UNIVERSITY OF OSLO
Department of Informatics

Reducing Latency in Internet Access Links with Mechanisms in Endpoints and within the Network

PhD Thesis

Naeem Khademi

May 28, 2015

To my parents

Abstract

Excessive and unpredictable end-to-end latency is a major problem for today's Internet performance, affecting a range of applications from real-time multimedia to web traffic. This is mainly attributed to the interaction between the TCP congestion control mechanism and the unmanaged large buffers deployed across the Internet. This dissertation investigates transport and link layer solutions to solve the Internet's latency problem on the access links. These solutions operate on the sender side, within the network or use signaling between the sender and the network based on Explicit Congestion Notification (ECN). By changing the sender's reaction to ECN, a method proposed in this dissertation reduces latency without harming link utilization. Real-life experiments and simulations show that this goal is achieved while maintaining backward compatibility and being gradually deployable on the Internet. This mechanism's fairness to legacy traffic is further improved by a novel use of ECN within the network.

Acknowledgements

First and foremost, I wish to express my utmost gratitude to my supervisors, Prof. Michael Welzl and Prof. Stein Gjessing who gave me encouragement and insight, amazing support, constructive feedback and active guidance through the entire duration of my studies. Finishing this journey would not have become possible without you.

I am also grateful to my friends and colleagues at University of Oslo, especially to Dr. David Hayes, Dr. Amirhossein Taherkordi, and Safiqul Islam who made this place an attractive place to do academic research.

I would like to especially thank Prof. Grenville Armitage at Swinburne University of Technology, Prof. Renato Lo Cigno at University of Trento and Prof. Godred Fairhurst at University of Aberdeen for their collaboration and guidance during my PhD studies. I would also like to extend my appreciation to all partners of European RITE project for their collaboration and useful discussions.

Finally I would like to thank my parents, Mohammad Khademi and Fatemeh Khademolhosseini who infused me with a love of science and creativity.

Naeem Khademi
Oslo, May 28, 2015

Contents

Part I

Overview

Chapter 1

Introduction

This dissertation studies the problem of large unmanaged buffers in today's Internet and how they affect the performance of delay-sensitive applications. It explains that since many of the end-hosts on the access links use Wi-Fi technology, it is therefore important to improve the performance of such applications for Wi-Fi networks in addition to wired networks.

It also categorizes the possible solutions as:

- Solutions based on changing the TCP sender behavior only
- Solutions based on changing the queuing policy in the middlebox – e.g. discarding packets by an Active Queue Management (AQM) mechanism
- Solutions that use explicit signaling between the middlebox and the sender and require the coordination of the two.

This chapter explains the context of the work, asks research questions and highlights the main contributions of this thesis. It also provides details on the research methods employed and the published results. This thesis consists of two parts. Part I is structured as follows: Chapter 2 presents the background on the problem of large buffers in the Internet and the existing mechanisms to mitigate that; Chapter 3 provides an overview of the solutions studied in this dissertation and Chapter 4 concludes the thesis. A complete set of published results can be found in Part II.

1.1 Context of the Work

This work focuses on finding solutions for solving the Internet's latency problem that is due to unmanaged and excessively large buffers deployed over decades, specifically on the last-hop access links where most of such induced latency is present (also known as "bufferbloat"). Network devices along the Internet paths require a certain

amount of buffering space on their egress links to allow for the rate mismatch between the ingress and egress links, mainly when there are short-term packet bursts. The buffering allows to avoid the arriving packets to be discarded if the egress link is busy for a certain amount of time. In the case of transports with congestion control mechanisms that react to packet loss such as the commonly deployed TCP protocol, the buffering can assure the egress link stays fully utilized once TCP's sending rate decreases.

The common rule-of-thumb for finding out how much buffering is needed for a single standard TCP flow with NewReno congestion control to fully utilize the link is to use the bandwidth×delay product (C.RTT), with delay being the Round-Trip Time (RTT) between the sender and receiver and bandwidth being the capacity of the bottleneck link (C). This rule-of-thumb, also known as the Bandwidth-Delay Product (BDP) rule, has been used for around two decades by network administrators and operators to set the buffer size in the network devices although in some cases, some network devices have been using buffers that are much larger than a typical BDP of an Internet path. While the BDP rule assures that a single standard TCP NewReno flow can fully utilize the link, it also has the consequence of inducing a maximum queuing delay that is equal to one RTT, increasing the total round-trip time to 2.RTT at maximum. In case of devices that use more than one BDP worth of buffering, the RTT can increase even further from a few hundred milliseconds to several seconds. This will adversely affect all other applications that share their traffic at such buffer – e.g. delay-sensitive applications like interactive multimedia, Voice-over-IP (VoIP), Skype, online gaming and even popular web traffic.

This forms a major challenge for the Internet today, where buffer sizes across the Internet paths are sometimes set with the pure goal of assuring 100% link utilization with no consideration of latency as a performance metric. This is mainly due to the fact that traditionally Internet Service Providers (ISPs) have been using "bandwidth" as a commercial sales argument without having incorporated latency guarantees in their Service-Level Agreements (SLAs). Over the last decades, the capacity of Internet paths has dramatically increased with the introduction of new link layer technologies and buffers have therefore grown even larger, and with it the latency.

1.2 Research Questions

The aforementioned problem of *bufferbloat* gives rise to the following Research Questions (RQs):

- RQ1: *How should one size and manage the network buffers on the access links appropriately to reduce the end-to-end latency?*

Buffer sizes need to be small but sufficient enough to accommodate short-term packet bursts and provide a good level of utilization at minimal cost for the end-to-end latency. The question of buffer sizing will depend on the Internet traffic's characteristics such as the number of concurrent flows, RTTs, and the link capacity. In addition, such buffers should be managed by mechanisms that inform the sender of the onset of congestion. This can provide the sender with the means to react early enough in advance, before the buffer is full. The choice of buffer management and sizing will also affect the trade-off between latency and utilization and should therefore carefully be investigated.

– RQ2: *Which component(s) in the network need to be modified?*

The solution to the bufferbloat problem may require modifications in one or more components in the network. This will affect the cost and practicality of the solution as well as its deployability. Ideally, any solution should require little or minimal modification of network devices and operate at the single point which is controlled by the user/operator. Possible solutions can be implemented at the sender (sender-based), within the network (network-based), or in both components. In this thesis, all possible scenarios have been investigated and their pros and cons have been evaluated.

– RQ3: *How deployable is the solution in the current Internet?*

To have a real impact on the Internet, any solution must take into account today's Internet traffic and design characteristics and require minimal change. A key success factor is *gradual deployment*, where the solution can be introduced in a part of the network without the rest of the Internet supporting it, yet provide an improvement. It should also be possible to tune and improve the scheme's parameters over time to provide a better performance as the Internet evolves. This thesis addresses all of the above RQs.

1.3 Main Contributions

This dissertation presents a number of solutions to the Internet's end-to-end latency problem that is due to the unmanaged and excessively large buffers deployed on the last-hop access links. It begins with the study, characterization and evaluation of end-to-end latency over 802.11 (Wi-Fi) access links since Wi-Fi networks are commonly deployed as the last-hop access link on the Internet. Wi-Fi networks also provide a challenge for end-to-end latency due to the multi-rate nature of their link layer. It takes into consideration and evaluates in-depth different link layer rate adaptation mechanisms deployed in commonly used Wi-Fi devices and identifies the mechanism that optimizes the link-layer-induced latency in 802.11 WLANs.

It also investigates and evaluates CAIA Delay-Gradient (CDG), for a novel use as a multimedia-friendly "background" transport protocol in the context of Wi-Fi access networks. CDG is a sender-based mechanism that aims to minimize the queuing delay using a delay-based TCP congestion control mechanism at the sender.

This dissertation extends its study further by investigating and evaluating the solutions that rely on the network's feedback to the sender (network-based solutions) known as Active Queue Management (AQM) mechanisms. It studies in depth the state-of-the-art as well as traditional AQM mechanisms that try to minimize the latency on the access links by actively marking or dropping packets in the buffer to notify the sender about the onset of congestion.

To more significantly reduce the end-to-end latency in the Internet, this thesis proposes "Alternative Backoff with ECN" (ABE), a novel mechanism that is a sender-side-only modification and can be deployed incrementally. ABE leverages the Explicit Congestion Notification (ECN) functionality implemented across AQM-enabled networks and updates the TCP congestion control's backoff mechanism at the sender (and hence is a solution based on the collaboration between the network and the sender). Using extensive evaluations, significant performance gains are shown without losing the delay-reduction benefits of deploying AQM mechanisms. ABE reduces the cost of reducing latency in the Internet in terms of utilization and thus enables to deploy AQM mechanisms at lower target queuing delay thresholds. Moreover, ABE offers a compelling reason to deploy and enable ECN across the Internet.

At the end, this thesis proposes and evaluates a solution that enhances ABE further in terms of backward compatibility, by achieving perfect fairness between traditional TCP senders and ABE senders for scenarios where perfect fairness is a desired goal.

1.4 Research Methods

This section presents the different research methods that were employed in this dissertation. A combination of analysis, even-driven simulation and real-life experiments have been used. Details on which methods has been used are provided in each of the published works that are presented in Section 1.5.

1.4.1 Analytic

Analysis was used in Section 2.1 of Paper V where an analytical model was developed to address the correlation between TCP congestion control's backoff mechanism, path characteristics and link utilization with regards to different values of the bottleneck buffer size. We validated our model against even-driven simulation.

Table 1.1: Network testbed platforms and their respective published work(s).

Tested Platform	Paper(s)
NDlab	II, IV
TEACUP [87]	V
Emulab [82]	II, III

1.4.2 Network Simulation

A popular and commonly-used discrete event network simulation tool, called The Network Simulator (ns-2) [62] was used in various parts of this dissertation. ns-2 provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless networks and is considered as the de facto standard networking simulation tool in the academic community. Papers I, II, V and VI use ns-2 in their evaluations.

The Linux TCP congestion control suite shipped with ns-2.35 was mostly used. The Linux TCP networking stack in ns-2.35 was updated with the code shipped from Linux 3.17.4 for the work done in Paper V. Moreover, the TCP code was updated to change TCP's backoff mechanism. Active Queue Management mechanisms such as CoDel [60] and PIE [64] were imported into ns-2.35, using the latest code (at the time of writing) provided by their publishers.

For the evaluations of our first and second papers (Paper I and Paper II), an extension to ns-2.29 was added to support the Adaptive Auto-Rate Fallback (AARF) rate adaptation mechanism in 802.11 WLANs.

1.4.3 Real-life Experiments

A variety of real-life experiments was carried out using various experimental platforms. Papers II, III, IV, V use real-life tests in their evaluations. The network testbed platforms and their employment in different parts of this dissertations are presented in Table 1.1.

NDlab testbed

NDlab is a dedicated network testbed for this research work, located at the Network and Distributed Systems Group, Department of Informatics, University of Oslo, Norway. It is a cluster of nodes stacked in a lab environment, where their close proximity provides a suitable situation for the experiments that require homogeneous channel conditions. Each node is a Dell OptiPlex GX620 machine equipped with 802.11 b/g and/or 802.11 b/g/n network cards and switched with 100 Mbps Ethernet links (all nodes are interconnected with wired and wireless connectivity). For more details about the hardware specifications and network setup, refer to the

material presented in respective sections of Papers II and IV (Section V of Paper II and Section 3.1 of Paper IV).

TEACUP testbed

The TEACUP testbed is a set of physical nodes located at the Centre for Advanced Internet Architectures (CAIA), Swinburne University of Technology, Australia. These nodes are HP Compaq dc7800 machines that can run either FreeBSD 10.1-RELEASE or openSUSE 13.2. These nodes are linked by a 64-bit software router based on openSUSE 13.2 that is run over a Supermicro X8STi.

To provide automation to conduct a large variety of real-life TCP performance tests, these nodes are controlled by a Python-based software called TEACUP (TCP Experiment Automation Controlled Using Python). The detailed presentation of the TEACUP software design is provided in [88]. The detailed CAIA network testbed specification and setup for TEACUP software is described in [87].

For the real-life experiments used in our work, refer to the experimental setup and network specifications presented in Appendix A.2 of Paper V.

Emulab

Emulab [19, 82] is a public network testbed available without charge to researchers worldwide, giving them a wide range of environments in which to develop, debug, and evaluate their systems. It is primarily run by the Flux Group at the School of Computing of the University of Utah. The name Emulab refers both to a facility and to a software system. There are also installations of the Emulab software at more than two dozen sites around the world. In our real-life experiments we used Emulab's 802.11 wireless network testbed at the University of Utah. Emulab's 802.11b/g testbed we used is a cluster of Pentium III 600 MHz Machines each with two wireless interfaces, plus a wired control network under the full control of the tester.

A more detailed explanation of Emulab's network setup in our real-life experiments is given in Section V of Paper II and Section IV.A of Paper III.

1.5 Published Works

1.5.1 Research Papers in the Thesis

Paper I:

Title: On the Uplink Performance of TCP in Multi-rate 802.11 WLANs

Authors: Naeem Khademi, Michael Welzl and Renato Lo Cigno

Venue: Proceedings of the 10th International IFIP TC 6 Networking Conference (IFIP NETWORKING), Lecture Notes in Computer Science, Volume 6641, pages 368-378, May 2011, Valencia, Spain.

My contributions: Main authorship; contribution to the idea of the paper; developed all ns-2 simulation scripts and codes and conducted all simulation tests; data analysis and producing results (e.g. graphs, figures, etc); contributed text and editorial work.

Summary: This paper investigates the cross-layer interaction between Distributed Coordination Function (DCF) coupled with Rate Adaptation (RA) mechanism at the link layer and uplink TCP in Wi-Fi networks and identifies that the poor choice of RA mechanism can significantly impede the TCP performance in uplink scenarios.

Paper II:

Title: Experimental Evaluation of TCP Performance in Multi-rate 802.11 WLANs

Authors: Naeem Khademi, Michael Welzl and Stein Gjessing

Venue: Proceedings of the 13th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (IEEE WoWMoM), page 1-9, June 2012, San Francisco, California, United States.

My contributions: Main authorship; contributed to the idea of the paper; developed all ns-2 simulation and real-life test scripts and codes and conducted all tests; data analysis and producing all results (e.g. graphs, figures, etc); contributed text and editorial work.

Summary: This paper evaluates various state-of-the-art link layer RA mechanisms in both uplink and downlink TCP scenarios and identifies the RA mechanism that can mitigate the problem of uplink TCP performance Wi-Fi networks.

Paper III:

Title: Using Delay-Gradient TCP for Multimedia-Friendly 'Background' Transport in Home Networks

Authors: Grenville Armitage and Naeem Khademi

Venue: Proceedings of the 38th IEEE Conference on Local Computer Networks (LCN), pages 509-515, October 2013, Sydney, Australia.

My contributions: Co-authorship; developed all real-life test scripts/codes (except the already-available CDG v0.1 code) and conducted all real-life tests; data analysis and producing all results (e.g. graphs, figures, etc); contributed text and editorial work.

Summary: This paper explores the novel use of a delay-gradient TCP for background TCP connections in home networks. It shows that this mechanism induces low latencies regardless of the bottleneck's internal buffer size (useful when coexisting with latency-sensitive traffic) while achieving a significant fraction of spare link capacity. It also shows that the delay-gradient mechanism does not gratuitously steal capacity from commonly deployed foreground TCPs.

Paper IV:

Title: The New AQM Kids on the Block: An Experimental Evaluation of CoDel and PIE

Authors: Naeem Khademi, David Ros and Michael Welzl

Venue: Proceedings of the 17th IEEE Global Internet Symposium, Computer Communications Workshops (IEEE INFOCOM Workshops), pages 85-90, April-May 2014, Toronto, Canada

My contributions: main authorship; contributed to the idea of the paper; developed all real-life test scripts and codes and conducted all experiments; data analysis and producing all results (e.g. graphs, figures, etc); contributed text and editorial work.

Summary: The extended version of this paper evaluates two state-of-the-art AQM mechanisms (CoDel and PIE) that are proposed to tackle the bufferbloat problem, in a variety of network traffic scenarios and under different parameter settings and compares them with a traditional AQM mechanism (Adaptive RED). It surprisingly finds that in many instances results were much more favorable towards Adaptive RED. While it does not call into question the need for new AQM mechanisms, it concludes that there are lessons yet to be learned from old designs.

Note: An extended technical report of this paper is included in this dissertation.

Paper V:

Title: Alternative Backoff: Achieving Low Latency and High Throughput with ECN and AQM

Authors: Naeem Khademi, Michael Welzl, Grenville Armitage, Chamil Kulatunga, David Ros, Gorry Fairhurst, Stein Gjessing and Sebastian Zander

Venue: TBD

My contributions: Main authorship; steering and coordination among all authors and maintaining the paper's coherence; contributed to the idea of the paper; developed simulation scripts and code and conducted simulation tests, all related to single flow bulk transfer; provided analytical model in problem and background section; data analysis of TEACUP testbed results related to the effect of slow-start overshoot; review, proofread and editorial work throughout the entire paper; contributed text related to the evaluation of single flow bulk transfer.

Summary: This paper proposes "Alternative Backoff with ECN" (ABE), which consists of enabling Explicit Congestion Notification (ECN) and letting individual TCP senders use a larger multiplicative decrease factor in reaction to ECN-marks from AQM-enabled bottlenecks. Using a mix of experiments, theory and simulations, significant performance gains in lightly-multiplexed scenarios have been shown, without losing the delay-reduction benefits of deploying CoDel or PIE.

Note: This paper is under anonymous peer-review at the time of writing.

Paper VI:

Title: Improving the Fairness of Alternative Backoff with ECN (ABE)

Authors: Naeem Khademi, Michael Welzl and Stein Gjessing

Venue: TBD

My contributions: Main authorship; contributed to the idea of the paper; developed all ns-2 simulation scripts and codes and conducted all simulation tests; data analysis and producing all results (e.g. graphs, figures, etc); contributed text and editorial work.

Summary: This paper proposes a modification to the AQM mechanism to provide fairness between ABE and standard TCP flows for scenarios where fairness may be needed. The proposed method has been evaluated using RED and is shown to provide complete fairness among TCP flows of different types under a variety of scenarios.

Note: This paper will be submitted for peer-review after the publication of Paper V since it relies on the mechanism proposed in Paper V.

1.5.2 Related IETF draft

- **Internet Draft I:**
- **Title:** AQM Characterization Guidelines (draft-ietf-aqm-eval-guidelines)
- **Authors:** Nicolas Kuhn, Preethi Natarajan, David Ros and Naeem Khademi
- **Status:** Active Internet Draft, Active Queue Management Working Group (AQM WG), Internet Engineering Task Force (IETF)
- **My contributions:** Editorship; contributed text and editorial work as well as the IETF meeting presentations and discussions.
- **Summary:** This draft documents various criteria for performing characterizations of AQM proposals. This document also helps in ascertaining whether any given AQM proposal should be taken up for standardization by the IETF. Several of the test scenarios described in this draft were inspired by the works in this thesis.

Chapter 2

Background

2.1 Bufferbloat (Unmanaged Buffers)

Using unnecessarily large buffers on the access networks over the last decades has led to the high latency on the Internet, most specifically with regards to latency-sensitive applications. This is referred to as "bufferbloat" [13]. The bufferbloat problem most profoundly concerns delay-sensitive applications such as interactive multimedia, VoIP, online gaming and real-time video-streaming and can highly affect the user-perceived performance even in the case of commonly used web traffic. This is because commonly-deployed standard loss-based TCP congestion control mechanism as well as the more aggressive CUBIC congestion control, the default in Linux kernel, tend to fill up any buffer – and this is problematic for flows belonging to delay-sensitive application that share the bottleneck queue with these TCP flows. The choice of buffer size on the access link has been driven by two factors: (1) the wrong traditional perception of commercial device vendors as well as the users which would translate more memory to better performance over the last years; (2) The common rule-of-thumb (attributed to [78]) of using a bandwidth-delay product of buffering, which is the amount of buffering needed to achieve 100% link utilization for a single TCP NewReno flow on the path in the congestion avoidance phase. Assumption one is completely wrong and assumption two aims for a maximal link utilization in a scenario when only a single TCP NewReno flow exists on the bottleneck and is unnecessarily utilization-oriented, since achieving full utilization will come at the cost of inflating the RTT to twice the path's intrinsic RTT.

Additionally, with applications that produce bursty traffic, the excessive size of buffers translates to large delay spikes and jitter and can harm the other types of traffic that share the bottleneck with such flows.

The work in [28] elaborates on the prevalence of bufferbloat. For instance the 2007 study of [17] conducted on 1894 broadband hosts connected through 11 major commercial cable and DSL in North America and Europe shows that DSL upstream

queues were frequently more than 600 ms, and those for cable generally exceeded one second. Another measurement study conducted in 2010 also revealed widespread and severe bufferbloat in the broadband edge networks [44]. However bufferbloat is not only limited to broadband edge networks and can in fact be more severe in Wi-Fi or cellular networks. This is because Wi-Fi or cellular networks are typically equipped with large buffers to compensate for link layer errors and are therefore more prone to bufferbloat. The measurements of [36, 37] identify the presence of bufferbloat in cellular networks and Section 2.2.2 elaborates on the bufferbloat in Wi-Fi networks.

2.2 Buffer Sizing

There has been several works that attempt to reduce the buffer sizes along Internet paths both in the network's core as well as in the access links. In this section we provide a range of existing solutions and guidelines on sizing the buffers on network devices.

2.2.1 Network's Core

In 2004, Appenzeller et al. argued that the rule-of-thumb (the bandwidth-delay product rule) is incorrect for core routers [6]. A large number of TCP flows are multiplexed together on a backbone link. Using a set of real-life experiments, simulations and analysis, it was demonstrated that a link with n flows, given that n is large enough, requires no more than $B = (\overline{RTT} \times C)/\sqrt{n}$ worth of buffering to maintain a link utilization level close to 100%, for long-lived or short-lived TCP flows where $\overline{RTT}$ is the average round-trip time of a flow passing across a link, and C is the data-rate of the link. This will consequently lead to a decrease in latency induced by core routers and a significant reduction in cost of router-design due to less Random-Access Memory (RAM) requirements. As an example, a 10 Gbps link in the network's core carrying 50,000 flows requires only 10 Mb of buffering which can be easily accommodated by a low-price on-chip Static RAM (SRAM). In many other network scenarios, this can account for more than 99% reduction in buffer size [6].

The work in [6] shows that when the number of long-lived flows (n) is large, congestion window ($cwnd$) of TCP flows become de-synchronized in losses and therefore the queue occupancy would follow a normal distribution in probability. It also shows that in the core, the buffer size will usually be determined by the number of long-lived flows (elephants) rather than short flows (mice).

In addition to the findings of [6], the authors of [85] argue that sacrificing a little utilization may significantly reduce latency and jitter and identify that the goal of 100% link utilization is not always worthwhile. The authors of [85] also mention

that small buffers can be problematic with regards to packet bursts with bursty traffic types and also during the slow-start phase which leads to the packet drops. They also regard this problem as a rather "philosophical" issue to either consider this as unfairness or not, or whether to alternatively use "rate-pacing". However, [4] reports synchronization problems with rate-pacing due to the delay in feedback caused by pacing.

However, [85] argues that setting the buffers small would reduce the feedback delay and therefore would diminish the possible synchronization by rate-pacing. These assertions of [85] have been investigated by a follow-up work in [69] which, using control-theoretic analysis of synchronization, indicates that small buffers actually induce de-synchronization.

Eventually, the work in [20] shows why very small buffers are sufficient, as long as one is willing to sacrifice a small amount of utilization. The analysis of [20] shows that if the packets of a TCP flow are sufficiently spaced out, then 10 to 20 packet buffers are enough, independent of the link speed. Packets are spaced naturally when flows enter through access links that are much slower than the backbone links. Alternatively, packets can be spaced out using TCP Pacing.

Since the publication of the aforementioned works, there have been plenty of works with the conclusions ranging from questioning the applicability of their results to proposing alternative schemes. The reader is encouraged to refer to [79] that provides a synopsis of these proposed buffer sizing strategies and broadly classifies them according to their desired objectives and discusses the pros and cons of their approaches.

2.2.2 Last-hop Access Links (Wired and Wi-Fi)

The last-hop access links, such as commonly used ADSL2+ [26] or CableLab's DOCSIS 3.0/3.1 cable modems [18] used in conjunction with widespread 802.11 Wi-Fi [84] access points and user devices at the customer's edge of the network, have different characteristics than routers in the network's core, . One major aspect here is the varying bit-rate (or bandwidth) at the link layer due to the usage of different Modulation and Coding Schemes (MCS). The multi-rate nature of these PHY/link layer technologies mandates different buffering requirements than the ones common for the links with static data rates. In addition, on the last-hop access link the number of active long-lived TCP flows is not as large as in the core and therefore the Appenzeller rule [6] is not applicable due to the lack of multiplexing among TCP flows.

In addition, in a scenario like Wi-Fi, the link has a half-duplex nature, resulting in forward and reserve traffic sharing the same medium on the channel and therefore affecting each other's performance – e.g. upload TCP data packets and download TCP ACK packets would be transmitted from a station and download data packets

and upload ACKs will be transmitted from the Access Point (AP). This also leads to the well-known unfairness in favor of uplink traffic and deterioration of downlink traffic since in the 802.11 DCF mode (which is commonly deployed) [84], each station including the AP will get an equal probability to access the wireless channel. The fairness issues in 802.11 WLAN due to the cross-layer interaction of TCP and DCF are discussed in details in [59, 10, 68, 49, 52].

The authors of [68] identified that the buffer size in the AP plays an important role in wireless channel bandwidth allocation. This is due to the fact that, assuming a simple saturated link scenario with static link rates and where all K stations are active, the AP gets only a $1/(K+1)$ share of transmission opportunities to transfer all downlink data while all other K stations get a $K/(K+1)$ share of transmission opportunities to upload data. This means, assuming AP's downstream buffer to be the bottleneck, most of the downlink data has to be buffered in that buffer before the AP receives its share of transmission opportunities. Such a buffer will be occupied by download data packets as well as upload ACK packets which do not have the same importance with regards to TCP's self-clocking mechanism. This has been regarded as "direction-based unfairness" in the literature [41].

There is already an extensive amount of work addressing this problem and proposing different solutions on how to tackle it [49, 50, 55, 41, 51]. As briefly mentioned before, another important aspect is the multi-rate nature of the 802.11 devices – e.g. changing the bit-rate by the AP or other stations participating in the data transmission and therefore varying bandwidth and delay would affect the buffering requirements in the AP due to the shared nature of the wireless medium. Different MCSs on the last-hop Wi-Fi link translate to a different share of the end-to-end latency and bandwidth and with regards to TCP may inflate the RTT to different values from a few milliseconds to several seconds and occasionally even more [13, 28]. This has become a complex problem since the varying bandwidth available on the wireless shared medium depends on many factors such as: the AP's downlink rate, each station's uplink rate, the number of actively competing stations, the number of competing TCP flows, the network's path characteristics, the traffic pattern on the wired side of the access network, etc.

However, and perhaps because of their complexities, many of the research works seem to have not explicitly taken into account the aforementioned factors. One of the exceptions is [51, 50] that proposes a solution which emulates a BDP by setting the AP's buffer size accordingly, using the active measurement of mean service time on the queue where its change would correlate to changes in the link-layer rate. This solution still relies on the validity of the common rule-of-thumb for AP buffer sizing.

Another problem with latency on Wi-Fi networks specifically and links with varying rate in general is that there are multiple buffers at different layers – e.g. physical transmission queues used in the 802.11 transmitter chipsets, buffers in the

MAC layer and firmware as well as the buffer at the interface, priority queues in QoS schemes, etc. Any or several of these buffers may contribute to the end-to-end latency if they happen to be constantly or momentarily the bottleneck. Some of these low-level buffers may not be accessible to control or modify by the user or the network administrator and can only be modified by the chipset vendor for instance.

One solution to provide a better performance level in terms of bandwidth and latency is to employ an optimal bit-rate selection (rate-adaptation) mechanism in 802.11 devices. Papers I and II investigate this. At the end we can conclude that there can be no static buffer size that always performs well under different types of traffic characteristics and network conditions and a more sophisticated and adaptive scheme rather than or in addition to buffer sizing is needed.

2.3 Active Queue Management (AQM)

Loss-based standard TCP (NewReno) and CUBIC (which is the default in Linux) congestion controls both fill up any buffer and would backoff their sending rate (*cwnd*) upon the arrival of three dup-ACKs, which TCP takes as an indication of packet loss. This means with any given buffer size and even when the buffer is properly sized according to the previous sections' requirements and constraints, TCP would reduce its rate only after the link has become fully congested. This has several drawbacks; Firstly, the tail drop, would normally trigger multiple losses and not only one, especially with bursty packet arrivals. Secondly, it leads to constantly or frequently full buffers, increasing latency as well as jitter – e.g. up to twice as the intrinsic RTT with one BDP buffer. Loss-based TCP has been the standard and predominant congestion control mechanism in the Internet since its early days in the late 80s.

The idea of Active Queue Management (AQM) [11] is to signal the onset of congestion to the TCP sender (or any other transport protocol that implements congestion control) early enough to allow the sender to backoff before the buffer is full, thus improving latency and jitter and reducing the level of burstiness in losses which in turn leads to the de-synchronization of the flows [25]. AQM signaling can happen through dropping packets or marking them using Explicit Congestion Notification (ECN) [70].

While early AQM designs such as Random Early Detection (RED) [25], Adaptive-RED (ARED) [24], CHOKe [66], BLUE [21], etc, had the goals of de-synchronization, fairness and low latency, the main objective of AQM came into attention once again in recent years in the light of Bufferbloat [13] community efforts, with a focus on controlling the end-to-end latency on the access links. These recent works include CoDel [60], FQ_CoDel [32] and PIE [64], which are briefly discussed in this section.

2.3.1 Random Early Detection (RED)

Random Early Detection (RED) was proposed by Floyd et al. in 1993 [25] and since then has served as the baseline AQM mechanism in the research community and networking industry. It has also been widely implemented with different variants in various network devices due to the recommendation of [11], although it has not seen wide-spread deployment. One of the reasons for this lack of deployment often mentioned by the network administrators and researchers is its complex parameter tuning and lack of adaptability to different network conditions and path characteristics. Authors of [7] mention that "with an appropriate set of parameters, RED is an effective algorithm. However, dynamically predicting this set of parameters was found to be difficult".

RED keeps two thresholds: a lower and an upper one, called th_min and th_max. When the weighted average queue size ($\bar{Q}$) is below th_min, RED does not mark/-drop any packet. However, when $\bar{Q}$ grows beyond th_min, RED begins to mark/-drop arriving packets randomly with a certain probability. This probability increases as $\bar{Q}$ increases and reaches a maximum value called p_{max} when $\bar{Q}$ becomes equal to th_max (see Eq. 2.2). RED will mark/drop any packet if $\bar{Q}$ goes beyond th_max. RED uses the weighted average queue size with a weighted averaging factor (w_q), instead of the instantaneous queue size to allow short-term fluctuations and provide better system stability. Upon every packet arrival, RED calculates the average queue size as

$$\bar{Q} = \begin{cases} (1 - w_q)\bar{Q} + w_q q, & \text{if } q > 0 \\ (1 - w_q)^{f(now - idle)}\bar{Q}, & \text{if } q = 0 \end{cases} \tag{2.1}$$

where q is the current instantaneous queue size, $idle$ is start of the queue idle time, now is the current time and $f(t)$ is a linear function of time t. If $th\_min < \bar{Q} < th\_max$, it calculates the dropping/marking probability (p_a) as:

$$p_b = p_{max}(\bar{Q} - th\_min)/(th\_max - th\_min) \tag{2.2}$$

$$p_a = p_b/(1 - count.p_b) \tag{2.3}$$

where $count$ is the number of packets since the last marked packet. Eventually RED marks the arriving packet with the probability of p_a. For a more detailed description of RED refer to [25]. The RED probability configuration is depicted in Figure 2.1.

RED variants and Adaptive RED (ARED)

As mentioned before, one of the main obstacles for RED deployment has been its parameter tuning complexity. Adaptive RED [24] is a variant of RED that tries to

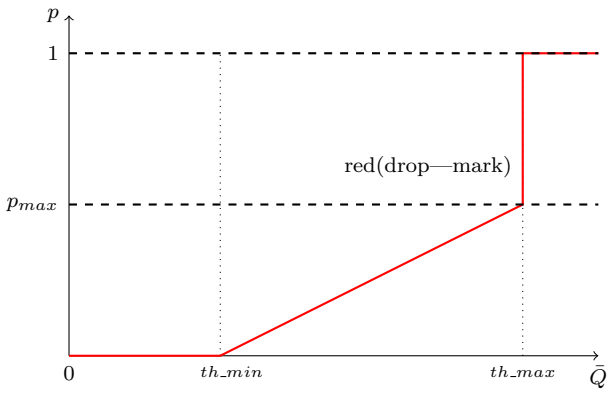

Figure 2.1: RED configuration.

tackle this problem by dynamically adjusting the RED's maximum marking/dropping probability (p_{max}) using an Additive-Increase Multiplicative Decrease (AIMD) function for the aim of a desired target $\bar{Q}$ that is equal to $(th\_min + th\_max)/2$. To the best of our knowledge, while ARED [24] was never documented in IETF or published as a peer-reviewed publication, it has been available in the Linux kernel and is being frequently cited and used by other works for comparison against different AQM mechanisms. A full description of ARED can be found in our work in [39] as well as Section 2.3 of Paper IV.

2.3.2 CoDel

Controlled Delay (CoDel) [60, 61] was proposed as an AQM solution to the Bufferbloat problem [13]. It tries to detect a standing queue by tracking, using timestamps, the minimum queuing delay (or sojourn delay) packets experience in a fixed-duration time interval, set to 100 ms by default. It assumes that a small target standing queuing delay is tolerable so as to achieve good link utilization. When the minimum queuing delay has exceeded the target delay value during at least one interval, a packet is dropped from the tail of the queue and a control law is used to set the next dropping time. Using the well-known relationship of dropping rate to TCP throughput [56], this time interval is decreased in inverse proportion to the square root of the number of drops since the dropping state was entered to ensure that TCP does not under-utilize the link.

When the queuing delay goes below the target delay, the controller stops dropping packets and exits the dropping state. In addition, no drops are carried out if the queue contains fewer than an MTU's worth of bytes. Additional logic avoids re-entering the dropping state too early after exiting it, and CoDel resumes the dropping state at a recent control level, if one exists. CoDel only enters the drop-

ping state when the minimum queuing delay has exceeded the target delay for an interval long enough to absorb normal packet bursts. This ensures that a burst of packets will not experience packet drops as long as the burst can be cleared from the queue within a reasonable period of time. A similar description of CoDel can also be found in Section 2.1 of Paper IV.

FQ_CoDel

A variant of CoDel known as "FlowQueue CoDel" (FQ_CoDel) [32] has been available since Linux kernel version 3.5 to provide better fairness and flow isolation as well as interactivity. FQ_CoDel is a hybrid combination of Deficit Round-Robin (DRR) [72] and CoDel with the aim of optimizing sparse flows (called "Flow Queuing") similar to SQF [14] and DRR++ [54] and its scheduling mechanism bears a resemblance to Stochastic Fair Queuing (SFQ) [57].

FQ_CoDel stochastically classifies the incoming packets into different sub-queues (up to maximum 1024 sub-queues by default) by hashing the 5-tuple of IP protocol number and source and destination IP addresses and port numbers, perturbed with a random number selected at initiation time. Each sub-queue is managed by an instance of the CoDel mechanism. On dequeue, it selects the sub-queue from which to dequeue using a round-robin scheme, in which each sub-queue is allowed to dequeue up to a configurable quantum of bytes (usually equal to one MTU) for each iteration. FQ_CoDel maintains two set of sub-queues: "new" and "old". The "old" sub-queues are the sub-queues that hold more than one quantum of bytes (and therefore build a standing queue) whereas the "new" sub-queues hold less than or equal to one quantum of data (and therefore with no standing queue). In one round-robin iteration, a "new" sub-queue is served and removed from the list and will be re-added each time a packet arrives for it and therefore will get priority over "old" sub-queues. In practice this distinction, would benefit applications that require low latency and interactivity, e.g. HTTP requests, SSH, DNS lookups, VoIP, etc. Flow isolation in FQ_CoDel also better protects flows from the impact of non-responsive flows such as Constant Bit-Rate (CBR) multimedia traffic. On the other hand there is a potential disadvantage of enforced fairness, e.g. when a set of flows multiplexed in a VPN tunnel are treated as a single flow by FQ_CoDel.

FQ_CoDel is implemented in and has been available since Linux kernel 3.4.0 and turned on by default on Cerowrt router firmware [1]. A detailed description of FQ_CoDel can be found in [32].

2.3.3 Proportional Integral controller Enhanced (PIE)

The Proportional Integral controller Enhanced (PIE) AQM scheme [64, 65] developed by Cisco, randomly drops a packet at the onset of congestion. Similar to CoDel, it uses queuing delay but in the form of estimated queuing delay instead of

the more commonly used queue length that is used in RED (and ARED). It uses the trend of latency over time (increasing or decreasing) to determine the congestion level. Different from CoDel that drops packets on departure (dequeue time) and requires timestamping, PIE drops packets on arrival (enqueuing time) with a random probability p and does not require timestamping, making it a more lightweight mechanism. Every t_{update} time units, PIE estimates the current queuing delay using Little's law [53] in Eq. 2.4 and derives p based on Eq. 2.5:

$$E[T] = N/\mu \tag{2.4}$$

$$p = p + \alpha * (E[T] - T_{target}) + \beta * (E[T] - E[T]_{old}) \tag{2.5}$$

where N is the current queue length, μ is the draining rate of the queue, $E[T]$ represents the current estimated queuing delay, $E[T]_{old}$ represents the previous iteration's estimation of the queuing delay, T_{target} is the target queuing delay, and α and β are the weights in probability calculation. The drop probability calculation incorporates the direction in which the delay is moving by employing a classic Proportional Integral (PI) controller design, similar to the one used in [34]. The α factor determines how the deviation of current latency from T_{target} affects the drop probability. The β factor makes additional adjustments depending on whether the latency trend is positive or negative. PIE has been available since Linux kernel 3.13.0. A full description of PIE can be found in [65].

DOCSIS 3.1 PIE

A customized version of PIE, called DOCSIS-PIE [83] is now being specified in increasingly popular DOCSIS 3.0 and 3.1 cable modems [18]. Implementation of DOCSIS-PIE is mandatory for DOCSIS 3.1 cable modems, and recommended for implementation in DOCSIS 3.0 cable modems where it is the default selection. A full description of DOCSIS-PIE is given in [83].

2.4 Explicit Congestion Notification (ECN)

Explicit Congestion Notification (ECN) [23] allows network devices (such as routers and switches) using an AQM mechanism to mark packets belonging to ECN-capable flows in the face of congestion instead of dropping them. Without ECN, a TCP sender relies on receiving three DupACKs to infer that congestion has occurred and a packet has supposedly been lost. Since these three DupACKs are caused by packets succeeding the dropped packet, they must reach the receiver in the same RTT in order to avoid having to wait a retransmission timeout (RTO) before the sender can react. ECN shortens this feedback time and also saves the packets that would otherwise be lost by dropping.

Despite being proposed in 2001 and widely implemented, ECN has not been widely used [76] partly due to the behavior of a subset of middleboxes that resulted in the failure of ECN-enabled TCP connections, rendering 8% of web servers unreachable in 2000 [63]. However, frequent measurements have over the course of several years shown this has reduced significantly over time along with a wide increase of client and server-side support for ECN [77]. ECN needs to operate in conjunction with an AQM mechanism. ECN marking is now supported by the CoDel and PIE in the Linux kernel and it is enabled by default in Cerowrt router firmware [1].

The standard mechanism of ECN has been specified in [70]. An introduction to ECN is provided in Section 7 of Paper IV and Section 1.2 of Paper V and a comprehensive set of benefits of using ECN is listed in Section 4 of Paper VI. ECN has been the subject of much interest in the research community and numerous proposals have been published suggesting to re-define the use of ECN code-points for different ways of performance improvement although none of them seem to inter-operate with the currently deployed standard ECN without requiring significant changes in the end-hosts and/or routers. A review of such proposals is given in Section 5 of Paper V.

Instantaneous Marking versus Averaging

Most AQM mechanisms such as RED maintain some form of averaging to smooth their measurements of queuing delay or length over a reasonable amount of time to provide better system stability and filter short-term bursts or noise in the measurements. While averaging has these benefits, it comes at the cost of delaying the congestion feedback to the sender and therefore the time until the sender reacts. By then the queue may have potentially become full, with increased chances of DropTail losses and more latency. Another alternative is to use "instantaneous marking" by e.g. setting the averaging factor in RED to 0 so that the congestion feedback will be sent to the sender immediately. Datacenter TCP (DCTCP) [5] leverages the use of ECN for instantaneous marking to provide an accurate feedback to the congestion control mechanism. However, DCTCP requires changes in both end-points as well as the middle-boxes and is not compatible with legacy ECN although first steps towards enabling this compatibility have already been taken [45]. The trade-off between instantaneous marking and averaging is a subject of further research and it not tackled in this thesis.

Chapter 3

Overview of the Contributions

In this chapter we elaborate on the solutions being investigated or proposed in this dissertation to tackle the latency problem in the Internet (a.k.a bufferbloat [13]). We categorize these solutions as: (1) sender-based: solutions that solely rely on the *sender* detecting the increase in network delay, either in the link layer or in the transport layer and reacting to it accordingly; (2) solutions that rely on the *network* itself trying to detect the increase in delay and signaling it back to a legacy sender, leveraging the standard behavior of the sender to reduce the delay; (3) solutions that require both *network signaling* and the *sender's collaboration* with the network.

This dissertation covers this three-fold solution space for the problem of latency in today's Internet and investigates the problems one by one. One key factor for the success of any solution is the possibility of gradual and easy deployment with partial benefits even when the entire Internet does not support a new mechanism. This requirement has guided the design of the solutions in this thesis. The upcoming sections elaborate on the aforementioned solution space and present research contributions in each of the three directions.

3.1 Solution: sender-based

A TCP sender can potentially try to predict or infer the onset of congestion in the network either at the link layer or the transport layer. The link layer delay often includes signal propagation delay, medium acquisition delay, serialization delay and possible frame retransmission delays. However, for instance in the case of Wi-Fi networks, these are often mainly dominated by the choice of Modulation and Coding Scheme (MCS) at the link layer which would determine the link speed. A sender needs to appropriately choose the right MCS (translated to a certain bit-rate) at the link layer for a given channel condition and network traffic scenario to minimize the frame transmission time and hence the overall latency experienced by end-points. We will elaborate on this later on in this section.

In the transport layer, a TCP sender can potentially try to detect the increase in delay at the onset of congestion (e.g. due to buffering) and react accordingly by reducing its rate. The commonly deployed loss-based TCP congestion control (standard NewReno [22] and Linux's CUBIC [29]) however, do not implement such mechanisms and are mostly ignorant towards the changes in delay or congestion level and would only react to packet losses. This means that on a perfect transmission medium they would only reduce the rate when the bottleneck buffer is full and a packet or more are dropped at the buffer's tail. This means, using loss-based congestion control, the buffer perpetually gets full and drains, which creates delay and jitter often in the order of several hundreds of milliseconds [73, 74] to multiple seconds [27, 51], which is harmful to coexisting delay-sensitive flows such as real-time multimedia, VoIP or web traffic.

On the other hand, an alternative class of *delay-based* congestion control algorithms exists that utilizes trends in the observed RTT (rather than packet losses) to infer the onset of congestion along an end-to-end path. This development began in 1989 with Jain's CARD (Congestion Avoidance using Round-trip Delay) [35] and several following algorithms [80, 81, 43, 75, 8, 46, 30] including the well-known TCP Vegas [12]. Such algorithms try to optimize their sending rate without forcing the filling of buffers to induce packet losses, by using measurements of delay. By not fully filling buffers, they can significantly lower the queuing delays compared to loss-based congestion control mechanisms, regardless of how much actual buffer space is available. A full review of such mechanisms is given in Section II of Paper III as well as in [40].

While trying to achieve a common goal, these delay-based mechanisms differ in their measurement method (e.g. using RTT, one-way delay, per-packet measurements, etc) as well as how they set their thresholds to infer congestion, and how they adjust their rate in response to inferred congestion (proportional, probabilistic, etc).

The problem with such *delay-threshold* mechanisms is that it is hard to find meaningful thresholds without the knowledge of the flow and network's path characteristics over time, since these mechanisms require to accurately estimate the path's intrinsic (or smallest possible) RTT (a.k.a base RTT) in order to be able to distinguish the queuing delay from propagation delay and also to be fair among different flows. In addition, a common major problem with delay-based mechanisms is unfairness when their flows share the bottleneck with loss-based TCP flows. This is because delay-based mechanisms react to the increase in queuing delay (although using different methods) by reducing their rate, while the source of delay increase is in fact a competing loss-based flow. This can lead to poor performance of delay-based mechanisms when their flows share the bottleneck with legacy Internet traffic.

CAIA Delay-Gradient (CDG) [31] is a sender-side algorithm compatible with all conventional TCP receivers. By using relative variations in RTT ("delay-gradient")

to infer congestion it eliminates the need to know a path's minimum, maximum or typical RTT levels, and eliminates reliance on pre-configured delay thresholds. CDG v0.1 has been available in the FreeBSD OS since kernel 9.2. Using a handful of mechanisms, CDG tries to gain a better share of the bandwidth compared to other delay-based mechanisms when coexisting with loss-based TCP traffic [31]. It is therefore a good and state-of-the-art candidate for a sender-based mechanism that aims to reduce the delay on the Internet. A full description of CDG is provided in Section III of Paper III.

CDG v0.1's performance in wired network scenarios has been evaluated in [31]. In our works in Paper III and [40], we evaluate CDG v0.1 in the context of Wi-Fi home networks. Using real-life experiments on an 802.11 testbed [82] with FreeBSD, we show that in homogeneous environments (no competing loss-based traffic), CDG is able to maintain the latency in the order of tens of milliseconds regardless of the bottleneck's internal buffer size and reduce the jitter while using large fraction of spare link capacity. However, Paper III shows that in heterogeneous environments, when sharing the bottleneck with loss-based NewReno or CUBIC traffic, CDG noticeably loses throughput in contrast to the CDG's performance in wired networks [31].

This is because CDG's design principles try to reduce the queuing delay (e.g. at the Wi-Fi access point whenever it happens to be the bottleneck) using the delay-gradient. However, the delay-gradient signal becomes very noisy when the source of extra induced delay is a mixture of the AP's queuing delay and medium access delay taking into account different MCSs (bit-rates) used by the rate adaptation mechanism at the link layer. Paper III leverages the good performance of CDG in homogeneous Wi-Fi environments as well as its "scavenger-like" nature (since it does not gratuitously steal capacity from commonly deployed "foreground" TCPs such as CUBIC and NewReno) in heterogeneous Wi-Fi environments to propose CDG v0.1 as multimedia-friendly "Background" transport in home networks.

In the end, we conclude that an optimal congestion control mechanism would probably need indications from the network itself about the onset of congestion to be able to distinguish the delay due to buffering from the link layer delay (e.g. medium access and transmission with different bit-rates).

3.1.1 Link Layer Considerations in 802.11 Networks

As mentioned in Section 3.1, one of the major challenges for sender-based mechanisms that try to reduce the end-to-end delay is to be able to distinguish the change in queuing delay from the delay changes related to the link layer mechanisms. This is a difficult issue in Wi-Fi networks where end-to-end delay is influenced by the use of varying MSCs (bit-rates) for frame transmission, varying channel acquisition delay in a shared Wi-Fi medium (contention delay), frame retransmission delay

due to noise and interference, etc., which all may contribute to a highly varying RTT even in short time-scales. This may not give the right indications about the source of delay for a sender-based congestion control mechanism that operates at the transport layer.

To minimize the delay at the link layer, and to optimize frame transmission at the MAC sub-layer, the choice of the "Rate Adaptation" (RA) mechanism is crucial. Several rate adaptation mechanisms have been proposed in the literature [33, 42, 67, 86, 3, 71] and a smaller set of them have been deployed in common Wi-Fi devices [38, 48, 58, 9]. The common problem facing RA mechanisms is to distinguish between the frame transmission failures due to noise and interference from the frame collisions, and only reacting to those related to the noise and channel condition. Several of the above RA mechanisms have tried to address this issue, although implementing some of them will require changes in the IEEE 802.11 standard and are therefore not suitable options for deployment in common Wi-Fi networks. We have elaborated on this in Section 2 of Paper I as well as Section III of Paper II.

The works in [15, 16] show that the choice of RA mechanism is trivial for downlink TCP with regards to the distinction between collision-related frame loss and the frame loss related to the channel quality. On the contrary, our work in Paper I demonstrates, using simulations, that *uplink* TCP performance can drastically deteriorate in the presence of a somewhat simplistic RA algorithm called AARF [48] because it cannot properly distinguish the frame errors due to collision from the frame errors due to channel quality.

Paper II extends this work to a set of RA mechanisms commonly deployed in 802.11 drivers for Atheros-based chipsets, available in the *madwifi* RA suite [2] (now superseded by `ath5k` and `ath9k` drivers). Using real-life experiments in two different testbeds (NDlab and Emulab [82, 19]), Paper II evaluates the RA mechanisms AMRR [48], SampleRate [9] and Minstrel [58] and identifies that in contrast to AMRR and SampleRate, Minstrel is able to keep the uplink performance at roughly the same level as downlink in various different scenarios, e.g. with a varying number of contending nodes, different RTT values and under different TCP congestion control variants, thus providing the optimal performance and rate selection with regards to the throughput.

Using a state-of-the-art RA mechanism such as Minstrel would therefore prevent unnecessary rate downshifts that would otherwise occur using AMRR or SampleRate, specifically in uplink scenarios and would therefore improve the end-to-end latency at link layer. We argue that in Wi-Fi networks, having the right RA mechanism at the link layer is a necessity, and a prerequisite for any mechanism in the transport layer that tries to minimize the end-to-end latency.

3.2 Solution: network-based

As explained in Section 3.1, an optimal congestion control mechanism aiming at reducing the queuing delay would probably need feedback from the network about the onset of congestion. This normally comes in the form of an Active Queue Management (AQM) mechanism marking/dropping packets in the network device that is a bottleneck, in order to notify the sender to reduce its sending rate to avoid a standing queue.

In Section 2.3 we provided a review of traditional as well as state-of-the-art AQM mechanisms that aim at reducing delay on the access links. Several AQM mechanisms have recently been proposed to address the Bufferbloat [13]. These include the two state-of-the-art AQMs, CoDel [60, 61] and PIE [64, 65] which share the goal of dynamically adapting to the network's conditions with a traditional AQM mechanism such as Adaptive RED [24].

Paper IV investigates the performance of CoDel, PIE and ARED in access link scenarios, where the bottleneck is most likely to reside, using real-life experiments conducted in NDlab testbed. It evaluates the performance of these AQM mechanisms in a variety of scenarios, including different congestion levels and RTTs in wired access network scenarios. In addition, it investigates the AQM performance issues in wireless networks and also elaborates on AQM behavior with and without Explicit Congestion Notification (ECN). The main contribution of Paper IV is its parameter sensitivity analysis of AQM mechanisms which has also been published in [39].

One of the key factors for AQM's performance as well as the success in its widespread deployment is its knob-free operation. As we noted before, RED's complex parameter tuning for varying network conditions was one of the main factors that prevented the network operators from turning it on despite its more than a decade long history of being available in network hardware. Both designers of CoDel [60] and PIE [64] claimed knob-free operation of their AQM mechanism. However in Paper IV, using a measurement-based parameter sensitivity study (also available in our work in [39]), we identified that both CoDel and PIE in fact maintain a set of parameters (with recommended default values) that do affect their performance.

The experimental results of Paper IV shows that in multiple scenarios ARED performs better than CoDel and PIE with regards to reducing queuing delay while maintaining goodput in moderately-to-highly multiplexed traffic scenarios while the latter AQM mechanisms are able to maintain a better trade-off between goodput and latency at the cost of extra latency in lightly multiplexed traffic scenarios.

In addition, Paper IV and Paper V demonstrate that the default parameter settings of CoDel and PIE (as well as ARED in Paper IV) leads to poor performance on paths with large RTTs (e.g. cross-continental paths or satellite links) where both latency and throughput might be crucial performance factors. This is because both

CoDel and PIE try to reduce the queuing delay to a value in the order of 5 ms to 20 ms using their default parameter settings on access links where the traffic might usually be lightly multiplexed. At such low dropping/marking AQM thresholds, the link may go under-utilized frequently.

In order to strike a good balance between reduction in latency and throughput, and with low marking thresholds such as the ones used in CoDel and PIE by default, a TCP sender would perhaps need to reduce its rate more conservatively in response to AQM congestion signals and only react in a standard way (by halving its rate) in the presence of packet losses induced by full buffers. However, a packet loss can be either an indication of a congestion signal from an AQM instance on the path or a full buffer and therefore it is not a good indicator for the sender to identify an AQM-induced congestion signal. There should be a way for the network device to explicitly signal the onset of congestion to the sender so that the sender can react accordingly.

3.3 Solution: signaling between the network and the sender

An ECN-mark is a clear indication of the presence of an AQM mechanism on the path. While there has been a significant rise in ECN support on the Internet over the recent years [76, 63, 77], a topic that is also covered in details in Section 2.4), the measurements of [76] only found one ECN-enabled router in the Internet despite the fact that ECN is almost 15 years old. A hypothetical observation of an ECN-enabled network device along an Internet path in the near future would most likely to be accompanied with the presence of a state-of-the-art AQM mechanism such as CoDel (or FQ_CoDel and its variants) or PIE on the same device which aims to maintain the queuing delay at the thresholds of 5 ms to 20 ms by default with short-term packet burst tolerance. Thus, receiving an ECN-mark can be taken as an indication of low marking thresholds for the AQM-enabled device and therefore the TCP senders can afford to reduce their sending rate by less (e.g. with a larger multiplicative decrease factor) in response to an ECN-mark whereas this is not the true in case of packet loss (three DupACKs).

As we explained in Section 3.2 using a larger TCP multiplicative decrease factor would provide a better latency versus throughput trade-off, most specifically on large RTT paths (e.g. when a path's intrinsic RTT is above 60 ms to 80 ms) which are not uncommon with cross-continental and inter-continental traffic.

Our work in Paper V leverages this potential and proposes a very simple mechanism called "Alternative Backoff with ECN" (ABE) which consists of enabling ECN and letting individual TCP senders use a larger multiplicative decrease factor in reaction to ECN-marks from AQM-enabled bottlenecks. The idea of ABE is

straightforward and requires only a minor modification to the TCP sender (changing the TCP's multiplicative decrease factor) which can be a success factor with regards to its widespread deployment on the Internet. It can be deployed incrementally and requires no flag-day. ABE achieves this using the ECN marking specified in RFC 3168 [70] for routers/middleboxes and the TCP receiver. It also defaults to the standard behavior whenever ECN is not supported along the path.

The idea of using a larger multiplicative decrease factor in response to ECN is not new, as the authors of [47] proposed to use a larger multiplicative decrease factor in conjunction with a smaller additive increase factor for ECN in 2002. However [47] assumes the AQM on the bottleneck to be RED (since this work was performed before the introduction of CoDel and PIE) which may not necessarily be deployed with low marking thresholds by default. Our work in Paper V differs from the work in [47] since it takes into consideration the default values of the state-of-the-art AQM mechanisms (CoDel and PIE) as being shipped and also solely relies on updating the multiplicative decrease factor.

Using a mix of real-life experiments run with the TEACUP [87] testbed, analysis and ns-2 simulations with standard NewReno and CUBIC flows, Paper V shows significant throughput gains in lightly-multiplexed traffic scenarios, without losing the delay-reduction benefits of CoDel or PIE. Paper V motivates and investigates our proposed ABE mechanism in several scenarios: first it shows the inter-relationship between the TCP's back-off mechanism, bottleneck buffer size and path's characteristics provided with analysis and simulation and also demonstrates how TCP performance degrades over CoDel and PIE bottlenecks on large RTT paths using real-life experiments. Second, it evaluates ABE in various scenarios using a combination of real-life experiments and simulations e.g. including bulk TCP transfers as well as short flows. Third, in doing so it also evaluates ABE flows' convergence time as well as their fairness with legacy TCP traffic when sharing a common bottleneck. It is also worth noting that ABE uses the same multiplicative decrease factor in both slow-start and congestion avoidance. In paper V, we show that using a larger factor at the end of slow-start can significantly benefit the short flows common for web traffic.

The significant and robust performance gain of ABE in all of the mentioned scenarios demonstrated in Paper V gives a compelling reason for AQM deployment with ECN turned on, all achievable with a simple and low-cost change (in terms of implementation effort) in the TCP sender's reaction to ECN-marks.

The results of Section 4.3 of Paper V show that ABE flows can coexist with standard TCP flows on the same bottleneck without any of the flows being starved and with a relatively good level of fairness (in terms of normalized throughput ratio) between both types of flows for the paper's recommended multiplicative decrease factors. However, a more strict fairness level among the two types of flows may be needed in certain scenarios e.g. for the sake of ensuring backward compatibility

with standard Internet traffic or depending on the operator's policy.

Paper VI proposes a certain way of configuring AQM mechanisms (using a showcase dual-threshold RED mechanism), which aims to provide total fairness between ABE flows and standard loss-based TCP flows. The idea behind this mechanism is to use two different AQM thresholds: a lower one for packets belonging to ECN-capable flows and a higher one for non ECN-capable flows allowing more queuing for the standard TCP flows in order to achieve high utilization. Using a set of mechanisms embedded in the dual-threshold mechanism, detailed in Section 2.3 of Paper VI, it is being assured that none of the two flow types get starved and the bottleneck does not go underutilized, while achieving strict fairness between ABE and standard TCP flows.

Chapter 4

Conclusion

This dissertation evaluated approaches to solve the problem of excessive latency on the Internet's access links. This problem is the result of the legacy loss-based TCP congestion control mechanism's interaction with unnecessarily large unmanaged buffers in access link network equipment. It is also the result of TCP's interaction with multi-rate link layer mechanisms commonly deployed on the access links. Section 1.2 provided three Research Questions (RQs). In the upcoming subsection we address each of these RQs.

4.1 Addressing Research Questions

– RQ1: *How should one size and manage the network buffers on the access links appropriately to reduce the end-to-end latency?*

RQ1 has been addressed with regards to buffer sizing in Section 2.2 which provides an overview of the existing methods to control the maximum buffer size for different network scenarios including the network's core and last-hop access links. However, for the last-hop access links, none of the discussed works in Section 2.2.2 provides a solution that reduces the latency to less than a bandwidth-delay product worth of buffering. With regards to the question of buffer management in **RQ1**, Paper IV evaluates state-of-the-art Active Queue Management (AQM) mechanisms for various parameter settings and network scenarios and explores the challenges for such mechanisms to successfully control the delay under various network conditions. We conclude that AQM mechanism should be used, but the performance with such AQM mechanisms is highly dependent on the end-point's behavior in reaction to AQM dropping/marking.

– RQ2: *Which component(s) in the network need to be modified?*

RQ2 examines which components in the network need to be changed in order to maintain a low latency level. In Chapter 3 an overview is given: (1) solutions that require a modified sender behavior to control the delay; (2) solutions that require the network device to maintain the latency low by actively managing its buffer and relying on legacy TCP congestion control behavior; (3) solutions that require explicit signaling between the sender and network device about the onset of congestion with a modified sender behavior in response to explicit signaling.

Papers I, II, and III fall in the category of solution set (1) and require a change in the sender's congestion control mechanism (Paper III) and potentially rate adaptation mechanism in the link layer (Paper I and Paper II). Paper IV belongs to category (2) and only requires a change in the bottleneck network device to support an AQM mechanism, and Papers V and VI belong to category (3) which require changes in both sender and bottleneck network devices. We conclude that while either modifying the network or the sender provides an improvement, the best solution requires modifying both.

– RQ3: *How deployable is the solution in the current Internet?*

RQ3 examines how deployable each solution is in the current Internet. In each of the categories mentioned above, depending on the magnitude and location of changes required in each of the components, the cost and the viability of its deployment path is different.

Papers I and II conclude that in order to optimize rate adaptation in the link layer, the sender's 802.11 network device should use a mechanism such as Minstrel [58]. This is normally not in control of the user or network operator but rather the vendor or the driver developer and requires hardware compatibility and support. However, we assume that, in order to have an optimal performance in Wi-Fi networks, it is necessary for the sender to use a suitable RA mechanism.

While CAIA Delay Gradient (CDG) v0.1 congestion control discussed and evaluated in Paper III is relatively easy to deploy by the user because of its availability in the the FreeBSD kernel, it still suffers from poor performance when sharing the bottleneck with standard loss-based TCP traffic. On the other hand, an AQM mechanism such as the ones evaluated in Paper IV should be deployed in the bottleneck network device which can mostly be controlled by the Internet Service Provider (ISP) or the home user, depending on its location and access level.

Our proposed methods in Papers V and VI require a change in both the sender and the network device. Paper V argues that this is achievable *today* because recent works (mentioned in Chapter 2) show a widespread ECN deployment in the endpoints as well as an increasing trend in AQM deployment with the parameter settings assumed in Paper V, reducing the cost and complexity of our solution's deployment. The proposed method of Paper V works with the state-of-the-art AQM mechanisms (e.g. (FQ_)CoDel and PIE) that are currently being deployed in

access link network devices. By using their support for ECN based on the currently existing specification in RFC 3168 [70], the required changes are reduced to a few lines of code in the sender's kernel that update the TCP multiplicative decrease factor in response to ECN.

4.2 Future Directions

DCTCP [5] and similar mechanisms leverage ECN to provide a more precise and more immediate feedback about congestion, some of which might not use ECN as defined in RFC 3168 [70]. Most state-of-the-art AQM mechanisms use some form of smoothing of the congestion signal for the sake of system stability. Delaying the congestion signal is a drawback of such mechanisms that can cause extra delay and potentially full buffers. With a more precise feedback it becomes possible to use an instantaneous signal instead of smoothing. The introduction of such mechanisms usually requires changes in bottleneck network devices, senders and receivers. The gradual deployment of our proposed methods in Papers V and VI can provide the necessary interim deployment path for such mechanisms (Section 6 of Paper V).

References

[1] CeroWrt Project. `http://www.bufferbloat.net/projects/cerowrt`.

[2] Multiband Atheros Driver for Wireless Fidelity. `http://madwifi-project.org`.

[3] P.A.K. Acharya, A. Sharma, E.M. Belding, K.C. Almeroth, and K. Papagiannaki. Congestion-Aware Rate Adaptation in Wireless Networks: A Measurement-Driven Approach. In *IEEE SECON*, San Francisco, California, USA, June 2008.

[4] Amit Aggarwal, Stefan Savage, and Tom Anderson. Understanding the Performance of TCP Pacing. In *IEEE INFOCOM*, Tel-Aviv, Israel, March 2000.

[5] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data Center TCP (DCTCP). In *ACM SIGCOMM*, New Delhi, India, September 2010.

[6] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. Sizing Router Buffers. In *ACM SIGCOMM*, Portland, Oregon, USA, September 2004.

[7] Fred Baker and Godred Fairhurst. IETF Recommendations Regarding Active Queue Management. Internet-Draft draft-ietf-aqm-recommendation, February 2015.

[8] Sumitha Bhandarkar, A. L. Narasimha Reddy, Yueping Zhang, and Dimitri Loguinov. Emulating AQM from End Hosts. In *ACM SIGCOMM*, Kyoto, Japan, August 2007.

[9] John C. Bicket. Bit-rate Selection in Wireless Networks. Technical report, Master's thesis, MIT, 2005.

[10] M. Bottigleliengo, C. Casetti, C.-F. Chiasserini, and M. Meo. Short-term Fairness for TCP Flows in 802.11b WLANs. In *IEEE INFOCOM*, Hong Kong, China, March 2004.

References

[11] 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. RFC 2309 (Informational), April 1998.

[12] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *ACM SIGCOMM*, London, United Kingdom, September 1994.

[13] Bufferbloat Project. `http://www.bufferbloat.net`.

[14] Giovanna Carofiglio and Luca Muscariello. On the Impact of TCP and Per-flow Scheduling on Internet Performance. *IEEE/ACM Transactions on Networking*, 20(2):620–633, April 2012.

[15] Jaehyuk Choi, Kihong Park, and Chong-Kwon Kim. Cross-Layer Analysis of Rate Adaptation, DCF and TCP in Multi-Rate WLANs. In *IEEE INFOCOM*, Anchorage, Alaska, USA, May 2007.

[16] Sunwoong Choi, Kihong Park, and Chong-kwon Kim. On the Performance Characteristics of WLANs: Revisited. In *ACM SIGMETRICS*, Banff, Alberta, Canada, June 2005.

[17] Marcel Dischinger, Andreas Haeberlen, Krishna P. Gummadi, and Stefan Saroiu. Characterizing Residential Broadband Networks. In *ACM SIGCOMM IMC*, San Diego, California, USA, October 2007.

[18] DOCSIS 3.1 Physical Layer Specification. `http://www.cablelabs.com/specification/physical-layer-specification/`.

[19] Network Emulation Testbed. `http://www.emulab.net/`.

[20] Mihaela Enachescu, Yashar Ganjali, Ashish Goel, Nick McKeown, and Tim Roughgarden. Part III: Routers with Very Small Buffers. *SIGCOMM Computer Communication Review*, 35(3):83–90, July 2005.

[21] Wu-chang Feng, Kang G. Shin, Dilip D. Kandlur, and Debanjan Saha. The BLUE Active Queue Management Algorithms. *IEEE/ACM Transactions on Networking*, 10(4):513–528, August 2002.

[22] S. Floyd, T. Henderson, and A. Gurtov. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 3782 (Proposed Standard), April 2004. Obsoleted by RFC 6582.

[23] Sally Floyd. TCP and Explicit Congestion Notification. *SIGCOMM Computer Communication Review*, 24(5):8–23, October 1994.

[24] Sally Floyd, Ramakrishna Gummadi, and Scott Shenker. Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management. Technical report, ICSI Networking Group, 2001.

[25] Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[26] G.992.5 : Asymmetric Digital Subscriber Line 2 Transceivers (ADSL2)-Extended Bandwidth ADSL2 (ADSL2plus). `http://www.itu.int/rec/T-REC-G.992.5/en`.

[27] Jim Gettys. The Criminal Mastermind: Bufferbloat! `http://gettys.wordpress.com/2010/12/03/introducing-the-criminal-mastermind-bufferbloat`, Dec 2010.

[28] Jim Gettys and Kathleen Nichols. Bufferbloat: Dark Buffers in the Internet. *ACM Queue*, 9(11):40:40–40:54, November 2011.

[29] Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: A New TCP-friendly High-speed TCP Variant. *SIGOPS Operating Systems Review*, 42(5):64–74, July 2008.

[30] D.A. Hayes and G. Armitage. Improved Coexistence and Loss Tolerance for Delay Based TCP Congestion Control. In *IEEE LCN*, Denver, Colorado, USA, Oct 2010.

[31] David A. Hayes and Grenville Armitage. Revisiting TCP Congestion Control Using Delay Gradients. In *IFIP/TC6 NETWORKING*, Valencia, Spain, May 2011.

[32] Toke Hoeiland-Joergensen, Paul McKenney, Dave Taht, Jim Gettys, and Eric Dumazet. FlowQueue-Codel. Internet-Draft draft-hoeiland-joergensen-aqm-fq-codel, November 2014.

[33] Gavin Holland, Nitin Vaidya, and Paramvir Bahl. A Rate-adaptive MAC Protocol for Multi-hop Wireless Networks. In *ACM MobiCom*, Rome, Italy, July 2001.

[34] C.V. Hollot, V. Misra, D. Towsley, and W.-B. Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *IEEE INFOCOM*, Anchorage, Alaska, USA, April 2001.

[35] R. Jain. A Delay-based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks. *SIGCOMM Computer Communication Review*, 19(5):56–71, October 1989.

[36] Haiqing Jiang, Zeyu Liu, Yaogong Wang, Kyunghan Lee, and Injong Rhee. Understanding Bufferbloat in Cellular Networks. In *ACM SIGCOMM CellNet*, Helsinki, Finland, August 2012.

[37] Haiqing Jiang, Yaogong Wang, Kyunghan Lee, and Injong Rhee. Tackling Bufferbloat in 3G/4G Networks. In *ACM SIGCOMM IMC*, Boston, Massachusetts, USA, November 2012.

[38] Ad Kamerman and Leo Monteban. WaveLAN-II: A High-performance Wireless LAN for the Unlicensed Band. *Bell Labs Technical Journal*, 2(3):118–133, Summer 1997.

[39] N. Khademi, D. Ros, and M. Welzl. The New AQM Kids on the Block: An Experimental Evaluation of CoDel and PIE. In *IEEE Global Internet Symposium, INFOCOM Workshops*, Toronto, Ontario, Canada, April 2014.

[40] Naeem Khademi and Grenville Armitage. Minimising RTT Across Homogeneous 802.11 WLANs with CAIA Delay-Gradient TCP (v0.1). Technical Report 121113A, Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, November 2012.

[41] Naeem Khademi and Mohamed Othman. Size-based and Direction-based TCP Fairness Issues in IEEE 802.11 WLANs. *EURASIP Journal on Wireless Communications and Networking*, 2010:49:1–49:13, April 2010.

[42] Jongseok Kim, Seongkwan Kim, Sunghyun Choi, and D. Qiao. CARA: Collision-Aware Rate Adaptation for IEEE 802.11 WLANs. In *IEEE INFOCOM*, Barcelona, Spain, April 2006.

[43] R. King, R. Baraniuk, and R. Riedi. TCP-Africa: An Adaptive and Fair Rapid Increase Rule for Scalable TCP. In *IEEE INFOCOM*, Miami, Florida, USA, March 2005.

[44] Christian Kreibich, Nicholas Weaver, Boris Nechaev, and Vern Paxson. Netalyzr: Illuminating the Edge Network. In *ACM SIGCOMM IMC*, Melbourne, Australia, November 2010.

[45] M. Kuhlewind, D.P. Wagner, J.M.R. Espinosa, and B. Briscoe. Using Data Center TCP (DCTCP) in the Internet. In *IEEE Workshop on Telecommunication Standards: From Research to Standards, Globecom Workshops*, Austin, Texas, USA, Dec 2014.

[46] Aleksandar Kuzmanovic and Edward W. Knightly. TCP-LP: Low-priority Service via End-point Congestion Control. *IEEE/ACM Transactions on Networking*, 14(4):739–752, August 2006.

[47] Minseok Kwon and Sonia Fahmy. TCP Increase/Decrease Behavior with Explicit Congestion Notification (ECN). In *IEEE ICC*, New York, New York, USA, May 2002.

[48] Mathieu Lacage, Mohammad Hossein Manshaei, and Thierry Turletti. IEEE 802.11 Rate Adaptation: A Practical Approach. In *ACM MSWiM*, Venice, Italy, October 2004.

[49] D.J. Leith and P. Clifford. Using the 802.11e EDCF to Achieve TCP Upload Fairness over WLAN Links. In *WiOpt*, Trentino, Italy, April 2005.

[50] Tianji Li and D. Leith. Adaptive Buffer Sizing for TCP Flows in 802.11e WLANs. In *ChinaCom*, Hangzhou, China, Aug 2008.

[51] Tianji Li, Douglas Leith, and David Malone. Buffer Sizing for 802.11-based Networks. *IEEE/ACM Transactions on Networking*, 19(1):156–169, February 2011.

[52] Xiaoyang Lin, Xiaolin Chang, and J.K. Muppala. VQ-RED: An Efficient Virtual Queue Management Approach to Improve Fairness in Infrastructure WLAN. In *IEEE LCN*, Sydney, Australia, Nov 2005.

[53] John D. C. Little. A proof for the queuing formula: L = λ w. *Operations Research*, 9(3):383–387, 1961.

[54] M.H. MacGregor and W. Shi. Deficits for Bursty Latency-critical Flows: DRR++. In *IEEE ICON*, September 2000.

[55] D. Malone, P. Clifford, and D.J. Leith. On Buffer Sizing for Voice in 802.11 WLANs. *IEEE Communications Letters*, 10(10):701–703, Oct 2006.

[56] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm. *SIGCOMM Computer Communication Review*, 27(3):67–82, July 1997.

[57] P.E. McKenney. Stochastic Fairness Queueing. In *IEEE INFOCOM*, San Francisco, California, USA, Jun 1990.

[58] Minstrel Description. `http://madwifi-project.org/browser/madwifi/trunk/ath_rate/minstrel/minstrel.txt`.

References

[59] Anthony C. H. Ng, David Malone, and Douglas J. Leith. Experimental Evaluation of TCP Performance and Fairness in an 802.11e Test-bed. In *ACM Workshop on Experimental Approaches to Wireless Network Design and Analysis (E-WIND), SIGCOMM Workshop*, Philadelphia, Pennsylvania, USA, August 2005.

[60] Kathleen Nichols and Van Jacobson. Controlling Queue Delay. *ACM Queue*, 10(5):20:20–20:34, May 2012.

[61] Kathleen Nichols, Van Jacobson, Andrew McGregor, and Jana Iyengar. Controlled Delay Active Queue Management. Internet-Draft draft-ietf-aqm-codel, October 2014.

[62] The Network Simulator NS-2. `http://www.isi.edu/nsnam/ns`.

[63] Jitendra Padhye and Sally Floyd. Identifying the TCP Behavior of Web Servers. In *In ACM SIGCOMM*, Stockholm, Sweden, August 2000.

[64] Rong Pan, P. Natarajan, C. Piglione, M.S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg. PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem. In *IEEE HPSR*, Taipei, Taiwan, July 2013.

[65] Rong Pan, Preethi Natarajan, Fred Baker, Mythili Prabhu, Chiara Piglione, Vijay Subramanian, and Bill Ver Steeg. PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem. Internet-Draft draft-pan-aqm-pie, September 2014.

[66] Rong Pan, B. Prabhakar, and K. Psounis. Choke - a stateless active queue management scheme for approximating fair bandwidth allocation. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 942–951 vol.2, 2000.

[67] Q. Pang, V.C.M. Leung, and S.C. Liew. A Rate Adaptation Algorithm for IEEE 802.11 WLANs Based on MAC-layer Loss Differentiation. In *IEEE BroadNets*, Boston, Massachusetts, USA, Oct 2005.

[68] S. Pilosof, Ramachandran Ramjee, d. raz, Y. Shavitt, and Prasun Sinha. Understanding TCP Fairness over Wireless LAN. In *IEEE INFOCOM*, San Francisco, California, USA, March 2003.

[69] Gaurav Raina, Don Towsley, and Damon Wischik. Part II: Control Theory for Buffer Sizing. *SIGCOMM Computer Communication Review*, 35(3):79–82, July 2005.

[70] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), September 2001. Updated by RFCs 4301, 6040.

[71] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. OAR: An Opportunistic Auto-rate Media Access Protocol for Ad Hoc Networks. *Wireless Networks*, 11(1-2):39–53, January 2005.

[72] M. Shreedhar and G. Varghese. Efficient Fair Queuing Using Deficit Round-Robin. *IEEE/ACM Transactions on Networking*, 4(3):375–385, Jun 1996.

[73] Lawrence Stewart, Grenville Armitage, and Alana Huebner. Collateral Damage: The Impact of Optimised TCP Variants on Real-time Traffic Latency in Consumer Broadband Environments. In *IFIP/TC6 NETWORKING*, Aachen, Germany, May 2009.

[74] Lawrence Stewart, David A. Hayes, Grenville Armitage, Michael Welzl, and Andreas Petlund. Multimedia-unfriendly TCP Congestion Control and Home Gateway Queue Management. In *ACM MMSys*, San Jose, California, USA, February 2011.

[75] Kun Tan, Jingmin Song, Qian Zhang, and M. Sridharan. A Compound TCP Approach for High-speed and Long Distance Networks. In *IEEE INFOCOM*, Barcelona, Spain, April 2006.

[76] Brian Trammell, Mirja Kühlewind, Damiano Boppart, Iain Learmonth, Gorry Fairhurst, and Richard Scheffenegger. Enabling Internet-wide Deployment of Explicit Congestion Notification. In *Passive and Active Measurement Conference (PAM)*, New York, New York, USA, March 2015.

[77] N. Vasic, S. Kuntimaddi, and D. Kostic. One Bit is Enough: A Framework for Deploying Explicit Feedback Congestion Control Protocols. In *COMSNETS*, Bangalore, India, Jan 2009.

[78] Curtis Villamizar and Cheng Song. High Performance TCP in ANSNET. *SIGCOMM Computer Communication Review*, 24(5):45–60, October 1994.

[79] Arun Vishwanath, Vijay Sivaraman, and Marina Thottan. Perspectives on Router Buffer Sizing: Recent Results and Open Problems. *SIGCOMM Computer Communication Review*, 39(2):34–39, March 2009.

[80] Zheng Wang and Jon Crowcroft. Eliminating Periodic Packet Losses in the 4.3-Tahoe BSD TCP Congestion Control Algorithm. *SIGCOMM Computer Communication Review*, 22(2):9–16, April 1992.

[81] David X. Wei, Cheng Jin, Steven H. Low, and Sanjay Hegde. FAST TCP: Motivation, Architecture, Algorithms, Performance. *IEEE/ACM Transactions on Networking*, 14(6):1246–1259, December 2006.

[82] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. *SIGOPS Operating Systems Review*, 36(SI):255–270, December 2002.

[83] Greg White and Rong Pan. A PIE-based AQM for DOCSIS Cable Modems. Internet-Draft draft-white-aqm-docsis-pie, January 2015.

[84] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. `http://standards.ieee.org/about/get/802/802.11.html`.

[85] Damon Wischik and Nick McKeown. Part I: Buffer Sizes for Core Routers. *SIGCOMM Computer Communication Review*, 35(3):75–78, July 2005.

[86] Starsky H. Y. Wong, Hao Yang, Songwu Lu, and Vaduvur Bharghavan. Robust Rate Adaptation for 802.11 Wireless Networks. In *ACM MobiCom*, Los Angeles, CA, USA, September 2006.

[87] Sebastian Zander and Grenville Armitage. CAIA Testbed for TEACUP Experiments Version 2. Technical Report 150210C, Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, February 2015.

[88] Sebastian Zander and Grenville Armitage. TEACUP v0.8 - A System for Automated TCP Testbed Experiments. Technical Report 150210A, Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, February 2015.

Part II

Discussion of Results