada, October 2008.

[92] M. Allman and V. Paxson. On estimating end-to-end network path properties. In *Proceedings of ACM SIGCOMM*, pages 263–274. ACM, Cambridge, MA, USA, August 1999.

[93] I. Psaras and V. Tsaoussidis. The TCP minimum RTO revisited. In *Proceedings of IFIP International Networking Conference*, pages 981–991. Springer-verlag, Atlanta, GA, USA, May 2007.

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

[95] N. Seddigh and M. Devetsikiotis. Studies of TCP's retransmission timeout mechanism. In *Proceedings of IEEE International Conference on Communications*, volume 6, pages 1834–1840. IEEE, Helsinki, Finland, June 2001.

[96] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, G. R. Ganger D. G. Andersen, and B. Mueller G. A. Gibson. Safe and effective fine-grained TCP retransmissions for datacenter communication. In *Proceedings of ACM SIGCOMM*, pages 303–314. ACM, Barcelona, Spain, August 2009.

[97] R. Ludwig and R. H. Katz. The Eifel algorithm: making TCP robust against spurious retransmissions. *ACM SIGCOMM Computer Communication Review*, 30(1):30–36, January 2000.

[98] H. Ekström and R. Ludwig. The Peak-Hopper: A new end-to-end retransmission timer for reliable unicast transport. In *Proceedings of IEEE Conference on Computer Communications*, volume 4, pages 2502–2513. IEEE, Hong Kong, China, March 2004.

[99] Z. Wen and K. L. Yeung. On RTO timer implementation in TCP. In *Proceedings of International Conference on Intelligent Systems Design And Applications*, pages 1350–1354. IEEE, Cairo, Egypt, December 2010.

[100] P. Hurtig, A. Brunstrom, A. Petlund, and M. Welzl. TCP and SCTP RTO restart. Internet draft (work in progress), Internet Engineering Task Force, June 2015. draft-ietf-tcpm-rtorestart-08.

[101] 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). Request for Comments 4138, Internet Engineering Task Force, August 2005.

[102] M. Mathis and J. Mahdavi. Forward acknowledgment: Refining TCP congestion control. In *Proceedings of ACM SIGCOMM*, pages 281–291. ACM, Stanford, CA, USA, August 1996.

[103] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, August 1999.

[104] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. Real-time transport protocol. Request for Comments 1889, Internet Engineering Task Force, January 1996.

[105] J. Li, S. Ha, and V. Bharghavan. HPF: A transport protocol for supporting heterogeneous packet flows in the Internet. In *Proceedings of IEEE Conference on Computer Communications*, volume 2, pages 543–550. IEEE, New York, NY, USA, March 1999.

[106] B. J. Dempsey, J. Liebeherr, and A. C. Weaver. On retransmission-based error control for continuous media traffic in packet-switching networks. *Computer Networks & ISDN Systems*, 28(5):719–736, March 1996.

[107] B. J. Dempsey. *Retransmission-Based Error Control for Continuous Media Traffic in Packet-Switched Networks*. PhD thesis, University of Virginia, VA, USA, May 1994.

[108] P. Conrad, E. Golden, P. Amer, and R. Marasli. A multimedia document retrieval system using partially-ordered/partially-reliable transport service. In *Proceedings of Multimedia Computing and Networking*. SPIE, San Jose, CA, USA, March 1996.

[109] P. D. Amer, C. Chassot, T. Connolly, and M. Diaz. Partial order quality of service to support multimedia connections: Reliable channels. In *Proceedings of High Performance Distributed Conference*, pages 272–280. IEEE, Spokane, WA, USA, July 1993.

[110] P. D. Amer, C. Chassot, T. Connolly, and M. Diaz. Partial order quality of service to support multimedia connections: Unreliable channels. In *Proceedings of International Networking Conference*. San Fransisco, CA, USA, August 1993.

[111] M. Tuexen, R. Stewart R. Seggelmann, and S. Loreto. Additional policies for the partial reliability extension of the stream control transmission protocol. Request for Comments 7496, Internet Engineering Task Force, April 2015.

[112] K. J. Grinnemo, J. Garcia, and A. Brunstrom. Taxonomy and survey of retransmission-based partially reliable transport protocols. *Computer Communications*, 27(15):1441–1452, September 2004.

[113] M. Molteni and M. Villari. Using SCTP with partial reliability for MPEG-4 multimedia streaming, November 2002.

[114] A. Balk, M. Sigler, M. Gerla, and M. Y. Sanadidi. Investigation of MPEG-4 video streaming over SCTP. In *Proceedings of World Multiconference on Systemics, Cybernetics, and Informatics*. IIIS, Orlando, FL, USA, July 2002.

[115] L. Wang, K. Kawanishi, and Y. Onozato. MPEG-4 optimal transmission over SCTP multi-streaming in 802.11 wireless access media. In *Proceedings of the Asia-Pacific Symposium on Information and Telecommunication Technologies*, pages 172 –177. IEEE, Bandos Island, Maldives, April 2008.

[116] M. N. El Derini and A. A. Elshikh. MPEG-4 video transfer with SCTP-friendly rate control. In *Proceedings of International Conference on Innovations in Information Technology*, pages 1–11. Dubai, UAE, September 2005.

[117] A. Argyriou and V. Madisetti. Streaming H.264/AVC video over the Internet. In *Proceedings of Consumer Communications and Networking Conference*, pages 169–174. IEEE, Las Vegas, NV, USA, January 2004.

[118] A. Argyriou. A novel end-to-end architecture for H.264 video streaming over the Internet. *Telecommunications Systems*, 28(2):133–150, February 2005.

[119] H. Huang, J. Ou, and D. Zhang. Efficient multimedia transmission in mobile network by using PR-SCTP. In *Proceedings of International Conference on Communications and Computer Networks*, pages 213–217. IEEE, Marina del Rey, CA, USA, October 2005.

[120] A. T. Connie, P. Nasiopoulos, Y. P. Fallah, and V. Leung. SCTP-based transmission of data-partitioned H.264 video. In *Proceedings of ACM workshop on Wireless multimedia networking and performance modeling*, pages 32–36. ACM, Vancouver, Canada, October 2008.

[121] C. Xu, Y. Qiao, E. Fallon, G. M. Muntean, P. Jacob, and A. Hanley. Comparative study of real-time multimedia transmission over multi-homing transport protocols. In *Proceedings of Management of Converged Multimedia Networks and Services*, pages 64–76. Springer-Verlag, Samos Island, Greece, September 2008.

[122] T. Maeda, M. Kozuka, and Y. Okabe. Reliable streaming transmission using PR-SCTP. In *Proceedings of Annual International Symposium on Applications and the Internet*, pages 278–279. IEEE, Seattle, USA, July 2009.

[123] L. Wang, K. Kawanishi, and Y. Onozato. Achieving robust fairness of SCTP extension for MPEG-4 streaming. In *Proceedings of International Symposium on Personal, Indoor and Mobile Radio Communications*, pages 2970–2974. IEEE, Tokyo, Japan, September 2009.

[124] K. H. Kim, K. M. Jeong, C. H. Kang, and S. J. Seok. A transmission control SCTP for real-time multimedia streaming. *Computer Networks*, 54(9):1418–1425, June 2010.

[125] I. E. G. Richardson. *H. 264 and MPEG-4 video compression: Video Coding for Next-generation Multimedia*. Wiley, 2003. ISBN 978-0-470-86960-4.

[126] J. Fitzpatrick, S. Murphy, M. Atiquzzaman, and J. Murphy. ECHO: A quality of service based endpoint centric handover scheme for VoIP. In *Proceedings of IEEE Wireless Communications and Networking Conference*, pages 2777–2782. IEEE, LV, USA, March 2008.

[127] J. Fitzpatrick, S. Murphy, M. Atiquzzaman, and J. Murphy. Evaluation of VoIP in a mobile environment using an SCTP based handoff mechanism. In *Proceedings of Mobile and Wireless communications summit*, pages 1–5. IEEE, Budapest, Hungary, July 2007.

[128] K. F. Chou, C. P. Young, S. Y. Chen, and L. C. Wang. The design and implementation of messenger on-the-drive. *Journal of Information Science and Engineering*, 23(3):697–708, January 2007.

[129] C. M. Huang and M. S. Lin. Partially reliable-concurrent multipath transfer (PR-CMT) for multihomed networks. In *Proceedings of IEEE Global Communications Conference*, pages 1–5. IEEE, New Orleans, LA, USA, November/December 2008.

[130] S. T. Kim, S. J. Koh, and Y. J. Kim. Performance of SCTP for IPTV applications. In *Proceedings of International Conference on Advanced Communication Technology*, pages 2176–2180. IEEE, Phoenix Park, Korea, February 2007.

[131] X. L. Wang and V. C. M. Leung. Applying PR-SCTP to transport SIP traffic. In *Proceedings of IEEE Global Communications Conference*, volume 2, pages 776–780. IEEE, St. Louis, MO, USA, December 2005.

[132] A. Deveriya. An overview of the syslog protocol. `http://www.informit.com/articles/article.aspx?p=426638`, December 2005. Accessed: 2015-10-05.

[133] C. Lonvick. The BSD syslog protocol. Request for Comments 3164, Internet Engineering Task Force, August 2001.

[134] D. New and M. Rose. Reliable delivery for syslog. Request for Comments 3195, Internet Engineering Task Force, November 2001.

[135] H. Tsunoda, T. Maruyama, K. Ohta, Y. Waizumi, G. M. Keeni, and Y. Nemoto. A prioritized retransmission mechanism for reliable and efficient delivery of syslog messages. In *Proceedings of Annual Communication Networks and Services Research Conference*, pages 158–165. IEEE, Moncton, New Brunswick, Canada, May 2009.

[136] J. Nagle. Congestion control in IP/TCP internetworks. Request for Comments 896, Internet Engineering Task Force, January 1984.

[137] T. Dreibholz, E. P. Rathgeb, I. Rungeler, R. Seggelmann, M. Tuxen, and R. Stewart. Stream control transmission protocol: Past, current, and future standardization activities. *IEEE Communications Magazine*, 49(4): 82–88, April 2011.

[138] R. Stewart, M. Tuexen, K. Poon, P. Lei, and V. Yasevich. Sockets API extensions for the stream control transmission protocol (SCTP). Request for Comments 6458, Internet Engineering Task Force, December 2011.

[139] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommendations on queue management and congestion avoidance in the Internet.

Request for Comments 2309, Internet Engineering Task Force, April 1998.

[140] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[141] K. Nichols and V. Jacobson. Controlling queue delay. *Queue*, 10(5): 20–34, May 2012.

[142] R. Pan, P. Natarajan, F. Baker, B. VerSteeg, M. Prabhu, C. Piglione, and G. White V. Subramanian. PIE: A lightweight control scheme to address the bufferbloat problem. Internet draft (work in progress), Internet Engineering Task Force, March 2015. draft-ietf-aqm-pie-01.

[143] T. Hoeiland-Joergensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet. FlowQueue-Codel. Internet draft (work in progress), Internet Engineering Task Force, July 2015. draft-ietf-aqm-fq-codel-01.

[144] Cisco WRED Guide. `http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/config_wred.pdf`. Accessed: 2015-10-05.

[145] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing residential broadband networks. In *Proceedings of ACM SIGCOMM Conference on Internet Measurement*, pages 43–56. ACM, New York, NY, USA, October 2007.

[146] B. Briscoe and J. Manner. Byte and packet congestion notification. Request for Comments 7141, Internet Engineering Task Force, February 2014.

[147] N. Ekiz and P. D. Amer. Transport layer reneging. *Computer Communications*, 52(1):82–88, October 2014.

[148] G. Dodig-Crnkovic. Scientific methods in computer science. In *Proceedings of Conference for the Promotion of Research in IT at New Universities and at University Colleges in Sweden*, pages 126–130. Skövde, Sweden, April 2002.

[149] D. J. Lilja. *Measuring computer performance: a practitioner's guide*. Cambridge University Press, UK, 2000.

[150] M. Carbone and L. Rizzo. Dummynet revisited. *ACM SIGCOMM Computer Communication Review*, 40(2):12–20, April 2010.

[151] M. Rajiullah and A. Brunstrom. Optimizing PR-SCTP performance using NR-SACKs. In *Proceedings of of the 2nd Baltic Congress on Future Internet Communications*, pages 222–229. IEEE, Vilnius, Lithuania, April 2012.

Towards a Low Latency Internet: Understanding and Solutions

Interactive applications such as web browsing, audio/video conferencing, multi-player online gaming and financial trading applications do not benefit (much) from more bandwidth. Instead, they depend on low latency. Latency is a key determinant of user experience. An increasing concern for reducing latency is therefore currently being observed among the networking research community and industry.

In this thesis, we quantify the proportion of potentially latency-sensitive traffic and its development over time. Next, we show that the flow start-up mechanism in the Internet is a major source of latency for a growing proportion of traffic, as network links get faster.

The loss recovery mechanism in the transport protocol is another major source of latency. To improve the performance of latency-sensitive applications, we propose and evaluate several modifications in TCP. We also investigate the possibility of prioritization at the transport layer to improve the loss recovery. The idea is to trade reliability for timeliness. We particularly examine the applicability of PR-SCTP with a focus on event logging. In our evaluation, the performance of PR-SCTP is largely influenced by small messages. We analyze the inefficiency in detail and propose several solutions. We particularly implement and evaluate one solution that utilizes the Non-Renegable Selective Acknowledgments (NR-SACKs) mechanism, which has been proposed for standardization in the IETF. According to the results, PR-SCTP with NR-SCAKs significantly improves the application performance in terms of low latency as compared to SCTP and TCP.

ISBN 978-91-7063-659-2

ISSN 1403-8099

DISSERTATION | Karlstad University Studies | 2015:41