January 2009

Networking Mechanisms for Delay-Sensitive Applications

Maxim Podlesny
Washington University in St. Louis

WASHINGTON UNIVERSITY IN ST. LOUIS

Department of Computer Science and Engineering

Dissertation Examination Committee:
Sergey Gorinsky, Co-Chair
Roger Chamberlain Co-Chair
Ron Cytron
Victor Gruev
Aleksandar Kuzmanovic
Robert Morley
Robert Pless

NETWORKING MECHANISMS FOR DELAY-SENSITIVE APPLICATIONS

by

Maxim Podlesny

A dissertation presented to the
Graduate School of Arts and Sciences
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

August 2009
Saint Louis, Missouri

ABSTRACT OF THE THESIS

Networking Mechanisms for Delay-Sensitive Applications

by

Maxim Podlesny

Doctor of Philosophy in Computer Science

Washington University in St. Louis, 2009

Research Advisor: Professor Sergey Gorinsky

The diversity of applications served by the explosively growing Internet is increasing. In particular, applications that are sensitive to end-to-end packet delays become more common and include telephony, video conferencing, and networked games. While the single best-effort service of the current Internet favors throughput-greedy traffic by equipping congested links with large buffers, long queuing at the congested links hurts the delay-sensitive applications. Furthermore, while numerous alternative architectures have been proposed to offer diverse network services, the innovative alternatives failed to gain widespread end-to-end deployment.

This dissertation explores different networking mechanisms for supporting low queueing delay required by delay-sensitive applications. In particular, it considers two different approaches. The first one assumes employing congestion control protocols for the traffic generated by the considered class of applications. The second approach relies on the router operation only and does not require support from end hosts.

Acknowledgments

The work on this thesis has been an exciting, challenging, interesting, and invaluable unique experience in my life. The thesis has been possible due to many people. First, I would like to express my great sense of gratitude to my research advisor Dr. Sergey Gorinsky for his advising and financial support. He has taught me how to find interesting research problems, how to come up with a solution, and how to deliver my ideas to other people in the research community. Working with my research advisor has been an invaluable experimence for my research career.

I am grateful to my thesis committee, Dr. Roger Chamberlain, Dr. Ron Cytron, Dr. Viktor Gruev, Dr. Aleksandar Kuzmanovic, Dr. Robert Morley, and Dr. Robert Pless. Having useful discussions with them and getting helpful comments from them allowed me to significantly improve and polish my thesis work.

I am grateful to many graduate students and faculty within my department for their feedback and discussions about my research, and for their various help. In particular, special thanks to Phillip Jones.

I am grateful to the Chair of the Department Dr. Gruia-Catalin Roman for helpful discussions and support. I am also grateful to our departmental Staff for everyday help during my stay in the Department.

Finally, I am grateful to my parents, my sister, and all my friends for their support and patience while I have been working on this thesis.

Maxim Podlesny

Washington University in Saint Louis
August 2009

Dedicated to my family.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Formulation

The growth of the Internet has led to a significant increase in networking applications and, which is much more important, to the divergence of their requirements for the network. For example, if a user wants to download from a Microsoft server and install an update for Windows, he will be interested in performing that operation as fast as possible, which means that to meet the needs of that user the download rate should be high. Another user, for example, wants to play a Counter Strike game with his friends living in another country. To deliver a good performance, network latency experienced by the players should be low. Below we provide the main metrics used for characterizing requirements of Internet applications:

- **Latency.** The time between sending of data by one end-host and receiving of data by the other end-host.
- **Data loss.** The part of sent data that have not been received by the end-host correctly.
- **Available capacity.** The capacity of a link that may be used for sending data of an application.
- **Jitter.** The variability of latency for different parts of transmitted data.

In Table 1.1 we show the requirements of the main types of applications for the network. As it is observed, Internet applications may have absolutely different, even opposite needs in terms of required capacity, delay, loss, and jitter. The problem is that the Internet still offers only a single best-effort service that cannot handle sufficiently well the whole set of existing network applications.

Application type	Delay sensitivity	Loss sensitivity	Available capacity sensitivity	Jitter sensitivity
File transfer	Low	Low	Low	None
Web traffic	Moderate	Low	Moderate	None
Transaction-based	Moderate	None	Low	None
Voice (VoIP)	High	Moderate	Low	High
Real-time Video	High	Moderate	High	High

Table 1.1: Requirements of Internet applications

In this dissertation, the efforts are concentrated on delay-sensitive applications. Latency, or one-way delay, is one of the crucial requirements for this type of Internet applications. One-way delay consists of two parts: propagation delay characterizing the transmission time of a signal through network links, and queuing delay describing waiting time of data packets in link buffers. Table 1.2 illustrates the requirements of delay-sensitive applications concerning one-way delay [49]. Current manufacturers of routers configure link buffers so that the maximum queuing delay is at least 250 ms [103]. However, if round-trip time (RTT) exceeds few hundred milliseconds, human perception of an interactive multimedia application might degrade dramatically.

1.2 Used Approach

1.2.1 Congestion Control Protocol

Transmission Control Protocol (TCP) [52] is the most prominent representative of a popular congestion control paradigm where a flow increases its transmission until the bottleneck

Application type	Maximum one-way delay
IPTV	100 ms
Video-on-Demand	50 ms
VoIP	150 ms
Video Conferencing	150 ms
Gaming	50 ms

Table 1.2: Delay requirements of delay-sensitive applications

link buffer saturates, causing the router to discard a packet. Since conventional routers employ First-In First-Out (FIFO) discipline for their link scheduling, such TCP-like probing for the available network capacity builds up long queues at shared bottleneck link buffers and hampers performance of some applications. For example, human perception of an interactive multimedia application might degrade dramatically after round-trip time (RTT) exceeds few hundred milliseconds.

The first part of the dissertation investigates how to provide low queuing delay through congestion control protocols. Achieving low queues, fair sharing, no losses, full bottleneck link utilization is a challenging problem, and a common solution is to operate in multiple modes so that each mode would allow to meet a particular requirement. MCP (Multimodal Control Protocol) follows this paradigm, and, in particular, to ensure convergence to fairness, MCP uses an innovative mechanism that allows a flow just joining the network to get fair-share transmission rate, which is the first contribution of the work. The main idea of the mechanism is the following: a new flow urges all existing flows sharing its bottleneck links to switch to a fairing mode dedicated to fairness improvement. As soon as flows have converged to fair sharing, flows go to the stable mode, which is the other contribution of the work. The idea of the stable mode is to transmit data at the constant rate, which allows to significantly decrease a queue size in the buffer of a bottleneck link. To make the stable fair rates independent of round-trip times and packet sizes, MCP employs rate-based control and uniform timing of adjustments.

MCP is targeted to improve performance in the environments in which the inter-arrival time for flows on a bottleneck link is less than the transmission time. While in the core it is problematic, prior research [76] indicates that congestion occurs during the ”first mile” or ”last mile” at the edge link of the network, with low degrees of multiplexing.

1.2.2 Service Differentiation

The mismatch between the single best-effort service of the current Internet and diverse communication needs of different distributed applications has led to numerous proposals of alternative architectures with diversified network services. Although those architectures [10, 12] propose a high variety of services, they have not been widely deployed. The experience with such proposals reveal that technical merits of an innovative architecture are neither the only nor the most important factor in determining its success. Economic and legacy issues become a crucial consideration because the Internet of today is a loose

confederation of infrastructures owned by numerous commercial entities, governments, and private individuals [19]. The multiplicity of the independent stakeholders and their economic interests implies that partial deployment of a new service is an unavoidable and potentially long-term condition. Hence, a successful architecture should provide ISPs (Internet Service Providers) and users with incentives to adopt the new service despite the partial deployment. The other important issue is that the proposed architectures require special mechanisms for billing customers and managing the services proposed by architectures, the design of which seems a challenging problem.

The second part of the dissertation explores the problem of supporting low queuing delay employing service differentiation. In particular, it puts attention on investigating a novel paradigm for network service differentiation that makes deployability the primary design concern. It explicitly postulates that partial deployment is unavoidable and that the new design should be attractive for early adopters even if other ISPs or users refuse to espouse the innovation. Besides, it is demanded that the benefits of the service diversification should not come at the expense of legacy traffic. The imposed constraints are potent. In particular, they imply that the new architecture may not assume even for the Internet edges that most ISPs will support admission control, traffic shaping, metering, billing, or any other mechanism added by the architecture.

The above design principles lead to the key idea of making performance itself an incentive for network service differentiation. While prior studies have established a fundamental trade-off between link utilization and queuing delay [44, 90], the Internet practice favors full utilization of bottleneck links at the price of high queuing delay. Unfortunately, delay-sensitive applications suffer dearly from the long queues created by throughput-greedy applications at shared bottleneck links. The RD (Rate-Delay) Services proposed in this dissertation resolve this tension by offering two classes of service: an R (Rate) service puts an emphasis on a high transmission rate, and a D (Delay) service supports low queuing delay. Each of the services is neither better nor worse per se but is simply different, and its relative utility for a user is determined by whether the user's application favors a high rate or low delay. Hence, the RD architecture provides the user with an incentive and complete freedom to select the service class that is most appropriate for the application. The user chooses the R or D service by marking a bit in the headers of transmitted packets.

The interest of users in the D service is considered as an indirect but powerful incentive for ISPs to adopt the RD services. By switching to the RD architecture, an ISP attracts additional customers and thereby increases revenue. In addition, the dissertation envisions

an RD certification program championed by early adopters. The RD certification will serve as a catalyst for virulent deployment of the RD architecture because being RD-certified will give an ISP a differentiation advantage over legacy ISPs when competing with them for users and provider peering agreements.

1.2.3 Buffer Management

While web applications and other sources of network traffic tend to generate short-lived flows, 85% of TCP flows are short-lived. The prevalence of short-lived flows makes their performance an important consideration, and FCT (Flow Completion Time) has been viewed as the most relevant metric for this type of traffic [24].

On the other hand, the existing Internet architecture does not offer effective support for keeping FCT of short-lived flows minimal. In particular, TCP congestion control is designed primarily to accommodate the needs of long-lived flows, which carry 80% of all TCP traffic. After ramping up transmission exponentially during the TCP slow-start mode, a long-lived flow is envisioned to utilize the bottleneck link capacity efficiently and fairly by operating mostly in the TCP congestion-avoidance mode. It is common for short-lived flows to terminate during the initial restrictive slow-start phase or suffer an even further increase of their FCT due to losses fomented by other traffic, such as long-lived flows in the congestion-avoidance mode.

The third part of the dissertation explores a new approach to reducing FCT of short-lived flows. The approach relies on the observation that while TCP serves 95% of Internet traffic, the longevity of a flow under this predominant transport protocol has a strong correlation with the main TCP operation mode of the flow. In particular, short-lived TCP flows tend to operate in the slow-start mode, and long-lived TCP flows operate largely in the congestion-avoidance mode. Thus, it proposes TMD (TCP Mode Differentiation) that classifies traffic according to the current TCP operation modes of the flows. TCP flows in the slow-start mode are categorized as class S (Start). Class A (Avoidance) comprises all the other traffic including TCP flows in the congestion-avoidance mode and UDP (User Datagram Protocol) [84] flows.

Based on the TMD classification, it investigates a buffer management scheme that dynamically splits the shared FIFO (First-In First-Out) link buffer between the two traffic classes. Specifically, the buffer space is partitioned so that the ratio of the per-flow buffer shares

for classes S and A is maintained around k, which is the only parameter in the design. To enable simple implementation of the TMD classification in routers, the sender marks the packets of its flow as belonging to class S by setting a bit in the packet headers. The default setting of the bit indicates that the packet should be treated as belonging to class A.

1.3 Main Contributions

1.3.1 MCP: Congestion Control Protocol

This chapter proposes a new congestion control protocol named Multimodal Control Protocol (MCP) [81, 82], which has the following main characteristics:

- Convergence of data flows to full and fair utilization of the network resources.
- Fair use of the network resources by data flows with heterogeneous RTTs or packet sizes.
- Low data losses at network links.
- Congestion control overhead is limited to few bits per packet.
- Scaled responsiveness to changes in communication demands and network capabilities.

A distinguishing property of MCP is stable transmission after converging to efficient fair states. To ensure convergence to fairness, MCP incorporates an innovative mechanism that enables a flow to urge all flows sharing its bottleneck links to operate in a fairing mode dedicated to fairness improvement. To make the stable fair rates independent of round-trip times and packet sizes, MCP employs rate-based control and uniform timing of adjustments.

1.3.2 RD Network Services: Service Differentiation Architecture

The proposed RD Network Services [83] assume partial deployment, which is one of the main goals. The other important issue that we pursue is no punishment of currently existed

legacy traffic due to the deployment of the new architecture. Third, we rely on the currently existed transport protocols like TCP or UDP.

Summarizing, the main contributions of the RD Services are:

- Incremental deployability within the current Internet.
- No changes required in end-to-end transport protocols.
- Preservation of the current structure of the IP datagram header.
- Elimination of the billing and management problems of previous Differentiated Services (DiffServ) designs.

Besides, the approach of designing for deployability holds promise for solving other types of networking problems.

1.3.3 TMD: Scheme for Accelerating FCT

The TMD buffer partitioning consistently provides short-lived TCP flows with better performance without reducing throughput of long-lived flows. In some settings, TMD reduces FCT of web-like flows by as much as 40%. While offering the significant performance benefits, the simple TMD scheme comes with a low overhead and is amenable to incremental deployment in the current Internet. In particular, TMD does not require any changes in the end-to-end TCP congestion control, packet structure, or FIFO link scheduling. The link buffer is partitioned via differentiated simple discard of packets upon their arrival to the buffer. The treatment of legacy traffic, which TMD views as belonging to class A, is similar to the single service of the existing Internet.

1.4 Dissertation Overview

The dissertation has the following structure. Chapter 2 discusses the background and related work. In particular, it considers different approaches that allow to support performance characteristics of the Internet applications. Chapter 3 talks about MCP, a congestion control protocol that aims at supporting low queuing delay. Chapter 4 describes the RD

Services, a new service differentiation architecture. Chapter 5 presents TMD, a new link buffer allocation scheme, for expediting FCT of short-lived flows. Finally, Chapter 6 summarizes the dissertation with the conclusions and future work.

Chapter 2

Background and Related Work

The goal of this chapter is to give a reader the general understanding of the networking techniques that are used for the Internet applications. First, it provides main definitions used throughout the rest of the dissertation. Second, the chapter explains the main functional principles of TCP [52, 54], the main transport protocol currently used in the Internet. Then, it surveys the research results in the area of supporting low queuing delay.

This dissertation distinguishes three main approaches that allow to support low queuing delay. In the first one, a congestion control protocol is a main tool providing low queuing delay. The second approach relies on Active Queue Management (AQM), which incorporates different techniques for packet scheduling, dropping, and buffer allocation. The third approach extends the second one, and, besides router functionality, relies on assistance from end-hosts.

2.1 Definitions

The dissertation contains the following main terms:

Congestion window. It is the maximum amount of data a TCP source can send without acknowledgment from a TCP receiver. In other words, it is a maximum amount of data that a TCP source can send in RTT.

Flow. A network flow is a sequence of data packets that share the source and destination IP (Internet Protocol) [85] addresses, the source and destination port numbers, and the same transport protocol.

Flow Completion Time (FCT). FCT is the interval between initializing a network connection and receiving the last data packet by a destination.

Goodput. A connection goodput is the ratio between the amount of received data passed by a destination host to an application and the interval during which the considered data have been passed.

Link utilization. A link utilization is the ratio between the amount of data departed to the link and the maximum amount of data that can be sent by the link during the same time.

Loss rate. A loss rate at a link is the ratio between the amount of data dropped at the link and the amount of data arrived at the link during the same interval.

Max-min fairness. A max-min fairness is the way to allocate the network resources to flows. It maximizes the sending rates of flows with lower rates and satisfies the following conditions:

1. The rate allocation is feasible, i.e., the sum of allocated sending rates of flows sharing the same link does not exceed the link capacity.
2. The allocated sending rate of a flow cannot be increased without violating the allocation feasibility or without decreasing a sending rate of a flow that has equal or smaller sending rate.

In particular, for a network with a single bottleneck link a max-min fairness assumes that all flows get the same part of the link capacity. If a flow needs a less amount of the link capacity, then the other flows share the rest of the link capacity equally. We will refer to max-min fairness as fairness.

Propagation Round Trip Time (RTT). A propagation RTT is the total time required for a data packet to come to the destination and an ACK packet to come back to the source disregarding delays at link queues.

Queuing delay. A queuing delay of a packet is the interval between the packet arrival at the link and the packet departure to the link.

Sending rate. A sending rate, or sending bitrate, is the ratio between the amount of data sent by a source and the interval during which the considered data have been sent.

Throughput. A connection throughput is the ratio between the amount of data received by a destination and the interval during which the considered data have been received.

2.2 Transmission Control Protocol (TCP)

TCP has been the dominant transport protocol in the Internet since being proposed. TCP relies on a window-based congestion control mechanism, which means that a source controls its data sending rate through congestion window. TCP in general is not max-min fair, because its sending rate depends on RTT. TCP is an end-to-end protocol, which means that TCP does not use any information from the network routers, i.e., TCP relies only on signals from the ends of a TCP connection. Next, this section will overview two main mechanisms of TCP: reliability and congestion control.

2.2.1 Reliability

The reliable delivery of data over a TCP connection is realized as follows. The data to be transmitted are split into small pieces named segments. The maximum size of a segment can not exceed a special parameter, MSS (Maximum Segment Size). Each segment has its own sequence number. On getting a data segment, a TCP receiver sends a special acknowledgment packet, or ACK, to the TCP source. The main purpose of the ACK is to confirm the successful delivery of the data segment to the receiver. There is a special timeout mechanism implemented at a source. The main purpose of the timeout is to control time interval, during which a source is waiting for an ACK. If the timer expires, a source resends a segment that induced the timeout. Besides, there is a special mechanism that controls integrity of segments that is realized through calculating a checksum. If the checksum at the receiver is different from the checksum calculated by the source, it means that a segment has been modified while traveling from the source to the receiver. In this case, the receiver drops such a corrupted packet, which leads to retransmission of the corrupted packet.

2.2.2 Congestion Control

To determine an appropriate data sending rate, a TCP source employs a special mechanism named congestion control. In particular, the TCP source can operate in two different modes: slow start and congestion control.

Slow start.

A TCP source starts transmitting a data flow in slow start. The initial congestion window is typically one segment. Each time a TCP source gets an ACK, it increases the congestion window by one segment. Thus, the congestion window is doubled each RTT. Mathematically, operating in slow start is described by Multiplicative Increase:

$$cwnd = cwnd * b_I \tag{2.1}$$

where $b_I = 2$. As soon as the congestion window achieves the window threshold, one of TCP parameters, a TCP flow continues operating in congestion avoidance.

Congestion avoidance.

While operating in congestion avoidance, a TCP source increases its congestion window by one segment per RTT. In other words, on each receiving of an ACK, the congestion window changes as follows:

$$cwnd = cwnd + \frac{1}{cwnd} \tag{2.2}$$

Thus, each subsequent successful delivery of a data packet leads to the increase of TCP congestion window. The behavior of TCP on data loss event will be discussed next.

There are two main signals of losses delivered to a TCP source:

- **Duplicate ACK.** This is the situation when a TCP source gets three ACKs of the packet that has already been acknowledged.
- **Timeout.** It takes place when a retransmission timeout expires. It has been discussed in Section 2.2.1.

On each loss event a TCP source may react in two different ways. It either decreases the congestion window by two, i.e., applying Multiplicative Decrease (MI) rule, or set congestion window to one packet. In particular, TCP NewReno [34], the currently used version, uses the MI rule on receiving a duplicate ACK, and sets the congestion window to one on entering timeout. The duration of the retransmission timeout is controlled through a special variable, RTO, which is estimated based on the value of RTT and its variance. Besides, on each expiration of retransmission timeout the RTO is doubled.

2.3 Congestion Control Protocols

Based on the type of feedback, congestion control protocols that support low queuing delay can be divided into two main groups.

Delay-based congestion control is represented by such protocols as Congestion Avoidance using Round-trip Delay (CARD) [56], TCP Vegas [13], and TCP Africa [65]. In this approach, a flow measures its RTT and curbs transmission when RTT increases. The reaction to raising delays is helpful for avoiding buffer overflows but unfortunately comes only after the link queue has started to grow. The problem with these protocols is their slow convergence and absence of min-max fairness in the networks with heterogeneous RTTs.

Explicit congestion feedback from routers enables a congestion control protocol to prevent queuing. Depending on whether the explicit feedback consumes few bits per packet or more, explicit congestion control protocols can be classified as limited-feedback and rich-feedback. Katabi et al. [60] designed XCP (eXplicit Control Protocol) employing explicit feedback from routers. However, XCP requires multiple bits as a feedback, which leads to a potential overhead in a network and per-packet calculations. Dukkipati et al. [23] propose a new transport protocol named RCP (Rate Control Protocol) for decreasing FCT of short-lived flows. The idea was to calculate a fair sharing rate by routers and deliver that information to sending hosts. The subsequent work [25, 57] aims to handle the aggressiveness of RCP, which revealed in the scenarios with quick changes of network conditions. The other example of rich-feedback designs include JetMax [114] also suffering from a feedback containing multiple bits. Examples of limited-feedback protocols are Explicit Congestion Notification (ECN) [91], Variable-structure congestion Control Protocol (VCP) [107], Multi-Level feedback Congestion control Protocol (MLCP [87]), and Binary Marking Congestion Control (BMCC) [86]. In addition, a common drawback of the

mentioned protocols is a relatively big queuing delay, which can negatively affect the performance of the applications with strict requirements concerning delay.

Besides, researchers have proposed changes to TCP for improving the performance of TCP. Hu and Steenkiste [50] propose Paced Start, a new TCP startup algorithm. The main idea is to use the difference between the data packet spacing and the ACK packet spacing for estimating the available capacity. In addition, some attention was dedicated to admission control. In particular, Ramesh and Kasera [92] propose two session-control algorithms, which searched for an optimal rate of session admissions allowing to maximize the flow completion rate and minimize FCT.

2.4 Router-Based Mechanisms

This approach distinguishes two main techniques: buffer management and scheduling algorithms.

A buffer management technique specifies the algorithm according to which routers enqueue and drop the arriving data packets. Active Queue Management (AQM) allows to achieve low queuing delay through supporting a particular length of queue size in a buffer. Random Early Detection (RED) [35] and its variants [69], [29] make decisions about dropping packets based on the average queue length. Loss Ratio based RED (LRED) [104] employs loss ratio, BLUE [30] uses the loss rate and link utilization, Proportional-Integral (PI) [48] relies on the queue length and input traffic rate. Georgiadis et al. investigate buffer management policies for delivering rate and delay guarantees and propose Guaranteed Rate (GR) [38] service. Small link buffers [4, 27, 40, 89, 106, 115] assure that a shared queue stays short. This approach also requires a complementary end-to-end congestion control to ensure that buffer overflow and link underutilization do not disrupt application performance. Evolutional TCP (E-TCP) [43] is a recent loss-driven proposal for such congestion control. However, the proposals of small buffers do not come up with a unique conclusion about how to configure link buffers. Kumazoe et al. [66] propose to add two parameters to packet headers, Maximum Transmission Unit Delay (MTQ) and Queue Delay To Live (QTL), and use these parameters by routers for identifying and dropping packets experiencing too large end-to-end delay.

Scheduling algorithms describe the rules according to which data packets depart from link queues. Exploring scheduling algorithms to support delay guarantees has also attracted a lot of interest from researchers. Currently existing scheduling algorithms can be divided into three groups. The first one incorporates round-robin algorithms. Whereas G-3 [16], SRR^{+}, SRR$^{\#}$ [17] support bounded queuing delay, Deficit Round Robin (DRR) [96], Smoothed Round Robin (SRR) [15], FRR [111] are able to provide only delay bounds in proportion to the number of active flows.

The second group of scheduling algorithms includes timestamp-based algorithms, where a virtual clock is maintained to emulate the concept of Generalized Processor Sharing (GPS) [78, 79]. In particular, WFQ (Weighted Fair Queueing) [21, 99, 100] as a part of Integrated Services (IntServ) [12] offers rate and delay guarantees to users. An improved version of WFQ is Worst-case Fair Weighted Fair Queueing (WF^{2}Q) [9] algorithm. The key problem for efficient implementation of WFQ and WF^{2}Q is a high complexity of computing the virtual time function [39]. Virtual Clock (VC) [112] allows to support end-to-end delay guarantees, but the problem of this scheme is the need to maintain per-flow state at each router. Core Jitter Virtual Clock (CJVC) [102] is able to support end-to-end delay guarantees without any per-flow state. Core-Stateless Fair Queuing (CSFQ) [101] is a fair queuing discipline that employs per-flow management only in edge routers. Core-Stateless Guaranteed Rate (CSGR) [62] algorithm provides the same delay guarantees as Guaranteed Rate (GR) [42] per-flow scheduling algorithm without maintaining per-flow state in core routers required by the latter. The extension of that work, Core-Stateless Guaranteed Throughput (CSGT) [63] architecture, provides the delay guarantees not only over long periods, but also over short time scales.

The third group uses the information about the total size of a flow or its already transmitted part. A number of size-sensitive link scheduling algorithms have been proposed [37, 94]. Avrachenkov et al. [6] come up with a scheduling algorithm named RuN2C that gives a priority to short-lived flows and relies on a size threshold for classifying flows. LAS (Least Attained Policy) [88] schedules the next packet to be served belonging to a flow that has received the least amount of service.

2.5 Service Differentiation Architectures

A prominent representative of the architectural innovations, IntServ (Integrated Services) [12], offers users a rich choice of services, including guarantees on end-to-end throughput and delay within a packet flow by means of admission control and link scheduling mechanisms such as WFQ and WF^2Q. While IntServ failed to gain ubiquitous adoption, early IntServ retrospectives attributed the failure to the complexity of supporting the per-flow performance guarantees, especially in busy backbone routers.

The proposal of DiffServ (Differentiated Services) [10] addresses the scalability concerns of IntServ by restricting complex operations to the Internet edges and offering just few services at the granularity of traffic classes, rather than individual flows. DiffServ maps the whole traffic into several classes, each of them gets the particular service specified by a Per-Hop Behavior (PHB). The set of PHBs for each data flow on its network path defines the service offered to the flow. Packets are assigned to a particular class by setting Differentiated Service Code Point (DSCP), the six most significant bits of the DiffServ Field in an IP header of a packet. Edge routers police the traffic through classifying, metering, marking, dropping, and shaping the traffic whereas core routers support PHBs. PHBs typically include two mechanisms: queue management and packet scheduling. Such an architecture allows to change policing and supporting mechanisms independently from each other. One can distinguish two different approaches that implement the idea of DiffServ.

The first one concentrates on ensuring particular performance characteristics. The Expedited Forwarding (EF) PHB [55] supporting Premium Service is used to provide the service characterized by low loss, low delay, low jitter, and ensured rate through domains. EF is attractive for such applications as VoIP, video [5], and other real-time services like online trading. EF traffic uses policing for ensuring its delay, loss, and jitter characteristics. Another PHB class, Assured Forwarding (AF) PHB [47], specifies different forwarding assurances for data packets. In more details, AF defines four classes of traffic, each of them is assigned to a certain amount of link capacity and buffer space. Within each class AF defines three drop priorities that specify a packet that should be dropped if congestion happens. The concept of AF called Assured Service is implemented on the basis of Random Early Detection (RED) [35] schema and is called RED with In and Out (RIO) [18]. The idea of RIO is based on applying different drop functions employed in RED for different classes of traffic. The Default PHB [74] is used to provide the classical best-effort service. Another AF implementation assumes use of Class Based Queuing (CBQ) [36], which is a

packet scheduling algorithm based on supporting hierarchical link sharing. In particular, the whole traffic is represented as a tree of different traffic classes, and each of them is assigned a queue and a certain capacity.

The second approach strives at providing relative differentiation of services. In particular, the approach defines Class-Selector PHBs [74] that assume eight different classes of traffic. Data packets from each subsequent class get the better service in the assigned capacity, packet delay, and loss rate than from the previous class. A model for relative delay differentiation is proposed in [22].

Alternative Best Effort (ABE) [51] proposes loss-delay differentiation. In particular, ABE offers a blue service with a smaller loss rate and a green service with a lower delay. The key problem with ABE is that it does not provide an explicit control over forwarding rates for blue and green packets. Most importantly, the blue service does not consistently provide a larger rate, e.g., by transmitting more aggressively, the green users can enjoy both a higher rate and lower queuing delay than those of the blue users. Besides, as the control rules of this architecture rely on loss rate and RTT estimations, it may be problematic to perform the estimations with good accuracy. Also, ABE requires per-packet calculations, which can negatively affect the link performance. Finally, while ABE considers it normal for a flow to mark some packets blue and other packets green, potential negative impact of such practices on legacy traffic raises a concern that the ABE design does not incorporate a sound strategy for incremental deployment. The lack of explicit rate-delay differentiation significantly weakens incentives for adopting ABE. Best Effort Differentiated Services (BEDS) [31] are similar to ABE and suffer from similar limitations.

In [75] the authors propose two architectures of service differentiation similar to Assured Forwarding services. The first one employs application-based differentiation, and the serving priority depends on the application. In particular, Telnet traffic gets the highest priority, web-like traffic - a smaller one, and ftp - the lowest priority. The second approach prioritizes short-lived flows, and classifies flows based on window of TCP flow. The problems with these designs are similar to the ones with the original DiffServ proposal, i.e., billing, management, and deployability. The Lottery Queuing [26] approach assumes applying lottery in making decisions about scheduling and dropping packets within routers to provide service differentiation. The question of providing several services for delay-sensitive applications with different delay requirements is considered in [8]. In [108] authors deliver the new routing mechanism, the key idea of which is to diversify routing paths. The proposed

architecture is scalable, compatible with the current ISP policies and incrementally deployable. Traffic Validation Architecture (TVA) [109] confines the impact of denial-of-service attacks. The key idea of the design consists of two points. The first one is the concept of capability that enables receivers to authorize senders. The second one assumes preferential forwarding of authorized traffic by routers.

There have been numerous proposals of new architectures for improving the performance of TCP flows. One of the approaches relies on isolation of short-lived flows from long-lived ones. In particular, Yilmaz and Matta [110] come up with the idea of isolating short-lived, long-lived TCP flows, and UDP flows, but configuring the buffers for the partitions remained an open issue. Matta and Guo [71] in their architecture have a similar problem. Moreover, their scheme does not accommodate the changes in the network as their queue service assumes a static configuration. Shaikh et al. [95] propose a new routing scheme, the main idea of which is to perform dynamic routing for long-lived flows while employing static routing for short-lived flows.

Another promising direction for improving the per-flow performance is to use different dropping policies for short-lived and long-lived flows [45, 46]. In particular, the idea of the mentioned work is to give a higher priority to flows with the current length smaller than a defined threshold. The proposed architectures employ RIO (RED with In and Out) [18] or WRR (Weighted Round Robin) [61] schemes. One of the most important issues related to this design is that configuring RED [35] is a challenging task. The other solution is to use DropTail queues and to drop packets only from long-lived flows [68] relying on the idea of closely paced packets from high-rate flows. The idea of using the history of drops of already arrived packets is used by Mahajan et al. [70], where authors propose to apply RED preferentially to high-rate flows using drop history of packets, and by Stanojevic and Shorten [98]. Pan et al. [77] rely on the assumption that high-rate flows occupy a relatively large amount of buffer. Kantawala and Turner [59] investigate new schedulers based on DRR (Deficit Round Robin) [96] algorithm for improving the performance of short-lived flows.

Chapter 3

MCP: Congestion Control Protocol for Low Queuing

This Chapter presents MCP which strives to maintain low queues and avoid congestion losses at network links. The multimodal MCP engages routers and hosts in limited explicit communication. A distinguishing property of MCP is stable transmission after converging to efficient fair states. To ensure convergence to fairness, MCP incorporates an innovative mechanism that enables a flow to urge all flows sharing its bottleneck links to operate in a fairing mode, dedicated to fairness improvement. To make the stable fair rates independent of round-trip times and packet sizes, MCP employs rate-based control and uniform timing of adjustments. The reported evaluation of MCP confirms achieving its design objectives.

3.1 Requirements and Design Principles

This section presents requirements for MCP and design principles that are the basis of the design.

3.1.1 Requirements

One of the common congestion control objectives is that packet flows generated by applications should utilize the network efficiently and share it fairly. By efficiency, this chapter means high utilization of bottleneck links. In the view of fairness, flows that contend for a bottleneck link should acquire equal shares of the link bitrate. Hence, the first design requirement is as follows:

Requirement 1 *Controlled flows should converge to utilizing the network efficiently and fairly, where fairness means equal rates on shared bottleneck links.*

TCP does not support such max-min fairness because the sending rate of a TCP flow depends on the packet size and RTT of the flow. Since local flows might compete for a bottleneck link with flows that span countries or even continents, a TCP-like dependence of the stable transmission rate on RTT is found unacceptable. Also, any dependence of the fair share on packet sizes is considered undesirable. Therefore, one can add the following clarifying requirement:

Requirement 2 *Heterogeneity of RTTs or packet sizes should not undermine fairness.*

While the propagation component alone can make RTT of a flow high, extra queuing in the network can render the RTT unacceptable for the served application. For example, human perception of an interactive multimedia application is likely to degrade dramatically when RTT exceeds few hundred milliseconds. Another negative aspect of long queuing in routers is the possibility of packet discard when the queue grows too large. Such losses are undesirable because recovery from the losses raises communication overhead (e.g., when the recovery relies on forward error correction) or boosts delays even further (e.g., when the recovery is through retransmission). Thus, the third design requirement is as follows:

Requirement 3 *At any network link, queuing should be low, and losses should be avoided.*

It is important that the communication overhead of the protocol does not consume a significant fraction of the network capacity. Moreover, to facilitate integration of the design with current protocols, such as the IP, it is required that the congestion control protocol uses no more than few bits in the header of each packet:

Requirement 4 *Congestion control overhead should be limited to few bits per packet.*

TCP enables a new flow to acquire the available network capacity promptly. Unfortunately, the reaction of existing TCP flows to increases in bottleneck link capacities or declines in competing traffic is less scalable. When the available capacity is large, TCP in congestion avoidance converges to high utilization of the bottleneck link slowly. It is postulated that the congestion control protocol should provide scalable responses to all possible changes in network conditions:

Requirement 5 *Response to changes in communication demands and network capabilities should scale well.*

3.1.2 Design Principles

Satisfying all of the above requirements with a single simple algorithm is difficult, if not infeasible. To provide a new flow with scalable acquisition of the available capacity, TCP uses the slow-start mode which unfortunately does not assure convergence to fairness and might cause significant losses by overrunning the buffer of the bottleneck link. For fairness convergence and smoother transmission increases, TCP also employs the congestion-avoidance mode. The design of TCP highlights the promise of a *multimodal* approach where each mode of operation pursues only a subset of all the design requirements. This approach constitutes the basis for the first design principle:

Principle 1 *Incorporate multiple modes of operation where each mode pursues a subset of the design objectives.*

Requirement 3 of low queuing in the network necessitates that flows react to incipient congestion of a link even before the router queue starts to build up. Hence, it is opted for a design where routers inform hosts explicitly about the bottleneck link utilization. In accordance with Requirement 4, such explicit feedback should not consume more than few bits in the header of any packet. This leads us to:

Principle 2 *To prevent queue buildups in routers, use the bottleneck link utilization as explicit few-bit feedback.*

As per Requirement 2, the stable transmission rate of a flow should be independent from RTT. To compensate the natural dependence of self-clocked protocols on RTT, the algorithm for adjusting the transmission window should explicitly account for RTT. VCP is an example of such solution. However, the RTT-aware transmission adjustment faces instability problems due to changes in RTT [107]. To satisfy Requirement 1, a different approach is selected where the control parameter is not the congestion window but the sending rate, and all flows adhere to uniform rules and timing for transmission adjustment. Furthermore, to make the transmission rate independent of packet sizes, the protocol should view the rate in terms of bits, not packets. Hence, the third design principle is as follows:

Principle 3 *Use the sending bitrate as the control parameter and employ uniform adjustment timing for all flows.*

TCP, VCP, and other protocols continue adjusting the transmission even when the bottleneck link is utilized efficiently and shared fairly. These further oscillations yield no meaningful improvement in fairness but cause such undesirable effects as long queuing at the bottleneck link or low utilization of the link capacity. This leads to the following design principle that distinguishes the approach used in this chapter significantly from the existing protocols:

Principle 4 *Incorporate a fairing mode and operate in it only as long as needed for convergence to sufficient fairness.*

After the bottleneck link is utilized efficiently, and the fairing mode ensures high fairness, the protocol supports Requirement 3 the best if the transmission is kept the most smooth, i.e., constant. Hence, the last design principle is as follows:

Principle 5 *Keep the transmission steady after achieving high efficiency and fairness.*

3.2 Design Details

Based on the principles formulated in Section 3.1.2, we now design Multimodal Control Protocol (MCP). First, this section describes MCP features in the order of their derivation from the design principles. Then, it summarizes the protocol with respect to the following four general categories of its operation: explicit communication format, router, sender, and receiver algorithms.

3.2.1 Link Utilization as Explicit Feedback

In accordance with Principle 2, MCP uses the bottleneck link utilization as explicit few-bit feedback. Computing the feedback involves computing the link utilization at every router on the data path of the flow. Below, this chapter describes when and how this information is computed and communicated to the sender.

Timing

The router measures utilization for each of its output links. In support of Principle 3, all routers adhere to uniform timing by measuring the link utilization periodically with the same period of duration T. According to statistics [58, 80], between 75% and 90% of all Internet flows have RTT less than 200 ms. Therefore, to alleviate unnecessary oscillations of the end-to-end control, the design uses 200 ms as the value of T.

Calculation

The router employs the following equation to compute utilization U of a link:

$$U = \frac{N + Q}{CT} \tag{3.1}$$

where N is the amount of data that has arrived for the link during the previous period, Q is the minimum size of the link queue during this previous period, and C is the link capacity. Including Q into the equation helps MCP to avoid persistent long queuing.

Communication

Because Principle 2 limits the amount of communication overhead, the router encodes the calculated link utilization into few bits. If E_1 and E_2 denote respectively encodings of utilizations U_1 and U_2, then $E_1 > E_2$ is equivalent to $U_1 > U_2$. The sender transmits each data packet with the lowest encoding. When the router receives a packet, the router compares encoding E of the local link utilization and encoding P contained in the packet header. If $E > P$, then the router resets the encoding in the packet header to E. After the packet reaches the receiver, the encoding in the packet header represents the bottleneck link utilization for the data path of the flow. The receiver echoes this encoding to the sender via an acknowledgment (ACK) packet.

3.2.2 Operational Modes

Scaling mode

Now, it is discussed how the sender benefits from received encoding E of bottleneck link utilization U. Whenever E indicates that the bottleneck link utilization is below 48%, the sender operates in a *scaling mode* designed for scalable increase of U into an area of relatively light underutilization. The scaling mode achieves this by using MI(2), multiplicative increase with factor 2. The choice of the factor ensures that upon leaving the scaling mode, MCP does not raise the bottleneck link utilization above 96%.

Overloaded mode

Another extreme on the bottleneck utilization spectrum is represented by the encoding that corresponds to $U \geq 0.98$. When U is at least 98%, the sender is in an *overloaded mode* and uses MD(0.5), i.e., multiplicative decrease with factor 0.5. The only exception to the decrease rule is discussed later in the context of another mode. The overloaded mode provides MCP with scalable response to severe overloads of the bottleneck link. The choice of factor 0.5 ensures that a decrease from the overloaded mode lowers the bottleneck link utilization to at most 49% and does not switch MCP into the scaling mode.

Fairing mode

MCP concerns itself with fairness improvement and uses a *fairing mode* for these purposes only when $U \in [0.48; 0.98)$. The fairing mode improves fairness by using AI(80 kbps), i.e., additive increase with coefficient 80 kbps. The measurement of the increase step in bits per second, rather than packets per RTT, is in conformance with Principle 3. The increase by 80 kbps corresponds to one 1000-byte packet per RTT of 100 ms, a common setting for TCP flows in congestion avoidance. In general, the duo of the fairing and overloaded modes provides fairness convergence via AIMD(80 kbps; 0.5) control similar to TCP congestion avoidance. According to Principle 4, flows should operate in the fairing mode only as long as needed to achieve sufficient fairness. Next section specifies how long the flows stay in the fairing mode, and how they decide when to switch into the fairing mode.

Time to stay in the fairing mode

To determine an appropriate longevity for operating in the fairing mode, the analysis in the classical Chiu-Jain model is conducted where n distributed users adjust their loads on a shared resource in response to uniform binary feedback that indicates whether the total load exceeds the target load [14]. Each user i uses AIMD(x; 0.5) to adjust its load $l_i(t)$ at time t, where $i = 1, \ldots, n$. The particular value of x is not essential for the analysis, and the analytical results also apply to the AIMD(80 kbps; 0.5) control employed by MCP. It is reasoned about fairness improvement under AIMD(x; 0.5) in terms of *increase-decrease cycles* where an increase-decrease cycle consists of adjustments between two peaks of the oscillating total load. We quantify fairness of the resource sharing at time t with the following fairness index:

$$F(t) = \min_{i,j=1}^{n} \frac{l_i(t)}{l_j(t)}. \tag{3.2}$$

The fairness index of 1 corresponds to the perfect fairness when all individual loads are equal.

To improve readability, the detailed descriptions of the used model and analysis is relegated to Section 3.2.4. The main conclusion from the analysis is that a small number of increase-decrease cycles is sufficient for AIMD(x; 0.5) to provide high fairness. For scenarios where the increase step is small in comparison to the fair share, and new users do not outnumber existing converged users, this chapter derives a lower bound for fairness improvement under AIMD(x; 0.5) and show that the fairness index grows after one increase-decrease cycle to at least 0.33, after two cycles to at least 0.60, after three cycles to at least 0.77, and after seven cycles to at least 0.98. Since 98% is viewed as sufficiently high fairness, MCP operates in the fairing mode for *seven increase-decrease cycles*. If during the process of fairness improvement the contending flows switch temporarily into the scaling mode (because of decline in competing traffic or increase in the bottleneck link capacity), the disruption does not affect the count of remaining increase-decrease cycles.

Mechanism for switching into the fairing mode

When a flow terminates, or the capacity of the bottleneck link changes, the fairness index does not decrease. Hence, the mechanism for switching into the fairing mode is important

primarily when new flows start. A less common but plausible scenario occurs when an application that has willingly generated a data flow at an unfairly low rate decides to increase the rate of the flow to the fair share of the bottleneck link capacity. Handling such scenarios is not straightforward. Without assistance from the network, it is extremely difficult for existing flows to detect that their bottleneck links have started to serve a new flow. Even the router of the bottleneck link has no effective implicit means to infer the desire of an existing slow flow to reclaim its fair share of the link capacity.

Due to above reasons, and because Principle 2 allows explicit limited feedback, MCP incorporates an explicit communication mechanism that enables a flow to urge all flows sharing its bottleneck links to operate in the fairing mode. More specifically, whenever a flow wants to improve fairness of the bottleneck link sharing, the sender of the flow sets a *fairing bit* in the headers of all data packets transmitted during the next seven increase-decrease cycles. When a packet with the set fairing bit arrives for being forwarded to a link, the router does the following for all data packets of *all* flows forwarded into the link before the end of the next period of the link-utilization measurement: before forwarding the packet into the link, the router checks the link-utilization encoding in the packet header; if the encoding corresponds to utilization range $[0.48; 0.98)$, the router sets the fairing bit in the header of the packet. This ensures that the receivers of all flows behind the shared bottleneck links learn about the need to switch into the fairing mode. Whenever the receiver of a flow receives a data packet with the set fairing bit, the receiver echoes the set fairing bit to the sender of the flow via the ACK packet. After the sender receives an ACK packet with the set fairing bit and link-utilization encoding that indicates the bottleneck link utilization between 48% and 98%, the sender switches into the fairing mode.

The above mechanism conceals a danger that a request to switch into the fairing mode might affect an unnecessarily large portion of the network. Consider the following example in the topology shown in Figure 3.1, where a through h denote hosts while A, B, C, and D are routers. Flows f_1 and f_2 traverse paths $gCDh$ and $aABCDd$ respectively. Then, flow f_3 starts and traverses path $eABf$. After the new flow requests the fairing mode, router A sets the fairing bit in packets of flow f_2, and this causes router C to set the fairing bit in packets of flow f_1. Consequently, all three flows start operating in the fairing mode, even though f_1 does not share any link with f_3.

To ensure that a request of the fairing mode affects only those flows that share bottleneck links with the requesting flow, MCP relies once again on explicit communication between hosts and routers and incorporates a *this-path bit*. Whenever the sender of a flow sets the

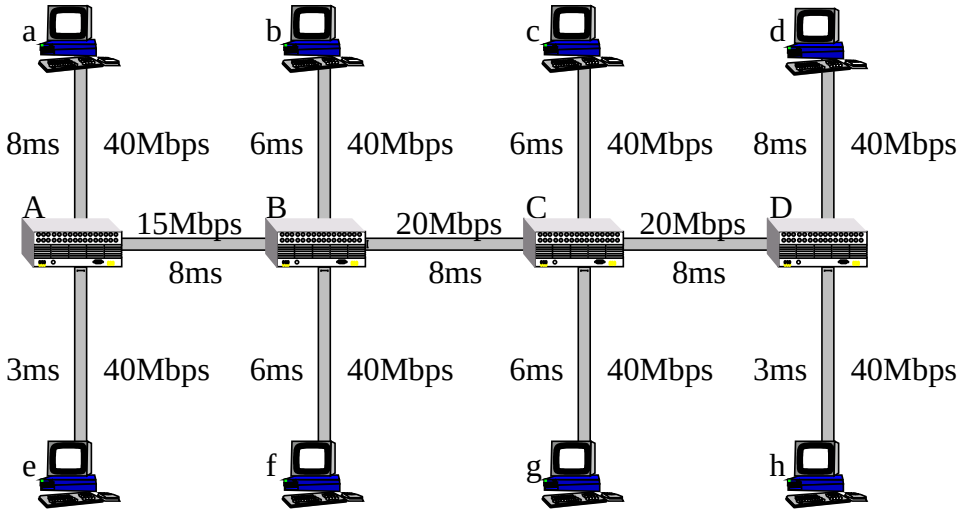


Figure 3.1: Network topology.

fairing bit in a data packet, the sender also sets the this-path bit in the packet. Routers never change this-path bits of forwarded packets. Only if both the fairing and this-path bits are set in the header of an incoming packet, the router propagates the set fairing bit to other flows. This prevents chain reactions that multiply set fairing bits needlessly.

Enhancing mode

From now on, this chapter discusses MCP operation when no flow demands the fairing mode but $U \in [0.48; 0.98)$. Although high fairness is achieved by this point, the bottleneck link utilization can be as low as 48%. To improve the utilization, range $[0.48; 0.98)$ is split into ranges $[0.48; 0.88)$ and $[0.88; 0.98)$. Hence, MCP distinguishes between only the following four ranges of link utilization: $[0; 0.48)$, $[0.48; 0.88)$, $[0.88; 0.98)$, and $[0.98; \infty)$. Their respective two-bit encodings are 00, 01, 10, and 11. When $U \in [0.48; 0.88)$, MCP is in an *enhancing mode* and uses MI(1.1), which ensures a scalable increase into the $[0.88; 0.98)$ range without overshooting into the overloaded mode.

Smoothing mode

After rising from the enhancing mode, MCP switches into a *smoothing mode* where $U \in [0.88; 0.98)$. The goal is to push the bottleneck link utilization further up, while being cautious not to cause a buildup of the link queue. The smoothing mode employs AI(80 kbps)

Mode	Bottleneck link utilization		Fairing bit	Control rule
	Range	Encoding		
Scaling	$[0; 0.48)$	00	0 or 1	MI(2)
Fairing	[0.48;0.98)	01 or 10	1	AI(80 kbps)
Enhancing	[0.48;0.88)	01	0	MI(1.1)
Smoothing	[0.88;0.98)	10	0	AI(80 kbps) until first overload then AD(80 kbps) once
Stable	[0.88;0.98)	10	0	constant
Overloaded	$[0.98; \infty)$	11	0 or 1	MD(0.5)

Figure 3.2: Modes of MCP operation

until the first overload (i.e., until U becomes at least 98%) and then applies AD(80 kbps) once to negate the previous overloading increase. As it is mentioned in Section 3.2.2, applying AD(80 kbps) is the only exception to using MD(0.5) for decrease. After the backtracking, MCP switches into a *stable mode*.

Stable mode

The stable mode is the ultimate regime of constant transmission prescribed by Principle 5. A flow stays in the stable state as long as $U \in [0.88; 0.98)$ and no flow requests a switch into the fairing mode.

3.2.3 Timing of Transmission Adjustment

As per Principle 3, MCP prescribes uniform timing for transmission adjustment by all flows. According to the control theory, the sender of a flow should adjust its transmission rate only after the feedback reflects the impact of the previous adjustment on the network [90]. In the proposed design, the sender increases its transmission once per $2T$, where T is the period of link-utilization measurement. Since the default value of T is 200 ms, each flow regardless of its RTT raises its transmission rate once per 400 ms. As soon as the feedback indicates nascent overload, the sender immediately curbs the transmission and restarts its transmission-adjustment timer. The reaction to sustained overload is at the standard pace of one decrease per $2T$.

The initial transmission rate for every flow is 80 kbps. To translate the current transmission rate into a specific schedule of packet transmission, MCP distributes the transmitted packets

uniformly during any inter-adjustment interval. After starting transmission of a packet of size S bytes, the sender schedules transmission of the next packet for $\frac{8S}{R}$ seconds later, where R denotes the current transmission rate in bps. Whenever the sender adjusts R (i.e., when the transmission-adjustment timer expires, or when the feedback indicates nascent overload), the sender also recalculates $\frac{8S}{R}$ and adjusts accordingly the remaining wait for the next packet.

3.2.4 Analysis of the Fairing Mode

This section reviews the classical Chiu-Jain model and then uses it to analyze for how long MCP flows should operate in the fairing mode.

Chiu-Jain model

In Chiu-Jain model [14], n distributed users share a single resource that has a target load C. The model is synchronous and employs a discrete timescale. Every instant on the timescale represents a moment when all the users adjust their loads on the resource. At time t, user i imposes a positive real load $l_i(t)$. Vector $\vec{l}(t) = (l_1(t), l_2(t), \ldots, l_n(t))$ captures all individual loads. The total load of the users is $L(t) = \sum_{i=1}^{n} l_i(t)$. By time t, the system provides all users with a uniform binary feedback

$$f(t) = \begin{cases} 0 & \text{if } L(t-1) \leq C, \\ 1 & \text{otherwise} \end{cases} \tag{3.3}$$

that indicates whether the total load of the users after the previous round of adjustments exceeds the target load.

Chiu and Jain applied their model to analyze behavior of Additive-Increase Multiplicative-Decrease (AIMD) algorithms that change the load of each user i as follows:

$$l_i(t) = \begin{cases} l_i(t-1) + x & \text{if } f(t) = 0, \\ y l_i(t-1) & \text{otherwise} \end{cases} \tag{3.4}$$

where coefficients x and y are constants such that $x > 0$ and $0 \leq y < 1$. A specific adjustment algorithm within the AIMD class is referred to as AIMD(x; y) where x and y represent respectively the AI and MD coefficients of the algorithm. Chiu and Jain showed

that the load of every user under an AIMD algorithm converges from any initial state toward the efficient fair state where the load of each user is $\frac{C}{n}$. The fairness index increases monotonically and converges to 1.

Sufficient longevity of the fairing mode

To reason about fairness improvement under AIMD(x; 0.5) after the total load reaches the target load, this chapter defines a notion of an *increase-decrease cycle* as a series of adjustments between two peaks of the oscillating load. If the increase step is sizable in comparison to the fair share, each increase-decrease cycle improves fairness significantly. Hence, we focus on more challenging scenarios where x is small with respect to $\frac{C}{n}$ and assume that the extent of overshooting the target load is always negligible. Then, since decrease with factor 2 releases one half of the target load, each increase-decrease cycle contains

$$m = \frac{C}{2nx} \tag{3.5}$$

increase steps. In considered cases, new users arrive to the system when existing users already utilize the resource efficiently and fairly. Also, we limit the analysis to settings where the new users do not outnumber the existing users. Then, the maximal individual load is initially at most twice the fair share:

$$l_{max}(t_0) \leq \frac{2C}{n} \tag{3.6}$$

where t_k denotes the k-th time the total load reaches the target load, and $l_{max}(t_k)$ represents the maximum individual load at time t_k. The k-th increase-decrease cycle transforms the maximal individual load into

$$l_{max}(t_k) = \frac{l_{max}(t_{k-1})}{2} + \frac{C}{2n}. \tag{3.7}$$

After $k-1$ increase-decrease cycles, the maximal individual load reduces by time t_{k-1} to

$$l_{max}(t_{k-1}) = \frac{l_{max}(t_0)}{2^{k-1}} + \frac{C}{n}\left(1 - \frac{1}{2^{k-1}}\right). \tag{3.8}$$

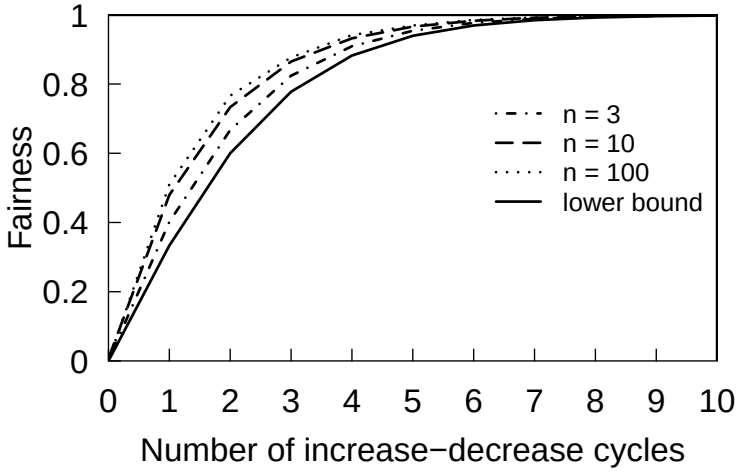


Figure 3.3: Speed of fairness convergence under AIMD(x; 0.5).

Taking into account Inequality 3.6, we derive

$$l_{max}(t_{k-1}) \leq \frac{C}{n}\left(1 + \frac{1}{2^{k-1}}\right). \tag{3.9}$$

From the definition of the fairing index, we derive that fairness after the k-th increase-decrease cycle of one decrease and m increases becomes

$$F(t_k) = F(t_{k-1}) + \frac{1 - F(t_{k-1})}{1 + \frac{l_{max}(t_{k-1})}{2mx}}. \tag{3.10}$$

Combining the above expression with Equation 3.5 and Inequality 3.9, we establish that the fairness index after the k-th increase-decrease cycle is bounded from below as:

$$F(t_k) \geq F(t_{k-1}) + \frac{1 - F(t_{k-1})}{1 + \frac{\frac{C}{n}\left(1+\frac{1}{2^{k-1}}\right)}{2\frac{C}{2nx}x}}, \tag{3.11}$$

i.e.,

$$F(t_k) \geq F(t_{k-1}) + \frac{1 - F(t_{k-1})}{2 + 0.5^{k-1}}. \tag{3.12}$$

Since the fairness index is at least zero, combination of $F(t_0) = 0$ and Inequality 3.12 provides an approximate lower bound for fairness after an arbitrary number of increase-decrease cycles. Figure 3.3 plots this lower bound together with graphs of the fairness

index under AIMD(80 kbps; 0.5) for different values of n in the system where the target load is $C = 100$ Mbps, one (new) user has initial load 80 kbps while the initial load of every other user is $\frac{C}{n-1}$. The line for $n = 2$ is indistinguishable from the lower bound. All the plotted graphs agree that seven increase-decrease cycles are sufficient to provide high fairness of at least 98%.

3.2.5 Summary of the MCP Design

This section summarizes MCP design in terms of its explicit communication format, router, sender, and receiver operation.

Explicit communication format

MCP allocates four bits in the header of each data packet for explicit communication between hosts and routers. Two of the bits are used to notify the sender about the bottleneck link utilization of its data path. The other two bits (fairing bit and this-path bit) enable the sender to urge all flows sharing its bottleneck links to operate in the fairing mode.

Router operation

Routers provide explicit feedback to senders through receivers. To form the feedback, each router periodically computes utilizations of its output links. Routers also set fairing bits in forwarded packets to disseminate to appropriate flows a request of operating in the fairing mode.

Sender operation

The sender operates in one of the following six modes: scaling, overloaded, fairing, enhancing, smoothing, and stable. The choice of the mode depends on explicit feedback in accordance with Figure 3.2.

Receiver operation

The receiver sends an ACK packet for every incoming data packet. The ACK packet echoes the fairing bit and encoding of the the bottleneck link utilization.

3.3 Performance Evaluation

This section reports the experimental evaluation of MCP and discusses the results. In particular, to estimate the performance characteristics of the proposed protocol, we conduct the simulations in version 2.29 of ns-2 [72].

3.3.1 Experimental Settings

General settings in the experiments are as follows: a packet size equals 1000 bytes; propagation delay of a bottleneck link is 8 ms; buffer size of a link is equal to the product of the link capacity and the minimum RTT among the flows in the simulation; link queuing discipline is FIFO. Unless stated otherwise, the capacity of a bottleneck link is 20 Mbps, and the capacities of non-bottleneck access links are 40 Mbps. To trace changes of a queue size in time, we sample the queue size every 10 ms. To plot the dependency of a queue size on a parameter, we measure the instantaneous value of the queue size. There are three types of conducted simulations. Simulations of the first kind illustrate how MCP behaves, e.g.,with respect to fairness convergence and bottleneck link utilization. The second class of simulations studies characteristics of MCP as functions of different network parameters. In these experiments, a single parameter is varied while all the other parameters are kept fixed. We run five simulations for each experimental settings, and calculate the average value. Besides, we plot the minimum and maximum values for each experimental settings. The third type of the experiments compares MCP with the existed protocols TCP and VCP.

3.3.2 Convergent Behavior

To illustrate the convergent behavior of MCP, this section presents the results of an experiment with three flows that have propagation RTT of 20 ms, 40 ms, and 100 ms respectively.

The network topology is a single-bottleneck dumbbell with the bottleneck link capacity of 20 Mbps. The duration of the experiment is 800 sec. Figure 3.4 reports the experimental results. The queue size at the bottleneck link never exceeds 7 packets. All three flows converge to stable transmission. In the stable state, the fairness index is 0.91, and the bottleneck link utilization reaches the high 0.97.

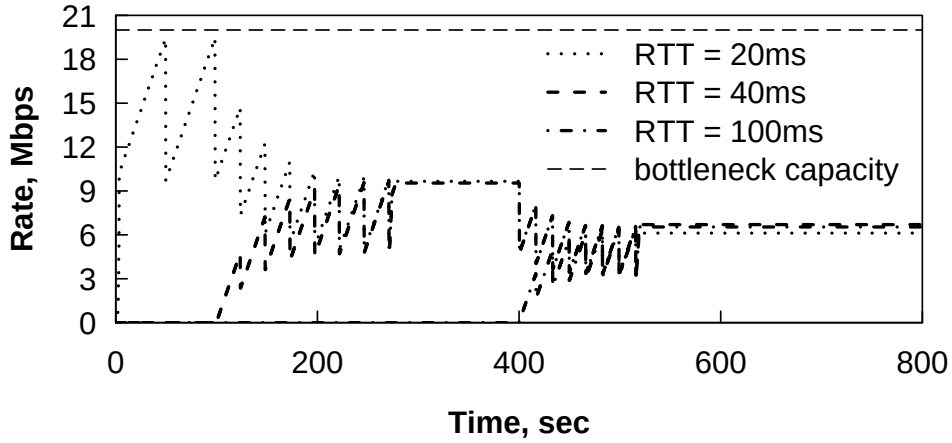
3.3.3 Scalability Properties

Influence of the Link Capacity

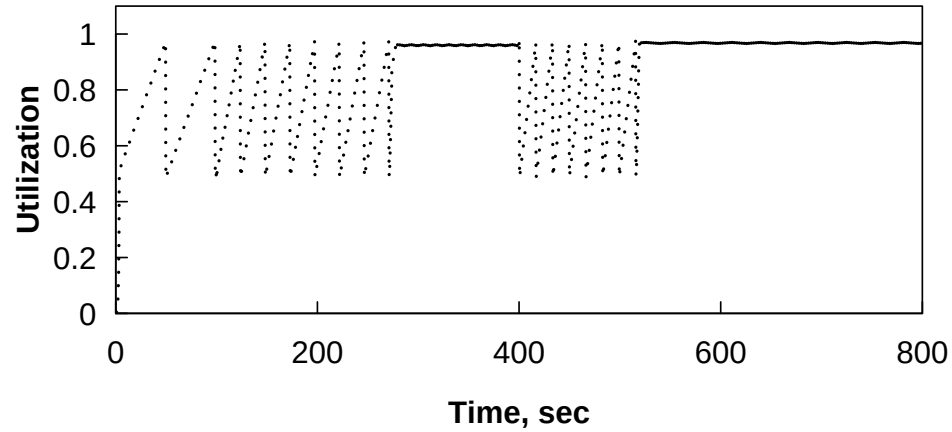
To study the impact of the bottleneck link capacity on MCP operation, simulations with 20 flows are performed. Propagation RTTs of the flows are uniformly distributed over the range from 20 to 100 ms. Each simulation lasts for 480 sec. The network topology is a single-bottleneck dumbbell where the bottleneck link capacity varies from 20 to 500 Mbps, and the capacities of non-bottleneck access links are scaled up proportionally. Arrival times of the flows are selected randomly from the interval between 0 and 10 sec. Figure 3.5 shows that the stable-state fairness, bottleneck link utilization, and peak queue size are relatively insensitive to the bottleneck link capacity: the average fairness index is between 0.776 and 0.886; the average link utilization lies is the range from 0.924 to 0.970; the peak queue size falls between 9.4 and 12.6 packets.

Impact of the Population Size

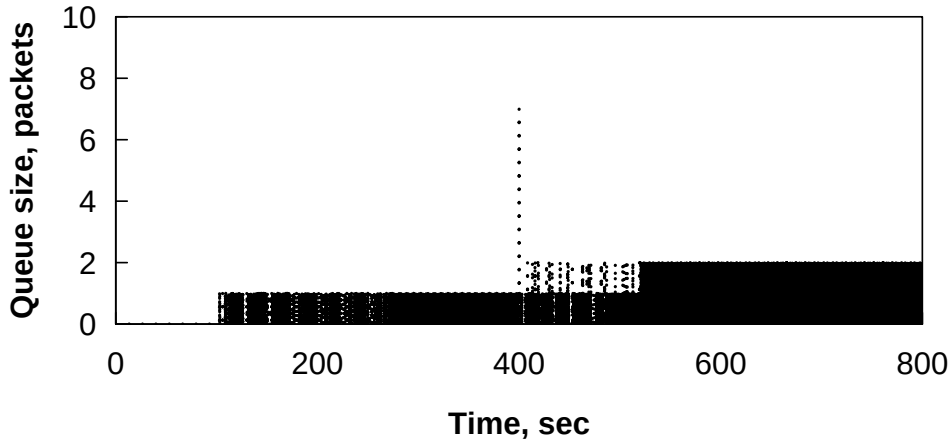
MCP performance with different numbers of flows is investigated in the single-bottleneck dumbbell topology where the bottleneck link has capacity 200 Mbps and propagation delay 24 ms while every access link has capacity 400 Mbps and propagation delay 3 ms. Hence, propagation RTT for each flow is 60 ms. The number of flows is changed from 50 to 600. Arrival times of the flows are randomly chosen from the interval between 0 and 10 sec. Figure 3.6 depicts how the number of flows affects the bottleneck link utilization and queue size. While the average link utilization remains relatively stable (it varies between 0.896 and 0.968), the queue size at the bottleneck link is significantly more sensitive to the flow count. As the number of flows increases, the average peak queue size rises from 17.8 to 931.4 packets, and the average peak queue size in the stable state grows from 6.2 to 152.8 packets. Although the buffer consumption per flow is less than 1.6 packets in general and



(a)



(b)



(c)

Figure 3.4: MCP convergence to fair high utilization and low queue size at the bottleneck link: (a) Transmission rates of the three flows; (b) Utilization of the bottleneck link; (c) Queue size at the bottleneck link.

less than 0.26 packets in the stable state, MCP design needs further improvements in order to maintain a low overall queue size regardless of the number of flows. Figure 3.6b also shows that MCP prevents packet losses in all the conducted simulations.

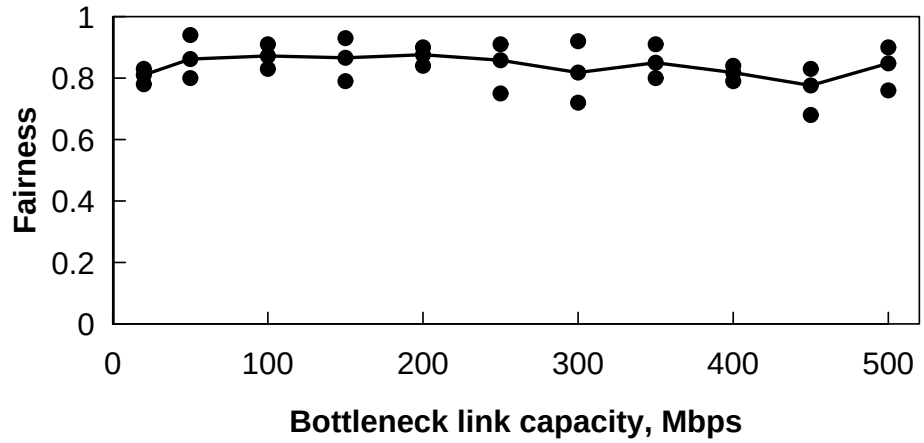
3.3.4 Heterogeneous Environments

Influence of Propagation RTTs

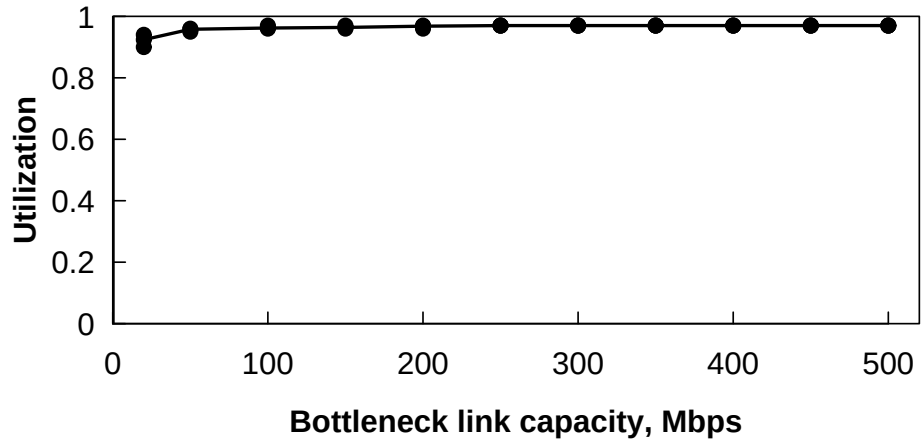
To evaluate MCP under different RTTs, simulations in the single-bottleneck dumbbell topology with 10 competing flows and the bottleneck link capacity 50 Mbps are conducted. This section uses $\frac{P_{max}}{P_{min}}$ as a control parameter where P_{max} and P_{min} respectively refer to the maximum and minimum propagation RTT of all the flows. P_{min} is always set to 20 ms, whereas P_{max} is modified in the range between 20 and 300 ms. Propagation RTTs of other flows are uniformly distributed between P_{min} and P_{max}. Arrival times for the flows are randomly picked from the range between 0 and 10 sec. The complete duration of the simulation is 120 sec. Figure 3.7 confirms that MCP stable-state operation is relatively independent of RTT heterogeneity: the fairness index varies from 0.880 to 0.934; the bottleneck link utilization lies in the range between 0.958 and 0.962; the average peak queue size at the bottleneck link changes only slightly from 7.2 to 11 packets.

Impact of Packet Sizes

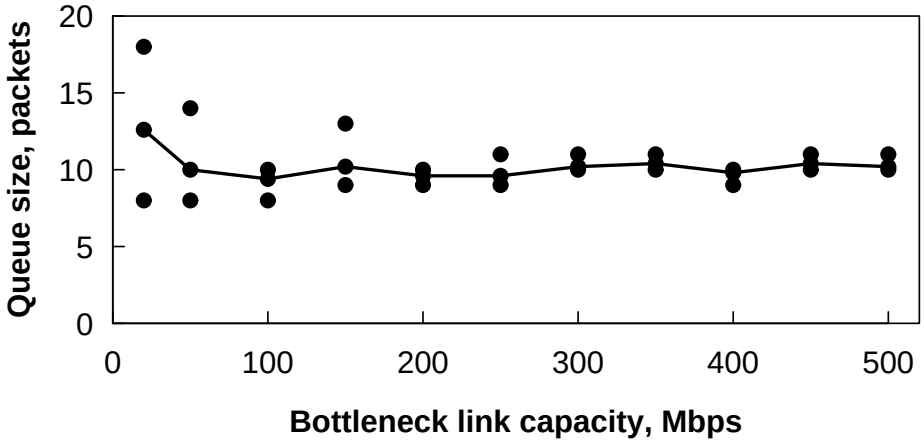
This section also examines the impact of packet-size heterogeneity. The topology is a single-bottleneck dumbbell with 5 flows and the bottleneck link capacity 50 Mbps. Arrival times of the flows are randomly chosen from the range between 0 and 5 sec. Each experiment lasts 200 sec. Propagation RTTs of the flows are uniformly distributed between 20 and 100 ms. In this set of the experiments, $\frac{S_{min}}{S_{max}}$ is employed as a control parameter where S_{max}, and S_{min} denote respectively the maximum and minimum packet size across all the flows. Packets within each particular flow are of the same size. S_{max} is always fixed to 1500 bytes, whereas S_{min} is varied between 500 and 1500 bytes. Other packet sizes are uniformly distributed between S_{min} and S_{max}. Figure 3.8 shows relative immunity of MCP stable-state operation to the heterogeneous packet sizes: the fairness index stays in the range from 0.742 to 0.930; the bottleneck link utilization varies from 0.962 to 0.970; the average peak queue size is almost constant, between 4.4 and 5.0 packets.



(a)

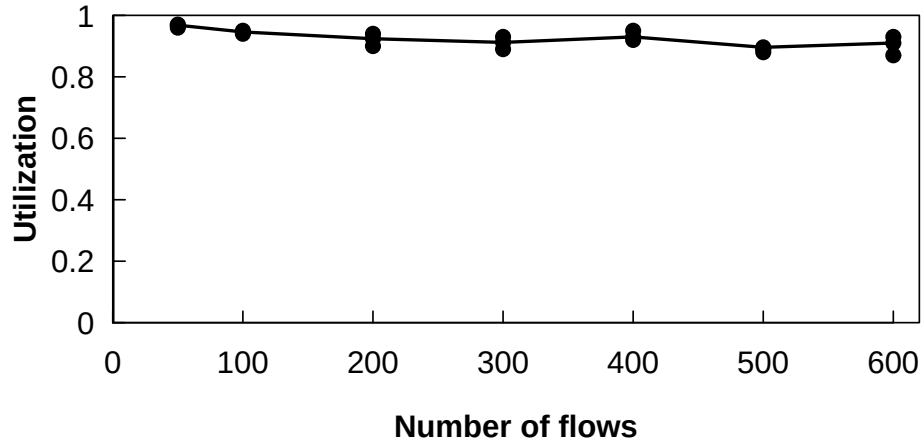


(b)

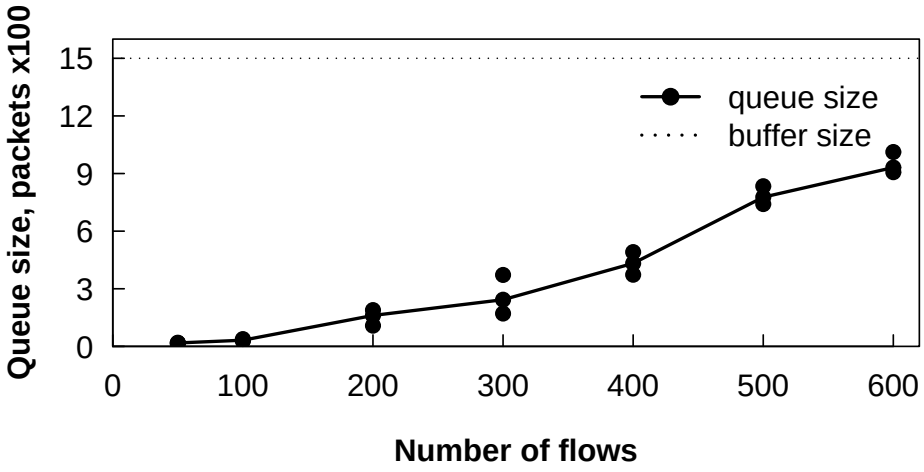


(c)

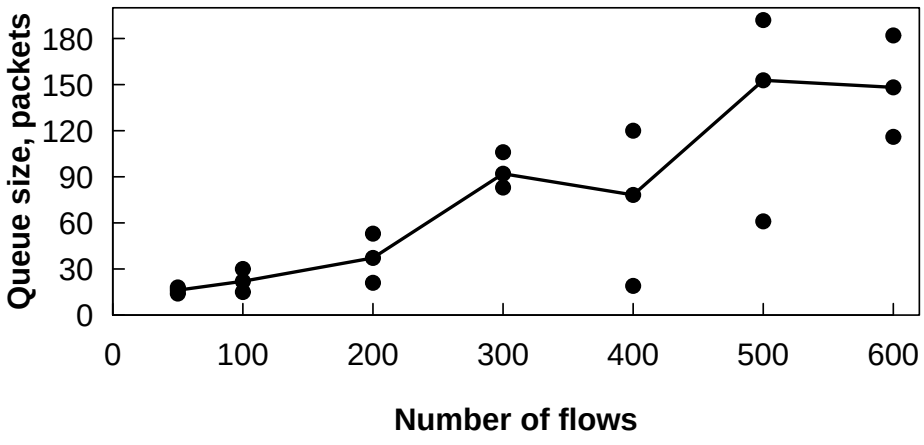
Figure 3.5: Low sensitivity to the bottleneck link capacity: (a) Fairness index; (b) Utilization of the bottleneck link; (c) Peak queue size at the bottleneck link.



(a)

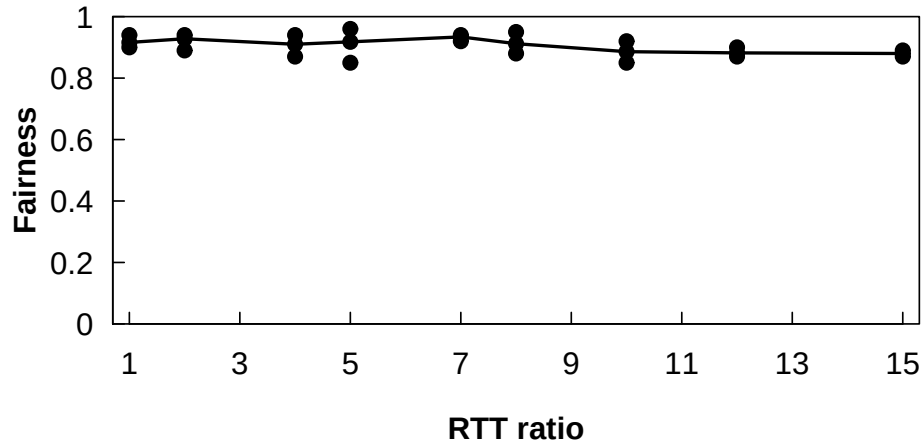


(b)

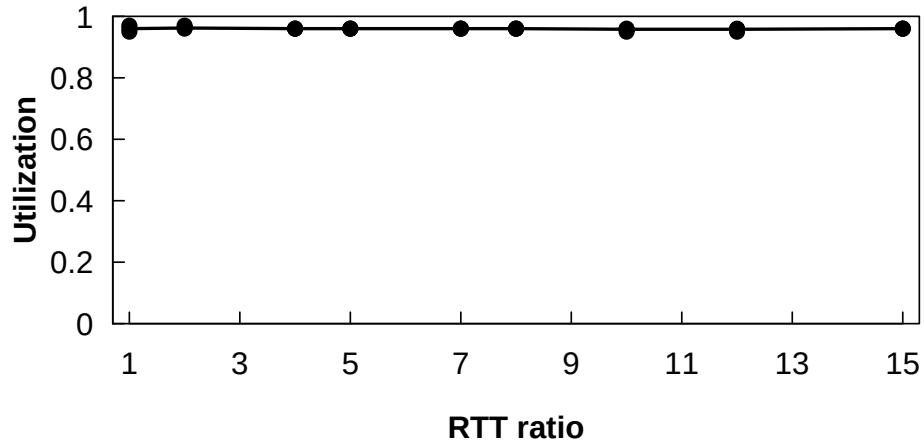


(c)

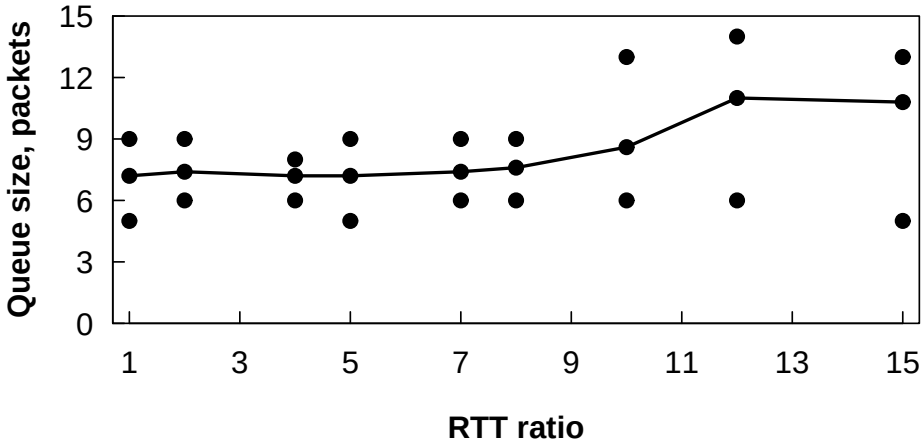
Figure 3.6: Dependence on the number of flows: (a) Utilization of the bottleneck link; (b) Peak queue size at the bottleneck link; (c) Peak queue size at the bottleneck link in the stable state.



(a)

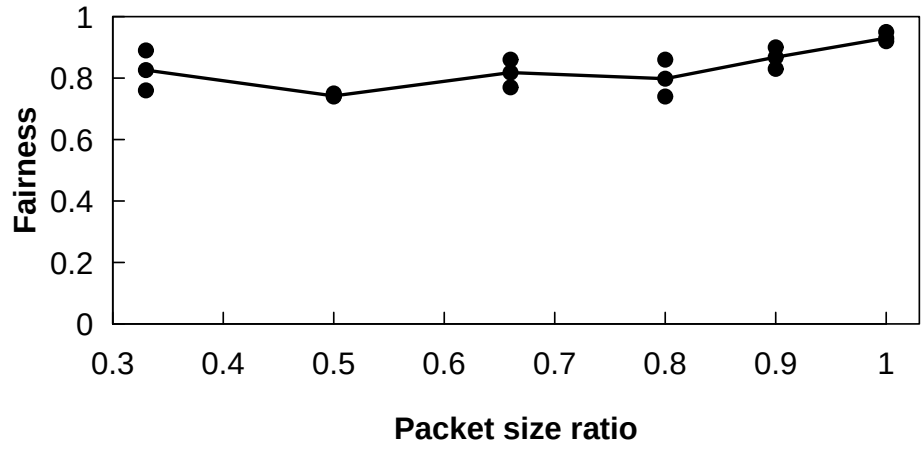


(b)

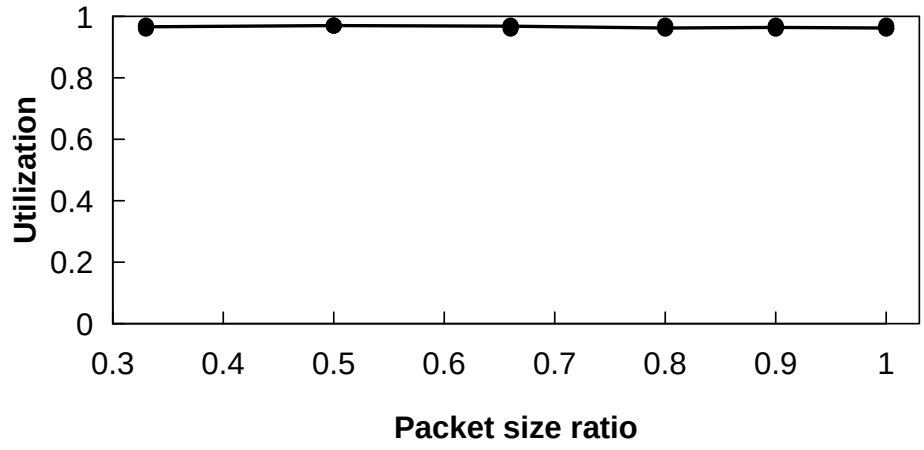


(c)

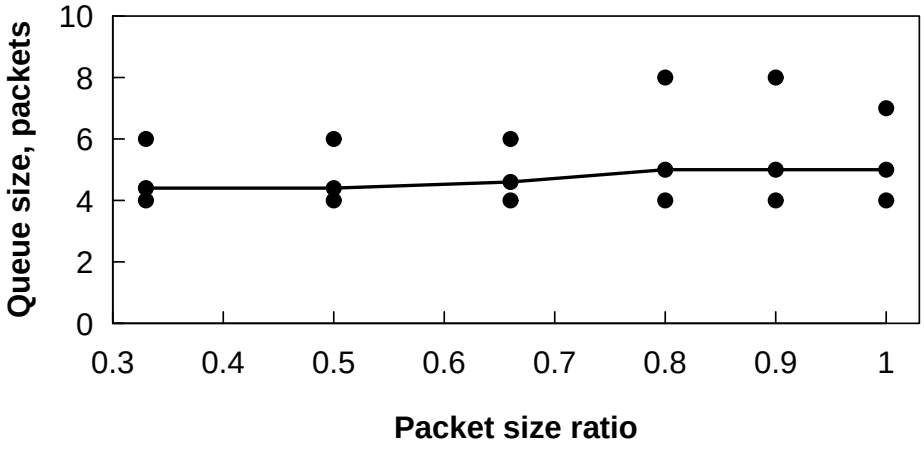
Figure 3.7: Independence of MCP stable-state operation from RTT: (a) Fairness index; (b) Utilization of the bottleneck link; (c) Peak queue size at the bottleneck link.



(a)



(b)



(c)

Figure 3.8: Independence of MCP stable-state operation from packet sizes: (a) Fairness index, (b) utilization of the bottleneck link; (c) Peak queue size at the bottleneck link.

3.3.5 Comparison with TCP and VCP

First, this section compares MCP, TCP, and VCP in terms of queuing at the bottleneck link. In each simulation of this series, the single-bottleneck dumbbell serves five flows that have propagation RTT of 20, 40, 60, 80, and 100 ms. The bottleneck link has capacity 20 Mbps and buffer for 50 packets of size 1000 bytes. The flows arrive 1 sec after each other. The simulation duration is 200 sec. The experiment is repeated for each of the three protocols. Figure 3.9 plots the queue size at the bottleneck link for two time scales: the whole experiment duration and zoomed interval between 90 and 95 sec. The graphs show that MCP significantly subdues the queuing: the peak queue size under TCP is 50 packets, under VCP is 20 packets, and under MCP is 6 packets. In the stable state of the above experiments, the bottleneck link utilization is 0.92 under MCP and oscillates between 0.89 and 1 under VCP.

Besides, an additional comparative evaluation of MCP and VCP is conducted in the single-bottleneck dumbbell topology where the bottleneck link capacity is 500 Mbps, and the capacities of the access links are 1 Gbps. In both experiments which last 50 sec, 1000 flows with propagation RTT 60 ms arrive at randomized moments between 0 and 1 sec. Figure 3.10 traces the bottleneck link utilization and queue size under each protocol. The peak queue size under MCP is 31% of the buffer size and close to the 39% under VCP. However, while the peak queue size under VCP stays at the same level in the stable state, MCP reduces its peak queue size in the stable state dramatically to 1.7% of the buffer size.

3.3.6 Behavior in the Complex Topology

To investigate MCP operation in scenarios with multiple and migrating bottleneck links, this section conducts an experiment in the parking-lot topology shown in Figure 3.1. The simulation lasts 600 sec and involves 4 flows with propagation RTT of 80, 60, 40, and 40 ms: flow f_1 arrives at time 0 and traverses path $aABCDd$, flow f_2 arrives at time 100 sec and traverses path $eABCDh$, flow f_3 arrives at time 300 sec and traverses path $bBCc$, and flow f_4 arrives at time 400 sec and traverses path $fBCg$. Flows f_1 and f_2 are bottlenecked at link AB before time 300 sec but the bottleneck migrates to link BC when flow f_3 arrives. Figure 3.11 shows that despite the migration of the bottleneck link, MCP always enables all the flows to converge to fair transmission rates.

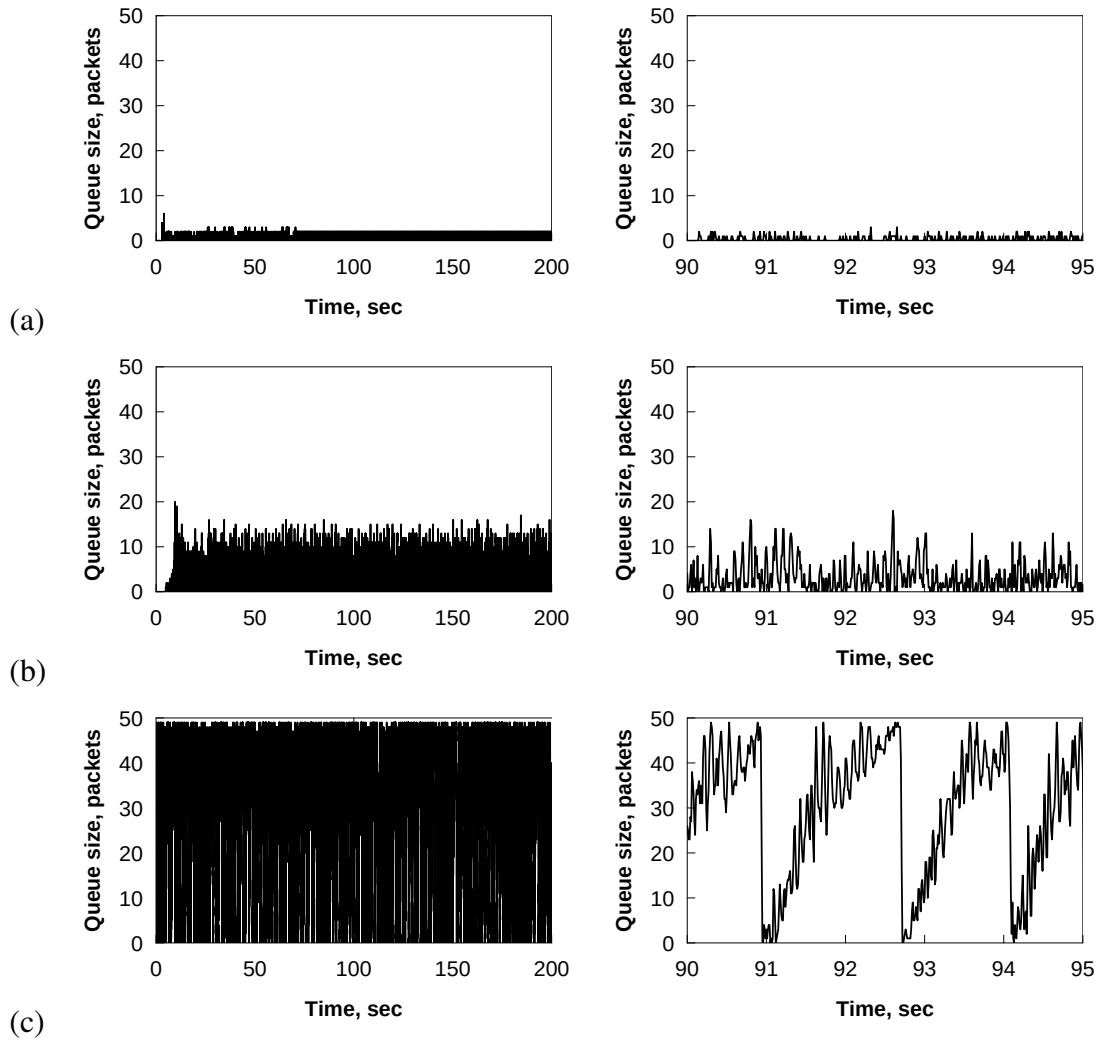
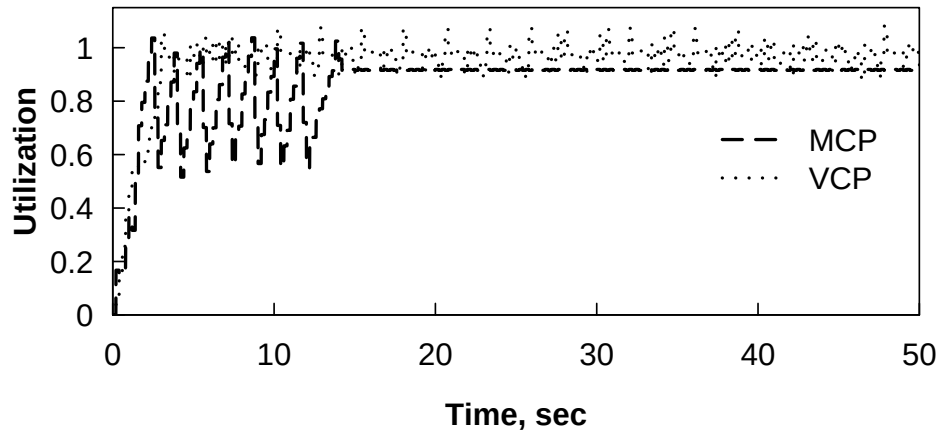
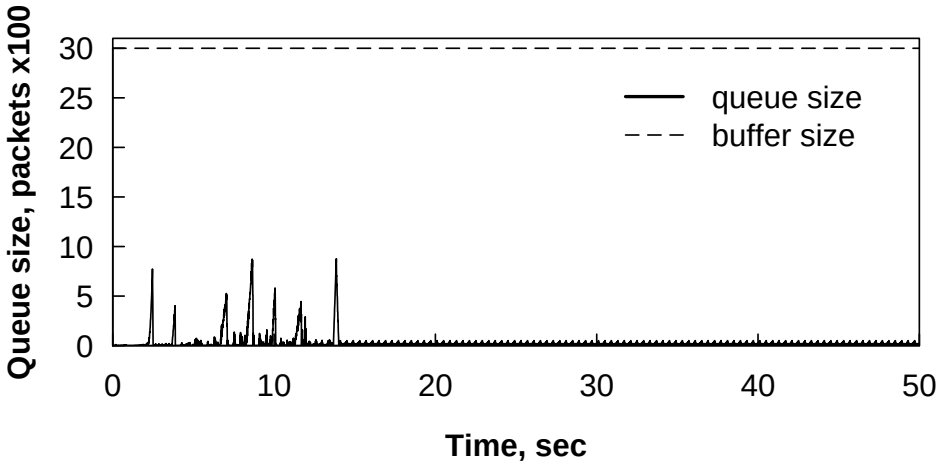


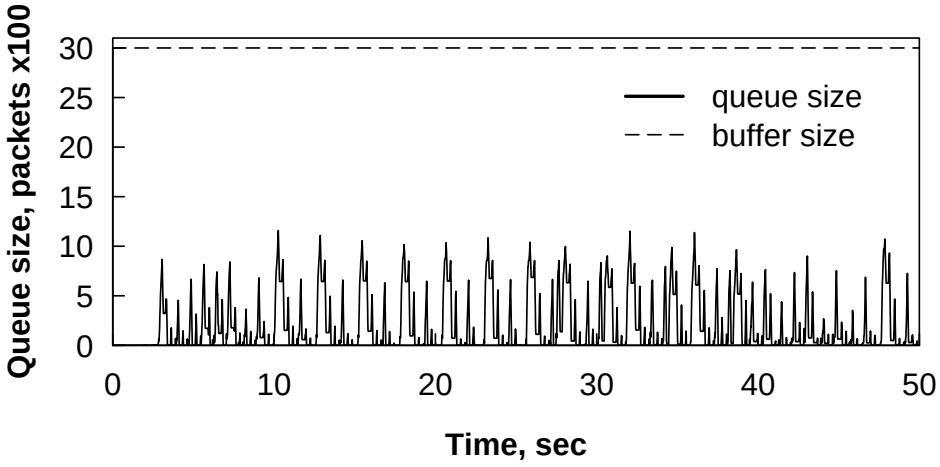
Figure 3.9: Queuing at the bottleneck link: (a) under MCP, (b) under VCP, (c) under TCP.



(a)



(b)



(c)

Figure 3.10: MCP versus VCP: (a) utilization of the bottleneck link, (b) queue size at the bottleneck link under MCP, and (c) queue size at the bottleneck link under VCP.

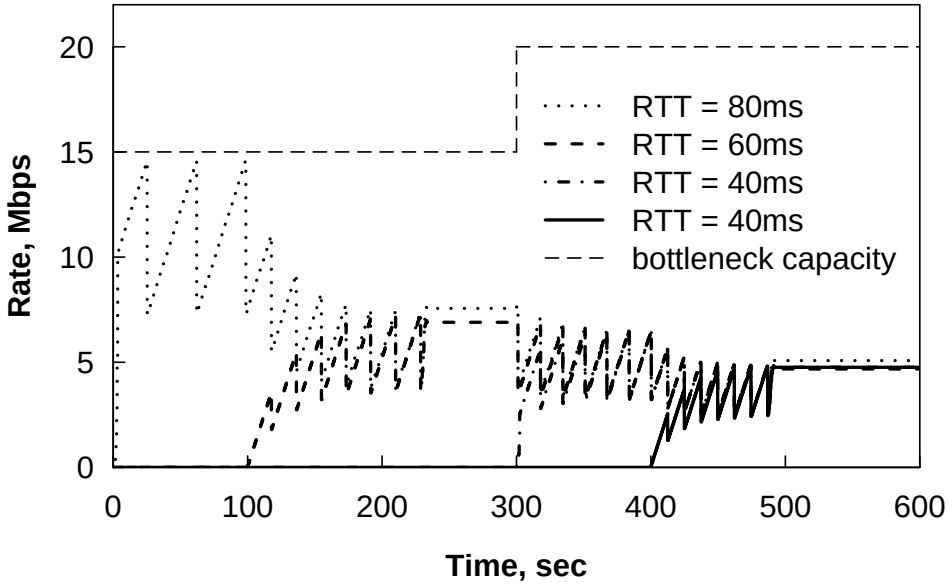


Figure 3.11: Transmission rates of MCP flows converge to fair rates after the bottleneck link migrates.

3.4 Summary

This Chapter presented the design and evaluation of MCP, a congestion control protocol for low stable-state queuing at bottleneck links. To achieve its design objectives, the protocol engages hosts and routers in limited explicit communication and exploits the insight that the transmission should be kept constant after converging to fair efficient rates. For convergence to fair transmission rates, MCP incorporates a novel explicit-communication mechanism that allocates fairing and this-path bits in the header of each data packet. These two bits enable the sender of a flow (e.g., of a new flow) to urge all flows sharing its bottleneck links to operate in the fairing mode for seven increase-decrease cycles of AIMD(80 kbps; 0.5) control. The analysis shows that the seven-cycle longevity of the fairing mode is sufficient to provide high levels of fair sharing. The evaluation of MCP and its comparison with TCP and VCP show that, by and large, MCP meets its design objectives.

Chapter 4

RD Network Services: An Architecture for Service Differentiation

With the Internet offering a single best-effort service, there have been numerous proposals of diversified network services that align better with the divergent needs of different distributed applications. The failure of these innovative architectures to gain wide deployment is primarily due to economic and legacy issues, rather than technical shortcomings. This Chapter proposes a new paradigm for network service differentiation where design principles account explicitly for the multiplicity of Internet service providers and users as well as their economic interests in environments with partly deployed new services. The key idea is to base the service differentiation on performance itself, rather than price. The proposed RD (Rate-Delay) services enable a user to choose between a higher transmission rate or low queuing delay at a congested network link. An RD router supports the two services by maintaining two queues per output link and achieves the intended rate-delay differentiation through simple link scheduling and dynamic buffer sizing. The conducted extensive simulations confirm effectiveness of the RD services geared for incremental deployment in the Internet.

4.1 Model and Principles

This chapter models the Internet as an interconnection of network domains owned and operated by numerous independent ISPs. ISPs generate revenue by selling network services to their direct customers. Users are the customers whose applications run at end hosts and send flows of packets over the Internet. In general, a network path that connects the end hosts of a distributed application traverses an infrastructure that belongs to multiple ISPs.

While different applications have different communication needs, the single best-effort service of the current Internet matches the interests of the users imperfectly. In response to this tension, various architectures with diversified network services have been proposed. Although technically brilliant, even the best of the proposals failed to gain wide deployment. This dissertation attributes the failures to ignoring the serious economic challenges of deploying a new service in a confederated infrastructure governed by numerous independent stakeholders. Instead of treating the deployment as an afterthought, this dissertation bases the design on principles that explicitly acknowledge the multiplicity of Internet parties and their economic rationale in deciding whether to adopt new services.

First, it is explicitly recognized that partial deployment is an unavoidable and potentially long-term condition for any newly adopted service. Hence, the new design should be attractive for early adopters even if other ISPs or users refuse to embrace the innovation:

Principle 1 *A new service should incorporate incentives for both ISPs and end users to adopt the service despite the continued presence of legacy traffic or other ISPs that do not espouse the new service.*

The above principle has a more specific but nevertheless important implication that the new design should not worsen the service provided to legacy Internet users. Doing otherwise is against the economic interests of ISPs due to the danger of losing a large number of current customers who keep communicating via legacy technologies. This consideration leads to the following principle:

Principle 2 *Adoption of a new service should not penalize legacy traffic.*

4.2 Conceptual Design

Below, the principles from Section 4.1 are applied to derive a conceptual design for Rate-Delay (RD) services, the solution to the problem of network service differentiation. As the name reflects, the RD services enable a user to choose between a higher transmission rate or low queuing delay at a congested network link.

Principle 1 prescribes providing both end users and ISPs with incentives for early adoption of the RD services. The constraint of the partial deployment excludes the common approach of pricing and billing, e.g., because a user should be able to opt for the RD services despite accessing the Internet through a legacy ISP that provides no billing or any other support for service differentiation. With direct financial incentives not being an option, the key idea of this chapter is to make the performance itself a cornerstone of the service differentiation. While the performance is subject to a fundamental trade-off between link utilization and queuing delay [44, 90], different applications desire different resolutions to the tension between the two components of the performance. Hence, the RD services consist of two classes:

- an R (Rate) service puts an emphasis on a high transmission rate;
- a D (Delay) service supports low queuing delay.

Each of the two services is neither better nor worse per se but is merely different, and its relative utility for a user is determined by whether the user's application favors a high rate or low delay. Since the network services are aligned with the application needs, each user receives an incentive to select the service of the most appropriate type, and the RD service architecture empowers the user to do such selection by marking the headers of transmitted packets.

An ISP finds the RD services attractive due to the potential to boost revenue by adding customers who are interested in the D service. This dissertation envisages an RD certification program championed by a nucleus of early adopters. The RD certification will serve as a catalyst for virulent deployment of the RD architecture because being RD-certified will give an ISP a differentiation advantage over legacy ISPs when competing with them for users and provider peering agreements.

To support the RD services on an output link, the router maintains two queues for packets destined to the link. This chapter refers to the queues as an R queue and D queue. Depending on whether an incoming packet is marked for the R or D service, the router appends the packet to the R or D queue respectively. The packets within each queue are served in the FIFO (First-In First-Out) order. Whenever there is data queued for transmission, the router keeps the link busy, i.e., the RD services are work-conserving.

By deciding whether the next packet is transmitted from the R or D queue, the router realizes the intended rate differentiation between the R and D services. In particular, the

link capacity is allocated to maintain a rate ratio of

$$k = \frac{r_R}{r_D} > 1 \tag{4.1}$$

where r_R and r_D refer to per-flow forwarding rates for packet flows from the R and D class respectively.

The router supports the desired delay differentiation between the R and D services through buffer sizing for the R and D queues. As common in current Internet routers, the size of the R buffer is chosen large enough so that the oscillating transmission of TCP (Transmission Control Protocol) [52] and other legacy end-to-end congestion control protocols utilizes the available link rate fully. The D buffer is configured to a much smaller dynamic size to ensure that queuing delay for each forwarded packet of the D class is low and at most d. The assurance of low maximum queuing delay is attractive for delay-sensitive applications and easily verifiable by outside parties. An interesting direction for future studies is an alternative design for the D service where queuing delay stays low on average but is allowed to spike occasionally in order to support a smaller loss rate.

In agreement with the overall design philosophy, parameters k and d are independently determined by the ISP that owns the router. The ISP uses the parameters as explicit levers over the provided RD services. The subsequent experimental study reveals suggested values for parameters k and d.

As per Principle 2, adoption of the RD services by an ISP should not penalize traffic from legacy end hosts. While the R service and legacy Internet service are similar in putting the emphasis on a high transmission rate rather than low queuing delay, the legacy traffic and any other packets that do not explicitly identify themselves as belonging to the D class are treated by an RD router as belonging to the R class, i.e., the router diverts such traffic into the R queue. Since those flows that opt for the D service acquire the low queuing delay by releasing some fraction of the link capacity, the adopters of the D service also benefit the legacy flows by enabling them to communicate at higher rates.

Due to the potentially partial deployment of the RD services, R and D flows might be bottlenecked at a link belonging to a legacy ISP. Furthermore, the R and D flows might share the bottleneck link with legacy traffic. This has an important design implication that end-to-end transmission control protocols for the R and D services have to be compatible with TCP.

4.3 Analytical Foundation

While Section 4.2 outlined the conceptual design of the RD services, this section now presents an analytical foundation for the specific implementation of the RD routers.

4.3.1 Notation and Assumptions

Consider an output link of an RD router. Let C denote the link capacity and n be the number of flows traversing the link. This section uses n_R and n_D to represent the number of flows from the R and D class respectively. Since the router treats legacy traffic as belonging to the R class, we infer that:

$$n_R + n_D = n. \tag{4.2}$$

For analytical purposes, it is assumed that both R and D queues are continuously backlogged and, hence:

$$R_R + R_D = C \tag{4.3}$$

where R_R and R_D refer to the service rates for the R and D queues respectively. Also, the analysis assumes that every flow within each class transmits at its respective fair rate, r_R or r_D:

$$R_R = n_R r_R \text{ and } R_D = n_D r_D. \tag{4.4}$$

The experiments in this chapter with dynamic realistic traffic including a lot of short-lived flows confirm that the above assumptions do not undermine the intended effectiveness of the RD services in practice.

The sizes of the R and D queues are denoted as q_R and q_D respectively and the buffer allocations for the queues as B_R and B_D respectively. If the corresponding buffer does not have enough free space for an arriving packet, the router discards the packet.

4.3.2 Sizing and Serving the R and D Queues

Combining equations (4.1), (4.3), and 4.4, we determine that the service rates for the R and D queues should be respectively equal to

$$R_R = \frac{kn_R C}{n_D + kn_R} \text{ and } R_D = \frac{n_D C}{n_D + kn_R}. \tag{4.5}$$

To ensure that queuing delay for any packet forwarded from the D queue does not exceed d, the buffer allocation for the queue should be bounded from above as follows:

$$B_D \leq R_D d. \tag{4.6}$$

Taking the second one of equations (4.5) into account, we establish the following buffer allocation for the D queue:

$$B_D = \frac{n_D C d}{n_D + k n_R}. \tag{4.7}$$

In practice, it is expected B_D to be much smaller than overall buffer B that the router has for the link. Manufacturers equip current Internet routers with substantial memory so that router operators could configure the link buffer to a high value B_{max}, chosen to support throughput-greedy TCP traffic effectively [103]. Thus, it is recommended to allocate the buffer for the R queue to the smallest of $B - B_D$ and B_{max} (and expect B_{max} to be the common setting in practice):

$$B_R = \min \left\{ B_{max};\ B - \frac{n_D C d}{n_D + k n_R} \right\}. \tag{4.8}$$

4.4 Design Details

The description of the design of the RD services is split into two parts. First, the section talks about the end hosts, and, second, it specifies the details of the routers.

4.4.1 End Hosts

As per the discussion at the end of Section 4.2, the RD services restrict end-to-end transmission control protocols to being compatible with TCP. The only extra support required from end hosts is the ability to mark a transmitted packet as belonging to the D class. This requirement is implemented by employing bits 3-6 in the TOS (Type of Service) field of the IP (Internet Protocol) datagram header [85]. To choose the D service, the bits are set to 1001. The default value of 0000 corresponds to the R service. Thus, the RD services preserve the IP datagram format.

Variable	Semantics
x	class of the service, R or D
n_x	number of flows from the x class
B_x	buffer allocation for the x queue
q_x	size of the x queue
L_x	amount of data transmitted from the x queue since the last reset of L_x
p	packet
t_p	arrival time of p
S	packet size

Table 4.1: Internal variables of the RD router algorithms in Figures 4.1, 4.2, and 4.3.

Parameter	Semantics
k	ratio of per-flow rates for R and D flows
d	upper limit on queuing delay of D packets
b	timestamp vector size
T	update period
E	flow expiration period

Table 4.2: Parameters of the RD router algorithms.

4.4.2 Routers

The main challenge for transforming the analytical insights of Section 4.3.2 into specific algorithms for RD router operation lies in the dynamic nature of Internet traffic. In particular, while equations (4.5), (4.7), and (4.8) depend on n_R and n_D, the numbers of R and D flows change over time. Hence, the RD router periodically updates its values of n_R and n_D. In the next three subsequent sections we describe the algorithms for processing a packet arrival, serving the queues, and updating the algorithmic variables at the RD router respectively. Table 4.1 summarizes the internal variables of the algorithms. In addition to the internal variables, a number of parameters characterize the RD router operation. Table 4.2 sums up these parameters.

```
p ← received packet;
x ← class of p;
S ← size of p;
if q_x + S ≤ B_x
      append p to the tail of the x queue;
      q_x ← q_x + S;
      if x = D
        t_p ← current time;
 else
      discard p
```

Figure 4.1: Router operation upon receiving a packet destined to the RD link.

Handling a packet arrival

Figure 4.1 presents the simple algorithm for dealing with packet arrivals. When the router receives a packet destined to the link, the router examines the seventh TOS bit in the packet header to determine whether the packet belongs to the R or D class. If the corresponding buffer is already full, the router discards the packet. Otherwise, the router appends the packet to the tail of the corresponding queue. Besides, if the enqueued packet belongs to the D class, the router remembers the arrival time of the packet until the packet reaches the head of the queue. Since the D buffer is typically small, storing the arrival times does not require significant memory.

Serving the R and D queues

The arrival times of enqueued D packets are used by the algorithm that serves the queues. The algorithm uses the times to ensure that queuing delay of forwarded D packets does not exceed upper bound d. More specifically, if the packet at the head of the D queue has been queued for longer, the router discards the packet. The situation might arise due to the dynamic nature of Internet traffic: since the population of flows changes, the service rate for the D queue might decrease after the packet arrives. The preliminary version of this algorithm did not include the last-moment enforcement of the queuing delay constraint. Experimental results for the preliminary version were similar to those reported in Section 4.6: while the loss rates did not differ much, the maximum observed queuing delay exceeded d by about 0.5 ms. It remains to be seen whether the strict enforcement of the

```
\* select the queue to transmit from *\
if q_R > 0 and q_D > 0
    if k n_R L_D > n_D L_R
        x ← R;
    else
        x ← D;
else \* exactly one of the R and D buffers is empty *\
    x ← class of the non-empty buffer;
p ← first packet in the x queue;
S ← size of p;
if x = D
    \* enforce the delay constraint of the D service *\
    while current time - t_p > d and q_D > 0
        discard p;
        q_D ← q_D − S;
        p ← first packet in the D queue;
        S ← size of p;
if p != null
    \* update the L variables *\
    if q_R > 0 and q_D > 0
        L_x ← L_x + S;
    else \* one of the R and D buffers is empty *\
        L_R ← 0; L_D ← 0;
    transmit p into the link;
    q_x ← q_x − S
```

Figure 4.2: Router operation when the RD link is idle, and the link buffer is non-empty.

queuing delay constraint is worth the price of tracking the arrival times of the enqueued D packets.

Figure 4.2 reports further details of the algorithm for serving the R and D queues. While the RD services are work-conserving, the router transmits into the link whenever the link buffer is non-empty. Since the router can transmit at most one packet at a time, the intended split of link capacity C into service rates R_R and R_D can be only approximated. The router does so by:

- monitoring L_R and L_D, the amounts of data transmitted from the R and D queues respectively since the last reset of these variables;
- transmitting from such queue that $\frac{L_R}{L_D}$ approximates $\frac{R_R}{R_D} = \frac{kn_R}{n_D}$ most closely.

More specifically, when $kn_R L_D > n_D L_R$, the router transmits from the R queue; otherwise, the router selects the D queue.

This chapter derived the above algorithm from the assumption that all flows within a class transmit at the same fair rate, r_R or r_D. While the assumption is clearly unrealistic, one specific problematic scenario occurs when the total transmission rate of the D flows is much less than $n_D r_D$, the maximum service rate for the D queue. Then, a throughput-greedy flow has an incentive to mark its packets as D packets and thereby achieve a much higher forwarding rate than the one offered by the intended R service. Although this scenario has not surfaced in the extensive simulations, and the unintended selection of the D service by the throughput-greedy flow does not disrupt the D service, this issue deserves close consideration. The future study will explore in detail the implications of the diversity in flow rates and user behaviors (including deliberate denial-of-service attacks) for the RD services.

Updating the algorithmic variables

Whereas n_R and n_D play important roles in the presented RD router algorithms, two approaches to computing the numbers of flows are compared: explicit notification from end hosts and independent inference by the router. Since the design principles allow a possibility that many users do not embrace the RD services, it is likely that the router serves many legacy flows and needs to do at least some implicit inference. Furthermore, since this chapter favors solutions with minimal modification of the current infrastructure, the router in the RD implementation estimates n_R and n_D without any help from end hosts.

To estimate the numbers of flows, the timestamp-vector algorithm [64] is applied separately to the R and D classes. The experiments confirm the excellent performance of the algorithm. Using a hash function, the algorithm maps each received packet into an element of the array called a timestamp vector. The timestamp vector accommodates b elements. The algorithm inspects the timestamp vector with period T and considers a flow inactive if the timestamp vector does not register any packets of the flow during last period E. Following the guidelines in [28] and assuming $E = 1$ s, 10^5 active flows, and standard deviation $\epsilon = 0.05$, we recommend $b = 18{,}000$ as the default setting for the timestamp vector size.

The RD router updates n_R and n_D with period T. At the same time, the router updates the buffer allocations for the R and D queues. Even if n_R or n_D is zero, the router allocates a

```
update n_R and n_D as per Section 4.4.2;
update B_R and B_D as per Section 4.4.2;
L_R ← 0; L_D ← 0;
if q_D > B_D
  discard all packets from the D queue;
  q_D ← 0;
else while q_R > B_R
        p ← last packet in the R queue;
        S ← size of p;
        discard p;
        q_R ← q_R − S
```

Figure 4.3: Update of the RD algorithmic variables upon timeout.

non-zero buffer for each of the queues. The experimental results suggest that the specific allocation split is not too important; in the reported experiments, the buffer allocations are initialized to $B_D = \frac{4Cd}{4+k}$ and $B_R = \min\{B_{max};\ B - B_D\}$, which correspond to the 1:4 ratio between the numbers of flows from the R and D classes. If both n_R and n_D are positive, the router updates the buffer allocations according to equations (4.7) and (4.8).

The update of B_R and B_D can make one of them smaller than the corresponding queue size. Figure 4.3 describes how the router deals with this issue. If the updated B_R is less than q_R, the router discards packets from the tail of the R queue until q_R becomes at most B_R. The discards ensure that the D service receives the intended buffer allocation. If B_D is decreased below q_D, the router flushes all packets from the D queue. Emptying the D buffer assures that neither of the packets will be queued for longer than d and thus need to be discarded after reaching the head of the queue. The longer queueing might occur otherwise because the decrease of B_D also proportionally reduces the service rate for the D queue. Although the D buffer is typically small, discarding the burst of packets might affect the loss rate negatively and be even unnecessary because it might be still possible to forward at least some of the discarded D packets in time despite the reduced service rate. While the experiments show acceptably low loss rates with this implementation of the algorithm, more subtle discard policies will be explored in the future work.

To select update period T, it is observed that reducing T increases the computational overhead. Also, the operation might become unstable unless T is much larger than d. However, with larger T, the design responds slower to changes in the network conditions. The experiments show that $T = 400$ ms offers a reasonable trade-off between these factors.

4.5 Stateless RD Router Implementation

Maintaining the per-packet state to track the packet arrival times constitutes a weakness of the original RD design because the added memory and processing requirements can deteriorate the router performance, make the router hardware more complex, and increase the router cost, e.g., due to relying on the expensive Static Random Access Memory (SRAM) [53]. This section proposes an alternative S-RD (Stateless Rate-Delay) router design that guarantees low queuing delay of the D service without keeping any per-packet state. The S-RD algorithm achieves this objective by configuring the D buffer to a smaller size.

4.5.1 Sizing the R and D Queues

The service rates for the R and D queues, i.e. R_R and R_D, are calculated using the same control equations (4.5) as for the RD services. To ensure that queueing delay does not exceed d, the size of the D buffer should be bounded as follows:

$$B_D = \lfloor R_D(d - w) \rfloor^+ \tag{4.9}$$

where:

$$w = \frac{2}{C}\Big(\frac{S_D^{max}}{\alpha} + S_R^{max}\Big). \tag{4.10}$$

Section 4.5.3 shows that configuring the buffer size of queue D through equations (4.9) and (4.10) guarantees the strict support of the delay constraint.

4.5.2 Serving the R and D Queues

Whereas the original RD algorithm handles the potential danger of exceeding the delay constraint if $q_D > B_D$ at the moment of the recalculating of control parameters, the RD buffer sizing misses one case of exceeding the delay constraint. In particular, this happens if $B_D^{old} > B_D$, where B_D^{old} is the previous value of the size of the D buffer, as draining rate of the D queue after recalculation decreases. The S-RD algorithm handles this through flushing the whole D queue.

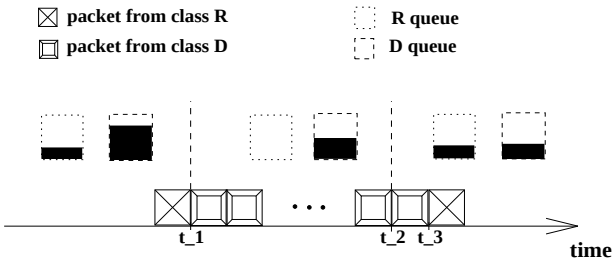


Figure 4.4: The exceeding of the delay constraint: combination of both and one queue backlogged.

To avoid overflow of the values L_D and L_R, they are periodically assigned to zero values. In particular, the assignment happens in two cases. The first one occurs if only one queue is backlogged. In this case, both L_D and L_R are zeros until both queues get backlogged again. The second one happens upon a timeout for recalculating the control parameters. The problem is that exceeding the delay constraint can occur right after the assignment of zero values to L_D and L_R, as it is discussed below.

Figure 4.4 illustrates the first case corresponding to only one backlogged queue. This section will refer to the value of the ratio between $L_D(t)$ and $L_R(t)$ as to $\beta(t)$. During time interval $[t_1 - \delta t; t_1)$, $\delta t > 0$, both queues R and D are backlogged. Then only one queue D gets backlogged during time interval $[t_1; t_2)$. Finally, both queues are backlogged again starting from time t_2. Let us denote the amount of traffic sent during time interval $[t_1; t_3]$ as $\delta L'_D$. Besides, let us assume that traffic is such that $\beta(t_1) < \alpha$, where α is defined as follows:

$$\alpha = \frac{n_D}{k n_R}. \tag{4.11}$$

Then if the following inequality:

$$\frac{L_D(t_1) + \delta L'_D}{L_R(t_1)} < \alpha \tag{4.12}$$

takes place, then it is possible to exceed the delay constraint for D packets that arrive after time t_3. The reason is that R_D is less than the required rate during time interval $[t_1; t_3)$, and that is "forgotten" by the algorithm as L_D and L_R are assigned to zero values at time t_1. Applying similar speculations, one can demonstrate exceeding the delay constraint in the second case related to the timeout expiration. To handle the described problems, we use a special parameter δL that tracks the amount of traffic that needs to depart from the D queue to avoid exceeding the delay constraint. Figures 4.5 and 4.6 give the detailed specification of the implementation of the S-RD router.

```
\* select the queue to transmit from *\
if q_R > 0 and q_D > 0
    if kn_R L_D > n_D L_R
        x ← R;
    else
        x ← D;
else \* exactly one of the R and D buffers is empty *\
    x ← class of the non-empty buffer;
p ← first packet in the x queue;
S ← size of p;
if p != null
    \* update the L variables *\
    if q_R > 0 and q_D > 0
        L_x ← L_x + S;
        δL ← (L_R n_D)/(k n_R) − L_D;
        if δL < 0 δL ← 0;
    else \* only D buffer is empty *\
    if q_R > 0 and q_D = 0
        L_R ← 0; L_D ← 0;
    else \* only R buffer is empty *\;
        if δL > 0 δL ← δL − S;
        if δL > 0 L_D ← −δL;
        else
            L_D ← 0;
        L_R ← 0;
    transmit p into the link;
    q_x ← q_x − S
```

Figure 4.5: Router operation when the S-RD link is idle, and the link buffer is non-empty.

4.5.3 Analysis of the S-RD Design

This section shows that the S-RD router design ensures the strict delay constraint without tracking packet arrival times or discarding packets at the head of the D queue.

We distinguish two different cases of backlogging at an S-RD link depending on the number of backlogged queues. Section 4.5.2 has already considered the first case characterized by one backlogged queue. We now examine the case when both the D and R queues are backlogged. Let us consider an arbitrary packet p from the D queue. We assume that p arrives at the D queue at time t_a and departs from the D queue at time t_d. Let us suppose

$B_D^{old} \leftarrow B_D$;
update n_R and n_D as in [64];
update B_R and B_D according to (4.8),(4.7);
if $\delta L > 0$ $L_D \leftarrow -\delta L$;
  else $L_D \leftarrow 0$;
$L_R \leftarrow 0$;
if $q_D > B_D$ **or** $B_D^{old} < B_D$
  discard all packets from the D queue;
  $q_D \leftarrow 0$;
else while $q_R > B_R$
    $p \leftarrow$ last packet in the R queue;
    $S \leftarrow$ size of p;
    discard p;
    $q_R \leftarrow q_R - S$

Figure 4.6: Update of the S-RD algorithmic variables upon timeout.

that:

$$\frac{L_D(t_a)}{L_R(t_a)} = \alpha + \delta(t_a), \tag{4.13}$$

$$\frac{L_D(t_d)}{L_R(t_d)} = \alpha + \delta(t_d), \tag{4.14}$$

where $L_R(t_a) > 0$ and $L_R(t_d) > 0$. We consider scenario where both the D and R queues are backlogged during time period $[t_a; t_d]$. We will refer to the traffic sent from the D and R queues during time period $[t_a; t_d]$ as δL_D and δL_R, $\delta L_R > 0$, respectively. We distinguish two cases of the S-RD buffer configuration. The first one corresponds to a buffer of zero size, and, therefore, gives no queuing delay. The second case reflects a non-zero buffer, i.e., $d - w > 0$, where d is the delay constraint, w is defined by equation (4.10). The analysis considers the second case. Next, we will prove the following:

Theorem 1 *Maximum queuing delay d-w is supported for any packet p within any traffic pattern if:*

$$\frac{\delta L_D}{\delta L_R} \geq \alpha. \tag{4.15}$$

Proof: Indeed, if inequality (4.15) takes place, then $\frac{R_D}{R_R} \geq \alpha$ during $[t_a; t_d]$. From $B_D = \alpha R_R(d - w)$ we conclude that the maximum packet delay does not exceed $d - w$. ■

As $\delta L_D = L_D(t_d) - L_D(t_a)$, $\delta L_R = L_R(t_d) - L_R(t_a)$, we rewrite inequality (4.15) as follows:

$$\frac{L_D(t_d) - L_D(t_a)}{L_R(t_d) - L_R(t_a)} \geq \alpha. \tag{4.16}$$

Let us denote the left part of inequality (4.16) as γ. Then, using equations (4.13) and (4.14) and performing simple transformation, we establish that:

$$\gamma = \alpha + \Big(\delta(t_d) + \frac{L_R(t_a)}{\delta L_R}(\delta(t_d) - \delta(t_a))\Big). \tag{4.17}$$

Therefore, inequality (4.15) takes place if and only if:

$$\delta(t_d) + \frac{L_R(t_a)}{\delta L_R}(\delta(t_d) - \delta(t_a)) \geq 0. \tag{4.18}$$

We now will prove the following:

Theorem 2 *$\frac{\delta L_D}{\delta L_R} \geq \alpha$ is supported for any traffic pattern and any packet p if and only if:*

$$\delta(t_a) \leq 0, \quad \delta(t_d) \geq 0. \tag{4.19}$$

Proof: First, we will prove that it is a sufficient condition. Indeed, if $\delta(t_a) \leq 0$, $\delta(t_d) \geq 0$, then inequality (4.18) is true for any values of $L_R(t_a)$ and δL_R, i.e., for any traffic pattern and any packet p.

Second, we will prove that it is a required condition. Let us suppose that it is not true. We need to consider all such possible cases:

Case 1: $\delta(t_a) \leq 0$, $\delta(t_d) < 0$. Then from inequality (4.18) we have that:

$$\frac{\delta L_R}{L_R(t_a)} + 1 \leq \frac{\delta(t_a)}{\delta(t_d)}. \tag{4.20}$$

As the left part of inequality (4.20) is larger than 1, and its right part can be smaller than 1, there is a contradiction.

Case 2: $\delta(t_a) > 0$, $\delta(t_d) \geq 0$. Then from inequality (4.18) we derive that:

$$\frac{\delta L_R}{L_R(t_a)} + 1 \geq \frac{\delta(t_a)}{\delta(t_d)}. \tag{4.21}$$

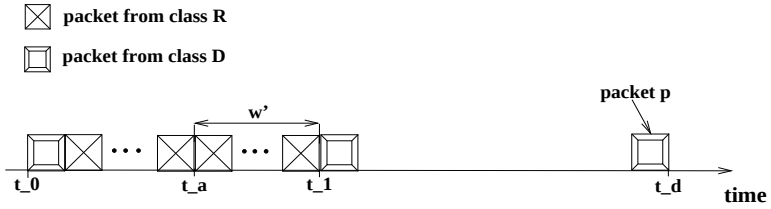


Figure 4.7: Schedule of packet departures when $\delta(t_a) > 0$, $\delta(t_d) \geq 0$.

As there exists a traffic pattern and packet p such that the left part of inequality (4.21) is smaller than 2, while the right part of inequality (4.21) is bigger than 2, there is a contradiction.

Case 3: $\delta(t_a) > 0$, $\delta(t_d) < 0$. Inequality (4.18) leads us to:

$$\frac{\delta L_R}{L_R(t_a)} + 1 \leq \frac{\delta(t_a)}{\delta(t_d)}. \tag{4.22}$$

As the left part of inequality (4.22) is bigger than 0, and its right part is smaller than 0, there is a contradiction.

Thus, we have shown that the made assumption is not true, which means than inequalities in (4.19) state a required condition. ■

From Theorem 1 and Theorem 2 we conclude that inequalities in (4.19) express a sufficient condition for supporting queueing delay of at most $d - w$ for any packet with an arbitrary traffic pattern at the S-RD link. Now let us consider packet p that fills up buffer of the D queue, i.e., the enqueing of that packet satisfies $q_D = B_D$.

Theorem 3 *Maximum queuing delay d-w is supported for p within any traffic pattern if and only if $\frac{\delta L_D}{\delta L_R} \geq \alpha$.*

Proof: The sufficiency follows from Theorem 1. This section will now prove the necessity. Indeed, if $\frac{\delta L_D}{\delta L_R} < \alpha$, then $\frac{R_D}{R_R} < \alpha$. From $B_D = \alpha R_R(d - w)$ we conclude that the maximum packet delay is bigger than $d - w$. ■

Now we consider all possible cases when the packet delay exceeds $d - w$ for such packet p.

Case 1: $\delta(t_a) > 0$, $\delta(t_d) \geq 0$. Figure 4.7 shows the schedule of packet departures in the considered case. According to Theorems 2 and 3, if packet p arrived at time t_1 and departed

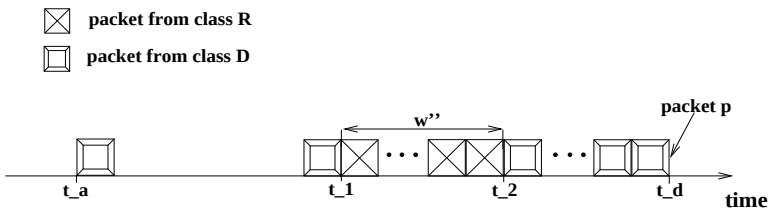


Figure 4.8: Schedule of packet departures when $\delta(t_a) \leq 0$, $\delta(t_d) < 0$.

at t_d then its queuing delay would not exceed $d - w$ as $\delta(t_1) < 0$, $\delta(t_d) \geq 0$. As in the interval $[t_a; t_1]$ there is no potential arrival time t'_a of packet p at which $\delta(t'_a) < 0$, queuing delay of packet p might be exceeded by the interval $[t_a; t_1]$. We will refer to the length of that interval as w', and to the amount of R traffic sent during this interval w' as X. Let us suppose that a D packet departing at time t_0 has size S_D, then the following inequalities take place:

$$L_D(t_0) \leq \alpha L_R(t_0), \tag{4.23}$$

$$L_D(t_0) + S_D > \alpha L_R(t_0), \tag{4.24}$$

$$L_D(t_0) + S_D \leq \alpha L_R(t_0) + X\alpha. \tag{4.25}$$

Lemma 1 *The worst-case solution to inequalities (4.23), (4.24), and (4.25) is at most:*

$$X = \frac{S_D^{max}}{\alpha} + S_R^{max}. \tag{4.26}$$

Proof: Indeed, X defined by equation (4.26) is a solution to the considered three inequalities. Next, we need to show that there is no smaller solution. Suppose that there exists X':

$$X' = X - \delta X, \tag{4.27}$$

where X is defined by equation (4.26), $\delta X > 0$, $\delta X < X$, δX is an integer, i.e., there exists δX such that X' satisfies inequalities (4.23), (4.24), and (4.25). Let us assume that the traffic scenario is such that inequality (4.23) becomes equality. Then from inequality (4.25) and $L_D(t_0) = \alpha L_R(t_0)$ we derive that:

$$S_D \leq S_D^{max} + \alpha S_R^{max} - \alpha \delta X. \tag{4.28}$$

Assuming that $S_D = S_D^{max}$ and $S_R^{max} < \delta X$, we conclude that inequality (4.28) is not valid. As it means that there exists a traffic scenario such that inequality (4.25) is not valid, there

is a contradiction. Finally, we mention that S_R^{max} in equation (4.26) reflects that traffic is in packets, i.e., not fluid. ■

From Lemma 1 we conclude that the maximum queuing delay in excess of $d - w$ in the considered case is as follows:

$$w' = \frac{1}{C}\Big(\frac{S_D^{max}}{\alpha} + S_R^{max}\Big). \tag{4.29}$$

Case 2: $\delta(t_a) \leq 0$, $\delta(t_d) < 0$. Figure 4.8 demonstrates how the packets are scheduled for this case. If during time interval $[t_1; t_2]$, instead of packets from class R, the link continued to serve the D queue up to packet p, then, according to Theorems 2 and 3, queuing delay of packet p would not exceed $d - w$ as $\delta(t_a) < 0$, $\delta(t'_d) \geq 0$, where t'_d would be its departure time. Therefore, queuing delay of packet p can exceed the delay constraint by the interval $[t_1; t_2]$. We will refer to the length of that interval as w'', and to the amount of D traffic sent during this time interval as Y. As in Case 1, Y is defined by the right part of equation (4.26). Therefore, w'' equals to w' defined by equation (4.29).

Case 3: $\delta(t_a) > 0$, $\delta(t_d) < 0$. As this case is a combination of the two previous ones, the maximum queuing delay in excess of $d - w$ is the sum of w' and w'':

$$w = \frac{2}{C}\Big(\frac{S_D^{max}}{\alpha} + S_R^{max}\Big). \tag{4.30}$$

As we have not used the information that p fills up the buffer of queue D while considering the three possible cases of the exceeding the $d - w$ delay, we have proved the following theorem:

Theorem 4 *Sizing the D buffer according to equations (4.9) and 4.10 ensures that the S-RD router algorithm supports maximum queuing delay* d.

4.6 Performance Evaluation of the RD Services

This section describes the performance evaluation of the RD Network Services. To evaluate the architecture, the networking simulator ns-2 [72], version 2.29, is used.

4.6.1 Experimental Settings

Unless explicitly stated otherwise, all flows employ TCP NewReno [33] and data packets of size 1 KB. Each link buffer is configured to $B = B_{max} = C \cdot 250$ ms, where C is the capacity of the link. Every experiment lasts 60 s and is repeated five times for each of the considered parameter settings. We calculate the average for each set of the parameters. The default settings include k = 2, d = 10 ms, b = 18,000, T = 400 ms, E = 1 s, T_{avg} = 200 ms, and T_q = 10 ms, where T_{avg} refers to the averaging interval for the bottleneck link utilization and loss rate, and T_q denotes the averaging interval for queuing delay. This section also averages the utilization and loss rate over the whole experiment with exclusion of its first five seconds.

In the experiments, we evaluate the original version of the RD services and stateless S-RD version. While queuing delay for the D service is at most d by the original design, all the experiments for the original version confirm that the maximum delay of D packets satisfies and closely approximates this upper limit. The utilization and loss rate characteristics are similar to the ones revealed by both the versions so for the S-RD version this section presents only the maximum queuing delay of D packets.

Simple Topology

To understand basic properties of the RD services, we use a traditional dumbbell topology. The experiments using such a topology include different transport protocols for D flows, both long-lived and short-lived traffic, diverse bottleneck link capacities, various settings for the delay constraint of the D service, Exponential and Pareto-distributed flow interarrival times, and sudden changes in the numbers of R and D flows. The core bottleneck and access links have capacities 100 Mbps and 200 Mbps, respectively. The bottleneck link carries 100 R flows and 100 D flows in both directions and has propagation delay 50 ms. The propagation delays for the access links are chosen so that propagation RTT (Round-Trip Time) for the flows is uniformly distributed between 104 ms and 300 ms. We will refer to the topology described in this section as a $simple\ topology$.

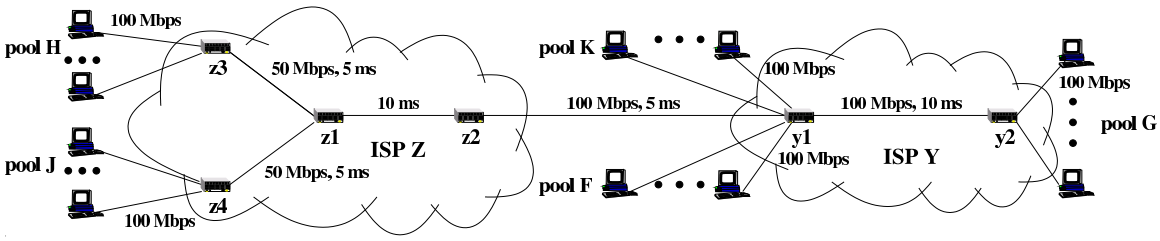


Figure 4.9: Shared settings of the multi-ISP topologies.

Complex Topology

The investigation of the RD services proceeds by examining their incremental deployability and other properties in topologies where multiple ISPs own the infrastructure. Figure 4.9 depicts the settings shared by the multi-ISP topologies, where both ISPs espouse the RD services. The network core belongs to ISP Z and ISP Y. Routers y1 and y2 of ISP Y offer the RD services with $k = 2$ and $d = 15$ ms. ISP Z configures all its four routers to offer the rate-delay differentiation with $k = 2$ and $d = 10$ ms. Backbone link z2-y1 connects the two ISPs and provides universal connectivity for all users. The users form five pools H, J, K, F, and G. Each user accesses his or her ISP through a personal link with capacity 100 Mbps. Every user from pools H, J, K, and F transmits a long-lived flow to a separate user in pool G. Hence, while the flows from K and F traverse the infrastructure that belongs only to ISP Y, both ISPs serve the flows from pools H and J. The propagation delays for the access links are chosen so that propagation RTT for the flows is uniformly distributed between 64 ms and M. In particular, propagation delay for both access links of each flow from pool H or J is chosen between 1 ms and $\frac{M}{4} - 15$ ms, and both access-link propagation delays for a flow from pool K or F are selected between 11 ms and $\frac{M}{4} - 5$ ms. The default setting for the maximum propagation RTT is $M = 300$ ms. The flows arrive according to a Poisson process. The average arrival rate is set by default to 100 flows per sec (fps) for creating a confident expectation that all the flows arrive before the measurement stage of the experiment. We will refer to the topology described in this section as a *complex topology*.

4.6.2 Various Transport Protocols for D Flows

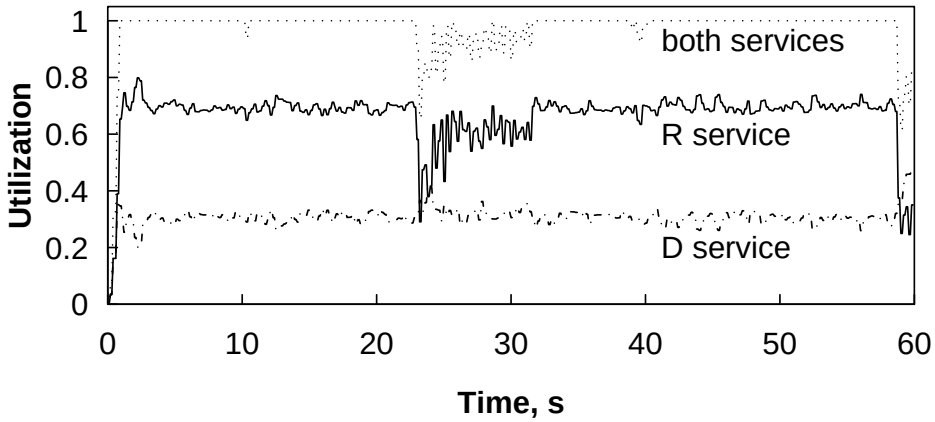
While the RD services restrict end-to-end transmission control to being compatible with TCP, this section illustrates how the RD design performs when the D flows employ TCP NewReno [33], Paced TCP [3, 20], or TFRC [32]. In this set of experiments, we use the

$simple\ topology$, and all flows stay throughout the experiment. With $k = 2$ and equal numbers of R and D flows, it is expected the R and D services to utilize the bottleneck link capacity fully with the 2:1 ratio. Figure 4.10 mostly confirms this expectation. In Figure 4.11, one can observe that the maximum queuing delay for R flows is about 375 ms, as expected for the link that allocates two thirds of its capacity C to the R flows and has the buffer sized to the product of C and 250 ms. Figure 4.12 demonstrates that the queuing delay for D flows fluctuates between 0 and d = 10 ms. Besides, the loss rate of R flows and D flows are shown in Figures 4.13 and 4.14, respectively. Due to slower detection of congestion and higher loss synchronization, Paced TCP yields larger losses than TCP NewReno. Among the three evaluated protocols, TFRC supports the smallest loss rate and most balanced rate differentiation. These superior properties make TFRC an attractive option for transmission control of D flows.

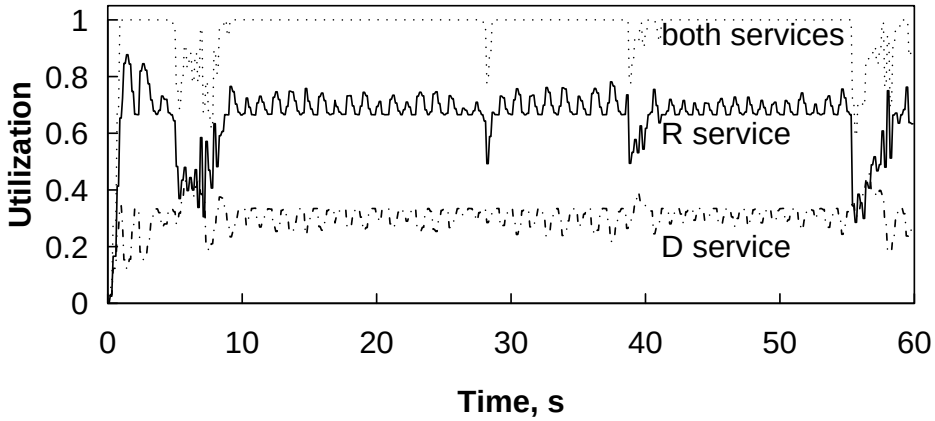
4.6.3 Scalability Properties

Influence of Link Capacity

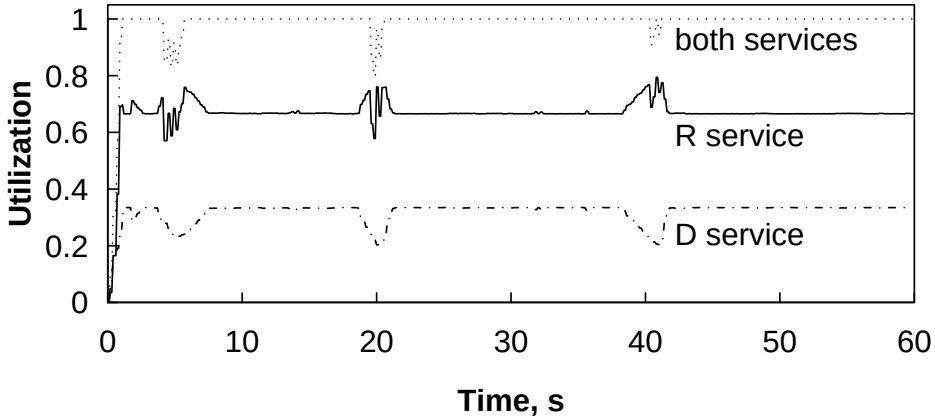
This series of experiments employs the $simple\ topology$ and varies the bottleneck link capacity from 1 Mbps to 1 Gbps while keeping the access link capacities twice as large. The traffic mix on the bottleneck link is enhanced with web-like flows from two sources: one source generates R flows, and the other transmits D flows. The sizes of the web-like flows are Pareto-distributed with the average of 30 packets and shape index of 1.3. The flows arrive according to a Poisson process. The average arrival rate for the web-like flows in this and next sections stays at 50 fps. Figure 4.15 shows that the rates of the R and D flows deviate from the intended 2:1 ratio significantly only for the lowest examined capacities close to 1 Mbps. The deviation occurs due to the extremely small buffering available for D packets in those settings. In particular, satisfying the 10-ms delay constraint at the 1-Mbps bottleneck link reduces the D buffer to about one packet, and the minimal buffering causes heavy losses and effectively shuts down the D service. As the bottleneck link capacity grows, the loss rate for the D flows decreases exponentially. Besides, Figure 4.15c confirms that there is no violation of the delay constraint for the whole range of the varied parameter if using the S-RD version.



(a)

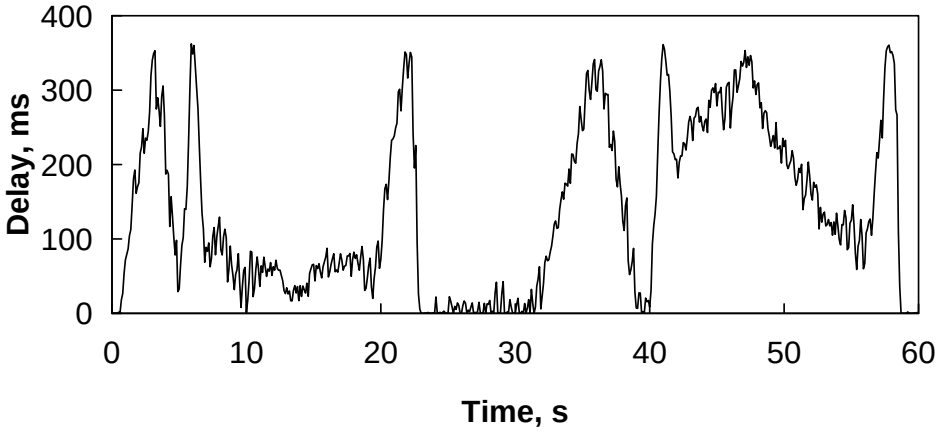


(b)

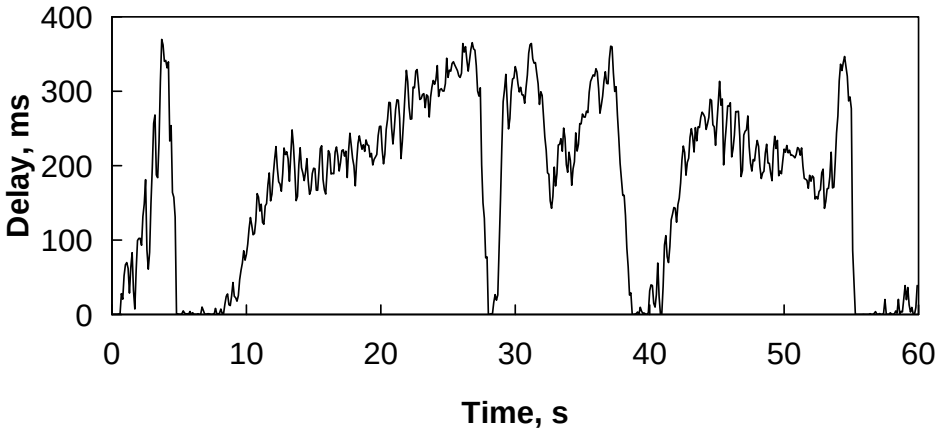


(c)

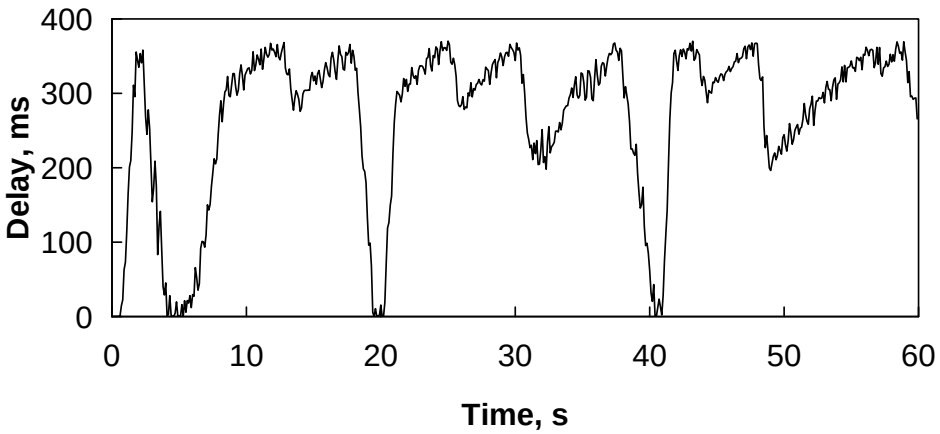
Figure 4.10: Bottleneck link utilization with different transport protocols for D flows: (a) TCP NewReno; (b) Paced TCP; (c) TFRC.



(a)

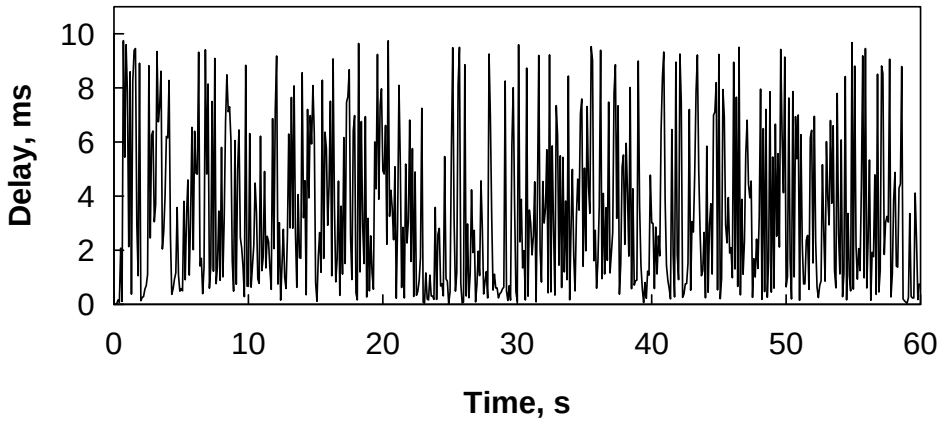


(b)

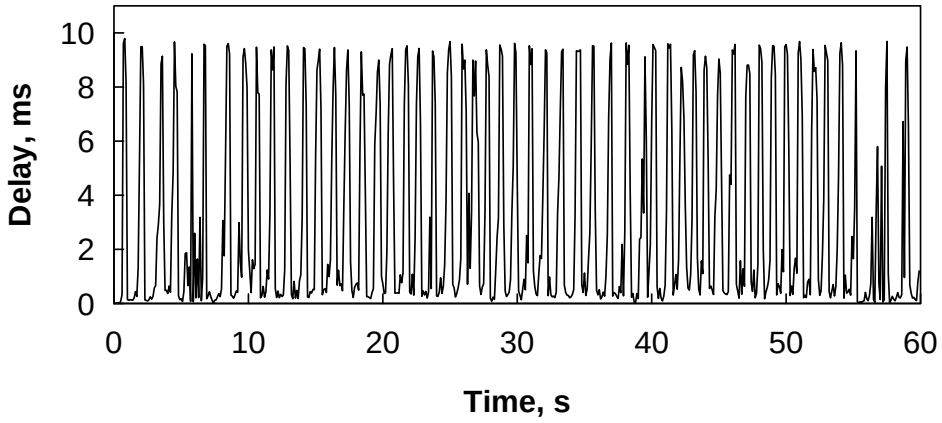


(c)

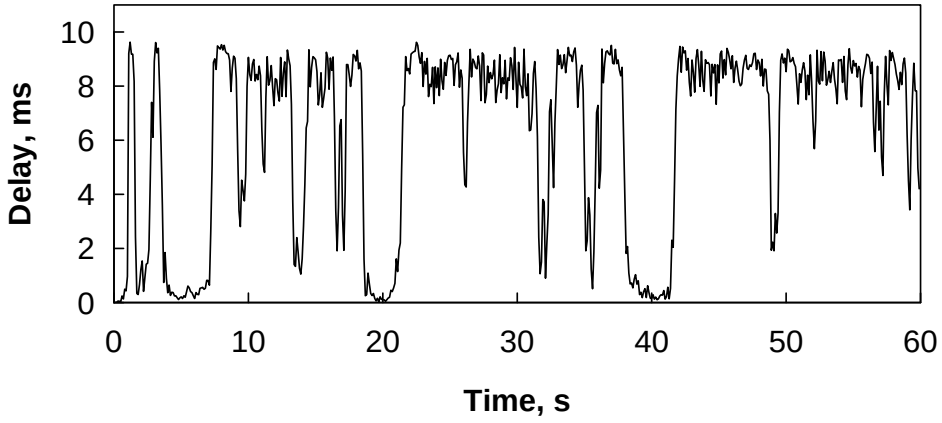
Figure 4.11: Queuing delay for R packets with different transport protocols for D flows: (a) TCP NewReno; (b) Paced TCP; (c) TFRC.



(a)

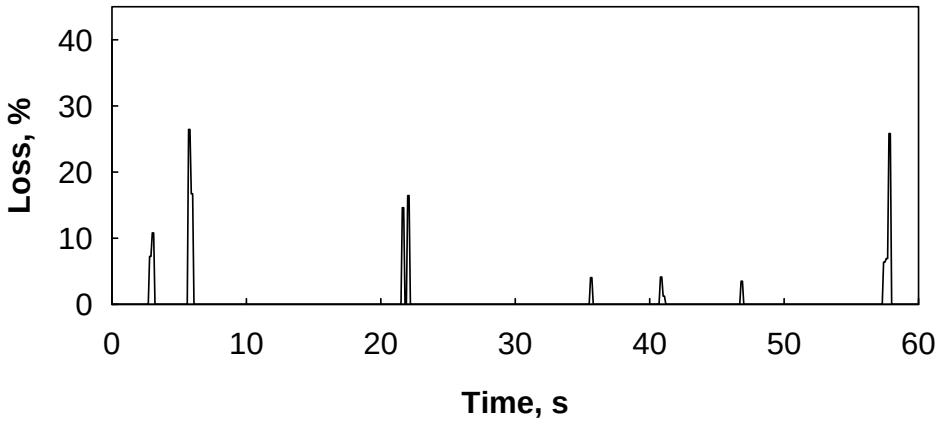


(b)

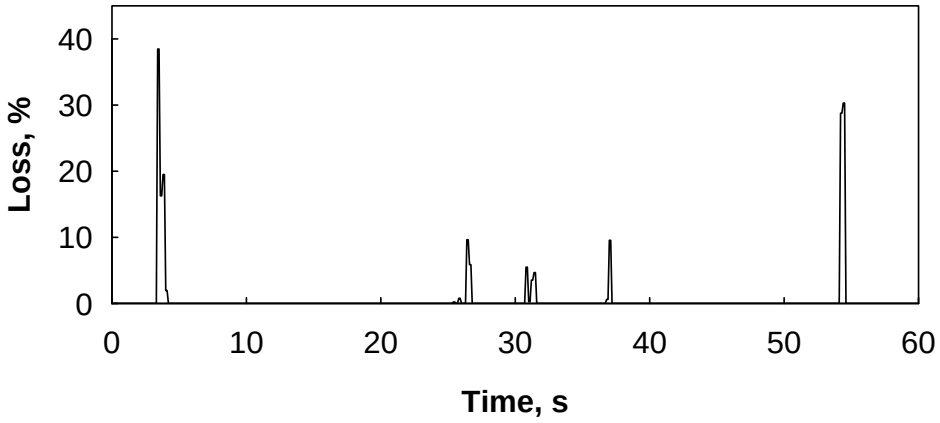


(c)

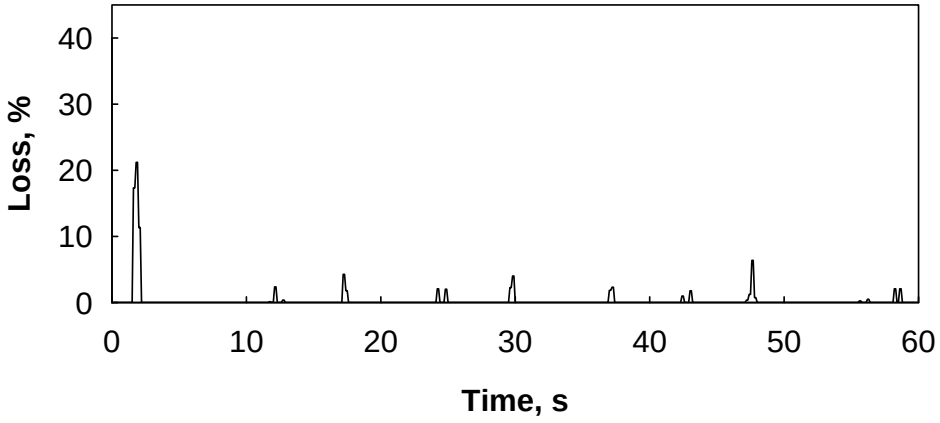
Figure 4.12: Queuing delay for D packets with different transport protocols for D flows: (a) TCP NewReno; (b) Paced TCP; (c) TFRC.



(a)

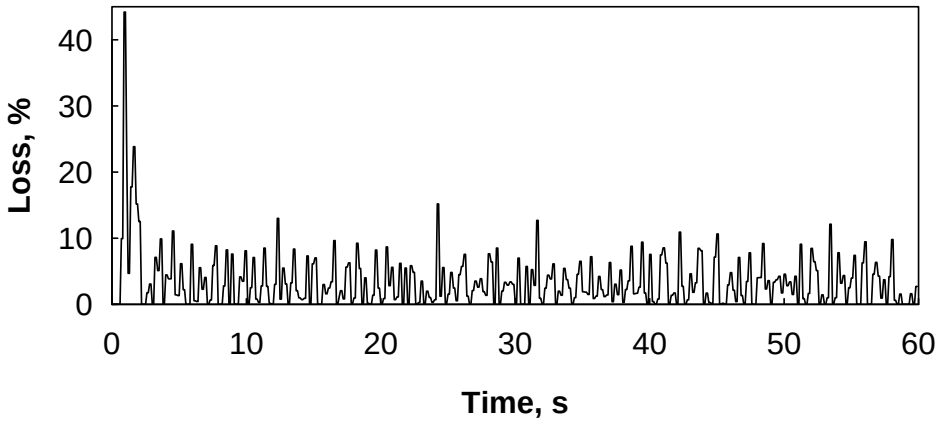


(b)

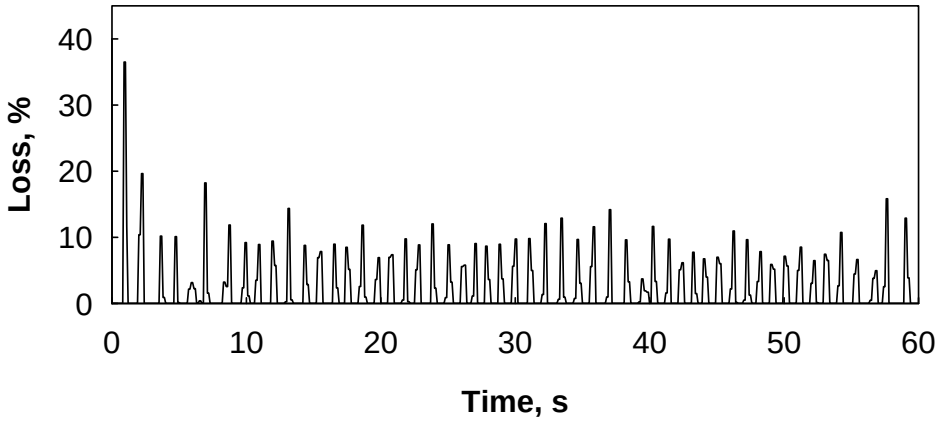


(c)

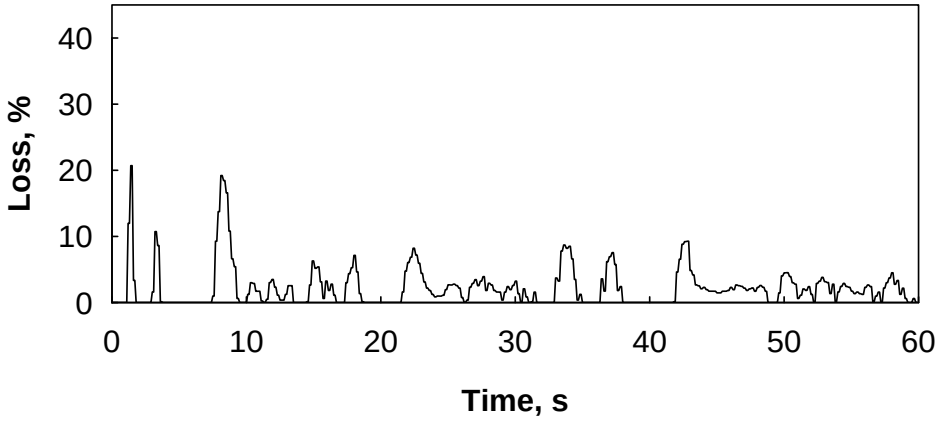
Figure 4.13: Loss rate for R flows with different transport protocols for D flows: (a) TCP NewReno; (b) Paced TCP; (c) TFRC.



(a)

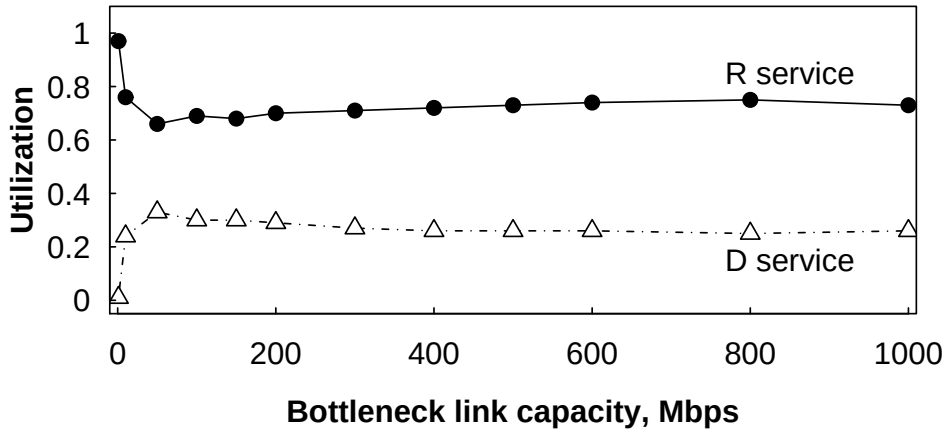


(b)

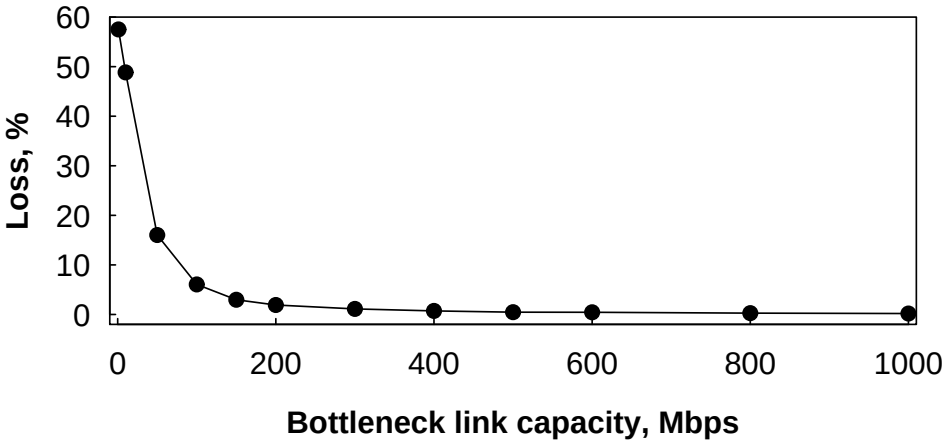


(c)

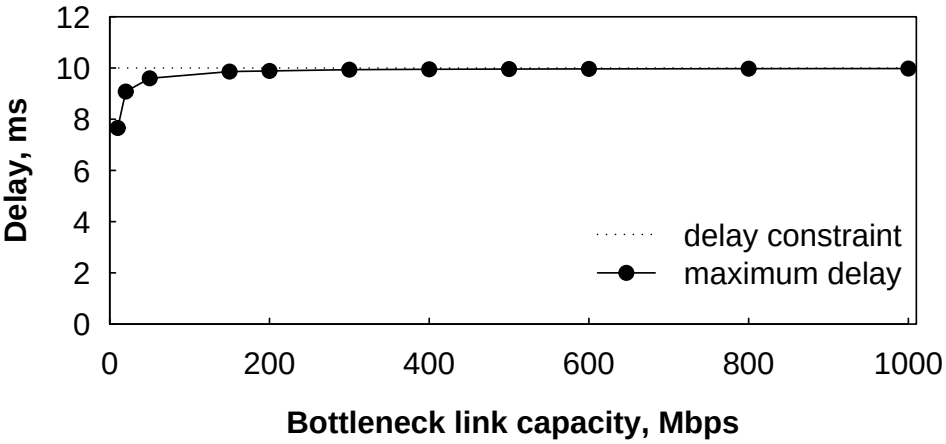
Figure 4.14: Loss rate for D flows with different transport protocols for D flows: (a) TCP NewReno; (b) Paced TCP; (c) TFRC.



(a)



(b)



(c)

Figure 4.15: Scalability of the RD services with respect to the bottleneck link capacity (S-RD version): (a) Link utilization; (b) Loss rate for D flows; (c) Maximum queuing delay for D flows.

Impact of the Population Size

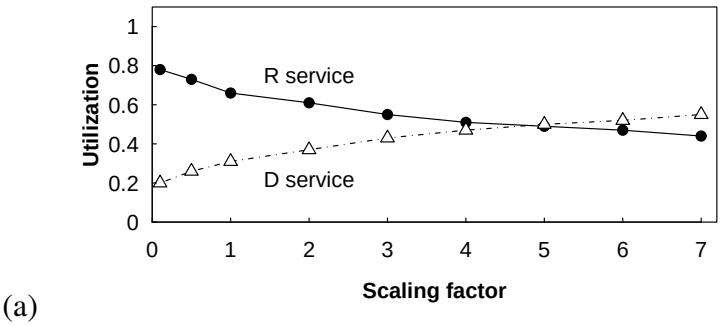
This section also explores population scalability of the RD services, i.e., examines how their performance scales when the numbers of R and D flows change in the $complex\ topology$. A scaling factor σ is used to modify the traffic mix as follows: the population of the long-lived flows includes 25 R flows and $\lceil 25\sigma \rceil$ D flows from pool H, 25 R flows and $\lceil 25\sigma \rceil$ D flows from pool J, 50 R flows from pool K, and $2 \cdot \lceil 25\sigma \rceil$ D flows from pool F. To preserve the expectation that all the long-lived flows arrive before the measurement stage of the experiment, average interarrival time is reduced to 3 ms for $\sigma > 3$. The long-lived traffic in the reverse direction mirrors again the forward-direction arrangement.

For either of bottleneck links z1-z2 and y1-y2, Figure 4.16 shows that increasing the number of long-lived D flows redistributes some of the link capacity from the R service to the D service. Due to the presence of the web-like flows, the redistribution depends on σ non-linearly. Also, since links z1-z2 and y1-y2 serve different numbers of flows, the D service gains parity with the R service in utilizing link z1-z2 with a larger scaling factor than for link y1-y2. As σ grows, the per-flow rates of the R and D flows decrease, and the loss rates of the services increase accordingly.

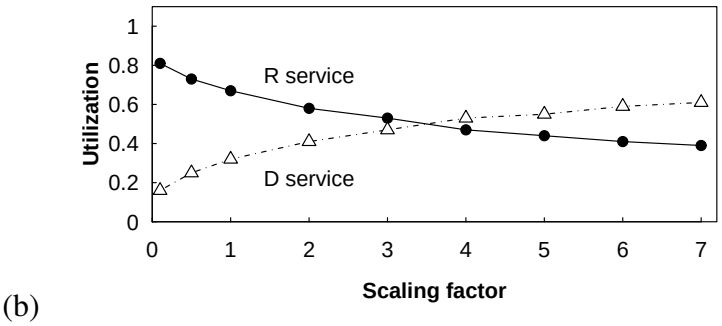
Finally, a similar study for scalability of the RD services with respect to the number of R flows is conducted. Once again, the long-lived traffic arrangement is symmetrical in the forward and reverse directions. In the forward direction, the long-lived traffic includes $\lceil 25\sigma \rceil$ R flows and 25 D flows from pool H, $\lceil 25\sigma \rceil$ R flows and 25 D flows from pool J, $2 \cdot \lceil 25\sigma \rceil$ R flows from pool K, and 50 D flows from pool F. Figure 4.17 plots utilization and loss rates for links z1-z2 and y1-y2. The analytical rationale for the observed performance profiles is the same as the above explanations for the scaling of the D population.

Sensitivity to the Delay Constraint

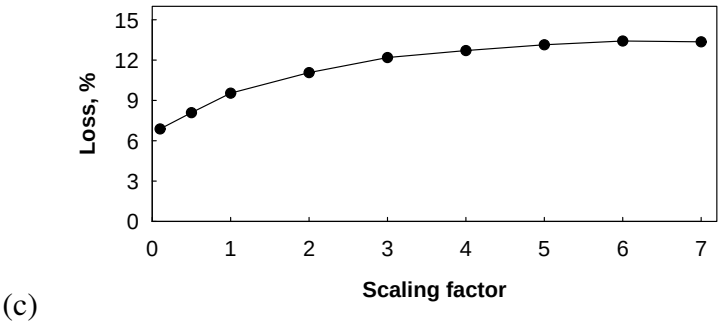
To examine sensitivity of the RD services to d, $simple\ topology$ is employed, and the delay constraint of the D service is varied from 3 ms to 15 ms. Figure 4.18 demonstrates that the per-flow rate ratio for the R and D flows stays close to the intended 2:1. As d increases, the loss rate for the D service decreases from about 8% to about 5% due to the increasing size of the D buffer. Moreover, Figure 4.18c shows that the stateless version of the RD services keeps the delay constraint.



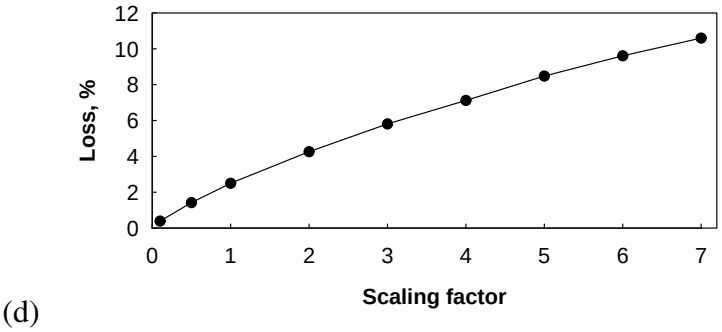
(a)



(b)

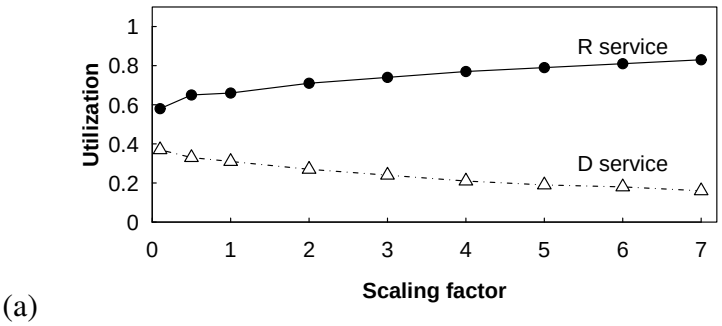


(c)

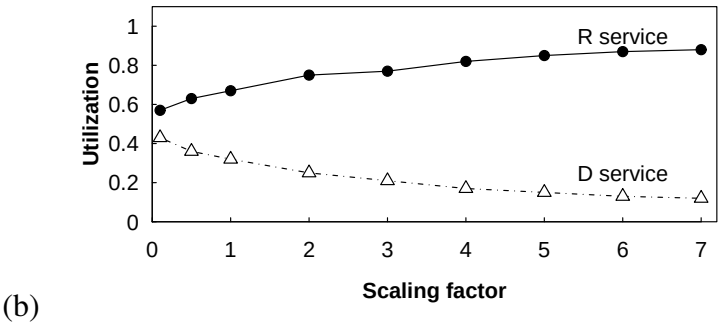


(d)

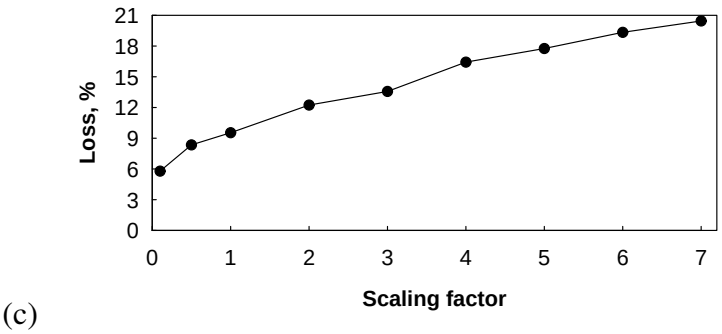
Figure 4.16: Scalability of the RD services with respect to the number of long-lived D flows: (a) Link z1-z2 utilization; (b) Link y1-y1 utilization; (c) Loss rate for D flows at link z1-z2; (d) Loss rate for D flows at link y1-y2.



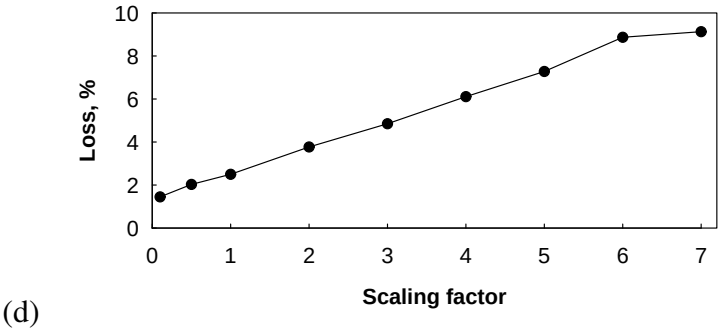
(a)



(b)

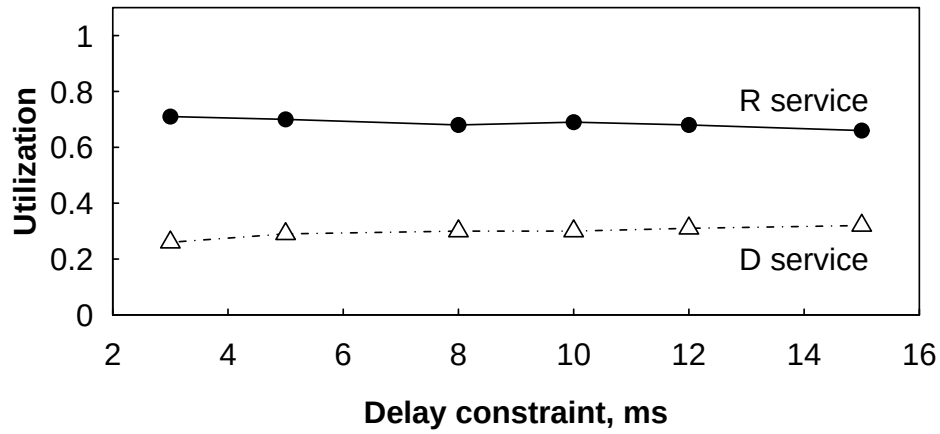


(c)

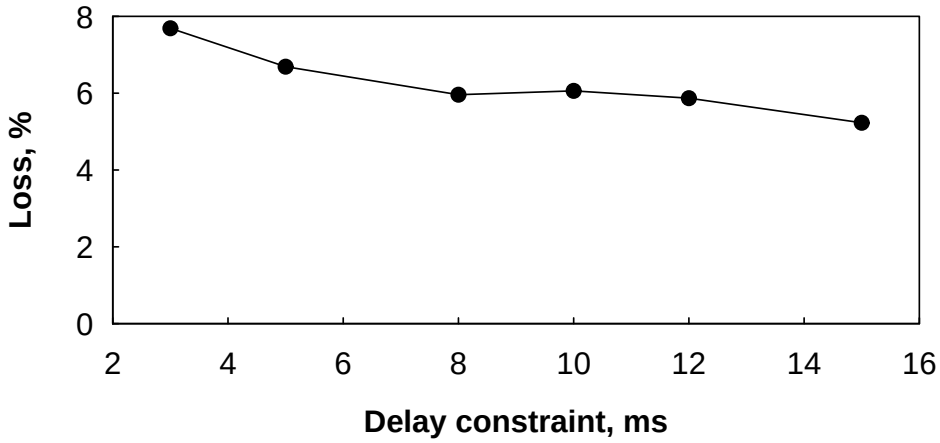


(d)

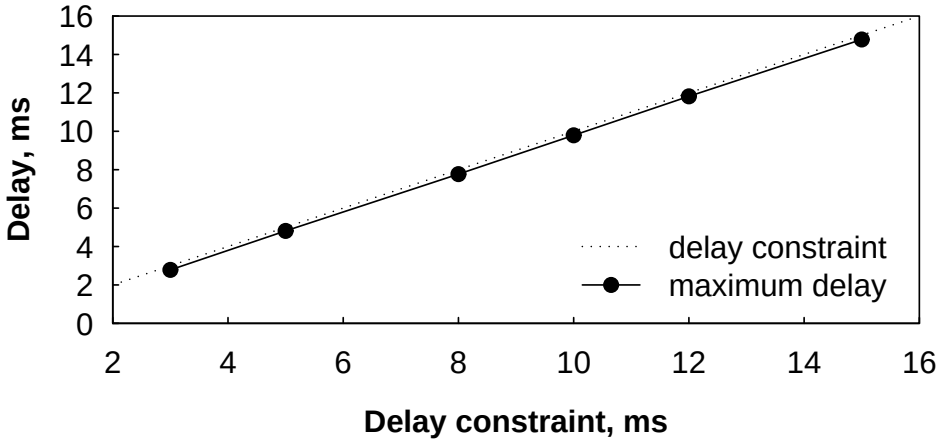
Figure 4.17: Scalability of the RD services with respect to the number of long-lived R flows: (a) Link z1-z2 utilization; (b) Link y1-y1 utilization; (c) Loss rate for D flows at link z1-z2; (d) Loss rate for D flows at link y1-y2.



(a)



(b)



(c)

Figure 4.18: Sensitivity to the delay constraint of the D service (S-RD version): (a) Link utilization; (b) Loss rate for D flows; (c) Maximum queuing delay for D flows.

4.6.4 Heterogeneous Environments

Influence of Propagation RTTs

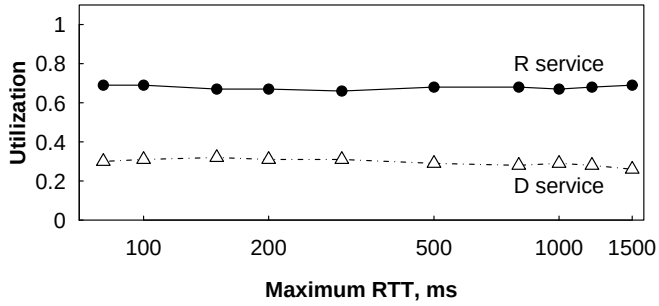
This set of the experiments is conducted in the $complex\ topology$. The long-lived traffic includes 25 R flows and 25 D flows from pool H, 25 R flows and 25 D flows from pool J, 50 R flows from pool K, and 50 D flows from pool F. For each of the flows, the reverse direction of its path carries another long-lived flow of the same class. Also, two sources in pool H transmit web-like R and D flows to pool G. The web-like traffic has the same characteristics as in Section 4.6.3. The capacity of link z1-z2 is set to 100 Mbps. Thus, the network contains two bottleneck links: z1-z2 and y1-y2.

To study the impact of propagation RTT on the RD services, M is changed from 80 ms to 1.5 s. As the maximum propagation RTT grows, the per-flow number of packets inside the network increases. Consequently, the TCP flows enjoy lower loss rates. Figure 4.19 confirms this expectation and also shows that the RD services consistently support the intended 2:1 per-flow rate ratio for the R and D flows.

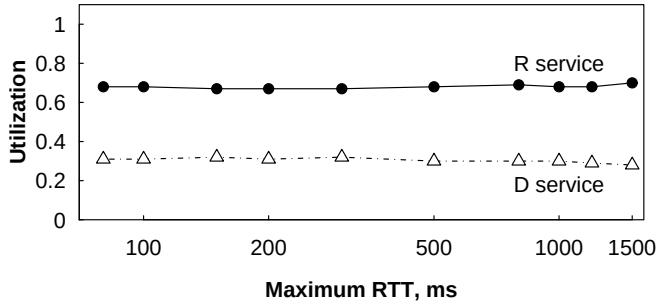
Besides, the same experiment is run for the S-RD version of the architecture. Figure 4.20 demonstrates that the throughput differentiation is supported over the whole range of the varied parameter. Moreover, the maximum queuing delay at both the monitored bottleneck links does not exceed the delay constraint.

Impact of the Packet Size

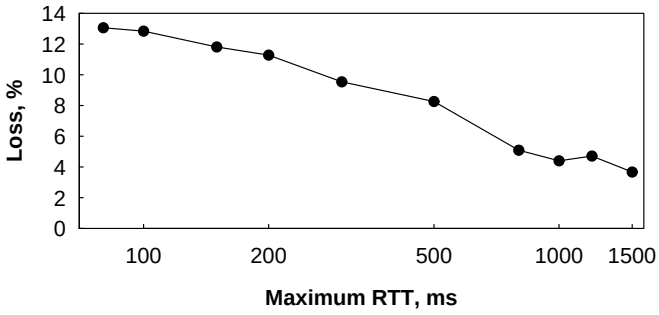
To estimate the influence of the packet size on the performance of S-RD implementation of the design, the sizes of packets in both the classes are varied in the $simple\ topology$. There are two sets of the experiments. In each set, the packet size for one class is fixed to 1000 bytes, and the packet size for the other class is modified in the range between 100 bytes and 1500 bytes. Figures 4.21 and 4.22 depict that the delay constraint is kept over the whole range of packet sizes from class R and class D, respectively. Whereas the size of packets from class R does not affect significantly the loss rate of class D, the loss rate of class D increases monotonically with the size of packets from class D. We explain it by the bigger contribution of part $\frac{S_D^{max}}{\alpha}$ than S_R^{max} in equation (4.10) into the D buffer size.



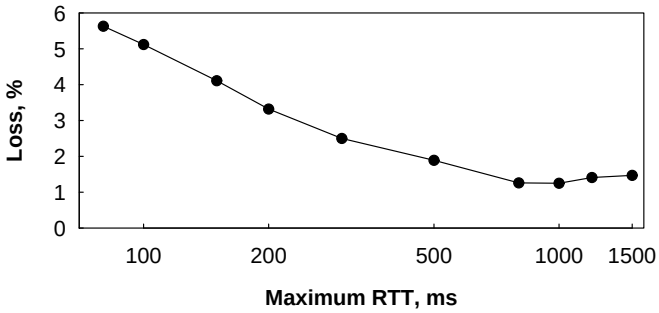
(a)



(b)

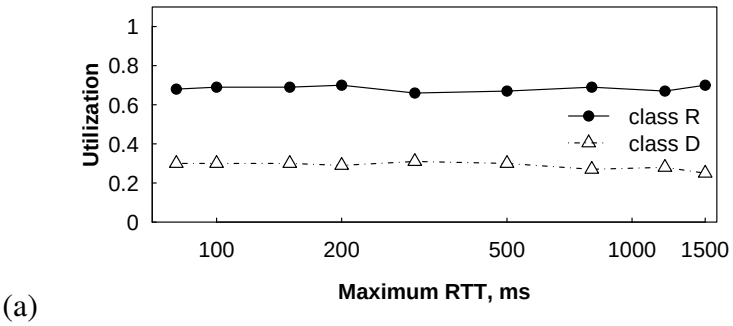


(c)

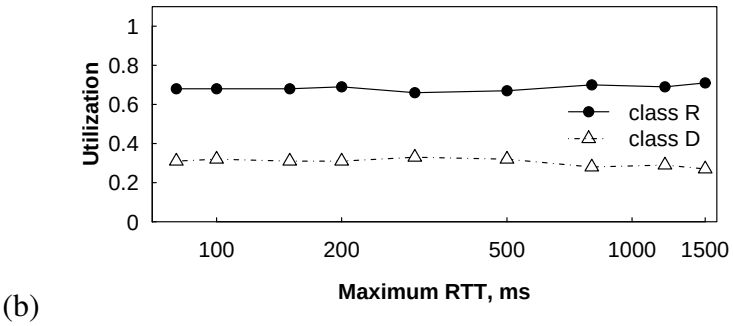


(d)

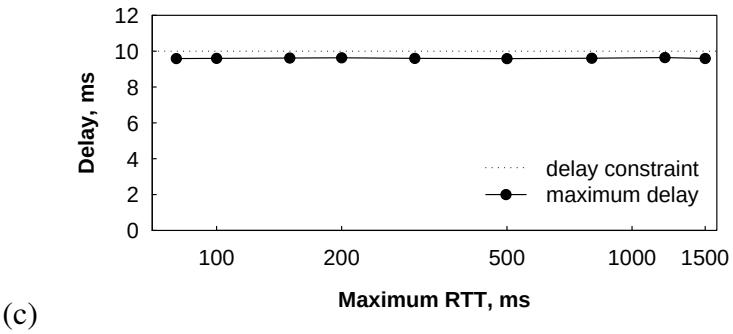
Figure 4.19: Impact of the propagation RTT diversity: (a) Link z1-z2 utilization; (b) Link y1-y1 utilization; (c) Loss rate for D flows at link z1-z2; (d) Loss rate for D flows at link y1-y2.



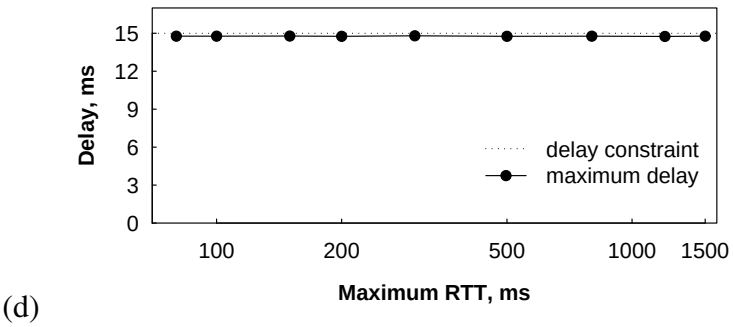
(a)



(b)

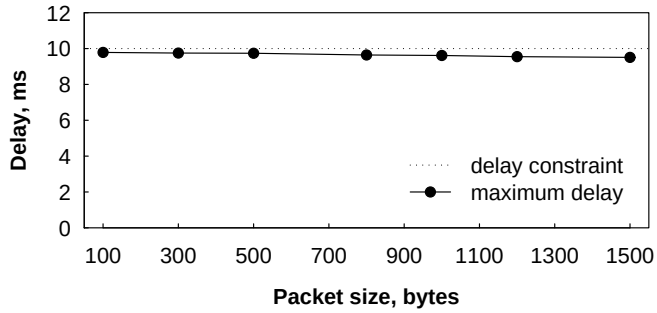


(c)

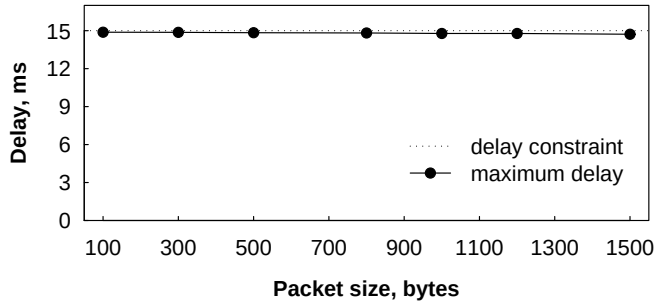


(d)

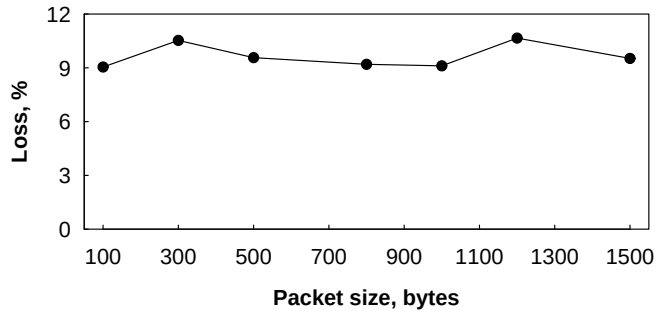
Figure 4.20: Impact of the propagation RTT diversity (S-RD version): (a) Link z1-z2 utilization; (b) Link y1-y1 utilization; (c) Maximum queuing delay for D flows at link z1-z2; (d) Maximum queuing delay for D flows at link y1-y2.



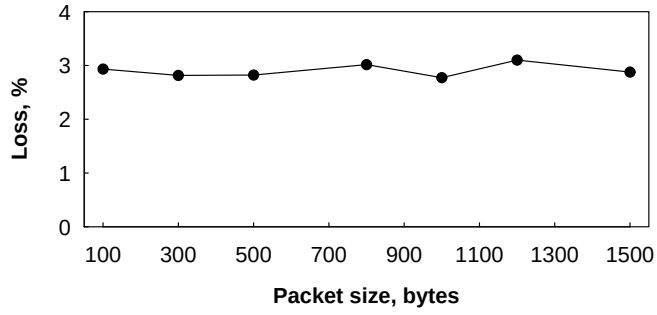
(a)



(b)

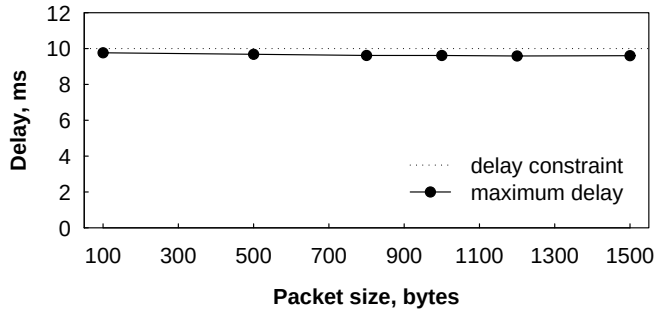


(c)

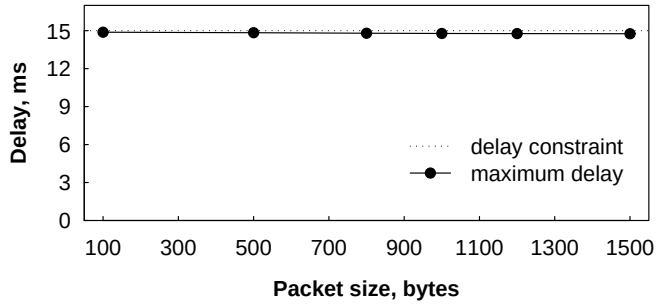


(d)

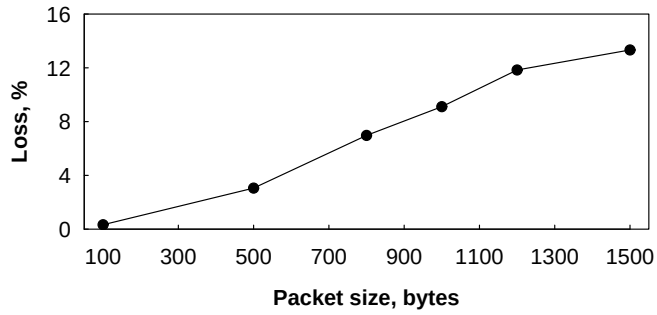
Figure 4.21: Sensitivity of the S-RD version to the size of packets from class R: (a) Maximum queueing delay for D flows at link z1-z2; (b) Maximum queuing delay for D flows at link y1-y2; (c) Loss rate for D flows at link z1-z2; (d) Loss rate for D flows at link y1-y2.



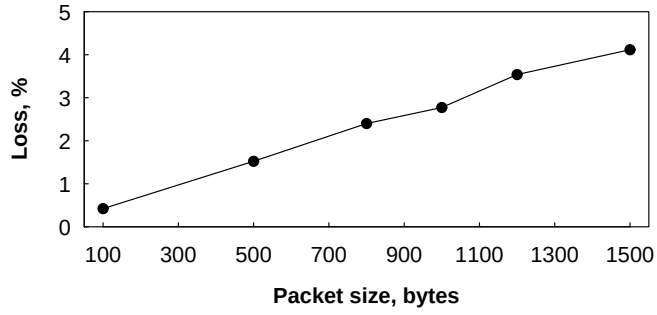
(a)



(b)



(c)



(d)

Figure 4.22: Sensitivity of the S-RD version to the size of packets from class D: (a) Maximum queueing delay for D flows at link z1-z2; (b) Maximum queuing delay for D flows at link y1-y2; (c) Loss rate for D flows at link z1-z2; (d) Loss rate for D flows at link y1-y2.

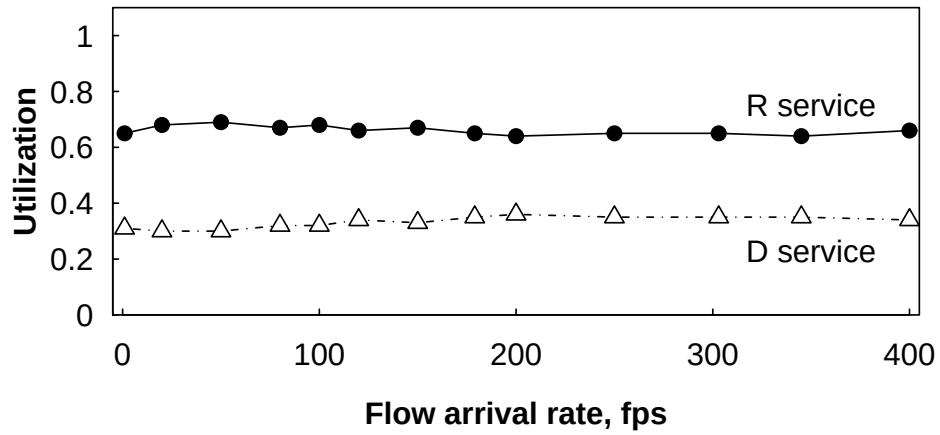
4.6.5 Dynamic Network Conditions

Influence of Web-like Traffic

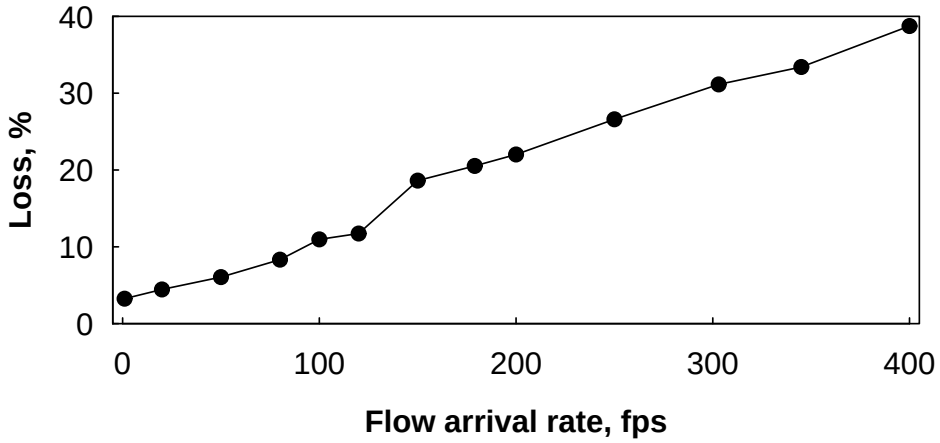
To see how web-like flows affect the RD services, the average arrival rate is varied from 1 fps to 400 fps in the $simple\ topology$. When the flows arrive more frequently, the traffic mix becomes more bursty and imposes higher load on the bottleneck link. As expected, these factors drive up the loss rate for the D service. As the approximate number of flows is the same, we expect that the ratio of the link utilizations by R flows and D flows should be 2:1, which is observed in Figure 4.23. More importantly, despite the increasing losses, the RD services still closely maintain the intended 2:1 per-flow rate ratio for the R and D flows. In addition, Figure 4.23c demonstrates that the maximum queuing delay for D flows revealed by the stateless version of the design does not exceed the expected 10 ms.

Heavy-tailed Flow Interarrival Times

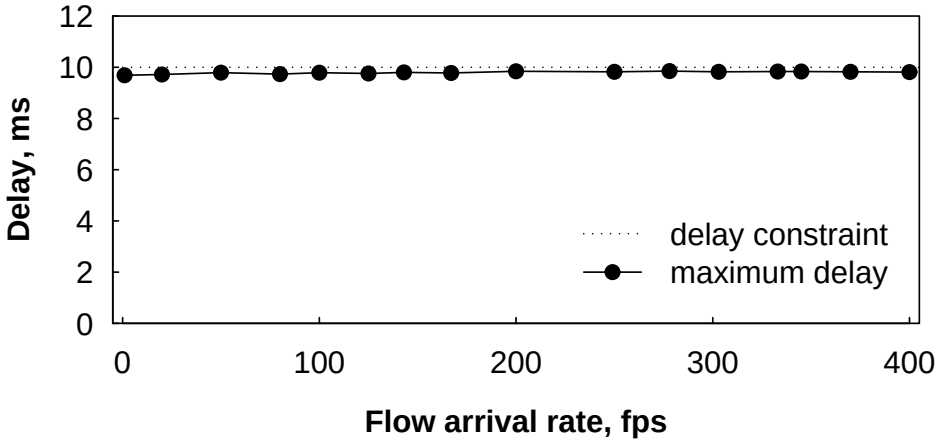
While the previous section experiments with the web-like traffic where the flow interarrival times adhere to the Exponential distribution, this section now modifies that arrangement to the Pareto distribution with the shape index of 1.1. The only other traffic besides the web-like flows comes from 50 R flows and 50 D flows that traverse the reverse direction of the bottleneck link throughout the experiment. The access links for the web-like flows have capacity 1 Gbps. The Pareto interarrival times make the traffic bursty and highly dynamic. Figure 4.24 reflects the high dynamism of the R and D flow counts by showing the widely fluctuating utilization of the bottleneck link by either R or D service. When the flows arrive at average rate 50 fps, their average cumulative load is low, and they rarely congest the bottleneck link. Arrival rate 100 fps makes the congestion instances more frequent and intense. Increasing the average arrival rate to 200 fps creates persistent overload of the bottleneck link. Together with the burstiness of the arrival process, the persistent overload causes heavy losses for the D service, which is shown in Figure 4.25. Despite the burstiness of the traffic, the delay constraint is always satisfied, as Figure 4.26 demonstrates.



(a)

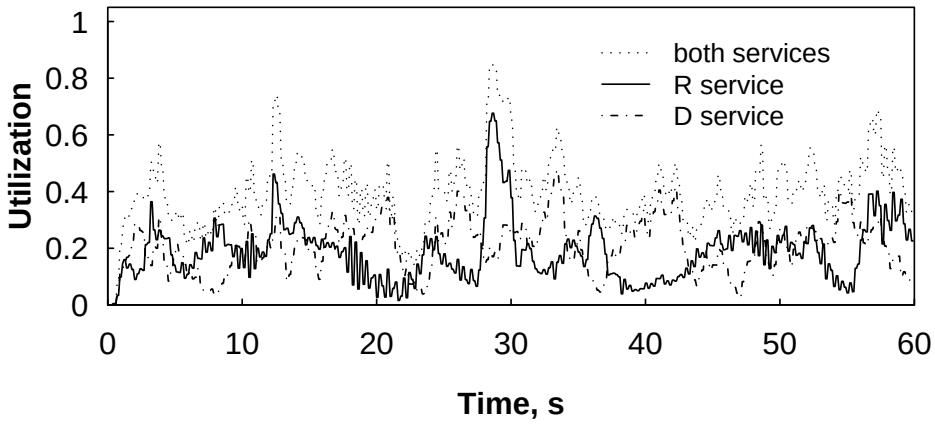


(b)

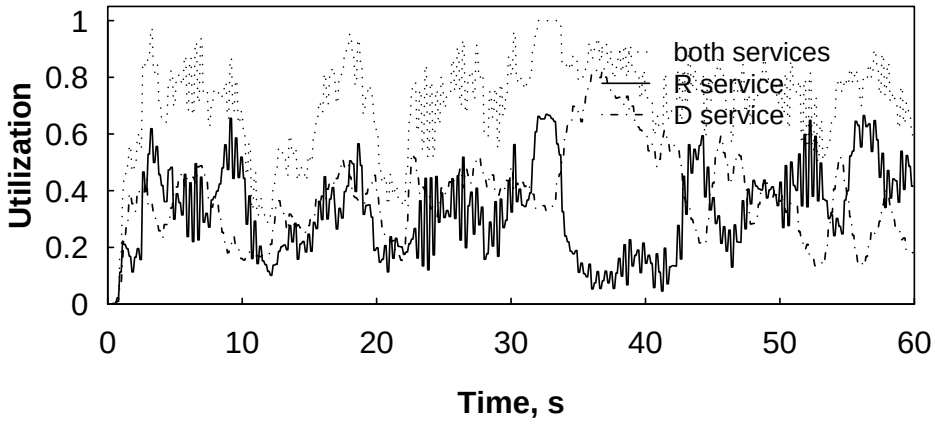


(c)

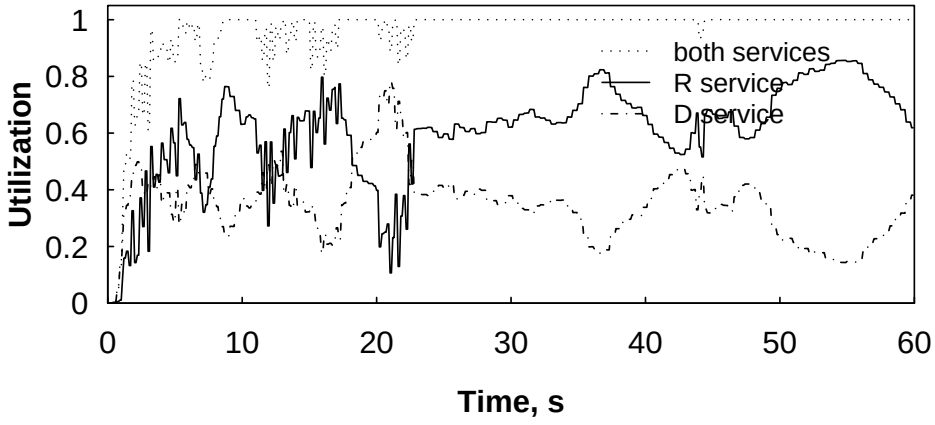
Figure 4.23: Impact of the web-like traffic (S-RD version): (a) Link utilization; (b) Loss rate for D flows; (c) Maximum queuing delay for D flows.



(a)

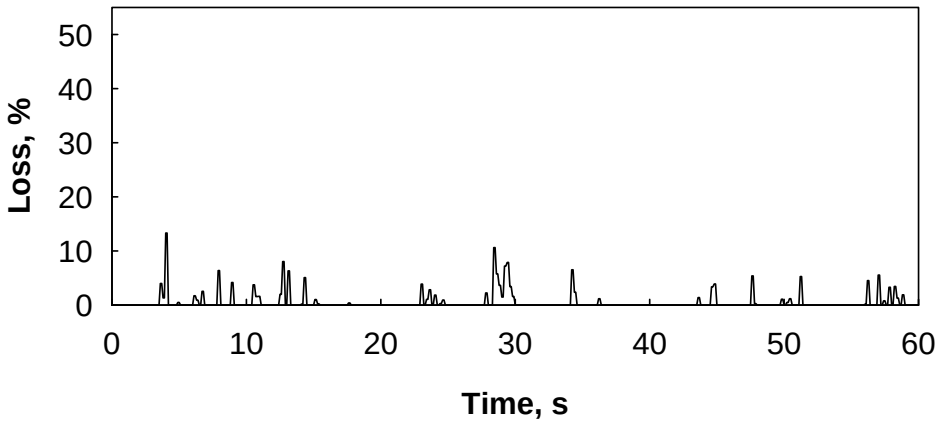


(b)

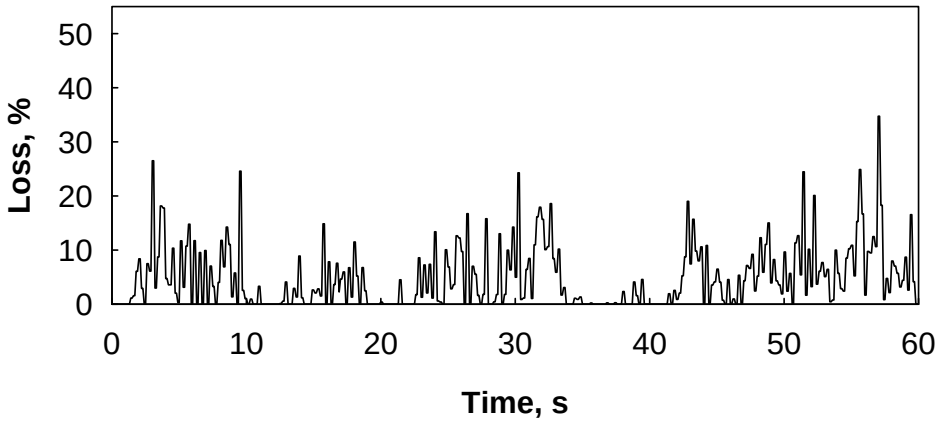


(c)

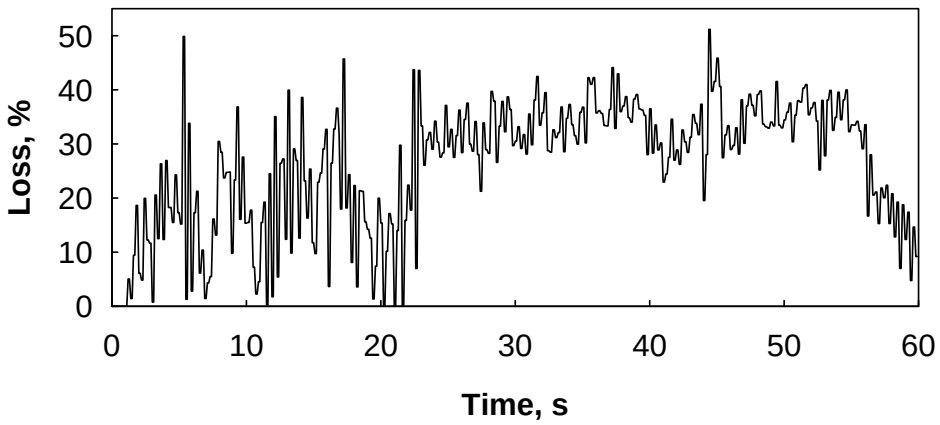
Figure 4.24: Bottleneck link utilization under the RD services with the Pareto distribution for the interarrival times of the web-like flows: (a) Average arrival rate 50 fps; (b) Average arrival rate 100 fps; (c) Average arrival rate 200 fps.



(a)

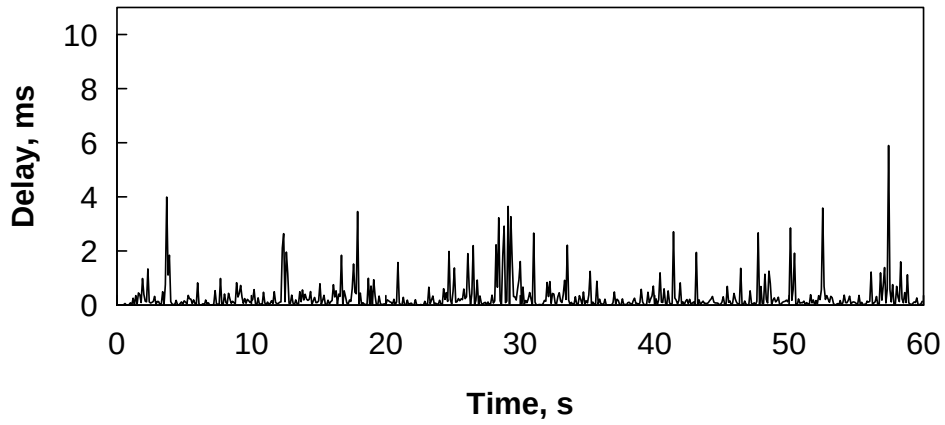


(b)

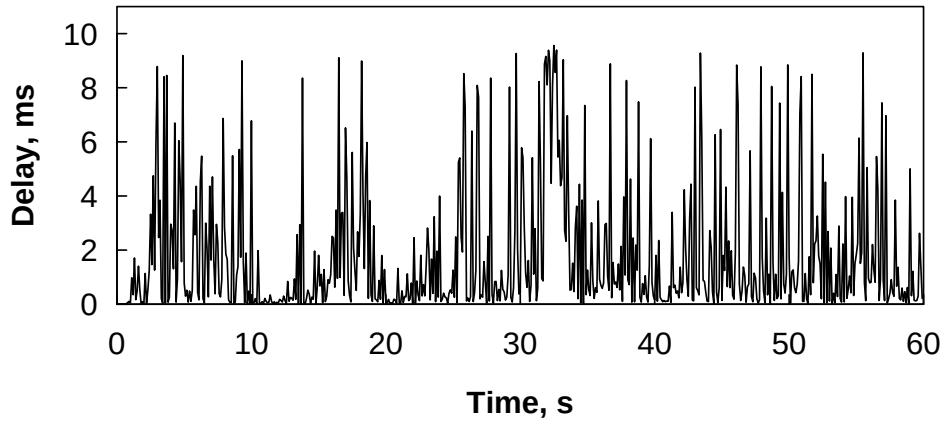


(c)

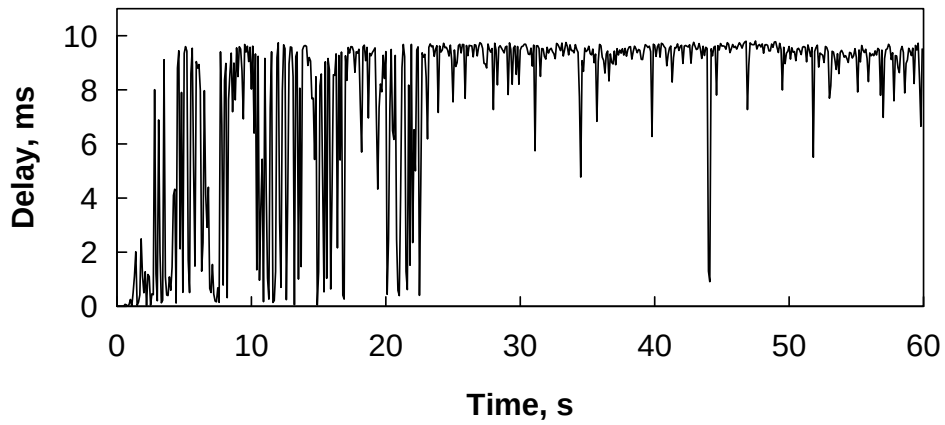
Figure 4.25: Loss rate for D flows under the RD services with the Pareto distribution for the interarrival times of the web-like flows: (a) Average arrival rate 50 fps; (b) Average arrival rate 100 fps; (c) Average arrival rate 200 fps.



(a)



(b)



(c)

Figure 4.26: Queuing delay for D packets under the RD services with the Pareto distribution for the interarrival times of the web-like flows: (a) Average arrival rate 50 fps; (b) Average arrival rate 100 fps; (c) Average arrival rate 200 fps.

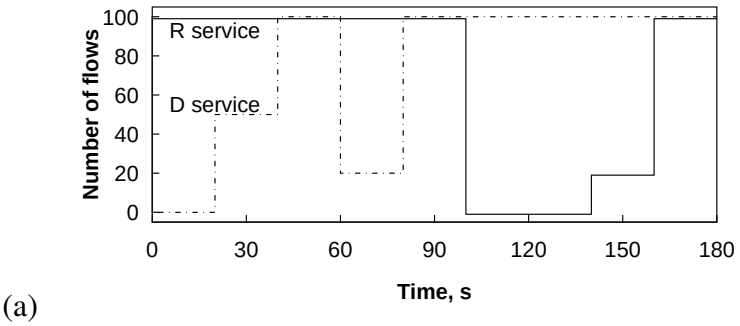
Sudden Changes in Network Conditions

To investigate how the RD services react to sudden changes in the network, the numbers of R and D flows are changed in the $simple\ topology$ as follows. 100 R flows start at time 0. 50 D flows join them 20 s later. 50 additional D flows arrive at time 40 s and thereby equalize the flow counts for the two services at 100. At time 60 s, 80 D flows finish. 80 other D flows arrive at time 80 s. All R flows leave at time 100 s but 20 new R flows start 40 s later. Finally, 80 extra R flows arrive at time 160 s and reestablish the parity in the numbers of R and D flows. Figure 4.27 shows that the RD design responds to the changes promptly and appropriately: reflecting the current ratio of the flow counts, the per-flow rate ratio for R and D flows becomes 4:1 at time 20 s, reduces to 2:1 at time 40 s, grows to 10:1 at time 60 s, and returns to 2:1 at times 80 s and 160 s, at the latter time by reverting from 1:2. The sudden changes in the flow counts cause sharp spikes in the loss rate for the D service but the spikes are short, and the losses decrease quickly to under 10%. The RD services utilize the bottleneck link fully except between 100 s and 140 s. During that interval, the link carries only D flows and is underutilized due to the small size of the D buffer.

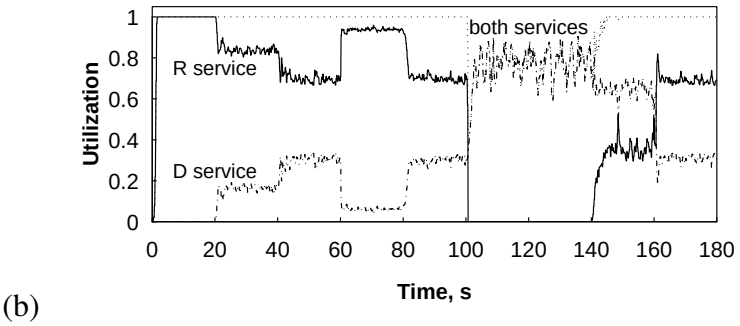
4.6.6 Legacy Traffic and Incremental Deployment

The design principles in Section 4.1 prescribe that a new service should attract adopters despite continued presence of legacy ISPs and without penalizing legacy traffic. This section experimentally verifies whether the RD services fulfill these design aspirations, and it uses the $complex\ topology$. Unlike ISP Y, ISP Z does not support the RD services and treats all traffic with the legacy service. 500 flows traverse the network: 125 flows come from pool H, other 125 flows originate at pool J, and the remaining 250 flows enter from pools K or F. Link z1-z2 has capacity 55 Mbps making link y1-y2 a bottleneck for all the flows. This section varies ρ, the percentage of D flows. The other $1 - \rho$ flows are either legacy or R flows. More specifically, $\lceil 125\rho \rceil$ D flows come from pool H, $\lceil 125\rho \rceil$ D flows originate at pool J, all $2 \cdot \lceil 125\rho \rceil$ flows from pool F indicate their preference for the D service, and the rest of the traffic consists of legacy and R flows.

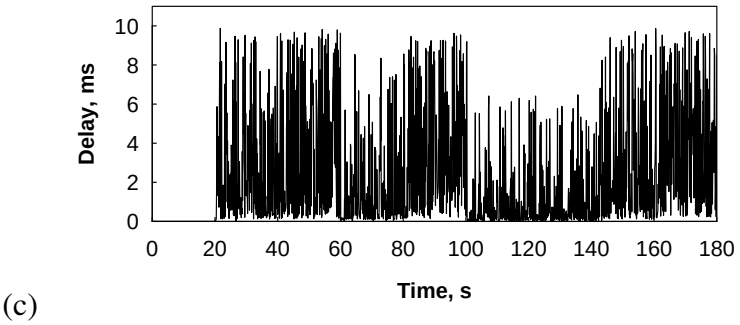
Figure 4.28a plots the per-flow rates achieved by the legacy and R flows and D flows at link y1-y2 of ISP Y. As those legacy flows that are interested in low delay opt for the D service and thereby increase the percentage of D flows, the per-flow rate for the remaining



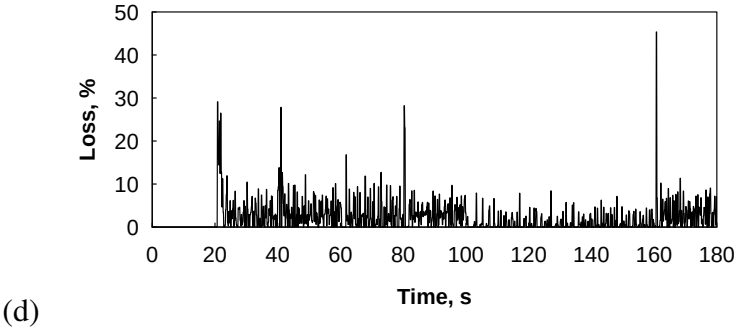
(a)



(b)

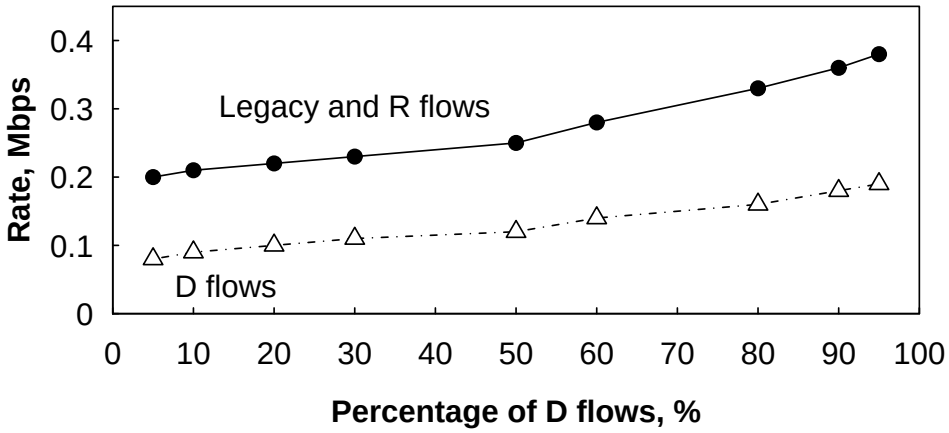


(c)

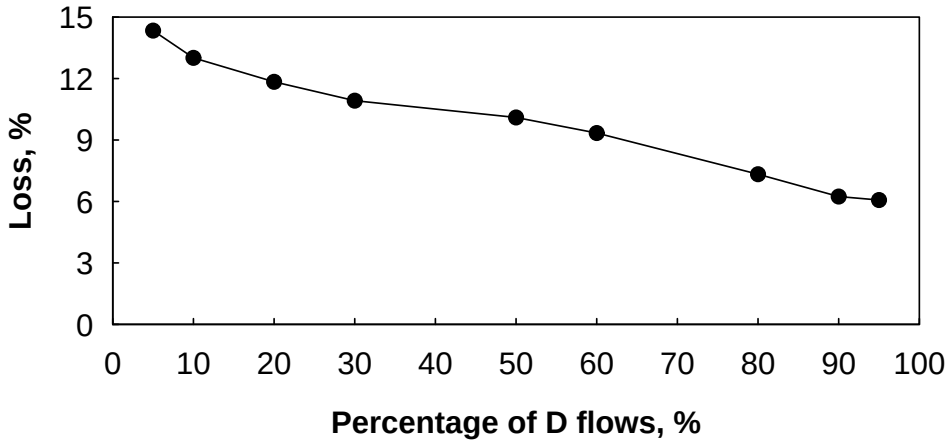


(d)

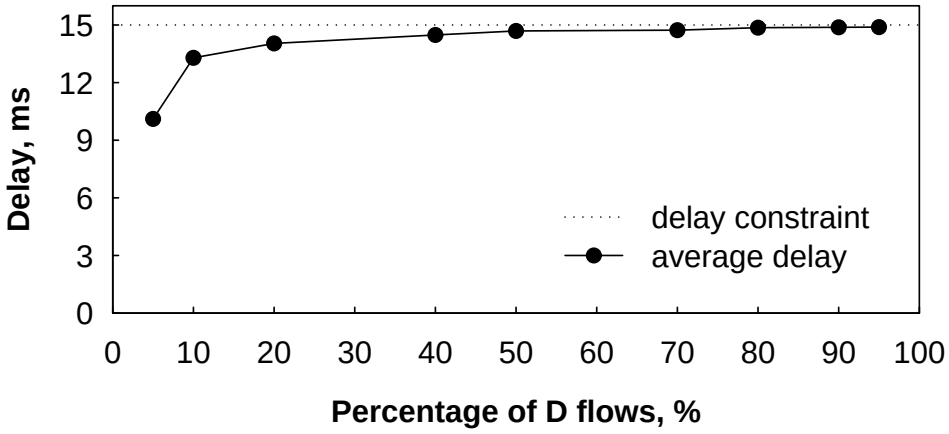
Figure 4.27: Reaction to sudden changes in the numbers of R and D flows: (a) Dynamics of the number of flows; (b) Link utilization; (c) Queuing delay for D packets; (d) Loss rate for D flows.



(a)



(b)



(c)

Figure 4.28: Impact of the incremental deployment on the performance at link y1-y2 of ISP Y: (a) Per-flow rates; (b) Loss rate for D flows; (c) Maximum queuing delay for D flows (S-RD version).

legacy flows consistently improves even though some of them enter the network through the legacy ISP Z. Hence, the legacy traffic not only avoids being penalized by the adopters of the D service in accordance with Principle 2 but also benefits itself by becoming able to communicate at higher rates. Besides, Figure 4.28 reveals that adoption of the RD services yields a win-win outcome for all users: as ρ grows, the per-flow rate increases for the D flows as well, and the increasing size of the D buffer reduces the loss rate of the D service. In addition, Figure 4.28 illustrates that the stateless version satisfies the delay constraint for all the values of the varied parameter. Therefore, whereas a user opts for the D service to acquire low delay, future adoptions of the D service by other legacy users make the service even more valuable, facilitating the virulent deployment of the RD services.

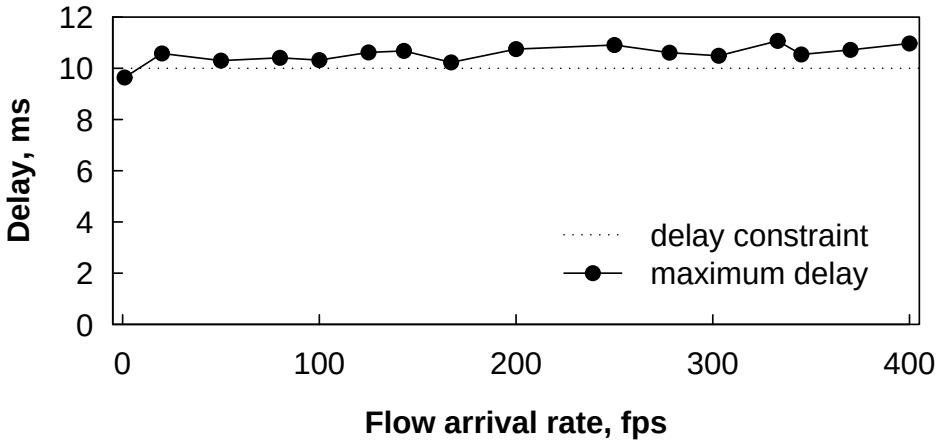
4.6.7 Impact on the Loss Rate

Finally, this section explores how the discard of all packets from the D queue at the moment of the recalculation of control parameters affects the loss rate of class D. Such flushing is done to address a possibility of exceeding the delay constraint. As a theoretical analysis of the influence of the flushing is difficult, we apply an experimental approach and conduct two sets of simulations employing the S-RD version of the design.

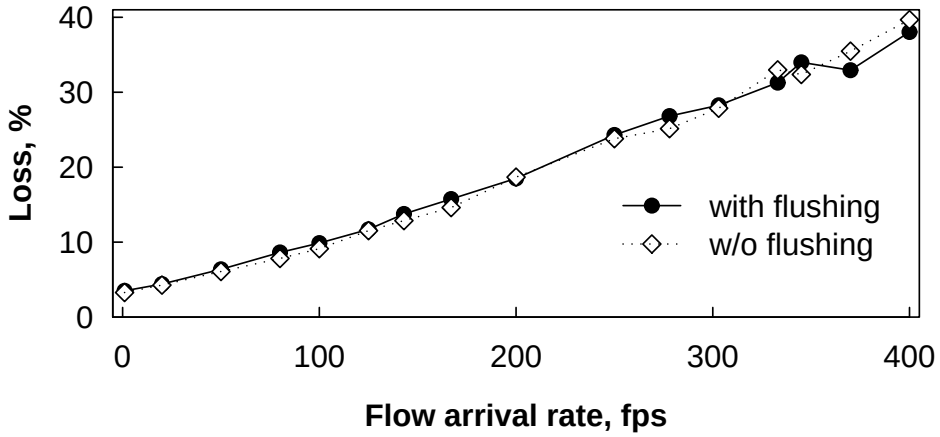
Figure 4.29 reports the results for the same experimental settings as in the part of Section 4.6.5 exploring influence of web-like traffic, with no flushing from the D queue. As expected the maximum queuing delay of class D exceeds the delay constraint for all the values of the arrival rate for web-like flows except for 1 fps. The loss rate of class D is approximately the same as with the flushing.

Figure 4.30 shows the results for the same experimental settings as in Section 4.6.6, which illustrates incremental deployment, with avoiding flushing from the D queue. One can observe that the maximum queuing delay of class D is bigger than the delay constraint for the half of the values of the percentage of D flows. The loss rate of class D is approximately the same as with the flushing except for the traffic with 5% of D flows, for which the loss rate increases by 3%.

The explanation of the loss rate behavior is as follows. Although with the flushing there are induced losses of class D, the dropping of packets creates the room in the buffer for new packets from class D, and this compensates for losses caused by the flushing. Thus, based



(a)



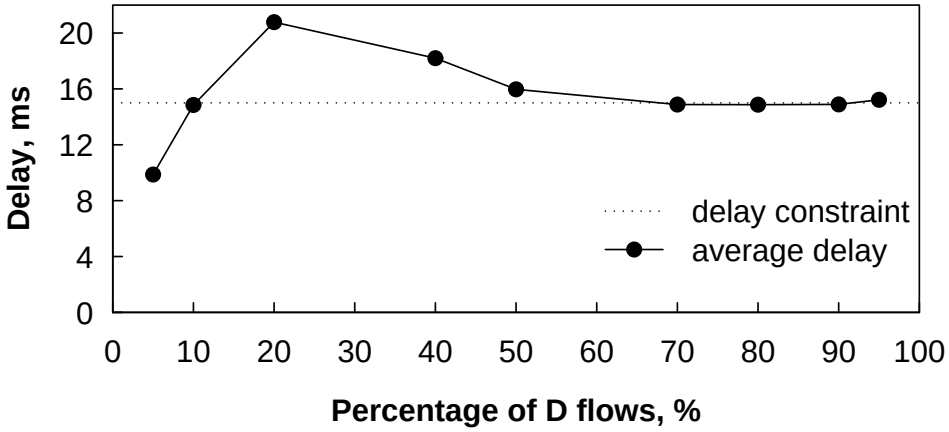
(b)

Figure 4.29: Influence of avoiding the "flushing" of the D queue with different intensities of web-like traffic: (a) Maximum queuing delay for class D; (b) Average loss rate for class D.

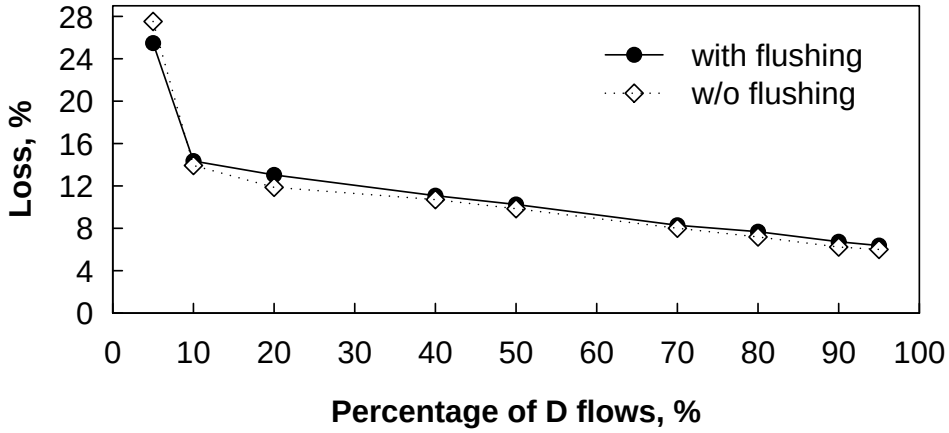
on the results of the experiments, we assert that flushing the D queue for ensuring the delay constraint does not lead to deterioration of the loss rate.

4.7 Performance Evaluation with Telephony Traffic

To evaluate the performance of the RD services for VoIP (Voice over Internet Protocol), we vary the intensity of the web-like traffic, propagation RTT, and the number of VoIP flows. We use the $simple\ topology$, and there are 100 flows from class R and 100 VoIP flows. To generate the VoIP traffic, we use the tool developed in [7]. RTTs of the VoIP flows are the same and equal to 150 ms. Besides, there are 100 flows from class R in the



(a)



(b)

Figure 4.30: Influence of avoiding the "flushing" of the D queue with the incremental deployment of the S-RD design: (a) Maximum queuing delay for class D; (b) Average loss rate for class D.

reverse direction. The value of d is 50 ms. In addition, there is one web server and one web traffic receiver connected to the bottleneck link for classes R and D. The size of web flows is described by the Pareto distribution with the average of 30 packets and shape index of 1.3. Web flows arrive with the intensity of 50 fps. We perform five experiments for each settings, and each experiment lasts for 70 sec. To encode the speech, we employ AMR (Adaptive Multi-Rate) Audio Codec [97] operating at audio bitrate of 12.2 kbps.

To evaluate the quality of the delivered service, we use the Mean Opinion Score (MOS) [1], a subjective score for voice quality ranging from 1, unacceptable, to 5, excellent. To estimate a MOS score, we apply the E-Model [2], which assesses VoIP quality accounting network characteristics. The E-Model uses the R-factor that is computed as a function of all of the impairments occurring with the voice signal, and is ranged from 0 to 100. The

R-factor range	MOS	Category of voice transmission quality	User satisfaction
90 - 100	4.34 - 4.50	Best	Very satisfied
80 - 90	4.03 - 4.34	High	Satisfied
70 - 80	3.60 - 4.03	Medium	Some users dissatisfied
60 - 70	3.10 - 3.60	Low	Many users dissatisfied
50 - 60	2.58 - 3.10	Poor	Nearly all users dissatisfied

Table 4.3: Categories of voice transmission quality

relationship between the R-factor and MOS score can be described through the following equation [2]:

$$MOS = 1 + 0.035R + 7 * 10^{-6}R(R - 60)(100 - R) \tag{4.31}$$

According to E-Model, R-factor is calculated as follows:

$$R = R_0 - I_s - I_{e,eff} - I_d + A \tag{4.32}$$

where R_0 captures the basic signal-to-noise ratio, I_s accounts the impairments occurring with the voice signal and does not depend on the transmission over the network, $I_{e,eff}$ describes impairments related to data loss and low rate codecs, I_d specifies the impairments induced by delay and echo, and A, "advantage factor", compensates the above impairments taking into account that a user may tolerate some decrease of voice quality in exchange for access advantage. For example, whereas for a wired phone A equals to zero, A becomes equal to ten for cellular in a moving vehicle. Table 4.3 maps the values of R-factor into MOS, the category of voice transmission quality, and user satisfaction. One should notice that connections with R-factor below 50 are not recommended.

To characterize the performance of the architecture for VoIP, we measure average MOS of all VoIP flows, the average utilization and loss rate of class R. While measuring MOS, first ten seconds of the experiment are neglected. All flows join the network during the first second. We compare the performance of the RD services with the performance of the DropTail routers.

4.7.1 Dynamic Network Conditions

Influence of Web-like Traffic

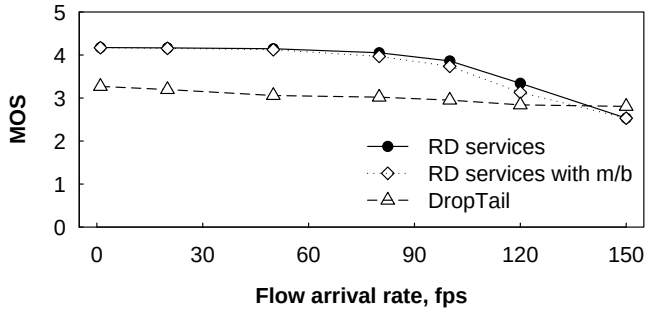
To study the influence of web-like traffic, we change the intensity of the web-like traffic in the interval between 1 fps and 150 fps. Besides, we keep tracking the performance of one long-lived throughput-greedy flow that initially uses R service and then switches to D service. The results are presented in Figure 4.31. One can observe that the RD services demonstrate better performance for VoIP over almost the whole range of the varied parameter, whereas the R traffic gets the same bottleneck link utilization and smaller loss rate comparing to the DropTail link. It means that the R flows get better goodput if employing RD services. One can also see that whereas the loss rate of class R with DropTail is bigger, the throughput is approximately the same with both the RD services and DropTail. Besides, the misbehaving throughput-greedy flow gains the performance after switching to class D for almost the whole range of the varied parameter, which is expected, as VoIP flows do not consume the whole throughput provided by RD routers.

Transient Behavior

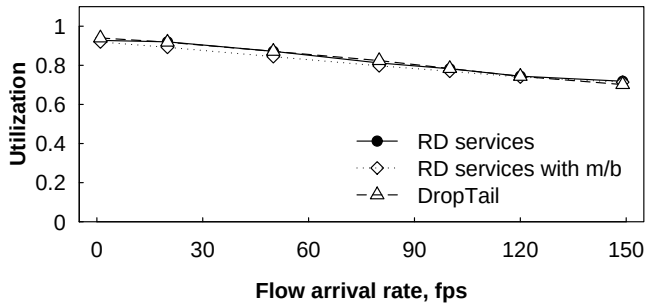
In this experiment, VoIP flows join the network during the whole experiment lasting for 600 sec. There are 500 VoIP flows that start coming to the network from the beginning. The arrival process is described by the Exponential distribution with the average 1 fps. Whereas the average MOS for DropTail is 2.97, MOS with the RD services is 4.16. The throughput of class D is 84.45% and 83.85% for RD services and DropTail, respectively. The loss rate of class D is 0.38% and 0.4% with the RD services and the DropTail link, respectively. Thus, one can observe that the RD services deliver better service for VoIP in the dynamic scenario.

4.7.2 Influence of Propagation RTT

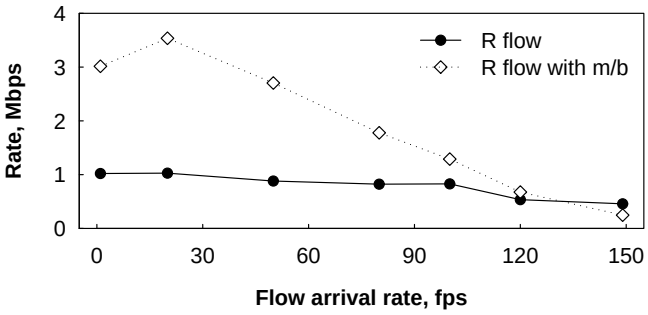
To explore how propagation RTT affects the quality of VoIP, we modify the propagation RTT of VoIP flows in the interval between 30 ms and 800 ms. The delay of the bottleneck link is 10 ms. In Figure 4.32, we notice that RD services reveal better performance for VoIP over the whole range of the varied parameter. In particular, at least medium quality of



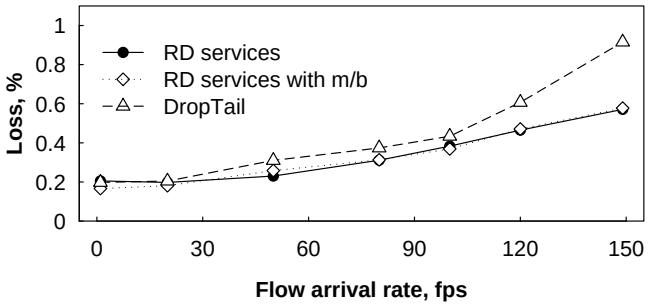
(a)



(b)

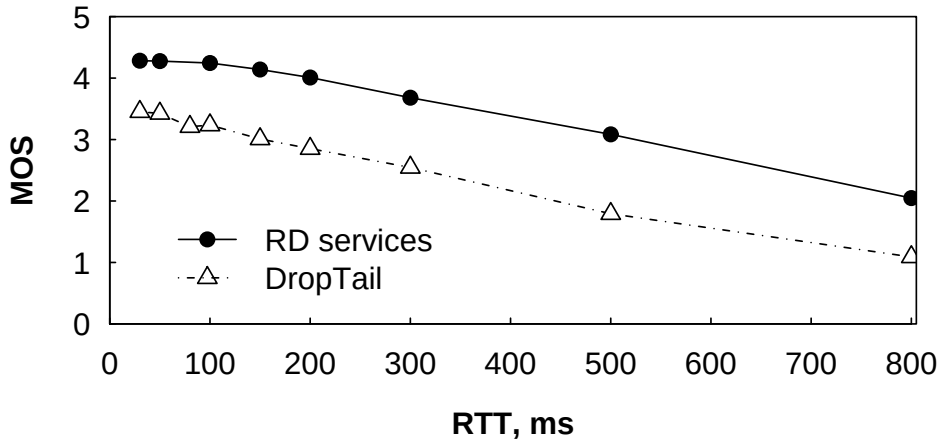


(c)

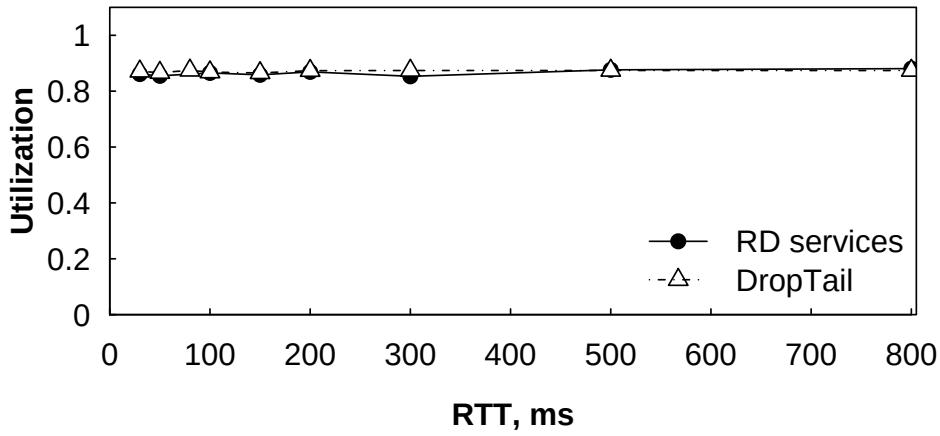


(d)

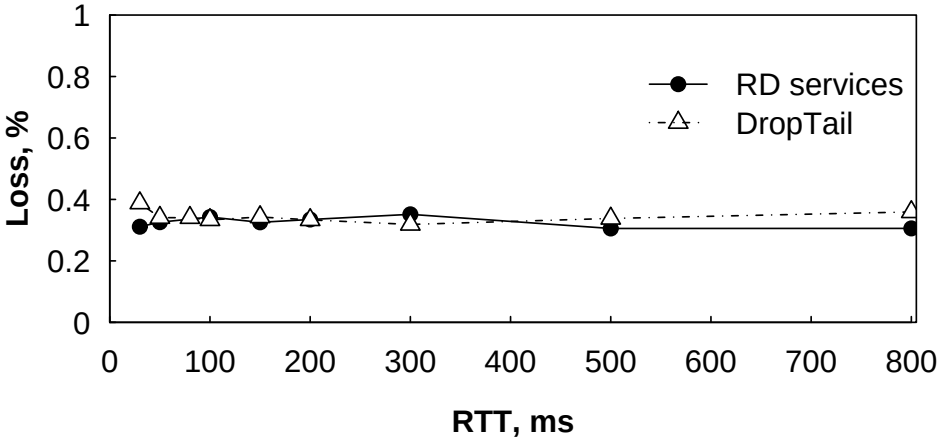
Figure 4.31: Performance for different web-like traffic intensities: (a) Average MOS; (b) average utilization of class R; (c) Average throughput of ”misbehaving” throughput-greedy flow; (d) Average loss rate of class R.



(a)



(b)



(c)

Figure 4.32: Performance for different propagation RTTs: (a) Average MOS; (b) Average utilization of class R; (c) Average loss rate of class R.

voice is supported for the propagation RTTs up to 300 ms with the RD link. On the other hand, the DropTail link can support the same quality of voice for the propagation RTTs no more than 50 ms. Besides, the R traffic gets the same bottleneck link utilization and loss rate as in the case of the DropTail link.

4.7.3 Scalability Properties

Influence of Long-lived R Flows

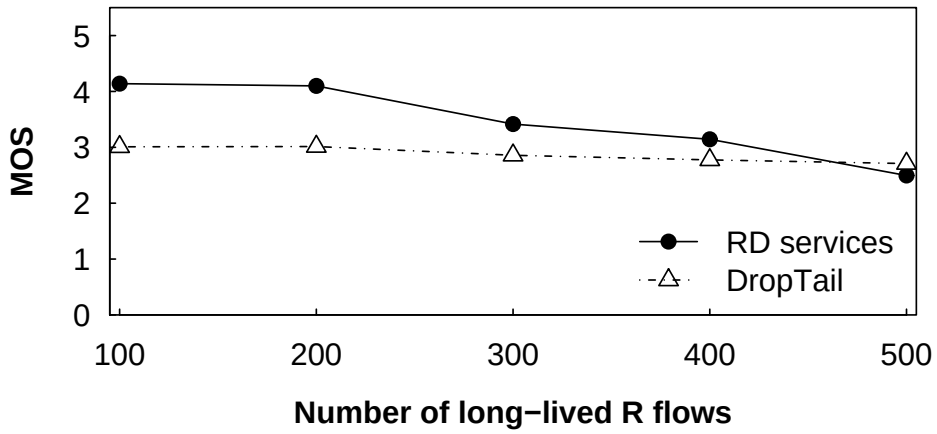
In this experiment, we change the number of long-lived R flows flows in the interval between 100 and 500. Figure 4.33 shows that VoIP flows receive better service comparing to the DropTail link through almost the whole range of the varied parameter, whereas R flows reveal the same performance concerning link utilization and loss rate. The deterioration of the voice quality with the increase of the number of R flows, when MOS reduces from 4.14 till 2.5, is because the decrease of the D buffer size increases the loss rate.

Impact of the VoIP Population Size

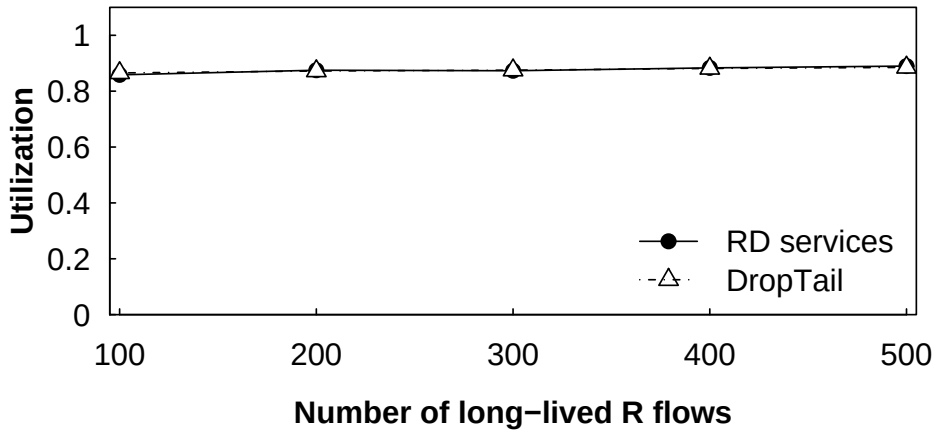
To examine the scalability of the design concerning the population of VoIP flows, we vary the number of them in the interval between 100 and 500. In Figure 4.34, we observe that the number of VoIP flows does not affect the quality of VoIP. In particular, MOS with the RD services stays in the range between 4.14 and 4.16 whereas MOS with the DropTail link is between 3.01 and 3.13. On the other hand, R flows reveal the same performance for both the schemes concerning the link utilization, which is between 82% and 87%, and loss rate, which values are between 0.32% and 0.38%.

4.7.4 Partial Deployment

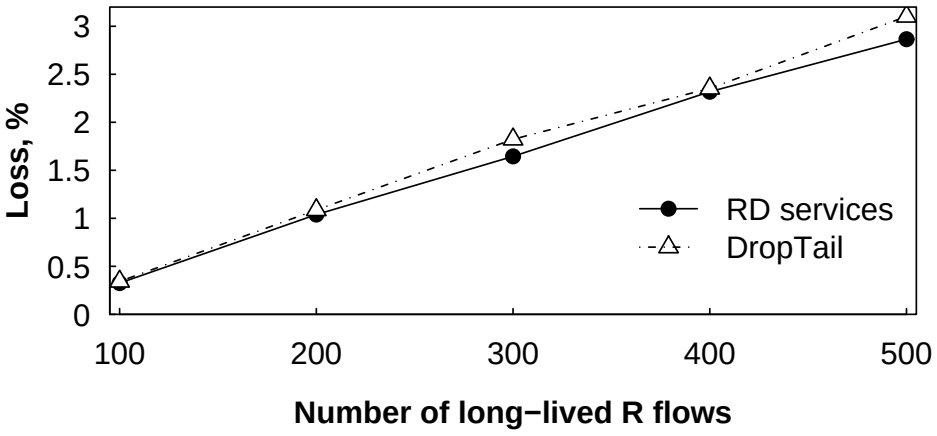
This experiment explores the situation when VoIP flows use the service provided by two different ISPs. There is one bottleneck within each ISP, 50 VoIP flows, 50 R flows going through both the ISPs, and two groups of 50 R flows each so that each group goes only through one ISP. In particular, we consider two deployment scenarios. The first one assumes that only one ISP has deployed the RD services, whereas in the second one both



(a)

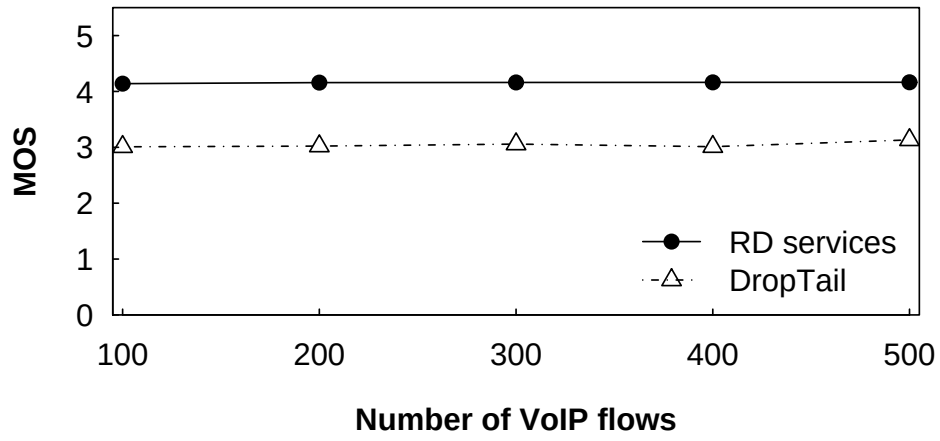


(b)

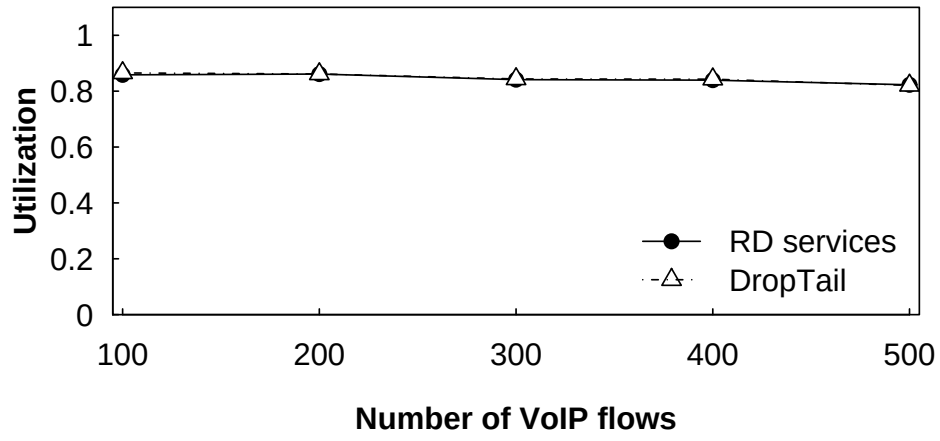


(c)

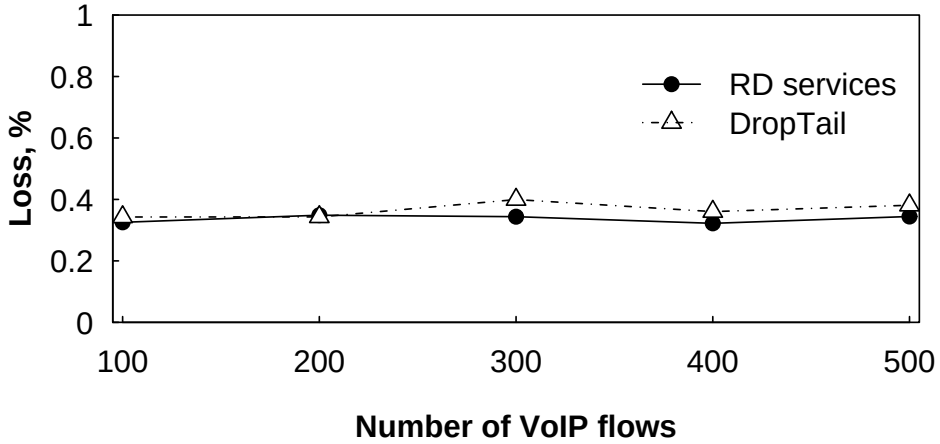
Figure 4.33: Performance for different numbers of R flows: (a) Average MOS; (b) Average utilization of class R; (c) Average loss rate of class R.



(a)

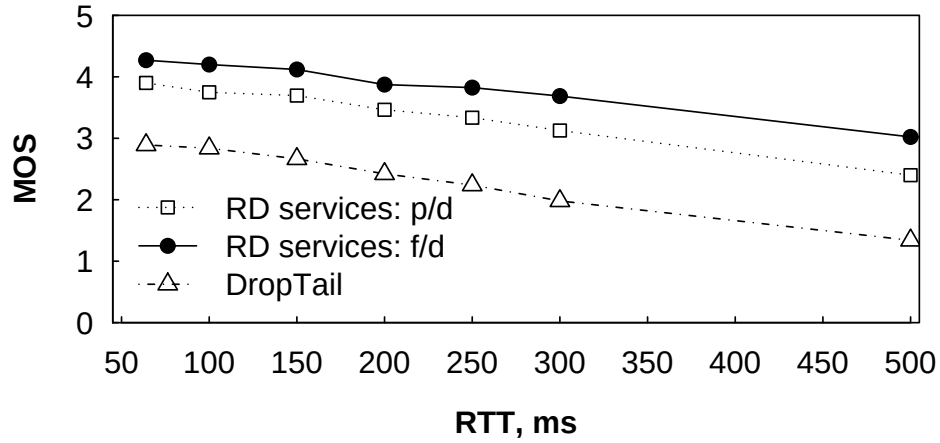


(b)

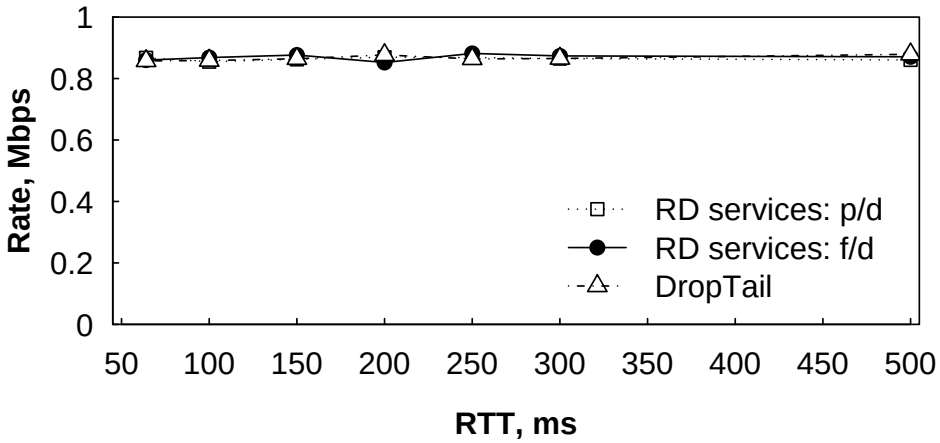


(c)

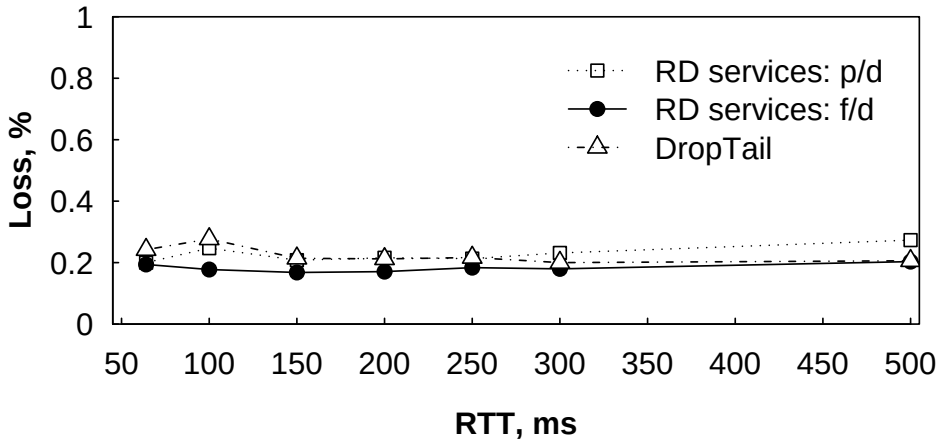
Figure 4.34: Performance for different numbers of VoIP flows: (a) Average MOS; (b) Average utilization of class R; (c) Average loss rate of class R.



(a)



(b)



(c)

Figure 4.35: Performance under partial deployment for different propagation RTTs: (a) Average MOS; (b) Average per-flow throughput of class R; (c) Average loss rate of class R.

the ISPs have adopted the proposed architecture. The propagation RTT of the VoIP flows is varied in the range between 64 ms and 500 ms. In Figure 4.35, we observe that even under the partial deployment of the RD services, which is labeled as ”p/d” in the graph, the VoIP flows get better service. Moreover, the full deployment of the design labeled as ”f/d” further improves the voice quality. More importantly, the improvements of VoIP quality do not affect the service delivered to the R class concerning the flow rates.

4.8 Summary

This Chapter reconsidered the problem of aligning network services with application needs. It argued that design of a new service should put an emphasis on providing both end users and ISPs with incentives to adopt the new service despite its partial deployment. It designed and implemented the RD services that offer two best-effort services of low queuing delay or higher throughput. The RD router supports the services with two queues per output link, one queue per traffic class. The extensive evaluation revealed that the design supports the intended delay-rate differentiation with a reasonable level of losses in a wide variety of settings.

Chapter 5

TMD: Accelerating Flow Completion through Dynamic Buffer Allocation

FCT (Flow Completion Time) has been recognized as a major performance metric for short-lived flows, e.g., for short-lived web flows. The problem of reducing FCT of short-lived flows without doing harm to long-lived flows is commonly tackled by treating the two types of traffic differently at a congested link. However, the existing solutions struggle with classifying flows as short-lived versus long-lived so that to perform successfully under various network conditions. While TCP (Transport Control Protocol) remains the dominant transport protocol in the Internet and for web applications in particular, this Chapter proposes to differentiate the forwarded traffic based on the current TCP operation modes of the flows. Because short-lived TCP flows tend to operate mostly in the slow-start mode, this Chapter proposes TMD (TCP Mode Differentiation), which categorizes slow-start TCP flows as belonging to class S and treats the other traffic as class A. Then, it explores a TMD buffer management scheme that dynamically partitions the shared FIFO (First-In First-Out) link buffer between the two traffic classes S and A. TMD imposes a low overhead and is amenable to incremental deployment in the Internet. Nevertheless, the extensive experimentation for a wide range of network settings demonstrates that TMD decreases FCT of short-lived flows significantly without reducing throughput of long-lived flows.

5.1 Analytical Foundation

To understand why the TCP mode might be a promising basis for reducing flow completion time, let $L = L_S(l_S) + L_A(l_A)$ denote the amount of data delivered by a TCP flow where $L_S(l_S)$ represents the data delivered in the slow-start mode, $L_A(l_A)$ captures the delivered

during congestion avoidance and the other (e.g., fast-recovery) modes, and l_S and l_A are respective loss rates. With $g_S(l_S)$ and $g_A(l_A)$ denoting respectively goodput of the TCP flow in the slow-start mode and all the other modes, one can express completion time of the flow as:

$$FCT = \frac{L_S(l_S)}{g_S(l_S)} + \frac{L_A(l_A)}{g_A(l_A)}. \tag{5.1}$$

While loss rates l_S and l_A are interrelated, a change in their balance might lead to new values l'_S and l'_A that provide the flow with smaller completion time:

$$FCT' = \frac{L_S(l'_S)}{g_S(l'_S)} + \frac{L_A(l'_A)}{g_A(l'_A)} < FCT. \tag{5.2}$$

5.2 Conceptual Design

This section explores whether the desired balance between the loss rates is achievable through dynamic partitioning of the congested link buffer. More specifically, TMD (TCP Mode Differentiation) design categorizes slow-start TCP flows as belonging to class S (Start), treats the other traffic as class A (Avoidance), and splits the link buffer between the two classes to maintain the ratio of per-flow buffer shares b_S and b_A for classes S and A at k:

$$\frac{b_S}{b_A} = k, \tag{5.3}$$

where k is the only parameter of the proposed scheme. Hence, if n_S and n_A represent respectively the number of flows from classes S and A, TMD splits total space B of the link buffer between classes S and A into partitions B_S and B_A sized as:

$$B_S = \frac{kn_SB}{n_A + kn_S} \quad \text{and} \quad B_A = \frac{n_AB}{n_A + kn_S}. \tag{5.4}$$

5.3 Design Details

5.3.1 Routers

A TMD router employs Equations 5.4 to partition the link buffer between classes S and A. When a packet arrives, the router determines the class of the packet. If the buffer partition

```
p ← received packet;
x ← class of p;
S ← size of p;
if q_x + S ≤ B_x
      append p to the tail of the link queue;
      q_x ← q_x + S;
 else
      discard p
```

Figure 5.1: TMD router operation upon receiving a packet destined to the link where B_x and q_x respectively refer to the buffer partition and queue sizes for traffic class x.

for this class has enough free space for the packet, the router append the packet to the tail of the link queue. Otherwise, the router discards the packet. Figure 5.1 depicts the above simple algorithm.

Whereas the dynamic buffer partitioning relies on n_S and n_A, the design estimates the numbers of flows for classes S and A using the timestamp-vector algorithm [64]. The algorithm maps each received packet into an element of the array called a timestamp vector employing a hash function. The timestamp vector accommodates b elements. The algorithm inspects the timestamp vector with period T and considers a flow inactive if the timestamp vector did not register any packets of the flow during the last period E. Following the guidelines in [28] and assuming $E = 1$ s, 10^5 active flows, and standard deviation $\epsilon = 0.05$, we use $b = 18,000$ as the default setting for the timestamp vector size.

A TMD router updates n_S and n_A with period T. Upon each update, the router adjusts the buffer partitions for classes S and A. Even if n_S or n_A is zero, the router allocates non-zero buffers for both classes. The experimental results suggest that specific partition sizes are not too important. In the reported experiments, the design initializes the buffer partitions to $B_S = \frac{4kB}{1+4k}$ and $B_A = \frac{B}{1+4k}$, which correspond to the 4:1 ratio between the numbers of TCP slow-start flows and all the other flows.

The TMD design partitions the buffer rigidly in the sense that if the partition for one class is full, arriving packets of this class are discarded regardless of the space availability in the other partition. An alternative implementation could allow the excessive traffic to utilize the unused space in the other partition with the obligation to return the space immediately (i.e., through discard of already enqueued packets) whenever traffic of the lending class needs the space. It is not clear whether the flexible alternative would have a positive impact

on flow performance. We do not undertake this alternative due to complexities associated with the discard of already enqueued packets.

Since the link is served from the single FIFO queue, packets within a TCP flow do not change their order while traversing the TMD router. By preserving the FIFO link scheduling, the TMD design avoids the well-known problems caused for TCP performance by packet reordering.

5.3.2 End Hosts

In principle, a router might be able to infer the current mode of a TCP flow without any assistance from the sender [11]. However, such inference would put a significant burden on the router. Instead, the design delegates the TMD classification to the senders. The TCP slow-start sender marks the packets of its flow as belonging to class S by setting one of the currently reserved and unused bits in the TCP segment headers. The default setting of the bit indicates that the packet should be treated as belonging to class A. Thus, TMD preserves the TCP segment format. Also, the incrementally deployable scheme views legacy flows as belonging to class A and provides them with a similar service as in the existing Internet.

Except for the class signaling, TMD does not require any support from end hosts. In particular, TMD does not modify end-to-end transport protocols.

5.3.3 TMD Partitioning Parameter

The only parameter of the TMD scheme is k, the ratio of the per-flow buffer shares for classes S and A. While robust setting of parameters has turned into a central concern for prior designs [35], we start with a qualitative discussion on how to set k. If k is too large, smaller FCT of short-lived flows comes at the expense of long-lived flows whose throughput drops due to insufficient buffer space. If k is too small, FCT of short-lived flows increases.

For further insights into setting k, this chapter experiments in ns-2 [72] with a non-partitioned FIFO DropTail buffer at the core bottleneck link of a single-bottleneck dumbbell topology. The bottleneck link has capacity 100 Mbps and propagation delay 50 ms. There are N long-lived flows with propagation RTTs distributed uniformly between 104 ms and 300 ms. One

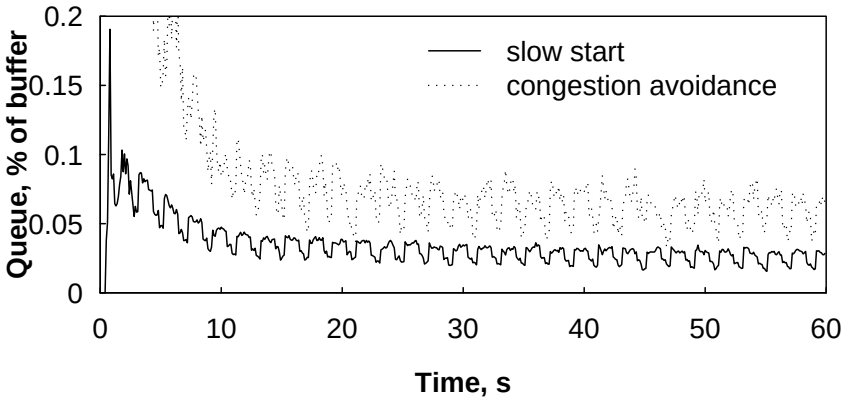
web server generates TCP traffic with propagation delay 100 ms. Web flows arrive according to the Exponential distribution with average intensity a. The sizes of web flows are Pareto-distributed with average b and shape index 1.3. All flows use packets sized to 1 KB. The long-lived flows start during the first second of the experiment. Web traffic is generated throughout the experiment. The duration of the experiment is 60 s. N, a, and b are varied, and the average per-flow queue sizes at the bottleneck link for slow-start flows and congestion-avoidance flows are measured. The measurements are performed every 0.1 s. Figure 5.2 shows that the ratio of the per-flow queue sizes is close to 0.5. These results for the non-partitioned buffer suggest that the explored range of k values for the partitioned TMD design should include $k = 0.5$.

5.4 Performance Evaluation

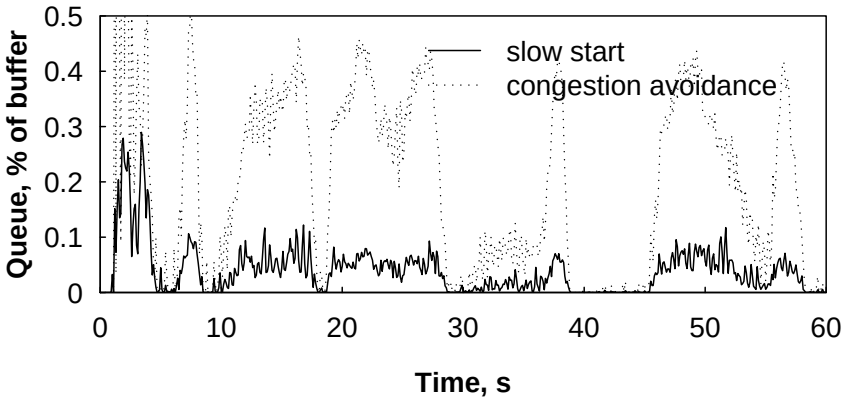
This section evaluates performance of the TMD design in ns-2 [72]. First, it takes a close look at setting k. Then, it uses the chosen value of k in the extensive evaluation of TMD.

5.4.1 Parameter Setting

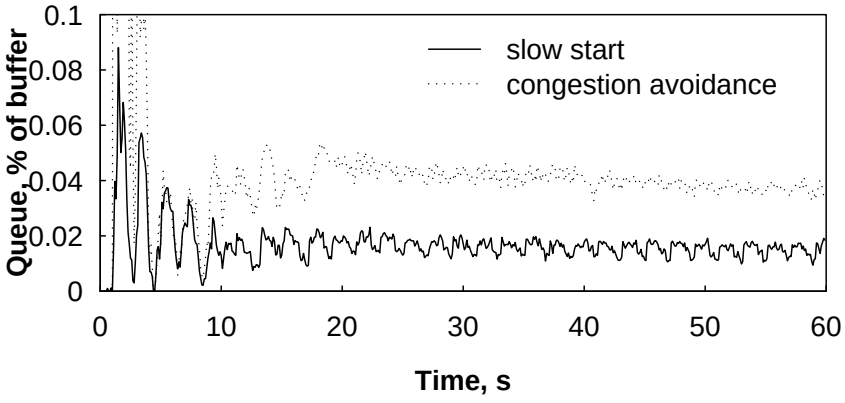
To find reasonable values for the TMD main control parameter k, we experiment in a single-bottleneck dumbbell topology with short-lived and long-lived flows. In each experiment, every short-lived flow has the same size and propagation RTT. For each experimental setting, the size of the short-lived flows is varied from 1 KB up to 150 KB. The packet size of all the flows is 1 KB. The bottleneck link capacity is 100 Mbps, each access link is 200 Mbps. The delay of the bottleneck link is 50 ms, and the maximum queuing delay at the bottleneck link buffer is 250 ms. There are 100 long-lived flows, and propagation delays of them are uniformly distributed in the range between 104 ms and 300 ms. The short-lived flows are generated by one web server, and they have propagation RTT equal to 100 ms. Five experiments are performed for each experimental settings, and the average value is calculated. This section reports the average flow completion time of the short-lived flows and the throughput of the long-lived flows in terms of the bottleneck link utilization. It runs the experiments for the DropTail link and the following values of k: 0.1, 0.5, 1.0, 2.0, 10.0.



(a)



(b)



(c)

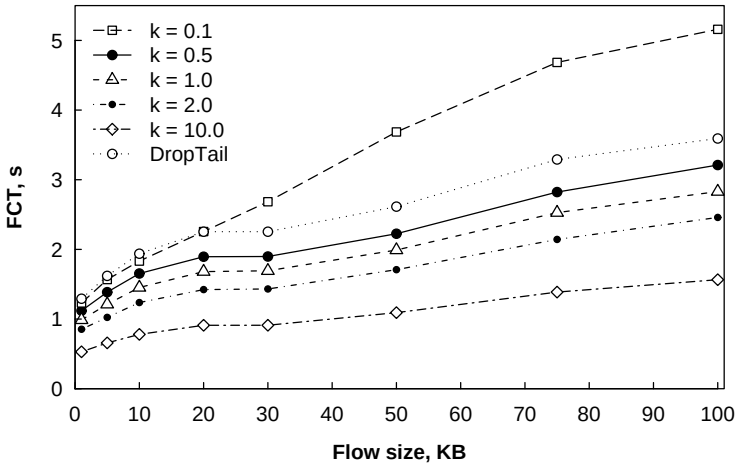
Figure 5.2: Average per-flow queue sizes for TCP slow-start flows and TCP congestion-avoidance flows with the non-partitioned FIFO DropTail buffer at the bottleneck link: (a) N = 0, a = 400 fps, b = 10 KB; (b) N = 500, a = 400 fps, b = 30 KB; (c) N = 100, a = 50 fps, b = 30 KB.

Different intensities of short-lived flows

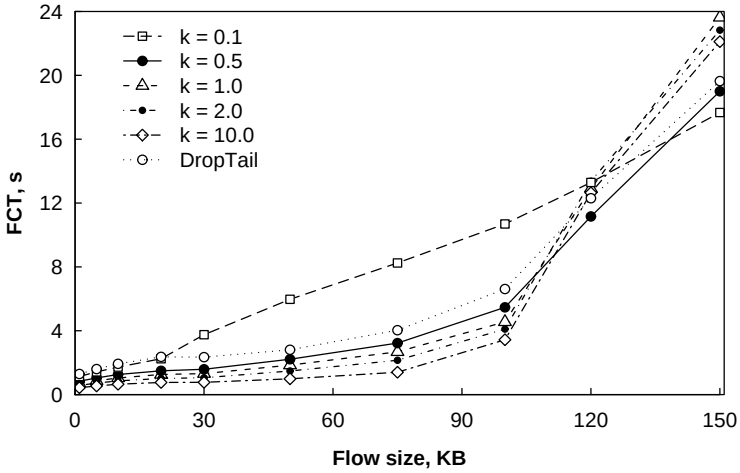
In this set of experiments, we use three different values of the intensity of web-like flows: a_1 = 20 fps, a_2 = 100 fps, and a_3 = 200 fps. Figures 5.3, 5.4, 5.5 report the results for the whole range of the varied parameter, and the average FCT for web-like flows with sizes between 1 KB and 30 KB. One can observe that for intensities a_1 and a_2 the performance of the short-lived flows corresponds to the qualitative description that has been given in Section 5.3.3. In particular, a big value of k such as 10.0 gives better FCT for the short-lived flows with sizes up to 30 KB comparing to the DropTail link, but FCT for bigger flows is larger comparing to the DropTail link. On the other hand, a small value of k such as 0.1 gives better FCT for the short-lived flows with size bigger than 70 KB for intensity of the short-lived flows a_3. In addition, FCT decreases if changing intensity a_3 by intensity a_2. Concerning intensity a_1, increasing the value of k improves FCT for all the values of flow sizes. It is explained that the value of the size of the short-lived flows starting from which bigger values of k deteriorate FCT, is bigger for a_1 comparing to a_2 and a_3 due to the smaller loss rate. Moreover, the average FCT of the short-lived flows with the DropTail link is bigger by up to 54% depending on the flow size for all the sizes of the short-lived flows for $k = 0.5$. The throughput of the long-lived flows is approximately the same for TMD with $k = 0.5$ and the DropTail link. On the other hand, the scheme with $k = 0.1$ reveals better performance of the long-lived flows for a_2, a_3 than the DropTail link, and deteriorates the throughput of the long-lived flows for a_2 if employing values of k bigger than 0.5. Thus, we conclude that $k = 0.5$ is close to optimal value in the design. As expected the scheme reveals better performance if the duration of slow start is comparable with the duration of congestion avoidance. The similarity of the throughput of the long-lived flows of the design with $k = 0.5$ and the DropTail link is explained that the long-lived flows operate in congestion avoidance for the vast majority of their existence. Thus, while not deteriorating the performance of long-lived flows, the TMD design improves the performance of short-lived flows.

Different propagation RTTs of web-like flows

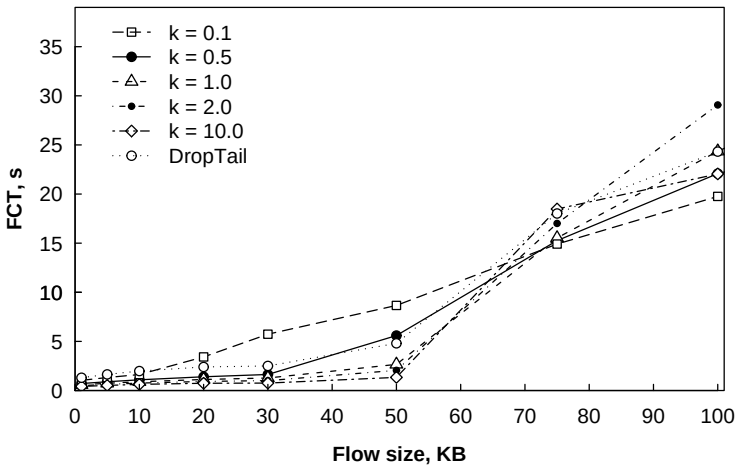
To evaluate the design for different propagation RTTs for web-like flows, the mentioned RTTs are modified between 30 ms and 300 ms. The intensity of the short-lived flows is 100 fps. The size of the short-lived flows is varied in the range between 1 KB and 150 KB. Figures 5.6 and 5.7 report the results for the whole range of the varied parameter,



(a)

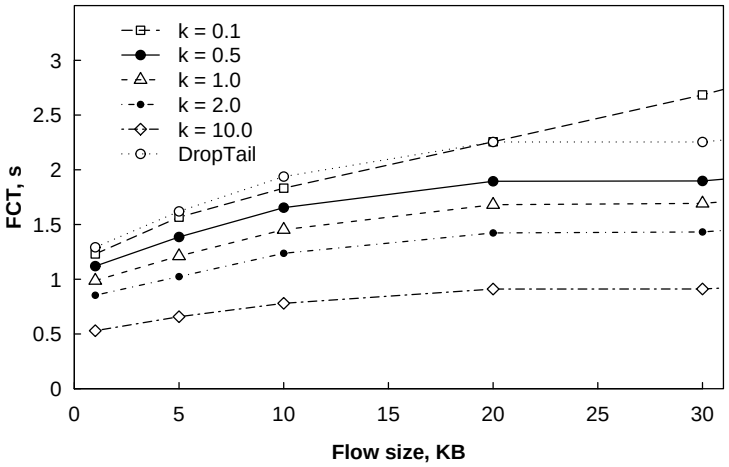


(b)

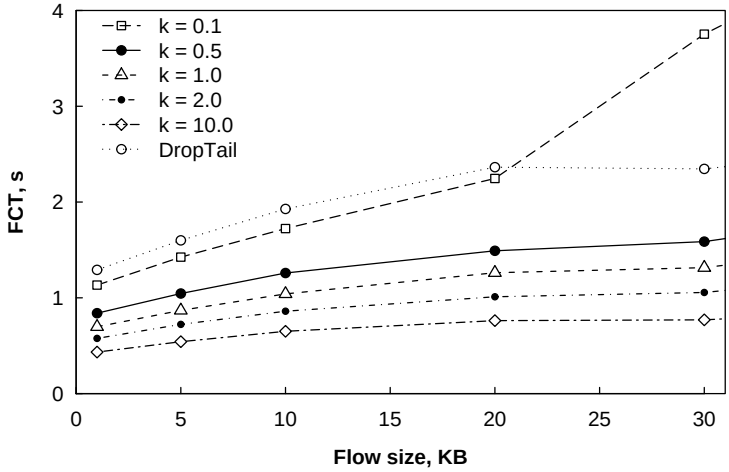


(c)

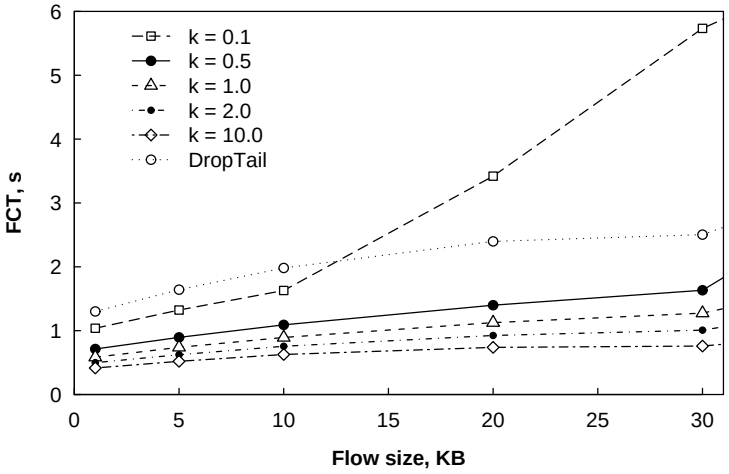
Figure 5.3: Average FCT of short-lived flows for their different intensities: (a) 20 fps; (b) 100 fps; (c) 200 fps.



(a)

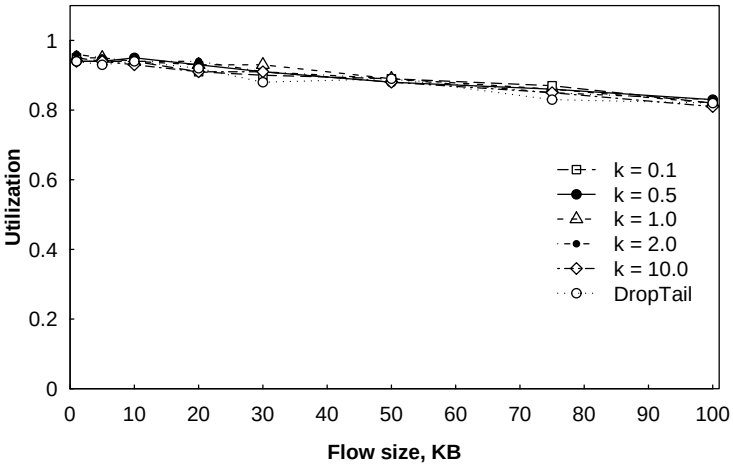


(b)

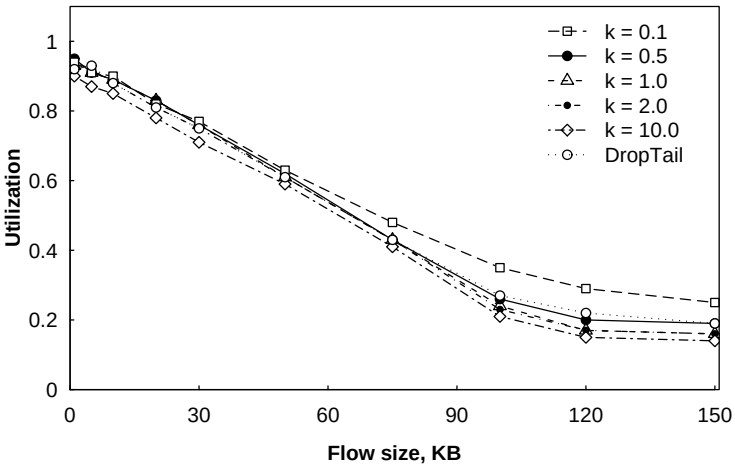


(c)

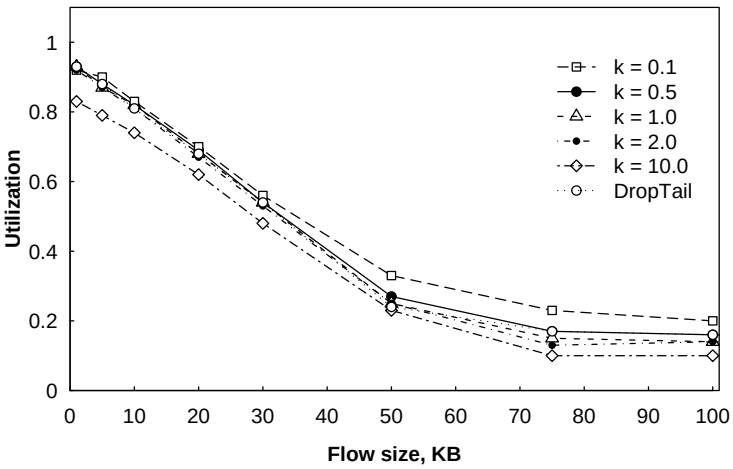
Figure 5.4: Zoomed average FCT of short-lived flows for their different intensities: (a) 20 fps; (b) 100 fps; (c) 200 fps sec.



(a)

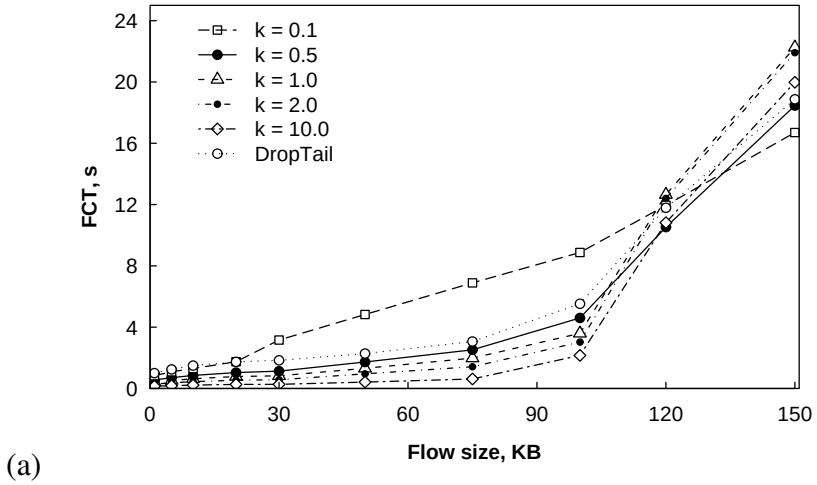


(b)

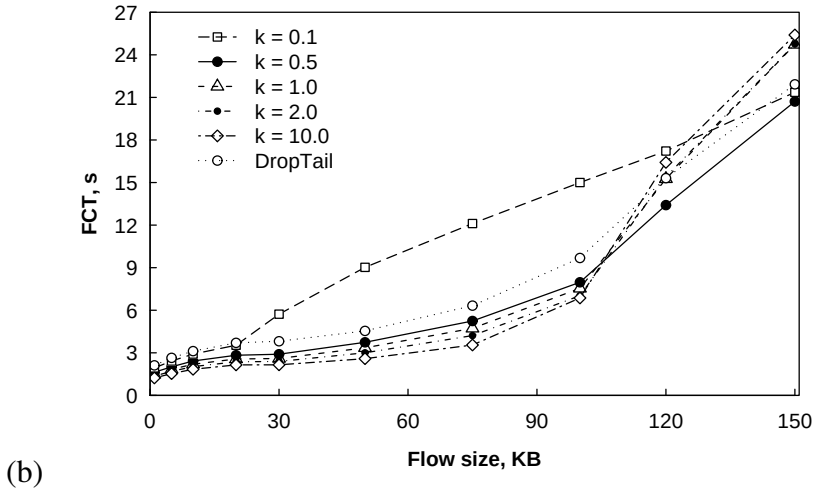


(c)

Figure 5.5: Average throughput of long-lived flows for their different intensities: (a) 20 fps; (b) 100 fps; (c) 200 fps.



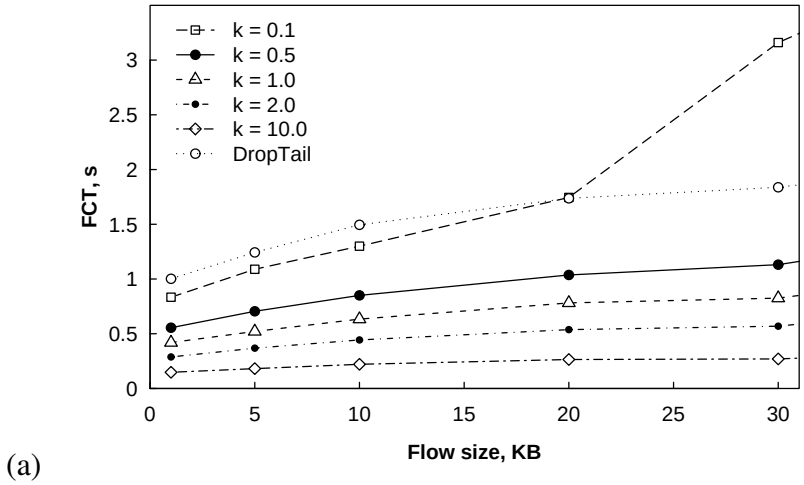
(a)



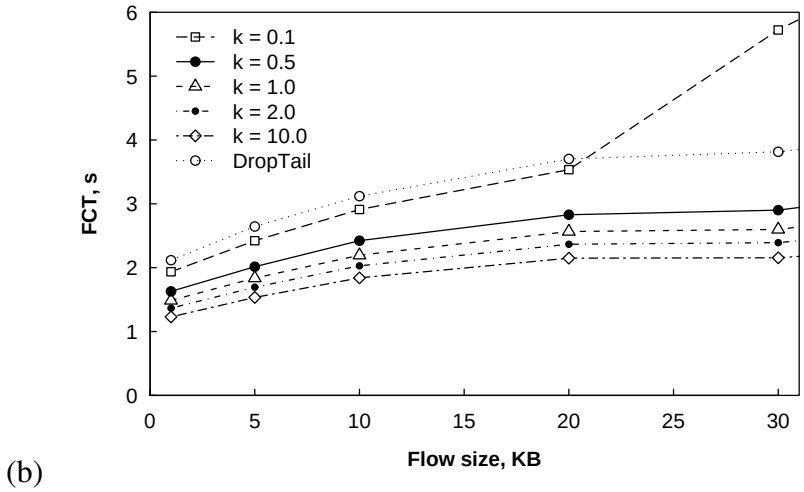
(b)

Figure 5.6: Average FCT of short-lived flows for their different propagation RTTs: (a) RTT = 30 ms; (b) RTT = 300 ms.

and the average FCT for the short-lived flows with sizes between 1 KB and 30 KB. The performance with the TMD link is qualitatively similar to the one revealed in Section 5.4.1. In particular, $k = 0.5$ decreases FCT of the short-lived flows for the whole range of their sizes without degrading the throughput of the long-lived flows. The relative improvement of FCT for $k = 0.5$ is bigger when the propagation RTT of the short-lived flows is 30 ms. In addition, bigger values of k improve FCT for the short-lived flows with sizes up to 100 KB, but deteriorate the throughput of the long-lived flows. On the other hand, smaller values of k gain the performance of the short-lived flows with sizes larger than 100 KB degrading FCT of the smaller short-lived flows. Again, one can observe that $k = 0.5$ seems to be the most appropriate value that decreases FCT of the short-lived flows of different sizes without diminishing the throughput of the long-lived flows.



(a)

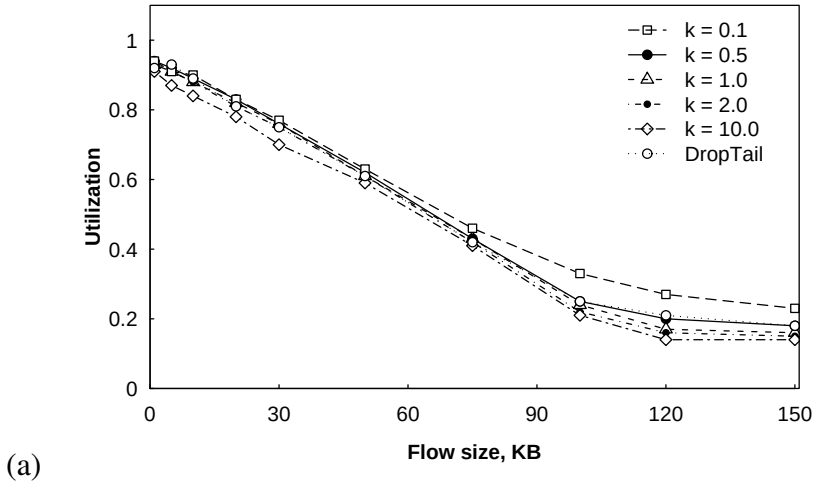


(b)

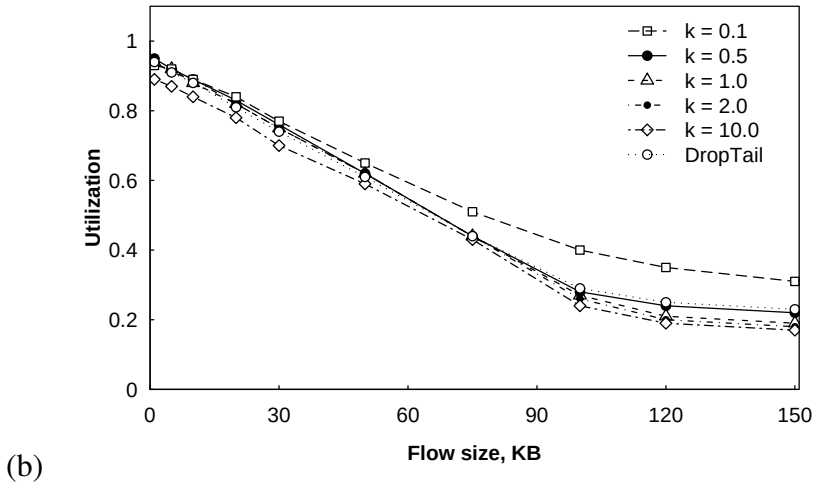
Figure 5.7: Zoomed average FCT of short-lived flows for their different propagation RTTs: (a) RTT = 30 ms; (b) RTT = 300 ms.

5.4.2 Experimental Settings

Now and till the end of this section, the performance of the TMD design is evaluated with $k = 0.5$ chosen in accordance with the above experimental study. This series of experiments uses the same settings as in Section 5.4.1, except for the bottleneck link propagation delay, which is set to 10 ms. We perform five experiments for each experimental settings, and calculate the average value. We report the average goodput of the web-like traffic as the average of the goodput of each web-like flow. We calculate the goodput of a web-like flow as the ratio between the flow size and its FCT. We also plot the minimum and the maximum values for each experimental settings. To compare the TMD design, we run the experiments with the same settings for the DropTail link.



(a)



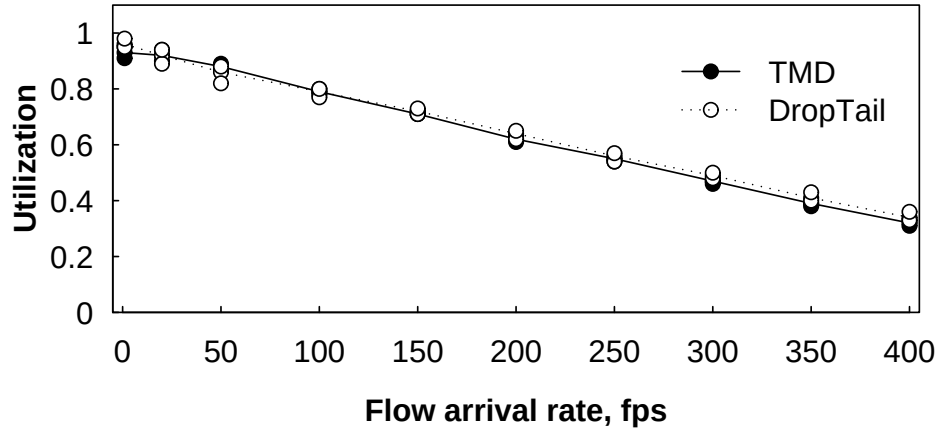
(b)

Figure 5.8: Average throughput of long-lived flows for their different propagation RTTs: (a) RTT = 30 ms; (b) RTT = 300 ms.

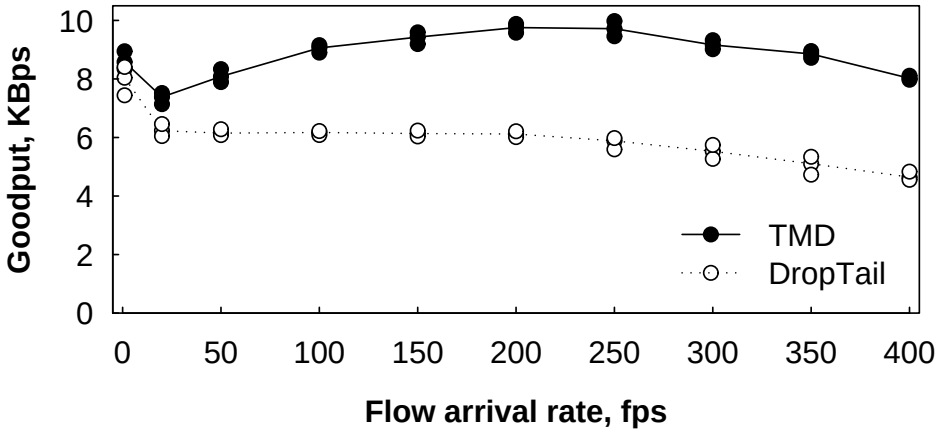
5.4.3 Dynamic Changes

Influence of Web-like Traffic

This section studies the influence of the intensity of the web-like traffic for the arrival rate of web-like flows between 1 fps and 400 fps. The flow size is described by the Pareto distribution with the average of 30 KB and shape index 1.3. Figure 5.9 demonstrates that the TMD link improves the performance of web-like flows by 20% for almost the whole range of the varied parameter. For the intensity of web-like traffic of at least 300 fps, the gain of the considered performance is up to 72%. The long-lived flows with the design demonstrate the same performance as with the DropTail scheme.



(a)



(b)

Figure 5.9: Influence of the intensity of web-like traffic: (a) Throughput of long-lived flows; (b) Goodput of web-like flows.

Heavy-tailed arrivals of flows

The purpose of these experiments is to explore the performance of the design for heavy-tailed arrivals of flows. Web-like flows join the network according to the Pareto distribution with the average intensity 100 fps and shape index 1.1. The flows sizes of the web-like flows are Pareto-distributed with shape index 1.3 and average size L_{avg}. This section runs the experiments for L_{avg} = 30 KB and L_{avg} = 100 KB. Figure 5.10 demonstrates better performance of the proposed design for the web-like traffic and the similar throughputs of the long-lived flows with the TMD and DropTail links. In particular, the improvement from TMD for the web-like flows is up to 62% for L_{avg} = 30 KB and up to 36% for L_{avg} = 100 KB.

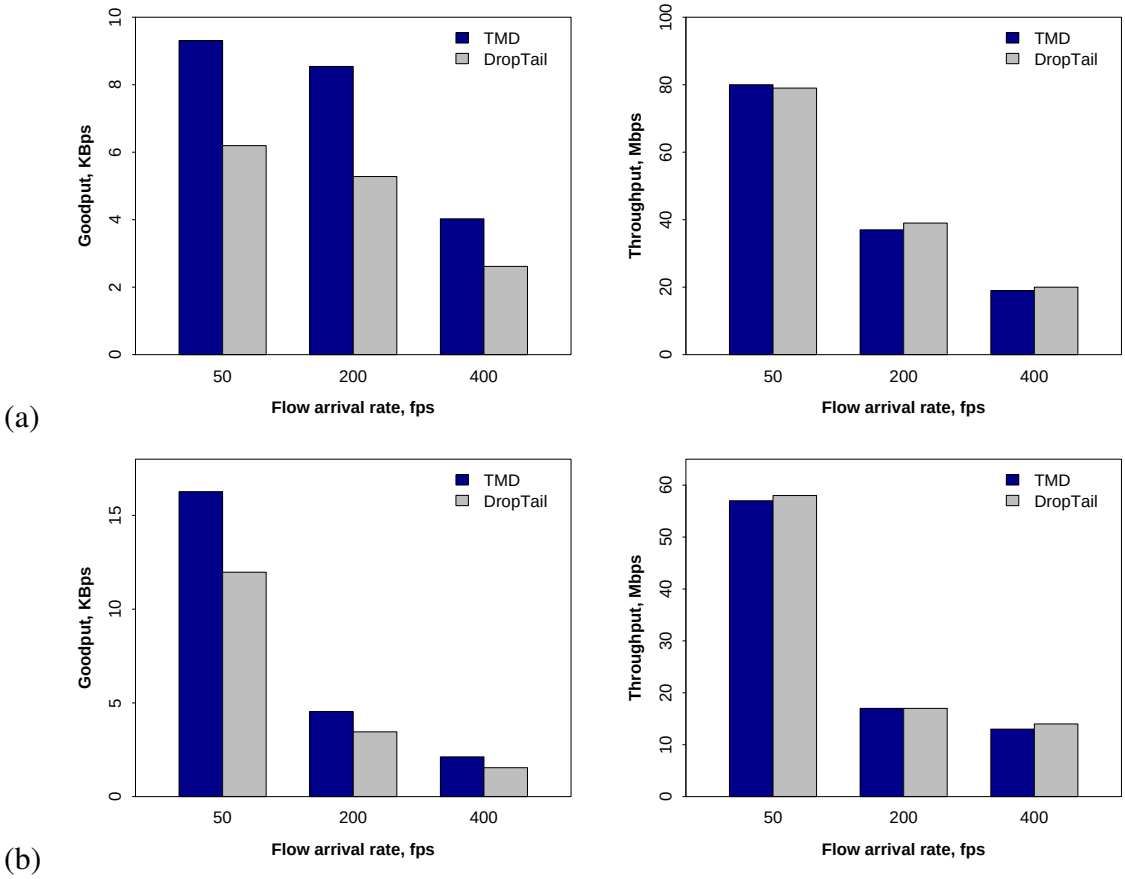
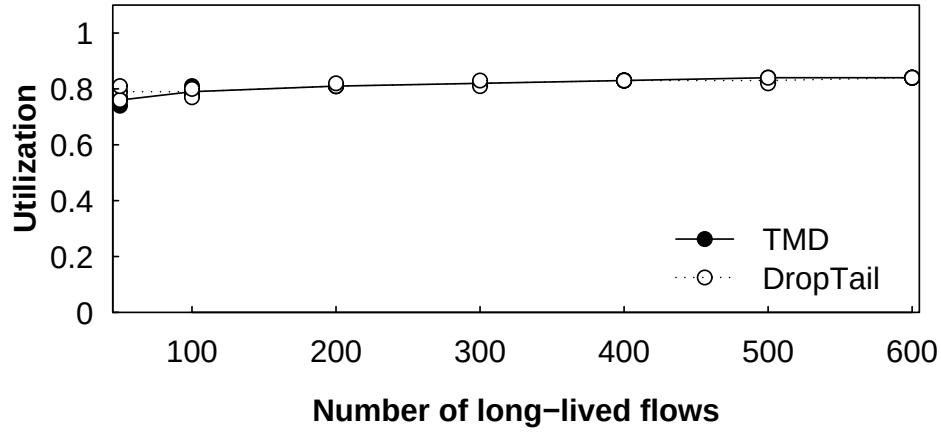


Figure 5.10: Average goodput of web-like flows and throughput of long-lived flows for different average sizes of web-like flows: (a) L_{avg} = 30 KB; (b) L_{avg} = 100 KB.

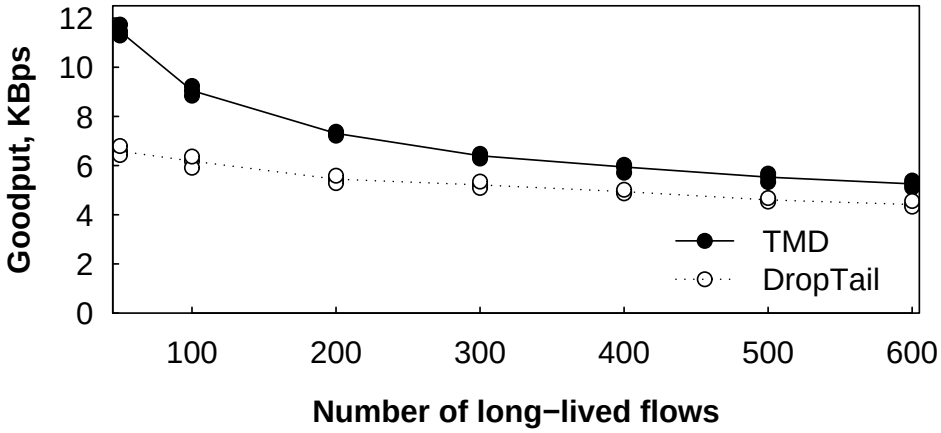
5.4.4 Scalability Properties

Population Scalability

To explore the performance concerning the population scalability, the number of the long-lived flows is varied between 50 and 600. The intensity of web-like traffic is 100 fps. Figure 5.11 demonstrates that whereas the long-lived flows have the same throughput with the TMD and DropTail links, the design improves the goodput of the web-like traffic by 19%-37%.



(a)

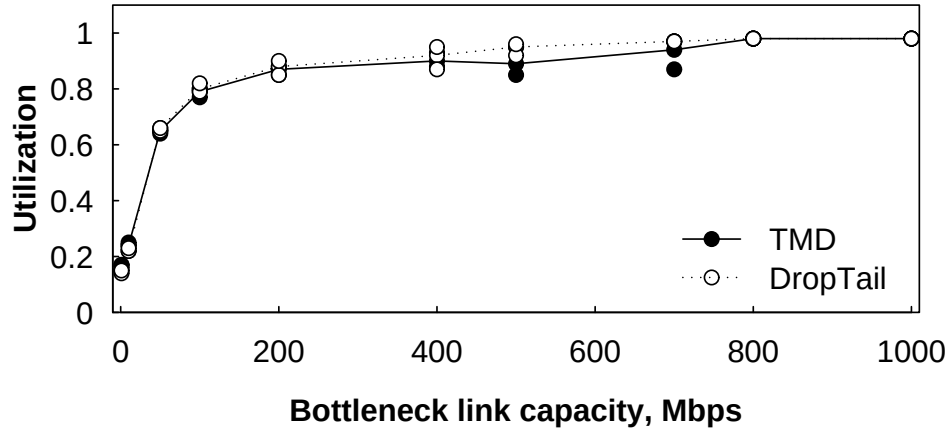


(b)

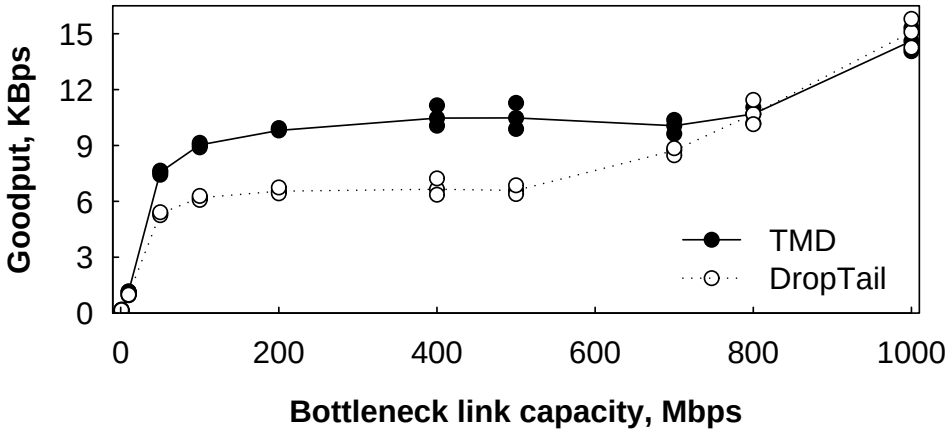
Figure 5.11: Scalability of the TMD design with respect to the number of long-lived flows: (a) Throughput of long-lived flows; (b) Goodput of web-like flows.

Scalability with the Bottleneck Link Capacity

To study the scalability concerning the bottleneck link capacity, the bottleneck link capacity is modified in the range between 1 Mbps and 1 Gbps. The propagation RTT of web-like flows is 100 ms. Figure 5.12 demonstrates that whereas the performance of TMD and DropTail schemes are similar for the long-lived flows except for the bottleneck link capacity of 500 Mbps, for which the scheme produces a slightly smaller throughput. However, the design reveals advantage over the DropTail link concerning the performance of the web-like traffic. In particular, TMD improves the goodput of the web-like flows by up to 55%. The decrease of the goodput of the web-like traffic for small values of the bottleneck link capacity down to almost zero is explained by the very high loss rate. The similarity of the performance of the web-like flows for large bottleneck link capacities is due to very



(a)



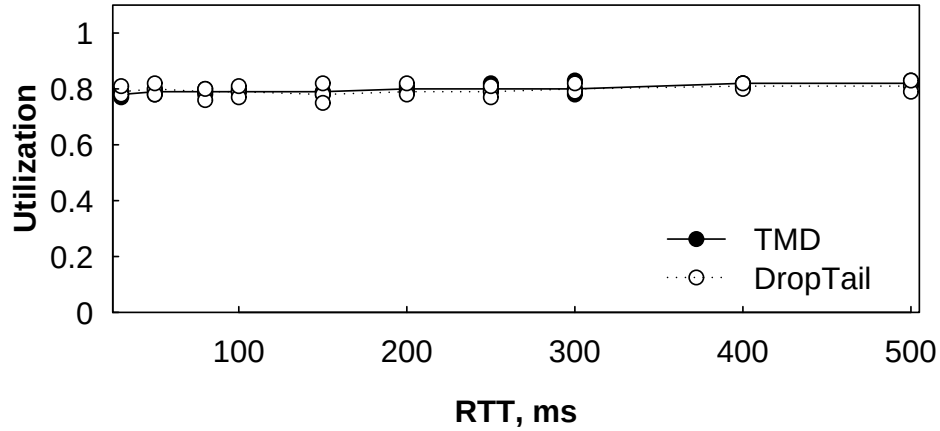
(b)

Figure 5.12: Scalability of the TMD design with respect to the bottleneck link capacity: (a) Throughput of long-lived flows; (b) Goodput of web-like flows.

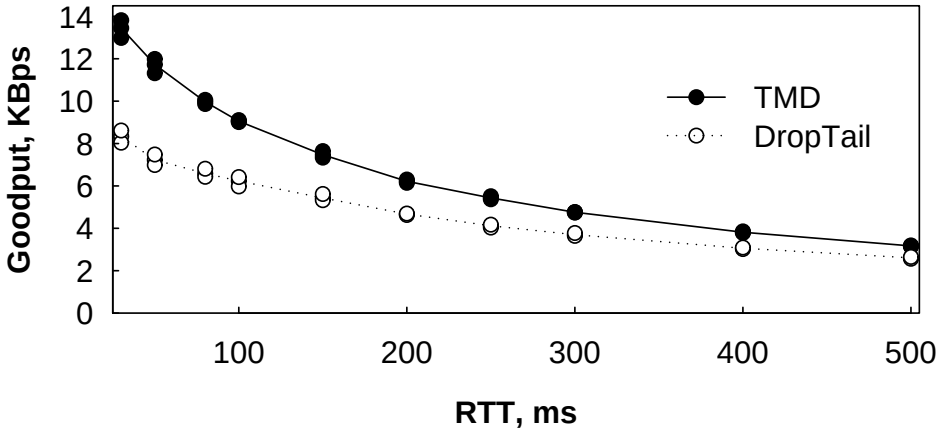
small loss rates, which limits the space for redistribution of losses between slow start and congestion avoidance.

5.4.5 Influence of the Propagation RTT of the Web-like Traffic

In this experiments, the propagation RTT of the web-like flows is changed in the range between 30 ms and 500 ms. The number of the long-lived flows is 100. Figure 5.13 shows that TMD scheme allows to improve the goodput of the web-like flows by up to 61%, and the performance with the TMD and DropTail links for the long-lived flows is similar. Besides, the performance of the web-like traffic with the TMD link becomes closer to one



(a)



(b)

Figure 5.13: Influence of the propagation RTT of web-like flows: (a) Throughput of long-lived flows; (b) Goodput of web-like flows.

with the DropTail link with the increase of propagation RTT because large propagation RTT dominates FCT.

5.4.6 UDP Flows

To investigate how TMD deals with non-elastic traffic, we enhance the bottleneck link traffic with three UDP flows. Each of these flows transmits at the rate of 10 Mbps. The duration of the experiment is 130 s. The first, second, third UDP flows start at 20 s, 40 s, and 60 s, respectively, and finish at 80 s, 100 s, and 120 s, respectively. Five experiments are run for each of the schemes, i.e., with the TMD and DropTail links. The average goodput of the web-like traffic for the proposed design and the DropTail link are 9.08 KBps and 6.22

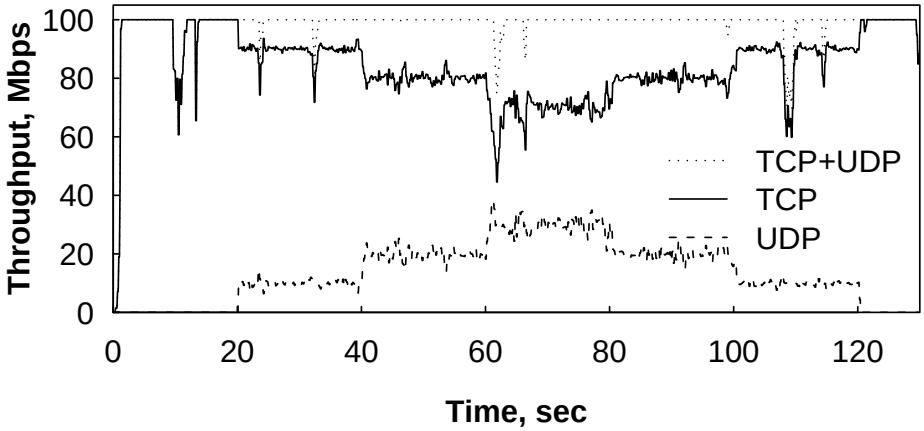


Figure 5.14: Coexistence with the non-elastic UDP traffic.

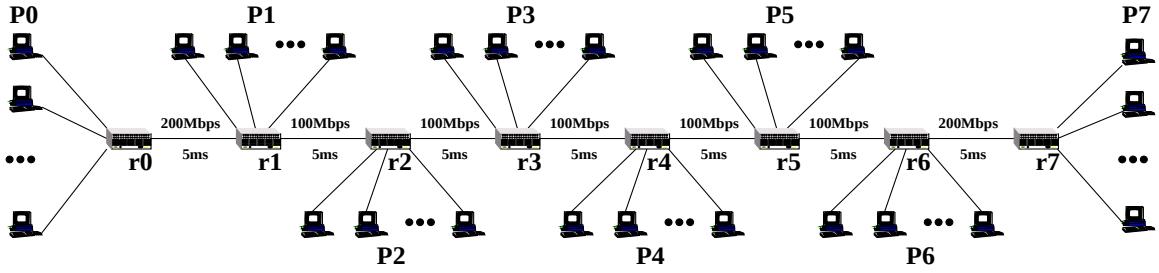
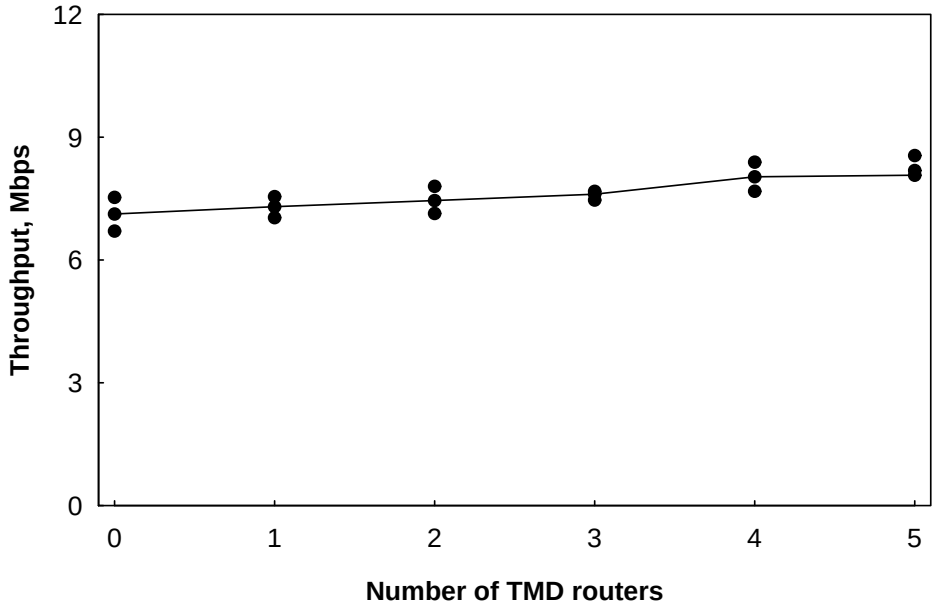


Figure 5.15: Multi-bottleneck topology.

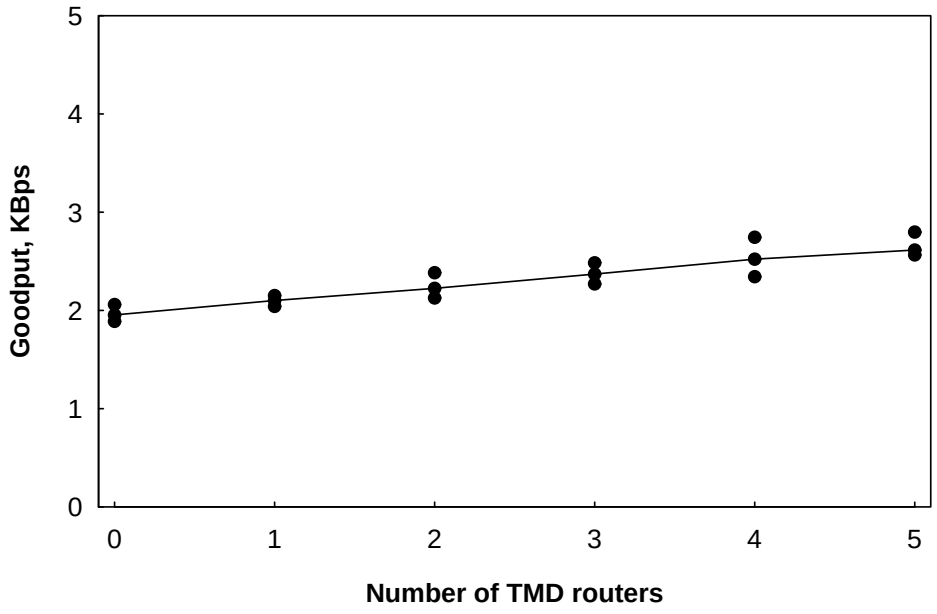
KBps, respectively, i.e., TMD improves the performance by 46 %. The average throughputs of the long-lived flows with the TMD and DropTail links are 64 Mbps each. Figure 5.14 reports the throughputs of TCP, UDP traffic, and the total throughput of the bottleneck link with the TMD scheme for one of the experiments. One can see that the proposed design handles the UDP traffic as expected and reacts to sudden changes in the bottleneck link capacity promptly and appropriately.

5.4.7 Incremental Deployment

To explore the performance of TMD under multi-bottleneck topology, a parking lot topology shown in Figure 5.15 is used. All access links are 200 Mbps. Propagation RTTs of the flows are uniformly distributed in the range between 74 ms and 300 ms. There are 20 long-lived flows going from pool P0 to P7, and 20 long-lived flows in the reverse direction. Each of the bottleneck links r1-r2, r2-r3, r3-r4, r4-r5, r5-r6 are shared by 20 long-lived



(a)



(b)

Figure 5.16: Incremental deployment in the multi-bottleneck topology: (a) Throughput of long-lived flows; (b) Goodput of web-like flows.

flows starting from pools P1, P2, P3, P4, P5, and destining to pools P2, P3, P4, P5, P6, respectively. Besides, a web-server in pool P0 generates traffic destined to pool P7, which is described by the same parameters as in Section 5.4.4. The number of bottleneck links with the deployed TMD scheme are varied between 0 and 5. Figure 5.16 reports the throughput of the long-lived flows going from pool P0 to P7 and the goodput of the web-like traffic. One can observe that wider deployment of the TMD design improves the performance of

the web-like traffic and the long-lived flows. Moreover, both the reported goodput and throughput are increasing functions of the number of the TMD links. Deploying the proposed scheme on all the bottleneck links allows to increase the throughput for the long-lived flows by 13 % and the goodput for the web-like traffic by 34%.

5.5 Security Considerations

Inappropriate classification of packets by the senders is an issue that warrants a discussion. In one such scenario, a TCP sender marks as A-class all its packets, including the packets transmitted in the slow-start mode. The deviation from the intended marking causes the slow-start packets of the flow to be forwarded through the buffer partition for class A. This outcome is similar to the current Internet service with a non-partitioned FIFO DropTail link buffer. Hence, the attempt to manipulate the TMD scheme does not reward the misbehaving sender with better performance.

In another extreme instance of inappropriate packet classification, a TCP sender marks as S-class all its packets, including the packets transmitted in the congestion-avoidance mode. With the default setting of $k = 0.5$, the per-flow buffer allocation for S-class flows is lower. Performance of the misbehaving flow might suffer from the smaller per-flow buffer allocation. Hence, the success of such attack is not assured and depends on n_S, n_A, and other information that is not available to the misbehaving sender. In contrast, following the TMD making guidelines consistently provides TCP flows with better performance. This property of the TMD design can be viewed as a strong incentive for TCP senders to classify packets as prescribed by the proposed scheme.

5.6 Summary

This Chapter reconsidered the problem of accelerating TCP flows concerning their completion times and proposed TMD, a new scheme for decreasing FCT. The key idea of the scheme is to dynamically allocate buffer space for short-lived and long-lived flows. The extensive evaluation demonstrated that TMD allows to significantly increase the goodput of short-lived flows without deteriorating the performance of long-lived flows.

Chapter 6

Conclusion

This Chapter summarizes the main results of the dissertation and potential directions for future research.

6.1 Summary

Chapter 3 described MCP, a new congestion control protocol based on the following key design principles: use of multiple operation modes, transmission at the constant rate in the stable state, and use of the explicit-communication mechanism for converging to fairness. The latter feature of the design is one of the MCP main contributions, and enables a flow to request competing flows to operate in a mode allowing convergence to the fair sharing of the network capacity. MCP uses several modes of operation for meeting all the requirements to the protocol. The proposed protocol makes data transmission stable for maintaining fairness, high utilization, and low queue size. MCP uses the sending bit rate as a control parameter, and this ensures that the flow throughput is independent of the flow RTT and packet size. Simulations showed that MCP achieves high utilization, reveals good convergence to fairness, flow throughput independent of flow RTT and packet size, and low buffer queue size.

Chapter 4 presented the RD Network Services, a new architecture for aligning network services with application needs. Based on the principles that a new service should incorporate incentives for users and ISPs to adopt the service despite partial deployment and without penalizing legacy traffic, the RD services were designed and implemented to empower each user to choose between a higher rate or low queuing delay. The RD design does not require changes in end-to-end transport protocols, preserves the IP datagram format, and relies on

simple router algorithms that are easy to implement even for high-capacity links. The extensive evaluation of the RD services confirmed their effective rate-delay differentiation as well as their fitness for incremental virulent deployment.

Chapter 5 proposed a solution to the problem of decreasing FCT of short-lived TCP flows without doing harm to long-lived flows. The advocated TMD design categorizes TCP slow-start flows as class S and all the other traffic as class A. The TMD buffer partitioning scheme for FIFO links was explored where the ratio of the per-flow buffer shares for S-class flows and A-class flows is maintained at k. Based on the analytic speculation and experimental validation, it was established that $k = 0.5$ was the preferred parameter setting that consistently satisfies the design objectives. The TMD scheme is simple, incrementally deployable, and does not require any changes in TCP congestion control, packet format, or FIFO link scheduling. The experiments reveal that TMD increases the goodput of web-like flows decreasing FCT of short-lived TCP flows by as much as 40%. Whereas the presented TMD design dynamically partitions the fixed-sized buffer between the two traffic classes, an interesting direction for further studies is to combine buffer partitioning with dynamic buffer sizing.

6.2 Future Work

Further work on MCP is needed to address the following three concerns about its original design:

- *Synchronous control* ensures that MCP flows having the same bottleneck link operate in the same mode. However, asynchrony of modes might be beneficial in some scenarios. For example, TCP might provide a new flow with faster convergence to a fair transmission rate than under MCP because the asynchronous TCP allows the new flow to operate in the aggressive slow-start mode whereas existing flows continue to operate in the congestion-avoidance mode, where TCP acquires the available capacity at a slower pace.
- *Vulnerability to host misbehavior* is another concerning property of MCP. In particular, by setting the fairing and this-path bits persistently beyond the prescribed seven increase-decrease cycles, a malicious host can make all flows sharing its bottleneck

link to keep operating in the fairing mode, causing needless oscillations of the bottleneck link utilization and queue size.

- *Handling short-lived flows* is not effective. While short-lived flows constitute more than 85% of all flows in the Internet, their frequent arrivals to the network make long-lived flows constantly switch into the fairing mode. Moreover, the main need of short-lived flows is a minimized FCT rather than convergence to the fare share. Therefore, MCP currently does not handle traffic consisting of short-lived flows perfectly.

The above concerns are not unique to MCP. For example, VCP also faces the issue of synchronous control. It is planned to examine whether randomizing the mode selection under some circumstances is able to realize the potential benefits of asynchrony without undermining the overall performance of MCP. The mechanism for requesting the fairing mode of MCP operation adds a new avenue for attacks to the already wide arsenal available to malicious hosts. Since senders and receivers might collude, effective protection against host attacks necessitates router assistance. Thus, lightweight router techniques for making MCP resilient to host misbehavior will be studied. Due to the mode synchronization and uniform timing of transmission adjustments, it seems easier for routers to detect misbehaving MCP flows than flows of other protocols where transmission rates depend on non-uniform packet losses, RTT, and packet sizes.

While the design principles of the RD Network Services led to the elegant effective solution for network service differentiation, it is believed that design for deployability holds great promise for solving other types of networking problems. Even within the conceptual framework of rate-delay differentiation, one can see numerous opportunities for further fruitful exploration. For example, whereas the strict enforcement of the delay constraint for the D service is a conscious attempt to encourage the service adoption only if the user is really interested in assuredly low queuing delay, it is worth to investigate whether delay should be allowed to spike occasionally as long as average low delay remains guaranteed.

A related issue is whether the RD architecture will induce any unintended behavior of users who seek to improve own service or deliberately disrupt services for other users. Although the two-queue design alleviates some denial-of-service attacks, the RD architecture inherits most security problems of the Internet. While securing the RD design is clearly an important area for future investigation, prior simple performance-based [41, 93] and other [105, 109] security proposals constitute promising starting points.

Our preliminary assessment suggests that denial-of-service attacks do not pose a significant danger to TMD. However, we plan to investigate the question of the attacks on the TMD design more extensively in future.

References

[1] Methods for Subjective Determination of Transmission Quality. ITU-T Recommendation P.800, August 1996.

[2] The E-model, a Computational Model for Use in Transmission Planning. ITU-T Recommendation G.107, June 2006.

[3] A. Aggarwal, S. Savage, and T. Anderson. Understanding the Performance of TCP Pacing. In *Proceedings IEEE INFOCOM 2000*, March 2000.

[4] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing Router Buffers. In *Proceedings ACM SIGCOMM 2004*, August 2004.

[5] W. Ashmawi, R. Guerin, S. Wolf, and M. Pinson. On the Impact of Policing and Rate Guarantees in Diff-Serv Networks: A Video Streaming Application Perspective. In *Proceedings ACM SIGCOMM 2001*, August 2001.

[6] K. Avrachenkov, U. Ayesta, P. Brown, and E. Nyberg. Differentiation Between Short and Long TCP Flows: Predictability of the Response Time. In *Proceedings IEEE INFOCOM 2004*, March 2004.

[7] A. Bacioccola, C. Cicconetti, and G. Stea. User-level Performance Evaluation of VoIP Using ns-2. In *Proceedings NSTools 2007*, October 2007.

[8] S. Bajaj, L. Breslau, and S. Shenker. Is Service Priority Useful in Networks? In *Proceedings ACM SIGMETRICS 1998*, June 1998.

[9] J. Bennett and H. Zhang. WF2Q: Worse-case Fair Weighted Fair Queuing. In *Proceedings IEEE INFOCOM 1996*, March 1996.

[10] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. IETF RFC 2475, December 1998.

[11] D. Blandford, S. Goldman, S. Gorinsky, Y. Zhou, and D. Dooly. Smartacking: Improving TCP Performance from the Receiving End. *Journal of Internet Engineering*, 1(1):6–21, January 2007.

[12] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. IETF RFC 1633, June 1994.

[13] L. Brakmo, S. Malley, and L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proceedings ACM SIGCOMM 1994*, August 1994.

[14] D. Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of Computer Networks and ISDN*, 17(2):1–14, July 1989.

[15] G. Chuanxiong. SRR: An O(1) Time Complexity Packet Scheduler for Flows in Multi-Service Packet Networks. *IEEE/ACM Transactions on Networking*, 12(6):1144–1155, 2004.

[16] G. Chuanxiong. G-3:An O(1) Time Complexity Packet Scheduler That Provides Bounded End-to-End Delay. In *Proceedings IEEE INFOCOM 2007*, May 2007.

[17] G. Chuanxiong. Improved Smoothed Round Robin Schedulers for High-Speed Packet Networks. In *Proceedings IEEE INFOCOM 2008*, April 2008.

[18] D.D. Clark and W. Fang. Explicit Allocation of Best Effort Packet Delivery Service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, August 1998.

[19] D.D. Clark, J. Wroclawski, K.R. Sollins, and R. Braden. Tussle in Cyberspace: Defining Tomorrow's Internet. In *Proceedings ACM SIGCOMM 2002*, August 2002.

[20] D. Wei. *A TCP pacing implementation for NS2. http://netlab.caltech.edu/projects/ns2tcplinux/ ns2pacing/.*

[21] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queuing Algorithm. In *Proceedings ACM SIGCOMM 1989*, September 1989.

[22] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional Differentiated Services: Delay Differentiation and Packet Scheduling. In *Proceedings ACM SIGCOMM 1999*, September 1999.

[23] N. Dukkipati, M. Kobayashi, and N.McKeown. Processor Sharing Flows in the Internet. *Thirteenth International Workshop on Quality of Service (IWQoS)*, June 2005.

[24] 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, 2006.

[25] N. Dukkipati, N. McKeown, and A. Fraser. RCP-AC: Congestion Control to Make Flows Complete Quickly in Any Environment. In *Proceedings High-Speed Networking Workshop: The Terabits Challenge, IEEE INFOCOM 2006*, April 2006.

[26] J. Eggleston and S. Jamin. Differentiated Services with Lottery Queueing. In *Proceedings IEEE IWQoS 2001*, June 2001.

[27] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden. Part III: Routers with Very Small Buffers. *ACM Computer Communication Review*, 35(3):83–90, July 2005.

[28] C. Estan, G. Varghese, and M. Fisk. Bitmap Algorithms for Counting Active Flows on High Speed Links. *IEEE/ACM Transactions on Networking*, 14(5):925–937, October 2006.

[29] W. Feng, D. Kandlur, D. Saha, and K. Shin. A Self-configuring RED Gateway. In *Proceedings IEEE INFOCOM 1999*, March 1999.

[30] W. Feng, K. Shin, D. Kandlur, and D. Saha. The Blue Active Queue Management Algorithms. *IEEE/ACM Transactions on Networking*, 10(4):513–528, 2002.

[31] V. Firoiu, X. Zhang, and Y. Guo. Best Effort Differentiated Services: Tradeoff Service Differentiation for Elastic Applications. In *Proceedings IEEE ICT 2001*, June 2001.

[32] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-Based Congestion Control for Unicast Applications. In *Proceedings ACM SIGCOMM 2000*, August 2000.

[33] S. Floyd and T. Henderson. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 2582, April 1999.

[34] S. Floyd, T. Henderson, and A. Gurtov. The NewReno Modification to TCP's Fast Recovery Algorithm, April 2004.

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

[36] S. Floyd and V. Jacobson. Link Sharing and Resource Management Models for Packet Networks. *ACM Transactions on Database Systems*, 3(4):365–386, August 1995.

[37] E. Friedman and S. Henderson. Fairness and Efficiency in Web Server Protocols. In *Proceedings ACM SIGMETRICS 2003*, June 2003.

[38] L. Georgiadis, R. Guerin, V. Peris, and R. Rajan. Efficient Support of Delay and Rate Guarantees in an Internet. In *Proceedings ACM SIGCOMM 1996*, August 1996.

[39] S.J. Golestani. A Self-Clocked Fair Queuing Scheme for High Speed Applications. In *Proceedings IEEE INFOCOM 1994*, June 1994.

[40] S. Gorinsky, A. Kantawala, and J. Turner. Link Buffer Sizing: A New Look at the Old Problem. In *Proceedings IEEE Symposium on Computers and Communications (ISCC 2005)*, June 2005.

[41] S. Gorinsky, S.Jain, H. Vin, and Y. Zhang. Design of Multicast Protocols Robust against Inflated Subscription. *IEEE/ACM Transactions on Networking*, 14(2):249–262, April 2006.

[42] P. Goyal, S. Lam, and H. Vin. Determining End-to-End Delay Bounds in Heterogeneous Networks. In *Proceedings NOSSDAV 1995*, April 1995.

[43] Y. Gu, D. Towsley, C. Hollot, and H. Zhang. Congestion Control for Small Buffer High Speed Networks. In *IEEE INFOCOM 2007*, April 2007.

[44] R. Guerin, S. Kamat, V. Peris, and R. Rajan. Scalable QoS Provision Through Buffer Management. In *Proceedings ACM SIGCOMM 1998*, September 1998.

[45] L. Guo and I. Matta. The War Between Mice and Elephants. In *Proceedings IEEE ICNP 2001*, November 2001.

[46] L. Guo and I. Matta. Differentiated Control of Web Traffic: A Numerical Analysis. In *Proceedings SPIE ITCOM 2002: Scalability and Traffic Control in IP Networks*, August 2002.

[47] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. IETF RFC 2597, June 1999.

[48] C. Hollot, V. Misra, D. Towsley, and W. Gong. Analysis and Design of Controllers for AQM Routers Supporting TCP Flows. *IEEE/ACM Transactions on Networking*, 47(6):945–959, 2002.

[49] J. D. Houle, K. K. Ramakrishnan, R. Sadhvani, M. Yuksel, and S. Kalyanaraman. The Evolving Internet - Traffic, Engineering, and Roles. In *Proceedings Research Conference on Communication, Information and Internet Policy (TPRC)*, September 2007.

[50] N. Hu and P. Steenkiste. Improving TCP Startup Performance using Active Queue Measurements: Algorithm and Evaluation. In *Proceedings IEEE ICNP 2003*, November 2003.

[51] P. Hurley, J.-Y. Le Boudec, P. Thiran, and M. Kara. ABE: Providing a Low-Delay Service within Best Effort. *IEEE Network*, 15(3):60–69, May/June 2001.

[52] Information Sciences Institute. Transmission Control Protocol, September 1981.

[53] S. Iyer, R. Kompella, and N. McKeown. Designing Packet Buffers for Router Line Cards. Technical Report, TR-02-HPNG-031001, October 2002.

[54] V. Jacobson. Congestion Avoidance and Control. In *Proceedings ACM SIGCOMM 1988*, August 1988.

[55] V. Jacobson, K. Nichols, and K. Poduri. An Expedited Forwarding PHB. IETF RFC 2598, June 1999.

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

[57] S. Jain and D. Loguinov. PIQI-RCP: Design and Analysis of Rate-Based Explicit Congestion Control. In *Proceedings IEEE IWQoS 2007*, June 2007.

[58] H. Jiang and C. Dovrolis. Passive Estimation of TCP Round-Trip Times. *ACM Computer Communications Review*, 32(3):75–88, July 2002.

[59] A. Kantawala and J. Turner. Queue Management for Short-Lived TCP Flows in Backbone Routers. In *Proceedings High-Speed Networking Symposium, IEEE Globecom 2002*, November 2002.

[60] D. Katabi, M. Handley, and C. Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. In *Proceedings ACM SIGCOMM 2002*, August 2002.

[61] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis. Weighted Round Robin Cell Multiplexing in a General-Purpose ATM Switch. *IEEE Journal on Selected Areas in Communications*, 9(8):1265–1279, October 1991.

[62] J. Kaur and H. Vin. Core-Stateless Guaranteed Rate Scheduling Algorithm. In *Proceedings IEEE INFOCOM 2001*, April 2001.

[63] J. Kaur and H. Vin. Core-Stateless Guaranteed Throughput Networks. In *Proceedings IEEE INFOCOM 2003*, April 2003.

[64] H. Kim and D. O'Hallaron. Counting Network Flows in Real Time. In *Proceedings IEEE GLOBECOM 2003*, December 2003.

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

[66] K. Kumazoe, M. Tsuru, and Y. Oie. Improving Delay Characteristics of Real-Time Flows by Adaptive Early Packet Discarding. In *Proceedings International Conference on Information Networking (ICOIN)*, January 2006.

[67] A. Kuzmanovic and E. Knightly. Low-Rate TCP-Targeted Denial of Service Attacks and Counter Strategies. *IEEE/ACM Transactions on Networking*, 14(4):683–696, August 2006.

[68] L. Le, J. Aykat, K. Jeffay, and F. Smith. Differential Congestion Notification: Taming the Elephants. In *Proceedings IEEE ICNP 2004*, October 2004.

[69] D. Lin and R. Morris. Dynamics of Random Early Detection. In *Proceedings ACM SIGCOMM 1997*, September 1997.

[70] R. Mahajan, S. Floyd, and D. Wetherall. Controlling High-Bandwidth Flows at the Congested Router. In *Proceedings IEEE ICNP 2001*, November 2001.

[71] I. Matta and L. Guo. Differentiated Predictive Fair Service for TCP Flows. In *Proceedings IEEE ICNP 2000*, October 2000.

[72] S. McCanne and S. Floyd. *ns Network Simulator. http://www.isi.edu/nsnam/ns/.*

[73] M. Mellia, I. Stoica, and H. Zhang. TCP Model for Short Lived Flows. IEEE Communications Letters, February 2002.

[74] K. Nichols, S. Blake, F. Baker, and D.L. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. IETF RFC 2474, December 1998.

[75] W. Noureddine and F. Tobagi. Improving the Performance of Interactive TCP Applications Using Service Differentiation. In *Proceedings IEEE INFOCOM 2002*, June 2002.

[76] A. Odlyzko. The Many Paradoxes of Broadband. *First Monday*, 8(9), September 2003.

[77] R. Pan, B. Prabhakar, and K. Psounis. CHOKe: A Stateless AQM Scheme for Approximating Fair Bandwidth Allocation. In *Proceedings IEEE INFOCOM 200*, March 2000.

[78] A. Parekh and R. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Single-node Case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, 1993.

[79] A. Parekh and R. Gallager. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: the Multiple Node Case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, 1994.

[80] V. Paxson. End-to-End Internet Packet Dynamics. In *Proceedings ACM SIGCOMM 1997*, September 1997.

[81] M. Podlesny and S. Gorinsky. "MCP: Few Bits for Fairing and Small Queues in the Stable State". In *Proceedings IEEE ISCC 2007*, July 2007.

[82] M. Podlesny and S. Gorinsky. Multimodal Congestion Control for Low Stable-State Queuing. In *Proceedings IEEE INFOCOM 2007*, May 2007.

[83] M. Podlesny and S. Gorinsky. RD Network Services: Differentiation through Performance Incentives. In *Proceedings ACM SIGCOMM 2008*, August 2008.

[84] J. Postel. User Datagram Protocol. IETF RFC 763, August 1980.

[85] J. Postel. Internet Protocol, September 1981.

[86] I. Qazi, L. Andrew, and T. Znati. Congestion Control using Efficient Explicit Feedback. In *Proceedings IEEE INFOCOM 2009*, April 2009.

[87] I. Qazi and T. Znati. On the Design of Load Factor based Congestion Control Protocols for Next-Generation Networks. In *Proceedings IEEE INFOCOM 2008*, April 2008.

[88] I. Rai, E. Biersack, and G. Urvoy-Keller. Size-based Scheduling to Improve the Performance of Short TCP Flows. *IEEE Networks*, 19:12–17, January/February 2005.

[89] G. Raina, D. Towsley, and D. Wischik. Part II: Control Theory for Buffer Sizing. *ACM Computer Communication Review*, 35(3):79–82, July 2005.

[90] K. K. Ramakrishnan and R. Jain. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks with a Connectionless Network Layer. In *Proceedings ACM SIGCOMM 1988*, August 1988.

[91] K.K. Ramakrishnan and S. Floyd. A Proposal to Add Explicit Congestion Notification (ECN) to IP. RFC 2481, January 1999.

[92] S. Ramesh and S. Kasera. Best-Effort Session Level Congestion Control. In *Proceedings IEEE ICNP 2007*, October 2007.

[93] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP Congestion Control with a Misbehaving Receiver. *ACM Computer Communications Review*, 29(5):71–78, October 1999.

[94] L. Schrage. A Proof of the Optimality of the Shortest Remaining Processing Time Discipline. *Operations Research*, 16(3):687–690, 1968.

[95] A. Shaikh, J. Rexford, and K. Shin. Load-Sensitive Routing of Long-Lived IP Flows. In *Proceedings ACM SIGCOMM 1999*, August 1999.

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

[97] J. Sjoberg, M. Westerlund, A. Lakaniemi, and Q. Xie. Real-Time Transport Protocol (RTP) Payload Format and File Storage Format for the Adaptive Multi-Rate (AMR) and Adaptive Multi-Rate Wideband (AMR-WB) Audio Codecs. IETF RFC 3267, June 2002.

[98] R. Stanojevic and R. Shorten. Drop Counters are Enough. In *Proceedings IEEE IWQoS 2007*, June 2007.

[99] D. Stidialis and A. Varma. Efficient Fair Queueing Algorithms for Packet-Switched Networks. *IEEE/ACM Transactions on Networking*, 6(2):175–185, 1998.

[100] D. Stidialis and A. Varma. Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms. *IEEE/ACM Transactions on Networking*, 6(5):611–624, 1998.

[101] I. Stoica, S. Shenker, and H. Zhang. Core-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks. In *Proceedings ACM SIGCOMM 1998*, September 1998.

[102] I. Stoica and H. Zhang. Providing Guaranteed Services Without Per Flow Management. In *Proceedings ACM SIGCOMM 1999*, September 1999.

[103] C. Villamizar and C. Song. High performance TCP in ANSNET. *ACM SIGCOMM Computer Communication Review*, 24(5):45–60, October 1994.

[104] C. Wang, B. Li, Y. Thomas Hou, K. Sohraby, and Y. Lin. LRED: A Robust Active Queue Management Scheme Based on Packet Loss Ratio. In *Proceedings IEEE INFOCOM 2004*, March 2004.

[105] D. Wetherall, U. Legedza, and J. Guttag. Introducing New Internet Services: Why and How. *IEEE Network*, 12(3):12–19, May-June 1998.

[106] D. Wischik and N. McKeown. Part I: Buffer Sizes for Core Routers. *ACM Computer Communication Review*, 35(3):75–78, July 2005.

[107] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman. One More Bit Is Enough. In *Proceedings ACM SIGCOMM 2005*, August 2005.

[108] X. Yang and D. Wetherall. Source Selectable Path Diversity via Routing Deflections. In *Proceedings ACM SIGCOMM 2006*, September 2006.

[109] X. Yang, D. Wetherall, and T. Anderson. A DOS-limiting Network Architecture. In *Proceedings ACM SIGCOMM 2005*, August 2005.

[110] S. Yilmaz and I. Matta. On Class-Based Isolation of UDP, Short-Lived and Long-Lived TCP Flows. In *Proceedings International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS) 2001*, August 2001.

[111] X. Yuan and Z. Duan. FRR: A Proportional and Worst-Case Fair Round Robin Scheduler. In *Proceedings IEEE INFOCOM 2005*, March 2005.

[112] L. Zhang. Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks. In *Proceedings ACM SIGCOMM 1990*, August 1990.

[113] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the Characteristics and Origins of Internet Flow Rates. In *Proceedings ACM SIGCOMM 2002*, August 2002.

[114] Y. Zhang, D. Leonard, and D. Loguinov. JetMax: Scalable Max-Min Congestion Control for High-Speed Heterogeneous Networks. In *Proceedings IEEE INFOCOM 2006*, April 2006.

[115] Y. Zhang and D. Loguinov. ABS: Adaptive Buffer Sizing for Heterogeneous Networks. In *Proceedings IEEE IWQoS 2008*, June 2008.

Vita

Maxim Podlesny

Date of Birth February 6, 1979

Place of Birth Perm, USSR/Russia

Degrees B.S. Summa Cum Laude, Applied Physics and Mathematics, June 2000

M.S. Summa Cum Laude, Applied Physics and Mathematics, June 2002

Ph.D., Computer Science, August 2009

Publications M. Podlesny, S. Gorinsky. RD Network Services: Differentiation through Performance Incentives, *in Proceedings* ACM SIGCOMM, August 2008

M. Podlesny, S. Gorinsky. Price of Asynchrony: Queuing under Ideally Smooth Congestion Control, poster paper, *in Proceedings* IEEE ICNP, October 2007

S. Gorinsky, M. Georg, M. Podlesny, and C. Jechlitschek. A Theory of Load Adjustments and Its Implications to Congestion Control, Journal of Internet Engineering, **1**(2): 82–93, October 2007

M. Podlesny, S. Gorinsky. MCP: Few Bits for Fairing and Small Queues in the Stable State, *in Proceedings* IEEE ISCC, July 2007

M. Podlesny, S. Gorinsky. Multimodal Congestion Control for Low Stable-State Queuing, *in Proceedings* IEEE INFOCOM, May 2007

August 2009