ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

TELECOM ParisTech

2016-ENST-00xx

EDITE - ED 130

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique et Réseaux »

présentée et soutenue publiquement par

Yixi GONG

le 4 Mars 2016

La quête de faible latence sur les deux bords du réseau : conception, d'analyse, de simulation et expériences

Directeur de thèse : **Dario ROSSI**

Jury

M. Robert BIRKE, Senior researcher, IBM	Rapporteur
M. David ROS, EU Coordinator in Communication Systems, Simula	Rapporteur
M. Marco AJMONE MARSAN, Prof., Politecnico di Torino	Rapporteur
M. Thomas Bonald, Prof., Télécom ParisTech	Examinateur
M. Tijani CHAHED, Prof., Télécom SudParis	Examinateur

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech
46 rue Barrault 75013 Paris - (+33) 1 45 81 77 77 - www.telecom-paristech.fr

THÈSE

The quest for low-latency at both network edges: design, analysis, simulation and experiments

Yixi Gong

Department of Network and Computer Science
Télécom ParisTech

This dissertation is submitted for the degree of
Doctor of Philosophy

February 2016

Acknowledgements

The past three years have been an incredible adventure that I believe to be a unique and important period in my life. Accompanied by those who always love me, what I acquire is not only academic accomplishment but also the courage to enjoy all sweet and bitter of life.

First of all, I would like to give my greatest acknowledgement to Dario. Thank you for leading me through such a splendid journey. Your professional at work, dedication to students, and positive attitude towards life all inspire and encourage me a lot. It is a fortune for me to have worked and learned with you. Thank you !

I am grateful to all the people I had the chance to work with, for their contributions and expertise : Claudio, Silvio, Dave, Emilio, Luca, and James. I would also like to thank the members of the jury, for attending my dissertation, and, in particular, the reviewers of the manuscript, for their suggestions and comments. Thanks also to all the colleagues and staff in Télécom ParisTech, LINCS, EIT Digital doctoral school for all help and collaboration in technical knowledge and administrative tasks. I would also like to thank Prof. Chen of Shanghai JiaoTong University for allowing me to visit, which was a great experience to meet an outstanding research group.

Last but not least, I would like to express my gratitude to my family. I dedicate all I have achieved to my parents who supported and encouraged me for so many years, and I hope this work will be the pride of you. I also want to thank my girlfriend for being on my side sharing every piece of joy and upset in these years ! Thanks to all my friends I met in Paris for the memorable moments we shared. For the everyday life in the lab, in hangouts, and also in travel.

Abstract

In the recent years, the innovation of new services over Internet is considerably growing at a fast speed, which brings forward lots of new challenges under varied scenarios. We replace encyclopedia with search engine, extend telephone functionality thanks to data connection, maintain social network from any time and anywhere, store and exchange music and video in the cloud, etc. The quality of user experience of such a diverse set of application can be mapped to network key performance indicators such as throughput (storage, video, file sharing) and latency (interactive service, voice, video of chat).

Bearing in mind of such context, the overall service performance depends in turn on the performance of multiple network segments. We investigated two representative design challenges in different segments : the two most important sit at the opposite edges of the end-to-end Internet path, namely, the end-user access network vs. the service provider data center network.

At the one end, due to the fact that regular users become data consumer and data producer at the same time, large amount of traffic is generated from “access network”. To ensure high throughput and low latencies, end users applications and equipment can leverage two complementary techniques : namely end-to-end congestion control algorithms vs. local scheduling and active queue management (AQM). Considering the bottleneck governed by an AQM, we discovered a potentially fateful interaction “reprioritization” between best-effort TCP and low priority congestion control (LPCC) protocols. We then quantify its generality with an extended set of simulation, followed by a thorough sensitivity analysis, and completed by an experimental campaign conducted on both controlled testbed and on the internet. Further analysis with control theory qualitatively characterizes this phenomenon, and therefore, help us propose a simple and practical system-level solution.

At the other end, on the service providers’ side, results are more being often dynamically calculated in the cloud. The design of “data center network” (DCN) becomes critical to the service quality and performance, and has been under active development for over a decade. Solutions proposed range from end-to-end transport protocol redesign to more intricate, monolithic and cross-layer architectures – yet, we remark the absence of DCN proposals based on simple fair-scheduling strategies. Our evaluation of FQ-CoDel in a DCN

environment shows that (i) average throughput is better with respect to that of end-to-end protocols tailored for DCN such as DCTCP, (ii) the completion time of short flow approaches that of state-of-art DCN proposals such as pFabric : good enough performance and striking simplicity make FQ-CoDel a serious contender in the DCN arena.

Abstract

Au cours de ces dernières années, les services Internet croissent considérablement ce qui crée beaucoup de nouveaux défis dans des scénarios variés. Nous avons témoignés divers changements tel que le remplacement des encyclopédies par des moteurs de recherche, l'extension des fonctionnalités de téléphone grâce à la liaison de données, l'utilisation de réseaux sociaux à tout moment depuis toutes sortes d'appareils, le téléchargement et l'échange de musique et des vidéos dans le nuage, etc. La qualité de l'expérience utilisateur d'un ensemble diversifié d'applications peut être liée aux indicateurs clés de performance des réseaux tels que le débit (stockage, vidéo, partage de fichiers) et la latence (service interactif, voix, conversations vidéo).

Compte tenu de ce contexte, la performance globale du service dépend à son tour de la performance des multiples segments de réseau. Nous étudions deux défis représentatifs de conception dans différents segments : les deux les plus importants se trouvent sur les bords opposés la connectivité de bout en bout des chemins d'Internet, notamment, le réseau d'accès pour l' utilisateur et le réseau de centre de données du fournisseur de services.

Du côté du réseau d'accès, les utilisateurs deviennent consommateur et producteur de données en même temps, ce qui génère une grande quantité de trafic sur ces réseaux. Pour garantir un débit élevé et une latence faible, les applications et les utilisateurs peuvent exploiter deux techniques complémentaires : les algorithmes de contrôle de congestion et la politique d'ordonnancement et la gestion active de file d'attente (AQM). Considérant le goulet d'étranglement régi par un AQM, nous avons découvert une interaction "reprioritization" potentiellement fatale entre un TCP "meilleur effort" et les protocoles de contrôle de file à priorité faible (LPCC). Nous avons ensuite quantifié sa généralisation avec d'une série de simulations suivie d'une analyse de sensibilité et complété par des méthodes expérimentales sur un banc de test contrôlé et sur Internet. Une analyse plus approfondie avec la théorie du contrôle caractérise qualitativement ce phénomène et nous a aidés à proposer une solution simple et pratique au niveau système.

Du côté des fournisseurs de services, les résultats sont plus souvent dynamiquement calculés dans le nuage. Par conséquent, la conception de "réseau de centre de données" (DCN) devient essentielle à la qualité de service. Elle a été en développement actif depuis

plus d'une décennie en utilisant des solutions variées. Nous les observons allant de la refonte de protocole de transport de bout-en-bout à l'architecture inter-couches complexe et monolithique. Cependant, on remarque l'absence de proposition basée sur une politique d'ordonnancement équitable. Notre évaluation d'un représentant "FQ-CoDeL" montre que, dans un environnement DCN, (i) le débit moyen atteint est mieux par rapport à celle des protocoles de bout-en-bout spécifiquement adaptés pour DCN tels que DCTCP, (ii) le délais de réalisation du flux court se rapproche de celle des propositions de pointe telles que pFabric. La bonne performance et la simplicité font de FQ-CoDeL un concurrent prometteur dans le domaine du DCN.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

It comes as no doubt that our daily activities increasingly require ubiquitous Internet access. In a typical day, we ask questions to Google and Bing, call friends with Skype or Hangout, socialize on Twitter and Facebook, upload pictures and tunes to Picasa and Google Music, backup or share data with BitTorrent, Dropbox and OneDrive, play video games on Xbox and Steam, and watch streaming shows through PC, mobile and even Internet TV set-top box, etc. A commonly acceptable way to evaluate the "goodness" of such services is quality of experience (QoE). Although different communities may use diverse language, such as business people talking about revenue per user and customer churn while behavioral scientists talk about happiness and experiences, engineers usually adopt network performance and quality of service (QoS) [1], which are the fundamental cornerstones. The innovation of new services over Internet is at such a lightning-fast pace that it presents quite a lot of new challenges, with the help of Fig. 1.1, we highlight two important ones worth attention.

First, regular users become data consumers and data producers at the same time. The prosperity of cloud-based service as shown in the "access network" of Fig. 1.1, constrained by mobile devices' hardware limitation, both computing and storage are moving from users' individual computers to some data centers. Data backup and sharing, whether it's client-server (CS) or peer-to-peer (P2P) architecture, generate large amount of traffic in the upload direction from users. Moreover, as a side effect of the proliferation of connected household devices, the need to synchronize data between the numerous appliances arises. As copies of the data are increasingly stored in some datacenters, this translates into frequent upload/downloads to/from the Cloud.

At the same time, the periphery of the Internet infrastructure was designed having in mind that users would mostly be data consumer (as opposed to data producer), as for instance

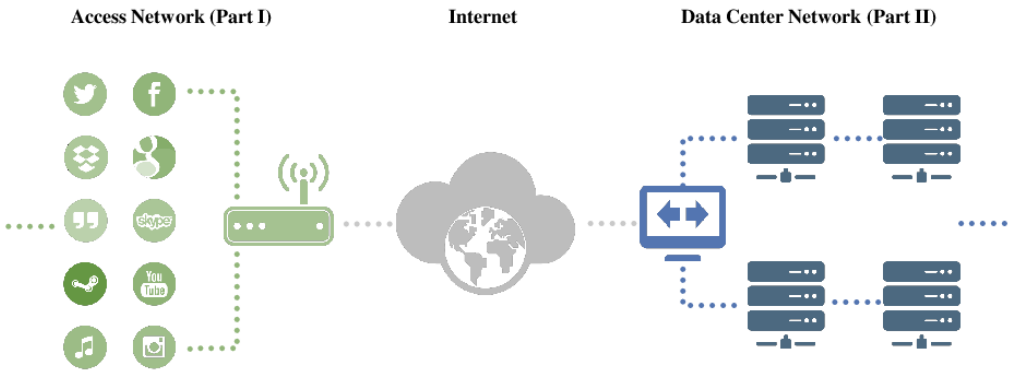


Figure 1.1 Representative network scenario considered in this thesis.

testified by the deployment of Asymmetric Digital Subscriber Line (ADSL) in Europe. While this fact was already challenged by peer-to-peer traffic (P2P), current data generation shift exacerbates the infrastructure asymmetry.

Second, more services are provided with dynamically calculated results from data centers instead of static information from database. As shown in the "data center network" of Fig. 1.1, instead of static information returned by a single server, currently there exists large data centers to calculate the result on-the-fly for users. Considering the fact that even "web search", one of the most common and frequently used service, relies on quite an amount of computing in data center before returning the best result that it can achieve in limited time according to service-level agreement (SLA). The rich web application such as online video game with massive users would require additional coordination between data centers.

However, traditional way of investigating user's quality of service in network context limitedly considers the bandwidth and latency between user and a single server. Awareness should be raised to include the network performance inside data center to be also a major component of QoS. Poor network design would accrue the cost of client churn either because of slow service or degraded result.

In the following Sec. 1.2 and Sec. 1.3, we further introduce the specific concern and design challenge facing these two dramatic transformations.

1.2 Concerns in access network

The Internet is a very heterogeneous ecosystem, where multiple protocol species coexist, evolve, and sometimes extinguish. TCP is a good example of this evolution. Over the years numerous species proliferated under this protocol family; it remains the most commonly adopted congestion control protocol over the Internet for the support of elastic data transfer and communication with weak real-time constraints such as video streaming. Since the most widespread flavors of TCP follow a loss-based design, an old problem which has resurged in recent years: namely, the "persistently full" buffer problem, which was nicknamed "bufferbloat" [2]. Bufferbloat refers to an excess of buffering that is exacerbated by two factors: (i) TCP loss-based design fills up the bottleneck buffer before the sender reduces its rate and (ii) Internet buffers are traditionally dimensioned sufficiently large to store up 250-500 msec of packets at the nominal line speed of the link. This simple rule-of-thumb approach may cause the unexpected increase of the packet round-trip time that may reach *several seconds*, where the real access bandwidth is significantly smaller than the nominal bandwidth under which the buffer has been designed (this may happen over heavily congested 3G/4G[3] and WiFi networks[4], or low-quality ADSL/Cable lines[5]).

For this problem, which has been well known since the 90s, two classes of solutions have been proposed. To better illustrate them, we first take a look at Internet "protocols" ecosystem: according the Internet (or OSI) terminology, the layering principle breakdowns it into an application-level (L7), a transport-level (L4), an Inter-networking (L3) and a data link (L2) level. For the sake of simplicity, we refer to engineers of the application and transport level protocols as the "upper-layer" engineer species, and to those of the network and host-to-network layers as "lower-layer" species.

First, to cope with bufferbloat, recent evolutions of the "lower-layer" have been specifically designed, and especially have started being deployed in both EU and US: indeed, infrastructural solutions to the bufferbloat such as scheduling (SFQ [6], DRR [7]) and Active Queue Management (RED [8], CoDel [4]) techniques, whose adoption has been so far limited, are now becoming commonplace for operators worldwide to improve the user experience of their ADSL/Cable lines customers (e.g., the French ISP Free employs SFQ since 2005 [9] and US follows suit with CoDel under active development in DOCSIS modems [10]).

Second, since early 2000, an ensemble of "upper-layer" congestion control algorithms, running over UDP, have been defined. These protocols typically support application executed in background, such as P2P file transfer or file synchronization, and are targeted to low-priority congestion control (LPCC) services, so that background applications do not subtract bandwidth from interactive applications. Notable examples of LPCC protocols are Microsoft Background Intelligent Transfer Service [11], TCP-LP [12], TCP-Nice [13] and

BitTorrent low extra delay background transport (LEBDAT) [14], which is especially relevant as BitTorrent is still credited as a primary contributor of uplink bandwidth [15].

While these trends on LPCC and AQM deployment are known facts, the temporal separation of the AQM vs. LPCC research topics results in the lack of in-depth investigation of their interaction. The impact on the dynamics of "upper-layer" LPCC schemes and "lower-layer" changes such as the replacement of simple DropTail buffer management policies with smarter adaptive policies is not completely clear.

In this thesis, we focus on the coexistence of best-effort TCP and Low Priority Congestion Control (LPCC) transiting a bottleneck link governed by AQM. Our work shows that a "reprioritization" phenomenon may occur with LPCC flows getting a bandwidth share comparable with the TCP share.

1.3 Concerns in data center network

In the last decade, data center networks (DCNs) have increasingly been built with relatively inexpensive off-the-shelf devices. DCNs are frequently assumed to be highly specialized environments, owned by a single entity that has full control of both the network architecture and the workload. DCN transport research has consequently explored a larger design space than that of the Internet. This characteristic leads to designs that generally do not focus on a single layer of the protocol stack but are more typically cross-layer.

Freedom in the DCN design space translates into a relatively crowded landscape of proposals, each of which is typically designed and tweaked with a specialized application scenario in mind. DCN proposals are further entangled as their design is tailored for very specific workloads, with diverse application patterns including, for instance, query-response [16, 17], map-reduce jobs [18, 19], or packet-size access to the DRAM of other machines [20]. Rarely, if ever, is a DCN design tested with a workload other than that for which the system was explicitly designed.

Beyond any reasonable doubt, the single-tenant assumption will be severely challenged in the near future. Increased user reliance on cloud applications will require DCNs to evolve towards multi-tenant system handling a significantly more heterogeneous set of applications and workloads. The range of applications will inevitably increase either because a single-tenant data center is used for new lines of business or simply because the data center is used by an increasing number of independent tenants. DCN workload will thus evolve beyond the typical mixture of short transactions and fat elastic transfers considered nowadays and will notably include a significant fraction of rate-limited flows with strict latency requirements.

As DCN resources are increasingly shared among multiple stakeholders, we must question the appropriateness of some frequently made assumptions. How can one rely on all end-systems implementing a common, tailor-made transport protocol like DCTCP [16], when end-systems are virtual machines under tenant control? How can one rely on applications truthfully indicating the size of their flows to enable the shortest flow first scheduling as in pFabric [17], when a tenant can gain better throughput by simply slicing a long flow into many small pieces? Expected future DCN usage clearly puts these assumptions in doubt and leads us to question the expected benefits of fragile DCN designs that rely on them.

In this thesis, we explore most important proposals in DCN design and evaluate the obvious yet surprising overlook, namely a scheduling mechanism providing fairness among flows, coupled with Active Queue Management (AQM) to upper-bound delay.

1.4 Contributions

The main contributions of this thesis are as follows.

- Demonstrate the negative interplay "reprioritization" when scheduling/AQM techniques and low-priority congestion control (LPCC) protocols are combined.
- Evaluate the generality of "reprioritization" phenomenona with simulation and controlled measurement experiments.
- Analyze the system dynamics of AQM vs. LPCC interaction with control theoretic fluid model.
- Propose a system-level deployable solution able to reinstate priorities among protocols.
- Quantify the performance of adopting fair scheduling and AQM techniques in data center network.
- Raise awareness in the community of fair scheduling in the data center network design landscape.

Tab. 1.1 contains an overview of the contributions of the thesis.

The first part of this thesis is dedicated to the analysis of "reprioritization" phenomenon when TCP-like and LPCC flows competing under a bottleneck governed by AQM. To this aim, different methodologies are exploited: (i) simulation program, (ii) experimental testbed, (iii) analytical model. In order to compare the performance under different combinations, various metrics are used, as the intra-protocol and inter-protocol fairness, the link utilization, the buffer occupancy, to cite a few.

In Chap. 2 we unveil the "reprioritization" phenomenon by a simple yet practical motivated scenario. By investigating at research history of AQM and LPCC, we discover the

Table 1.1 Synopsis of the thesis

	Chapter	Methodology	Comment	Publication
Part. I	Chap. 3	Simulation, experimental testbed	"Reprioritization" phenomenon assessment and sensitivity analysis	[W1, W2]
	Chap. 4	Analytical model	System dynamics and stability region analysis	[P1, J1, S1]
Part. II	Chap. 6	Simulation	Fair queuing performance measurement in DCN	in preparation

independent evolution of engineering at different network layers as a potential source of the problems mentioned, resulting in a lack of joint research over the years. We also present the related work of topics involved in Part. I.

In Chap. 3 we use `ns2` to evaluate a large spectrum of AQM and LPCC combination via packet-level simulation. We find that "reprioritization" holds under any considered LPCC and AQM. The choice of a specific (LPCC,AQM) combination has only very limited impact on the system performance. Moreover, a careful sensitivity analysis is carried out considering extended network scenarios.

In Chap. 4 we model the system as a Delay Differential Equation (DDE) that we study both as an open loop and as a closed loop system. Using the open-loop model, we fully characterize the "reprioritization" regions; we then numerically characterize the stability of the linearized system around the equilibrium using the closed-loop model. The results of our model are validated by packet-level simulation with good agreement. Therefore, we discuss a very simple yet effective way to avoid such "reprioritization" phenomenon.

The second part of the thesis is focused on data center network (DCN) proposals. With the goal of reducing a more service-oriented metric, the flow completion time, we study the influence of adopting a fair queuing scheduler via simulation.

In Chap. 5 we explore the landscape of DCN proposals in a comprehensive survey. Furthermore, we point out that the absence of a flow scheduler like FQ-CoDel in the design space is a surprising and unjustified omission that we hope to investigate in the following chapter.

In Chap. 6 we compare the performance of representative DCN designs among. We carry out careful calibration before running the simulation campaign, and discover the proposed FQ-CoDel as a suitable solution for data center network with great potential.

Finally, in Chap. 7 we discuss the main results achieved in this thesis, with a glance at the future directions and evolutions of this study.

Part I

Access network

Chapter 2

Background

In this chapter, we first in Sec. 2.1 motivate our study on the interaction of active queue management(AQM) and low priority congestion control(LPCC) in the access network environment. Sec. 2.2 describes related work in concerned research fields and methodologies used both in Chap. 2 and Chap. 3.

2.1 Motivation

The Internet is a rather complex ecosystem in which different species (i.e., hardware equipment), speaking a plethora of languages (i.e., protocols) with numerous dialects (i.e., software implementations) coexists. Equipment vendors, normalization fora, and software developers let these species, languages and dialects exist and evolve. Of course, within each category a finer grained taxonomy is possible. For instance, taking the Internet "protocols" category, the layering principle allows to breakdown the protocol ecosystem into, according to the Internet (or OSI) terminology, an application-level (or L7), a transport-level (L4), an Inter-networking (L3) and a host-to-network (L2) level. Let us, for the sake of simplicity, refer to engineers of the application and transport level protocols as the "upper-layer" engineer species, and to engineers of the network and host-to-network layers as "lower-layer" species.

Recent evolution in the "upper-layer" has confirmed TCP to be still the most commonly adopted congestion control protocol over the Internet for the support of either elastic data transfer or communication with weak real-time constraints such as video streaming. Nevertheless, in the recent years an ensemble of application-specific Layer-7 congestion control algorithms, running over UDP, have been defined. With few exceptions [1], these protocols typically support application executed in background, such as P2P file transfers, file synchro-

1. The most notable of which is represented by Google's QUIC, an application-layer protocol over UDP, targeted for TCP replacement in the context of SPDY/HTTP.

nization etc., and are targeted to low-priority congestion control (LPCC) services, so that background applications do not subtract bandwidth from interactive applications.

Notable examples of LPCC protocols are represented by Microsoft Background Intelligent Transfer Service [2], TCP-LP [12], TCP-Nice [13] and BitTorrent low extra delay background transport (LEBDAT) [14], which is especially relevant as BitTorrent is still credited as a primary contributor for uplink bandwidth. According to Brahm Cohen, and as confirmed by our measurement [21], LEDBAT is now "the bulk of all BitTorrent traffic, [...] most consumer ISPs have seen the majority of their upload traffic switch to a UDP-based protocol" [22]. Concerning the penetration of BitTorrent traffic, while downlink traffic is nowadays dominated by video streaming, BitTorrent remains the top-1 contributor in the uplink direction. As pointed out by a recent report [15] from the Canadian broadband management company Sandvine, BitTorrent is the top-1 application on uplink traffic, and can be credited for over one-third of all upload traffic in North America, Latin America and Asia Pacific. Additionally, BitTorrent represents no less than 10% of the aggregated uplink and downlink traffic, and is still the top-1 application in terms of aggregated traffic volume in the Asia Pacific region. Finally, as median Internet traffic increases, so does the overall BitTorrent traffic.

Different from standard TCP, which reduces the source sending rate only in the occurrence of packet loss events, LPCC congestion control schemes decrease the source sending rate as soon as the estimated packet delay grows beyond a given target. As a result, the LPCC schemes exhibit a less aggressive profile than TCP when they compete for the bottleneck bandwidth, which is typically placed at the access link (especially in upstream in today's scenarios), thus leave to the TCP flows most of the bottleneck bandwidth.

However, this is true only when TCP flows are able to fill the bottleneck buffer inducing a significant increase of the packet delay, i.e. when buffers are dimensionally sufficiently large and DropTail packet discarding policies are adopted.

With this regard, we recall that Internet buffers are traditionally sufficiently large to store up to 250-500 msec of packets at the nominal line speed of the link. This simple rule-of-thumb approach, however, may be the cause of the "bufferbloat" phenomenon, i.e., the unexpected increase of the packet round-trip time that may reach *several second*. In such cases, the real access bandwidth is significantly smaller than the nominal bandwidth for which the buffer has been designed (this may happen over heavily congested 3G/4G [3] and WiFi networks [4], or low-quality ADSL/Cable lines [5]).

To cope with bufferbloat, recent evolutions of the "lower-layer" have been specifically designed, and have started being deployed in both the EU and US: indeed, infrastructural

2. https://msdn.microsoft.com/en-us/library/aa363167.aspx

solutions to the bufferbloat such as scheduling (SFQ [6], DRR [7]) and Active Queue Management (RED [8], CoDel [4]) techniques, whose adoption has been so far limited, are now becoming common for operators worldwide to improve the quality of experience of their ADSL/Cable lines customers (e.g., the French ISP Free employs SFQ since 2005 [9] and US follows suit with CoDel under active development in DOCSIS modems [10]).

From the above observation, we gather an increasing adoption trend of both AQM techniques and LPCC protocols, which already coexist in the current Internet. Yet, studies have so far focused on AQM or LPCC, and only seldom considered these two aspects together. As such, interactions between AQM and LPCC is, at this stage, poorly understood.

In this thesis, we show a potentially fateful interplay between AQM and LPCC: namely, AQM resets the relative level of priority between best-effort and low-priority congestion control protocols. In other words, current scavenging protocols can successfully realize a lower-than-best-effort priority only if the bottleneck buffer operates according to a DropTail discipline. Intuitively, this arises from the fact that one of the typical design goals of AQM is to enforce fairness among flows, to penalize the most aggressive heavy-hitter flows and to protect the newly starting and short-lived ones. This is in sharp contrast with LPCC's design goal, which instead aims at utilizing the excess capacity without interfering with standard TCP transfers. As this interplay resets the relative level of priority among congestion control protocols, we refer to this issue simply as "reprioritization" in the following.

Fig. 2.1 illustrates the "reprioritization" phenomenon: we stress that, while the picture depicts simulation results gathered in a very specific case, the remainder of this part will show the phenomenon to hold under a large range of AQM+LPCC scenarios (using both real-world experiments and `ns2` simulations). The picture shows a breakdown of the link utilization when 5 TCP NewReno (each of which is represented with a different shade of red) and 5 LEDBAT (blue) backlogged flows sharing the same bottleneck. Capacity is set to 10 Mbps and the buffer has room for 500 packets (600 ms worth of delay); for the sake of simplicity, delays are homogeneous across flows. The left plot reports the case of a DropTail queuing discipline, while the right one reports the case of RED. The plots are annotated with further statistics concerning the average queue size in packets $E[Q]$, the capacity share exploited by the aggregate TCP%, and the average link utilization η.

In the DropTail case, the LEDBAT protocol operates in a lower-than-best-effort mode: we see that this delay-based scavenging protocol successfully exploits the spare capacity left unused by NewReno (as shown in [23]). The TCP aggregate uses the bulk of the capacity (TCP%=99%), with a fair share among TCP flows (due to homogeneous delay). However, the queuing delay approaches half a second because nearly 400 full-size TCP packets are queued on average. Clearly, bufferbloat would be even worse for lower (e.g., ADSL-like)

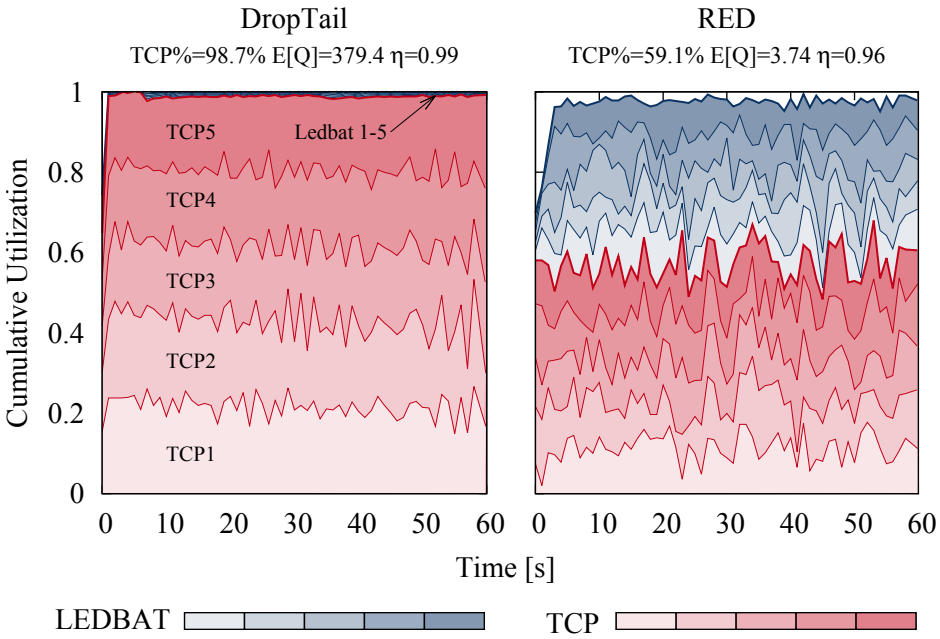


Figure 2.1 Illustrated example of problems arising from the interaction of AQM and CC techniques (ns2 simulation for AQM=RED, CC=TCP+LEDBAT).

capacities, or larger buffer sizes (common defaults for home gateways are well in excess of 1000 packets).

In the RED case, while it is successful in limiting the queue size (less than 4 packets on average), this comes at the cost of (i) a slight 3% reduction of the link utilization, (ii) a complete reset of the relative level of priority between flows. In the case depicted in the figure, the share is now equal among all LEDBAT and NewReno flows, so that LEDBAT operates in a best-effort mode, and is thus as aggressive as TCP. *While an AQM fixes the bufferbloat, it destroys the relative priority among CC protocols.*

While the interaction between LEDBAT and AQM is pointed out in [24] and mentioned by the draft [14], we believe both the depth and the extent of the problem to be underestimated. As for the depth, Sec. 3.3 not only confirms the phenomenon to hold in the real world via Internet experiments on multiple AQM and LPCC techniques, but also shows that the *relative level of priority seldom reverses under AQM.* As for the extent, in reason of the limited availability of actual implementation of AQM and LPCC in the Linux kernel, we are forced to complement our experimental methodology with a simulation based one: Sec. 3.2 summarizes results of over 3,000 simulations, showing that the phenomenon just illustrated *is a fairly general problem, arising from the interaction of AQM and any LPCC protocol.*

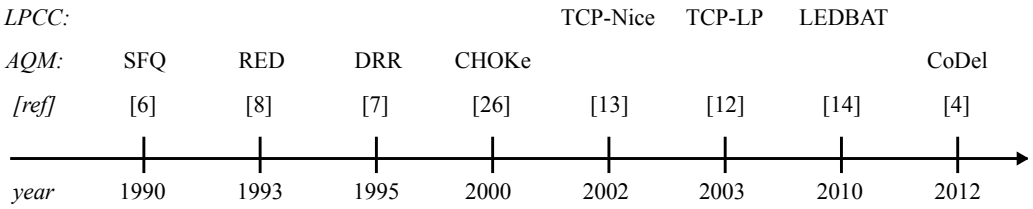


Figure 2.2 Timeline of Active Queue Management (AQM) and Low Priority Congestion Control (LPCC) algorithms.

2.2 Related work

It would be extremely cumbersome to comprehensively retrace over 20 years of Internet research in few pages. A historical viewpoint is sketched in [25]: we extend this viewpoint by reporting in Fig. 2.2 a timeline of research in scheduling/AQM algorithms and LPCC protocols. The timeline clearly shows a temporal separation of these two research topics, which in our opinion helps understand why the AQM vs. LPCC interaction assessed in this thesis was only barely exposed before. In this section, we overview the related work considering (i) AQM and scheduling, notably RED (ii) LPCC protocols, notably LEDBAT (iii) the AQM vs. LPCC interaction, (iv) fairness of capacity share, (v) fluid modeling of TCP and AQM.

2.2.1 Active Queue Management (AQM)

Traditional routers use drop-tail discipline in queues holding packets to be scheduled on each interface, and drop packets only when the queue is full. This mechanism tends to cause global synchronization between flows and penalize bursty flows.

To overcome these issues, AQM disciplines drop or mark packets before the queue is full in a probabilistic way, thus provide endpoints with an earlier congestion indication. As a result, AQM disciplines are able to maintain a shorter queue length than drop-tail queues, which serve as a practical method to counter “bufferbloat”.

Studies on scheduling and active queue management were very popular during the 90s (e.g., SFQ [6], RED [8], DRR [7]), and declined after the beginning of the early 2000s (e.g., CHOKe [26]).

Despite numerous AQM proposals, they have so far encountered limited adoption. The difficulties in tuning RED [27] are well known, and the computational cost of Fair Queuing was, back in the 90s, considered to be prohibitive (see [25] for a historical perspective). The situation has however started to change, with operators worldwide implementing AQM

policies in the upstream of the ADSL modem (e.g., in France, Free implements SFQ since 2005 [9], and Orange starts to deploy SQF [28]) to improve the quality of user experience.

We currently see a resurgence of the topic, in terms of novel proposals (e.g., CoDel [4], AFpFT [29]) and further research [30–32], as also testified by the very recent proposal to create a dedicated IETF AQM WG [33].

We provide the brief mechanism of AQM/Schedulings considered in our work, namely, SFQ, DRR, RED, CHOKe, and CoDel.

SFQ (Stochastic Fair Queuing)

Stochastic Fairness Queueing (SFQ) [6] is a simple implementation of the fair queueing algorithms family. SFQ does not shape traffic but only schedules the transmission of packets based on conversation (or flow). The goal is to ensure fairness so that each flow is able to send data in turn, thus prevent any single flow from drowning out the rest. The performance is impacted by the total queue length and the number of buckets.

SFQ works by dividing traffic into a large but limited number of FIFO queues, one for each conversation, using a hashing algorithm. Queues are serviced in a round-robin fashion, without considering packet lengths. When there are no free buffers to store a packet, the packet at the end of the longest queue is dropped. The limited bucket makes its fair scheduling guaranteed as "stochastically", which would under certain cases put multiple conversations under the same bucket. To prevent such situation from becoming noticeable, SFQ changes its hashing algorithm quite often so that any two colliding conversations will only do so for a small number of seconds. At the same time, using a hashing algorithm makes SFQ less accurate than other fair queueing scheduling policies, but also requires less calculations while being almost perfectly fair. And thus is suitable for high-speed network implementation.

DRR (Deficit Round Robin)

Deficit Round Robin (DRR) [7] is another approximation of fair queueing to correct the unfair behavior exhibited with previous approximations. It can handle packets of different size without having knowledge of their mean size by keeping track of credits for each flow. This queue mechanism used a well-designed idea to get better performance and can also be implemented in a cost-effective manner.

DRR inherits the queue assignment behavior of SFQ, and to overcome the major problem of ordinary round-robin servicing of queues, which is the unfairness caused by possibly different packet sizes used by different flows, it assigns a quantum to each queue. Each can

send a packet of size that can fit in the available quantum. If not, the idle quantum gets added to this queue's deficit and the packet can be sent in the next round. The quantum size is a very vital parameter in the DRR scheme, determining the upper bound on the latency as well as the throughput.

RED (Random Early Drop)

Random Early Drop (RED) [8] is one of the most important representative AQMs used in this thesis. Due to its relatively simple design, it has been in long time the AQM considered in quite a few researches and under continuing improvement.

A traditional RED monitors the average queue size and compares it with two parameters min_{th} (minimum threshold) and max_{th} (maximum threshold). If the buffer is shorter than min_{th}, no packet will be dropped; on the contrary, all packets will be discarded if the buffer is larger than max_{th}. When the buffer length is within the defined range, it drops packets based on statistical probabilities with another parameter p (probability).

In spite of its success in combating TCP global synchronization and "bufferbloat", it requires careful tuning [34] of its parameters in order to provide good performance. Therefore some work has been done aiming to improve the classical proposal, WRED (Weighted RED) [35], ARED (Adaptive RED) [36], RRED (Robust RED) [37], to name a few.

CHOKe (CHOose and Keep for responsive flows, CHOose and Kill for unresponsive flows)

CHOKe [26] is a variation of RED which tries to penalize flows that submit more packets than is allowed by their fair share. By doing this, the scheme aims to approximate the fair queueing policy. It is stateless and easy to implement with a minimum overhead.

While RED reduces the delays experienced by most flows, the percentage of packets dropped from each flow over a period of time is almost the same. Thus, it is unable to penalize unresponsive flows. The idea behind CHOKe is to use the "sufficient statistic" contents of the FIFO buffer as an indicator of the incoming traffic to identify misbehaving flows. Based on RED's mechanism and parameters, the difference is that once the queue passes min_{th}, before marking any incoming packets with a dropping probability, it randomly draws a packet from the FIFO buffer and compares it with the arriving packet. If they both belong to the same flow, then they are both dropped, else it is admitted with a probability computed as in RED. Therefore, packets of misbehaving flows are dropped more often than packets of well-behaved flows considering that it is more likely to draw packet belonging to misbehaving flows and encounter incoming packet from them.

CoDel (Controlled Delay), FQ-CoDel (FlowQueue-CoDel)

Controlled Delay (CoDel) [4] is a queue management algorithm designed to overcome bufferbloat in network links by setting limits on the delay that network packets suffer. CoDel claims to have addressed some fundamental misconceptions in the traditional AQM algorithm such as RED, it maintains that the packet sojourn time, instead of average queue size, should be used as an indicator of network congestion condition.

The algorithm is independently computed at each hop. The algorithm operates over an *interval*, which is initially set to 100ms. Per-packet queuing delay is monitored through the hop. As each packet is dequeued for forwarding, the packet sojourn time in the queue is calculated. The lowest queuing delay for the interval is stored. When the last packet of the interval is dequeued, if the lowest queuing delay for the interval is greater than a *target* delay (5ms), this single packet is dropped and the interval used for the next group of packets is shortened (in accordance with the inverse square root of the number of successive intervals in which packets were dropped due to excessive queuing delay). Otherwise, the packet is forwarded and interval is reset to initial *interval*.

Due to the controlling of packet delay, CoDel claims to be insensitive to round-trip delays, dynamically changing link rates, and traffic loads. With its implementation for Linux kernel and open-source router firmware, it has been exhaustively tested and is adopted as the standard AQM/scheduling in both Linux distribution (Linux 3.6 and later), open-source firmware (OpenWrt, CeroWrt, dd-wrt, IPfire, etc), commercial routers (Qualcomm, Netgear), and operators products (Free.fr) [38].

FlowQueue-CoDel (FQ-CoDel) was presented in 2012 as a hybrid packet scheduler/AQM algorithm for fighting bufferbloat and reducing latency across the Internet. It is based on a two-tier Deficit Round Robin (DRR) queue scheduler, with the CoDel AQM algorithm operating on each sub-queue. Unlike DRR, there are two sets of flows - a “new” list for flows that have not built a queue recently, and an “old” list for flow-building queues. As a consequence, the short flows that don’t build up more than a quantum of bytes before being visited by the scheduler can get priority over long flows that build up queues not empty out each round. This mechanism reduces the impact of the head-of-line blocking from bursty traffic. It provides isolation for interactive/low-rate traffic such as DNS, web, and video streaming traffic.

2.2.2 Low Priority Congestion Control (LPCC) protocols

Congestion and flow control may have different goals, such as controlling the streaming rate over TCP connections as done by YouTube or Netflix, or aggressively protecting user

QoE as done by Skype over UDP, or to provide *low-priority* bulk transfers service toward the Cloud (e.g., Picasa background upload or Microsoft Background Intelligent Transfer Service BITS, etc.).

The standard TCP congestion control needs losses to back off: this means that, under a drop-tail FIFO queuing discipline, TCP necessarily fills the buffer. As uplink devices of low-capacity home access networks can buffer up to hundreds of milliseconds, this may translate into poor performance of interactive applications (e.g., slow Web browsing and bad gaming/VoIP quality). Low priority congestion control protocols tackle this problem by using congestion indicator other than packet loss that enables it to react faster than standard TCP.

Studies on low-priority congestion control protocols started in the early 2000s, with several contributions such as TCP-Nice [13], TCP-LP [12], 4CP [39, 11] and LEDBAT [14]. While it is out of scope for this thesis to provide a full overview of the above protocols, we refer the reader to [40] for a more thorough survey. Finally, a simulation based analysis of the impact of LEBDAT parameters on its behavior has been recently carried out in [41]. It focuses on a DropTail bottleneck-link shared by LEBDAT and TCP New Reno flows, proposing a set of LEBDAT parameters that minimize the overall LEBDAT bandwidth share.

Protocols mentioned above share the same low-priority spirit of LEDBAT. We carried out a simulation-based comparison of TCP-Nice, TCP-LP, and LEDBAT in [42], showing that LEDBAT has the lowest level of priority.

In terms of adoption, while TCP-LP and TCP-Nice have been around in the Linux kernel for about a decade, they have seldom been used [3]. However, ignited by the ease of application-layer deployment, scavenging congestion control services are now becoming popular: examples of this trend are represented by Picasa's background upload option and the adoption of an LPCC by BitTorrent. Indeed, BitTorrent recently abandoned TCP in favor of LEDBAT, a "low extra delay background transport" protocol implemented at the application layer over a UDP framing.

In terms of AQM vs. LPCC trend, it is worth to highlight an interesting similarity between the most recent approaches of either class (i.e., CoDel and LEDBAT), as both control queuing delay explicitly: they both employ a *target delay* parameter upon which dropping decisions or congestion window modifications are based respectively.

We report, in the time order, the mechanism of LPCCs considered in our work, namely, TCP-Nice, TCP-LP, and LEDBAT.

3. In recent kernels, TCP-Nice is no longer available as a kernel module, whereas TCP-LP it available but not among the TCP flavors allowed by default via `net.ipv4.tcp_allowed_congestion_control`

TCP-Nice

TCP-Nice [13] is intended to provide an abstraction of infinite free network bandwidth for background data. The primary goals are (i) to eliminate interference with regular demand traffic, as well as avoid self-interference and (ii) to reap a significant fraction of the spare network bandwidth available. Unlike traditional TCP Reno, TCP-Nice uses increasing RTT as congestion signal instead of packet loss. It derives TCP Vegas's multiplicative decrease of congestion window to ensure an aggressive responsiveness to early congestion detected. By only modifying sender-side congestion control, TCP-Nice can approximate simple prioritization at the routers without modifying the routers and receiver at all.

Different from LEDBAT, that reacts to instantaneous one-way delay (OWD) variations, TCP-Nice instead reacts to RTT variations, thus could possibly reduce the sending window due to growing delay in the reverse path similar to TCP Vegas [43].

TCP-LP (TCP Low Priority)

TCP Low Priority (TCP-LP) [12] shares the same principle as TCP-Nice to utilize the spare network bandwidth without interfering with foreground traffic by inferring congestion earlier than TCP. It modifies the loss-based behavior of NewReno with an early congestion detection based on the distance of the one-way delay (OWD) from a weighted moving average calculated on all observations. In case of congestion (the smoothed OWD exceeds a threshold within the range of the minimum and maximum delay), the protocol halves the rate and enters an inference phase, during which, if further congestion is detected, the congestion window is set to one and normal NewReno behavior is restarted. This differs from LEDBAT's aim of explicitly bounding the maximum delay introduced in the bottleneck queue, which is particularly important for VoIP, video conferencing, gaming and all other interactive delay-sensitive applications.

LEDBAT (Low Extra Delay Background Transport)

Low Extra Delay Background Transport (LEDBAT) [14] was initially introduced in 2008 by BitTorrent, and later recognized on the Web. It implements a distributed congestion control mechanism, tailored for the transport of non-interactive traffic with lower-than-best-effort priority, whose main design goals are:

- Saturate the bottleneck when no other traffic is present, but quickly yield to TCP and other UDP real-time traffic sharing the same bottleneck queue.
- Keep delay low when no other traffic is present, and add little to the queuing delays induced by TCP traffic.

- Operate well in drop-tail FIFO networks, but use explicit congestion notification (e.g., ECN) where available.

These goals are achieved by reacting to congestion notification earlier than TCP, and by reducing its transmission rate to avoid harming current traffic: while TCP infers congestion from packet losses, LEDBAT infers congestion from increasing buffering delay, hence prior than losses occur.

Its congestion control algorithm is based on the One-Way Delay (OWD) estimation: queuing delay is estimated as the difference between the instantaneous delay and a base delay, taken as the minimum delay over the previous observations. Whenever the sender detects a growing OWD, it infers that queue is building up and reacts by decreasing its sending rate with reference to a *target* delay parameter. This way, LEDBAT reacts earlier than TCP, which instead has to wait for a packet loss event to detect congestion.

2.2.3 Joint AQM and LPCC studies

From our knowledge, only [24] merely mentions, without being the main focus, the interplay of AQM and LEDBAT via an experimental approach: in one of the tests, authors experiment with a home gateway that implement some (non-specified) AQM policy other than DropTail. When LEDBAT and TCP are both marked in the same "background class" the "TCP upstream traffic achieves a higher throughput than the LEDBAT flows but significantly lower than" that under DropTail [24]. This fact is also recognized by the LEDBAT RFC, which states that under AQM it is possible that "LEDBAT reverts to standard TCP behavior, rather than yield to other TCP flows" [14]. Hence, the interplay of AQM and LPCC has been anecdotally covered, though a broad and deep study is missing so far.

2.2.4 Fairness

Our main focus concerns fairness of the capacity share among heterogeneous control protocols on a bottleneck governed by AQM. While fairness is a long studied subject, its investigation generally considered rather different settings. First, it has often been tackled in the *intra-protocol* case [44, 23, 45, 21]: i.e., heterogeneous settings of a single protocol flavor. For instance, [44] studies RTT unfairness of TCP Reno.

Fairness in the *inter-protocol* case, thus closer to ours heterogeneous control protocols settings, has long been studied as well [46, 32, 29]. Old works focused especially on undesirable side effect of delay-based congestion control of Vegas, that makes it back off in the presence of TCP Reno [46]. Even more recent work on the topic studies different issues than ours. Authors of [32, 29] focus on several high-speed variants of TCP: in

their case, fairness between the different protocols is thus desirable, while in our settings *unfairness would be desirable* (as it would imply that low-priority property is maintained). Complementary to this work, authors in [29] design and analyze an AQM scheme (named AFpFT after Approximate Fairness through Partial Finish Tag), that they show via `ns2` simulations to reinstate fairness in the heterogeneous protocols case [32].

2.2.5 Fluid modeling

Fluid models have been extensively used in the literature to investigate the dynamics of TCP flows [47–53]. While several lines of research do exist (e.g., [52, 53] model the network as a time-delay linear system and TCP congestion control algorithms as Smith predictor controllers whose aim is to track an exogenous reference signal which models the TCP congestion window), the mainstream approach, closer to ours, is to model TCP window dynamics by means of either Delay Differential Equations (DDE) or Partial Differential Equations (PDE), [47–50]. We point out that, since generally a single dominant TCP flavor is modeled [47, 48, 52] (optionally including unresponsive background traffic [49] or short-lived connections [50]), the novelty in this thesis lies in the definition of a fluid model of *heterogeneous* responsive sources, notably including LEDBAT.

Chapter 3

Hands-on investigation

As motivated in Chap. 2, the "reprioritization" phenomenon could take place when TCP and LEDBAT are competing under a bottleneck governed by RED. In this chapter, we exploit the generality of this phenomenon using various methodologies described in Sec. 3.1, followed by results gathered in `ns2` simulation Sec. 3.2 and experiment testbed under both controlled and uncontrolled environment in Sec. 3.3. We also perform sensitivity analysis to further refine the understanding of its behavior by varying network scenarios. We finally draw the conclusion in Sec. 3.4.

3.1 Methodology

Simulation campaign

Our `ns2` simulations aim to test the validity of the reprioritization phenomenon under the largest possible set of scenarios.

We include a number of representative AQM techniques: namely, SFQ [6], RED [8], DRR [7], CHOKe [26] and CoDel [4]. Similarly, we consider a number of representative LPCC techniques such as TCP-Nice [13], TCP-LP [12] and LEDBAT [14]. As for standard best-effort TCP, we consider the IETF NewReno variant [1]. Notice that some of these modules are not directly available in `ns2` (version 2.33), therefore we patch it to support CHOKe [26], CoDel [4], LEDBAT and TCP-Nice (the last two LPCCs use our own open source implementations [54]).

1. We point out that, in recent times both Linux and Windows have drifted from IETF recommendation, selecting Cubic and Compound as default TCP flavors respectively. Yet, as both Cubic and Compound are designed to be more aggressive than NewReno, we henceforth expect reprioritization to hold for these flavors as well.

A couple of points are worth stressing. Although we are aware of the fallacies of RED, we believe that it needs to be considered as a reference benchmark (to which indeed CoDel is compared in [4]). Additionally, note that RED is one of the few AQM policies available on common recent Linux kernels shipped with standard distributions, we believe it would not make sense to exclude it from this study even due to its known performance issues [34, 27]. Similarly, we are aware of the latecomer unfairness issues of LEDBAT [23, 45], but its widespread use makes it necessary to be considered in the mix.

Experimental campaign

The experimental campaign is carried out to confirm the occurrence of the reprioritization phenomenon in the real world.

Since our aim is not to propose any new AQM or LPCC, nor to replicate the full set of simulation campaign, we select the ones that are already available in recent Linux 3.2 kernel: namely, RED [8] and SFQ [6] for AQM and TCP-LP [12] and LEDBAT for the LPCC protocol family. As for the LEDBAT, we employ the `libUTP` [55] application-level implementation of BitTorrent that we already analyzed in [56].

Incidentally, while our choice is forced by the availability of AQM and LPCC implementation in Linux, in reason of results of the simulation campaign we expect the performance of other AQM+LPCC combinations to fall into the boundaries defined by {RED,SFQ}$\times${TCP-LP,LEDBAT}. On the one hand, this is due to the fact that RED vs. SFQ yield to small vs. large queuing delays respectively; on the other hand, it is known that TCP-LP and LEDBAT have the most and the least aggressive behavior in the LPCC family [42].

Throughout this chapter, we express system performance mainly in terms of the link utilization η, the share of the link exploited by the TCP aggregate $TCP\%$, and the average queue length $E[Q]$. For convenience, we can either express $E[Q]$ in number of packets (possibly normalized over the buffer size $E[Q]/Q_{max}$ to gauge the bufferbloat intensity), or as the actual packet sojourn time in the queue (a more direct measure of the user quality of experience).

To facilitate cross-comparison and independent validation of our results, we make our scripts and datasets for both simulation an experimental campaign available to the research community at [57].

In particular, the network scenario analyzed in the following two chapters is shown in Fig. 3.1. N_w TCP flows and N_z LPCC flows compete at the bottleneck governed by AQM, which has B as buffer size, and C as link capacity.

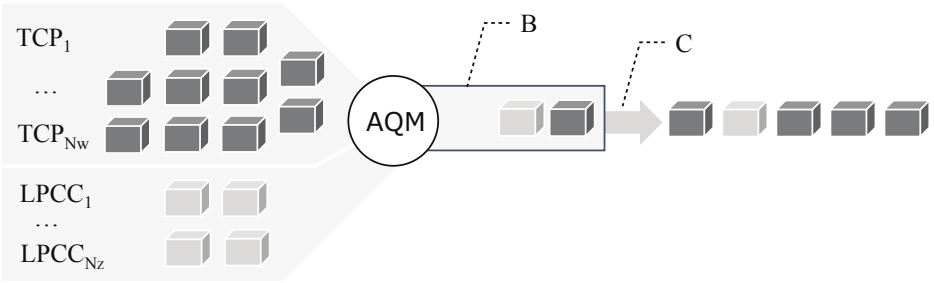


Figure 3.1 Network scenario.

3.2 Simulation results

In this section, we aim at conveying straightforward the most relevant message we gather from simulation. We therefore incrementally extend the breadth of our results by initially considering the impact of AQM policies in Sec. 3.2.1, then the joint impact of AQM and LPCC in Sec. 3.2.2, and finally the sensitivity analysis in Sec. 3.2.3 considering the impact of other network parameters such as buffer size, link capacity, heterogeneous RTT delay, and flow duty cycle.

For simplicity, we consider an equal number $N = 5$ of best-effort TCP NewReno and low-priority flows sharing a link having capacity $C = 4$ Mbps and a buffer size $Q_{max} = 400$packets (corresponding to a maximum bufferbloat of 1.2 seconds). All flows are backlogged, start at time $t = 0$, and have a homogeneous $RTT = 50$ ms. Simulations last for 60s, and 10 runs are repeated for each parameter settings. A larger number of settings is reported in Sec. 3.2.3.

3.2.1 Impact of AQM policy

For the time being, we fix the LPCC to LEDBAT and examine the impact of different AQM techniques. Fig. 3.2 reports the bufferbloat intensity $E[Q]/Q_{max}$ (top), link utilization η (middle) and TCP% breakdown (bottom) for varying AQM policies, and for DropTail as a comparison (right). As expected, at the price of a slight decrease in the link utilization, AQM reduces the intensity of the bufferbloat: compared to a persistently full DropTail buffer, queue size and delay is from 14 to 77 times smaller, using SFQ and RED respectively. This implies that, under AQM the queuing delay is always less than 85 ms (SFQ, the worst case [2]), and generally much lower. However, we see that the capacity share of the TCP aggregate can drastically reduce 35%-46% (under CoDel and CHOKe respectively). Notice that this

2. It has to be noted that, strictly speaking, SFQ is a scheduling discipline and not an AQM mechanism: as such, longer delay is expected since no “early” drops occur.

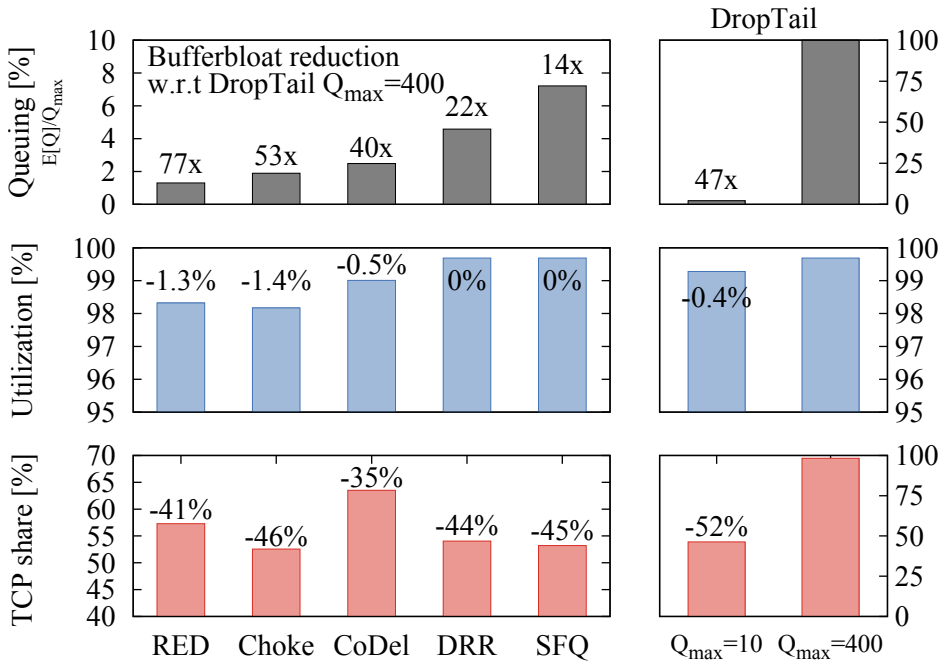


Figure 3.2 Impact of AQM policies on the bufferbloat intensity $E[Q]/Q_{max}$ (top) link utilization η (middle) and TCP% breakdown (bottom).

result alone extends the validity of the phenomenon observed under unknown [3] AQM policies in [24]. Importantly, we point out that simply reducing the DropTail buffer size is not a solution to the problem either: notice that (i) as shown in Fig. 3.2, shorter DropTail buffer behaves like AQM, as bufferbloat is reduced at the expense of reprioritization and (ii) in the case of varying link bandwidth such as WiFi (where rates can vary from few to several tens of Mbps), there is no fixed buffer size that would not translate into bufferbloat. Intrinsically, when the queue length is short, either due to AQM or tiny buffers, LPCC cannot reliably infer congestion signals from delay measurement.

Finally, Fig. 3.3 shows a Kiviat chart of the impact of AQM policies on drop rate of different flow types, considering average drop rate (top axis), TCP drop rate (bottom right axis) and LPCC drops (bottom left axis). Compared to DropTail, all AQMs increase drop rate of all flows (due to early drop decisions) but not enough to influence the link utilization. Interestingly, DropTail penalizes TCP as much as LPCC, whereas AQMs generally penalizes TCP more than LPCC – the root cause of reprioritization. Drop rates for TCP are worst in the case of CHOKe, which is designed to leverage power of two choices to especially penalize

3. Authors just mention that "different prioritized traffic classes" are implemented in the "modern home gateways" used in their experiments.

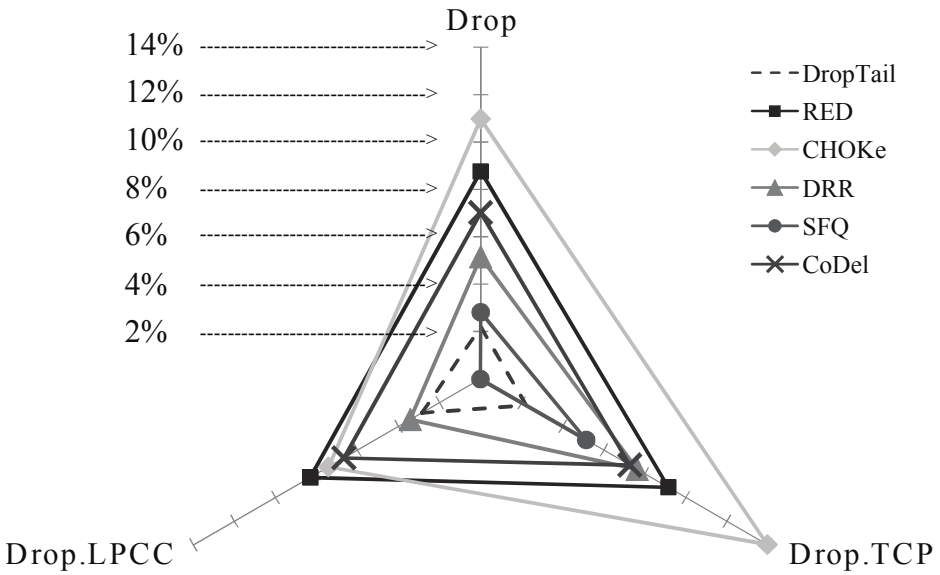


Figure 3.3 Impact of AQM policies on the drop rate.

heavy-hitter flows. In the case of SFQ scheduling, notice that LPCC almost experiences no losses.

Summarizing, reprioritization of LEDBAT holds under any AQM. Additionally, the use of tiny buffers is not helpful in terms of reinstating priorities between LPCC and best-effort TCP.

3.2.2 Impact of AQM policy and LPCC protocol

We now explore the full product of LPCC flavors and AQM techniques. Under DropTail, it is known [42] that LEDBAT achieves the lowest priority against best-effort TCP, followed by TCP-Nice and TCP-LP. We illustrate results as a parallel coordinate plot in Fig. 3.4. Having noticed that link utilization is subject to small variations, we consider the two main metrics of interest: namely the queuing delay and the TCP share. For convenience, we normalize the bufferbloat over the queue size and report its normalized intensity $E[Q]/Q_{max} \in [0,1]$ on the left y-axis, and report the normalized TCP share of the link capacity $\frac{TCP}{TCP+LPCC} \in [0,1]$ on the right y-axis. In the parallel coordinate plot, each (LPCC, AQM) pair is represented as a line, joining the delay and TCP share figures. For reference purposes, we put a light-gray strip representing the ideal case where the queue is short but priorities are unaltered.

We see that the three (LPCC,DropTail) combinations appear as horizontal lines on the top of the figure: this happens since, under DropTail, bufferbloat has maximal intensity while TCP monopolizes the bottleneck. We see that this holds for any LPCC protocol, with a slight

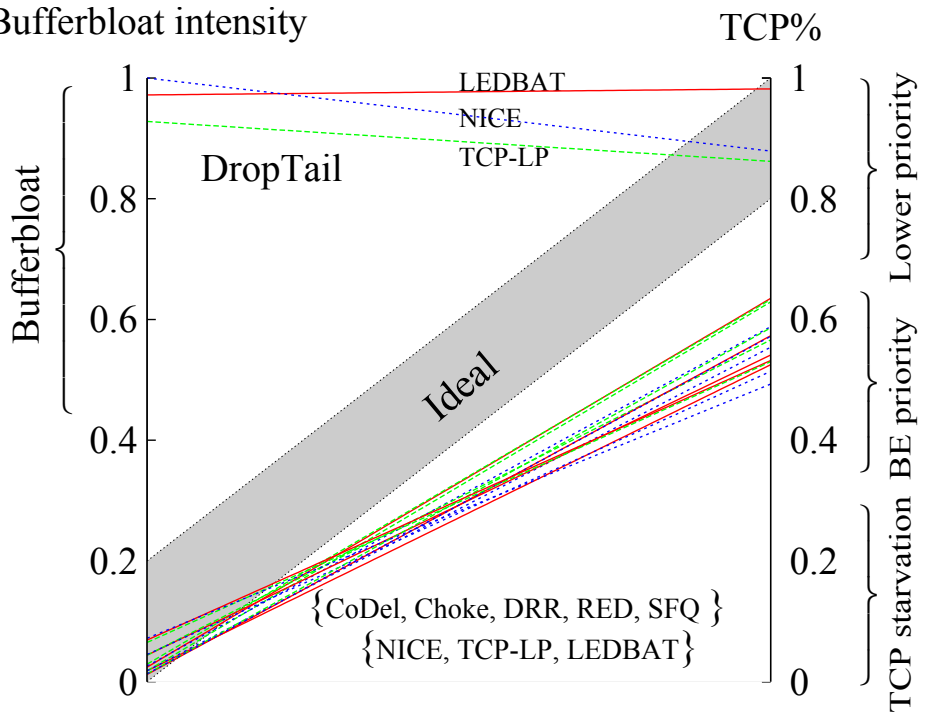


Figure 3.4 Joint impact of AQM and LPCC on the queuing delay and TCP breakdown.

separation of the curves reflecting the order of priority observed in [42] (from top to bottom, the least to the most aggressive LPCC with respect to TCP).

Excluding DropTail, we also see that all (LPCC,AQM) combinations are very close, as AQM implies low bufferbloat but jeopardizes CC priorities. Specifically, as under AQM all TCP and LPCC flows have roughly the same priority, the lines fall in the best-effort (BE) priority range, below the ideal strip, with best-effort TCP getting slightly more than half of the link share.

Summarizing, reprioritization holds under any considered LPCC and AQM. The choice of a specific (LPCC,AQM) combination has only very limited impact on the system performance – and, in any case, is not helpful in reinstating priorities between LPCC and best-effort TCP.

3.2.3 Sensitivity analysis

To further extend the breadth of our findings, we conduct over 3,000 simulations to investigate the impact of other network parameters such as buffer size Q_{max}, link capacity C, heterogeneous RTT delay, and flow duty cycle. We first discuss each of the above factors

Table 3.1 Sensitivity analysis parameters

Variable parameters

Parameter	Default	Range	Subsection
Qmax (pkt)	400	100, 200, 300, 400	Buffer size and capacity
C (Mbps)	4	0.25, 0.5, 1, 2, 4, 6, 8, 10	Buffer size and capacity
Workload (%)	100	10 - 100 (interval 10)	Flow duty cycle
RTT (ms)	6	2, 4, 6, 8, 10	RTT delay heterogeneity

Fixed parameters

Parameter	Value
AQM	DropTail, RED, CHOKe, DRR, SFQ, CoDel
LPCC	LEDBAT
No.TCP flows	5
No.LPCC flows	5
Duration (s)	60
No.Runs	10

separately, and finally compactly report their impact. We anticipate that under all explored circumstances [4] the early outlined phenomenon remains valid.

The parameters used in our analysis are listed in Tab. 3.1 (as parameters listed in the table are by no means exhaustive, we invite the reader to use the provided scripts [57] shall they reproduce these experiments). We let 10 flows (5 TCP + 5 LEDBAT) competing at the bottleneck at the same time for 60s, results are obtained from the average of 10 runs for each simulation. We will explain the varied parameters investigated in detail in its corresponding sub-section.

Buffer size and capacity

We explore over 30 combinations of link capacity varying in the 250Kbps to 10Mbps range, and buffer size between 100 and 400 packets (detailed values can be found in Tab. 3.1).

First, when fixing the value of buffer size, we observe a decrease of packet drop rate as the link capacity increases, while the link utilization is mostly unaffected by capacity variations. This is expected since TCP ramps up to exploit the full capacity both in the case of DropTail or AQM.

4. We avoid reporting some of which (such as scenarios with a larger number of flows, or dynamic flow arrivals patterns, etc.) in full details, to privilege clarity over completeness.

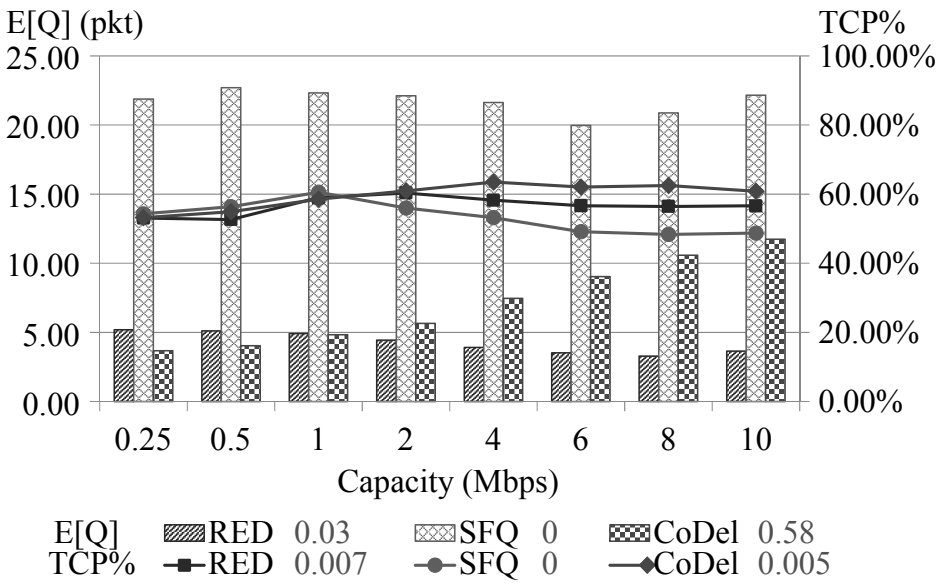


Figure 3.5 Sensitivity analysis: average buffer occupancy and TCP% share for varying capacity (x-axis) and buffer sizes (standard deviation across buffer size is reported in the legend).

If we instead fix the capacity, then the impact of the buffer size is limited in case of AQM. This is again expected, since AQM is configured to keep the buffer short irrespectively of its size: in other words, the average queue size is $E[Q] \approx Q_{max}$ under DropTail for any Q_{max}, whereas $E[Q] < Q_{max}$ under any AQM technique irrespectively of Q_{max}.

We elaborate more on capacity variations in Fig. 3.5, which reports the average queue size $E[Q]$ (in packets, left y-axis) and TCP% breakdown (right y-axis) as a function of the link capacity (x-axis) for some AQM mechanisms (namely we limitedly report RED, SFQ and CoDel to avoid cluttering the picture). The picture reports the average of the metrics over all buffer sizes, and the legend is further annotated with the largest standard deviation across buffer size (i.e., for any given capacity, we evaluate the standard deviation of the metrics of interest across different buffer sizes, and report the largest across all capacities). As the standard deviation is very low, this confirms reprioritization results to hold across the explored scenarios.

As expected, capacity does not have an impact on average queue size $E[Q]$ except for CoDel, which controls the packet-sojourn time: as a result, larger capacities translate into a larger number of packets to be allowed in the buffer under the same target delay configuration (hence, a larger standard deviation). The TCP% breakdown is also unaffected under any capacity and buffer size combination tested, confirming TCP% manages to maintain its priority level over LPCC only under DropTail.

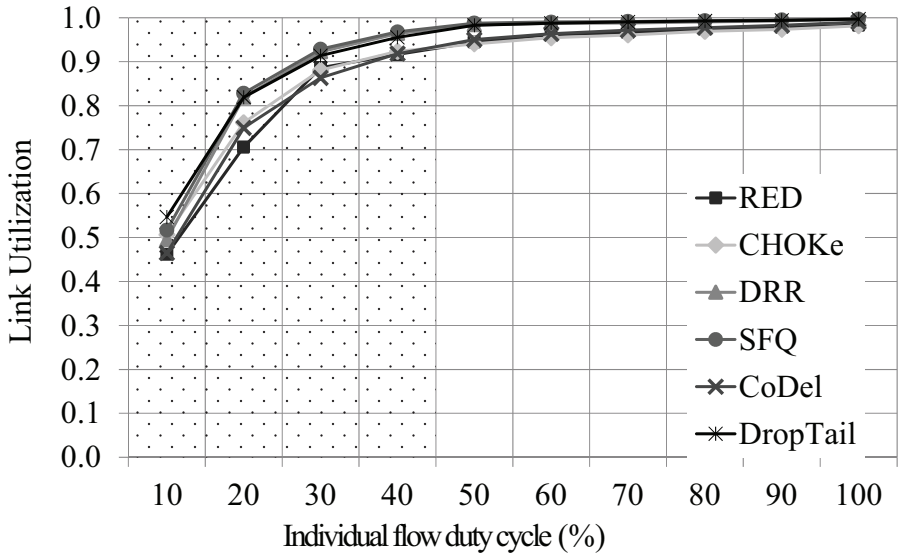


Figure 3.6 Sensitivity analysis: Impact of flow duty cycle on the link utilization.

Flow duty cycle

We next observe that networks are rarely used by backlogged flows: hence, we include duty cycle model simulating different workloads to make our simulation more realistic. We engineer the scenario to simulate different workloads, by controlling the individual flow duty cycle (from 10% to 100% with a 10% step). For each flow, we turn it ON and OFF alternatively during the entire simulation duration. During each ON period, the flow continuously transmits data, and during an OFF period, the flow remains idle and resets its window parameters. At the end of each OFF period (including the initial state), we randomly generate the duration of next ON period, which follows an exponential distribution with average λ. Also, the same procedure is employed at the end of each ON period, generating the duration of next OFF period with average duration μ. Thus, the individual flow duty cycle is defined as $\lambda/(\lambda+\mu)$.

By design, the duty cycle model affects link utilization, so that we can individuate two regimes. Regimes are illustrated as two colored regions in Fig. 3.6, which reports the average link utilization as a function of the individual flow duty cycle. When individual duty cycle is low, the aggregated load remains, on average, well below the link capacity. Also, as flows are often idle, they will operate in isolation, only seldom interact. Still, due to TCP bursty nature, the buffer can occasionally be filled with a burst of packets belonging to the same congestion window, possibly triggering AQM reactions. In the underload regime, we therefore expect each flow to get a fair share of the link. As individual flow duty cycle increases, and the

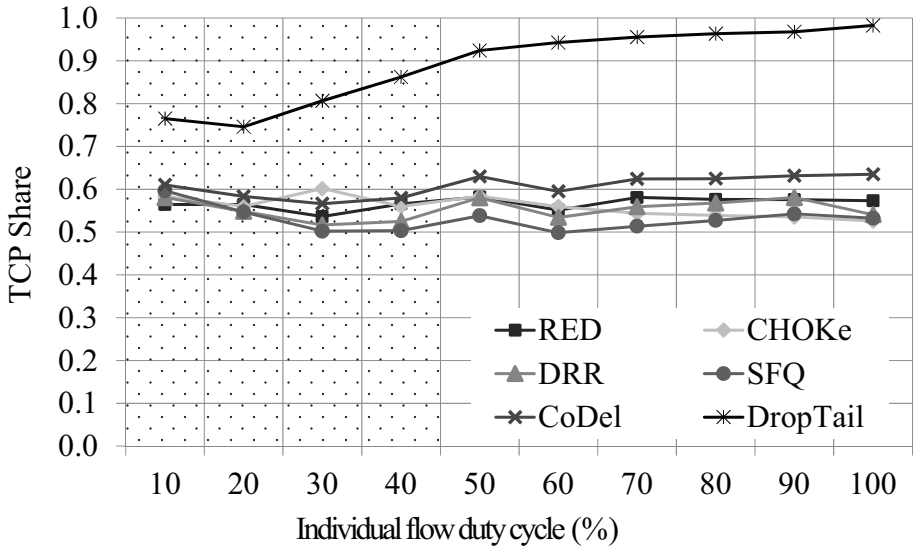


Figure 3.7 Sensitivity analysis: Impact of flow duty cycle on the TCP breakdown.

aggregated load approaches the link capacity, packets of different flows now mix in the buffer, and we expect AQM decisions to penalize the most aggressive flows – which as we previously observed will induce reprioritization. Also in the overload regime, we therefore expect each flow to get a fair share of the link.

As shown in Fig. 3.7, the expected behavior holds. In both the underload zone (shaded left part of Fig. 3.6 and Fig. 3.7) and overload region (white right part), TCP and LPCC always fairly share the link. In case only one flow is active at a time, it will monopolize the bottleneck in both TCP and LPCC cases. When two (or more) flows are active at the same time, in case one of these flows is best-effort TCP, then its AIMD dynamics will let the queue increase and trigger AQM reaction, to the advantage of LPCC flows.

Extending results of Fig. 3.3, we report per-protocol drop rate under different AQMs and workloads in the scatter plot of Fig. 3.8. The diagonal line corresponds to an equivalent drop rate of packets of both TCP and LPCC: as it can clearly be seen, under all considered duty cycle conditions TCP is more heavily penalized with respect to LPCC. Note that the line of CoDel is very close to the diagonal, which means its penalization of TCP is the slightest while SFQ has the heaviest penalization. Additionally, we see that fairness in the loss rate translates in higher TCP share: notice indeed that the loss behavior is reflected in Fig. 3.7, showing that CoDel and SFQ grant the highest and lowest TCP% breakdown respectively.

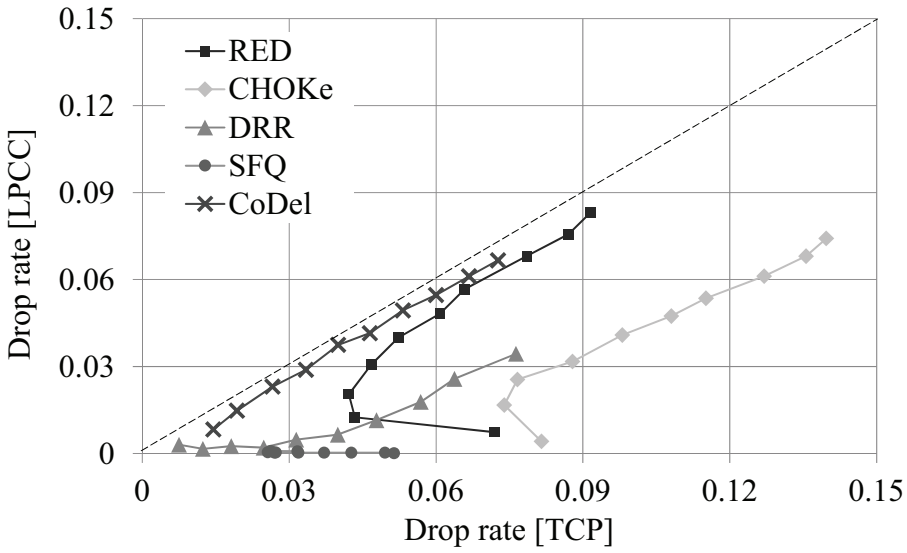


Figure 3.8 Sensitivity analysis: Impact of flow duty cycle on drop rate.

RTT delay heterogeneity

Finally, we acknowledge that rarely flows have homogeneous delay in the real life. Yet, as it is known that TCP exhibits RTT unfairness, it makes sense to carefully build the scenario so to avoid RTT unfairness biasing our conclusions. First, we observe that congestion control protocol in use end-to-end (i.e., TCP vs. LPCC) is not correlated with the RTT, as this is an end-host decision: it follows that TCP and LPCC aggregates have no a priori reason to have different RTT distributions. Second, we observe that as the queuing delay is low due to AQM, the propagation delay is the dominant component of the RTT, which controls the rate at which acknowledgments are received, which in turn controls how fast the sending rate can change: it follows that the relative difference of RTT between heterogeneous flows, more than the absolute RTT value, is important to gauge response to RTT heterogeneity.

As such, indicating by abuse of language with TCP and LEDBAT the set of flows using that congestion control protocol, we engineer the scenario so that (i) both the TCP and the LPCC sets have the same $E[RTT]$, (ii) the $E[RTT]$ of both sets is the same as in the homogeneous case, (iii) within one set, there are no flows having the same RTT delay, (iv) across two sets, there are one TCP and one LEDBAT flows having exactly the same RTT delay. Specifically, we assign $RTT_k = 2k$ ms for the k-th flows (with $k \in [1,5]$) in the TCP and LEDBAT sets (so that $E[RTT] = 6ms$ is the same as the default setting).

Unsurprisingly, we verify that RTT heterogeneity does not affect our conclusion in terms of reprioritization. While RTT unfairness affects the performance of individual flows

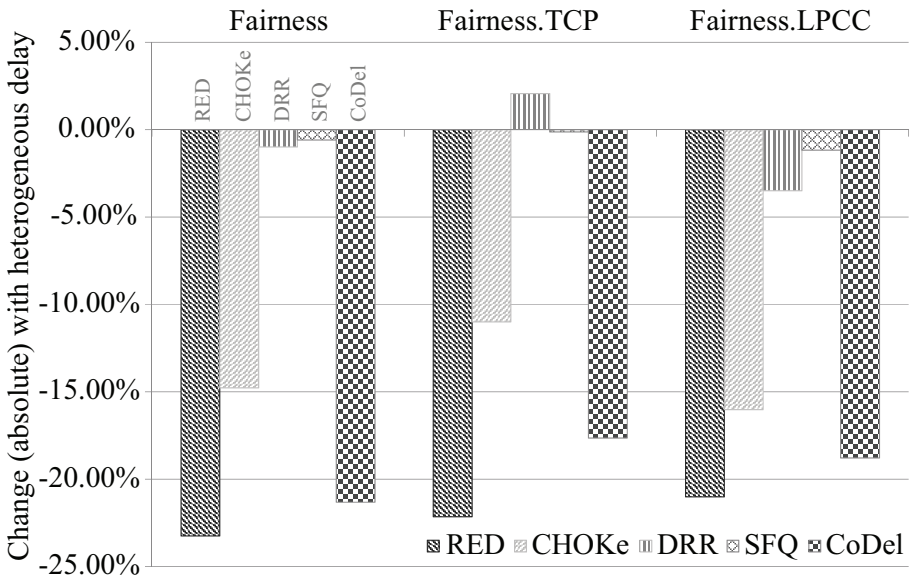


Figure 3.9 Sensitivity analysis: Impact of heterogeneous RTT delay on inter/intra-protocol fairness.

within an aggregate, we see that each aggregate as an entirety can fairly compete with each other on the RTT (i.e., there is a balanced mix of opportunistic flows with small RTT and penalized flows with large RTT in both sets). We resort to the classical Jain fairness index $F_X = (\sum_{i=1}^{N} T_i)^2/(N\sum_{i=1}^{N} T_i^2)$, with T_i throughput of the i-th connection, is calculated within homogeneous TCP and LPCC aggregate (intra-protocol fairness), as well as over the total set of flows (inter-protocol fairness). We visualize the impact of RTT heterogeneity in Fig. 3.9, where we report the relative error in terms of fairness with respect to the homogeneous scenario.

Fig. 3.9 shows that RTT heterogeneity decreases both inter-protocol and intra-protocol fairness up to 20%. Though different AQMs have rather noticeable different impact on fairness decrease, each of them influences TCP and LPCC aggregate in a similar way.

It is more interesting to observe the intra-protocol fairness, i.e., with respect to a set of flows using homogeneous congestion control protocol, which may change significantly depending on the AQM policy. We find that intra-protocol fairness of TCP (F_{TCP}) and LPCC (F_{LPCC}) are quite close with respect to each AQM. To rule out the impact of congestion control protocols heterogeneity, we measure the average Jain fairness index $F = (F_{TCP} + F_{LPCC})/2$ as an indicator of average intra-protocol fairness [5]. We have that fairness under CoDel

5. Notice that $\frac{1}{2}F_{TCP} + \frac{1}{2}F_{LPCC}$ differs from evaluating the fairness over the whole aggregate $F_{\not\!/} = TCP \cup LPCC$, as in the latter case we would be measuring the inter-protocol effect as well.

Table 3.2 Sensitivity analysis report (*Coefficient of variation (CoV)*)

		η	$E[Q]$	$TCP\%$
AQM		0.0073	**0.6916**	0.080
LPCC	DropTail	0	0.038	0.072
	AQM	0.0047	0.034	0.048
Link capacity	DropTail	0.0003	**1.136**	0.035
	AQM	0.0042	**0.604**	0.050
Flow duty cycle	DropTail	0.259	**1.309**	0.120
	AQM	0.313	**1.067**	0.057
RTT delay	DropTail	0.0003	0.022	0.001
heterogeneity	AQM	0.0248	0.052	0.048

($F = 0.81$) is only slightly better than RED ($F = 0.78$) and significantly smaller than SFQ ($F = 0.99$). Notice that CoDel fairness range is coherent with [4], which, however, does not address a comparison with SFQ and reports significantly smaller fairness values for RED (under a more heterogeneous scenario).

From the result, we can conclude that the extent up to which heterogeneous RTT delays can affect inter-protocol and intra-protocol fairness, is closely related to AQM drop decision-making mechanism. Due to the fact that AQM like RED and CHOKe work on the entire buffer status, it follows that with heterogeneous delay, packets will be dropped at more varied intervals, resulting in a lower fairness both *intra-protocol* and *inter-protocol*. For instance, CHOKe uses sampling to find heavy hitters, dropping both packets if the sampled packet and the newly incoming one both belong to the same flow (whereas RED would only drop the new packet): hence, this design choice increases the fairness among flows (with respect to RED) even when heavy hitters are flows opportunistically exploiting RTT heterogeneity. However, scheduling decisions of the DRR and SFQ policies affect individual flow (or stochastic flow aggregates), thus they can always ensure almost perfect fairness under these two scenarios.

Summary

Tab. 3.2 summarizes the results of the sensitivity analysis reporting the coefficient of variation $CoV(X) = \sigma(X)/E[X]$ of the metric X of interest (i.e., link efficiency η, average queue size in packets $E[Q]$ and the TCP aggregate $TCP\%$) due to each of the studied parameters (i.e., link capacity C, buffer size Q, flow duty cycle, RTT delay, etc.). Notice that metrics of interest all have different units and value ranges: CoV describes the dispersion in a

way that does not depend neither on the metric unit, nor on its scale. Hence, this choice allows a relative comparison across metrics, and makes CoV especially suitable in our context.

For parameters other than AQM, we include the result calculated both under DropTail and AQM: i.e., to exclude the impact due to AQM (shown in the first row of the table) we compute the $CoV(X)$ considering each AQM policy in isolation, then report the average $E[CoV(X)]$. The result shows that most values of the CoV are small, which confirms that results are only minimally affected by any of the tested parameters, and especially for the TCP%, confirming the generality of the reprioritization phenomenon illustrated in Fig. 3.4.

We highlight (in bold) CoV value larger than 0.5 in Tab. 3.2, which are of straightforward interpretation. It can be seen indeed that AQM policy can have noticeable impact on average queue size $E[Q]$ (recall Fig. 3.2) due to both their inner working mechanism (e.g. RED limits the queue explicitly while SFQ schedules the flows but allows a proportional increase of packets in the queue with increasing flows) and the variation of their default parameters (untested in this sensitivity analysis due to known difficulty in tuning AQM [27], and so not accounted for in CoV). Obviously, buffer size influences heavily the average queue size under DropTail (CoV=1.136), whereas flow duty-cycle translates into a wider range of queue sizes under both DropTail (CoV=1.309) and AQM (CoV=1.067), as queues are possibly empty unlike in backlogged workload.

Summarizing, reprioritization holds under most networking scenarios. While the specific scenario settings may have an impact on fairness and queue size statistics, they have nevertheless only very limited impact on the reprioritization phenomenon.

3.3 Experimental results

In this section, we present the results gathered from experimental testbed. For any AQM and LPCC combination, we explore some experimental settings, including varying bottleneck capacity C, configuration of AQM parameters, number of flows N in the bottleneck, in both an emulated testbed (Sec. 3.3.1) and in a real Internet deployment (Sec. 3.3.2).

3.3.1 Testbed experiments

In the testbed experiments, we directly connect two PCs through a crossover Ethernet cable. Capacity limitations are either emulated through a Hierarchical Token Bucket (HTB) of the standard Linux traffic control `tc` suite, or natively by forcing $C = 10$ Mbps PHY Ethernet through `ethtool` (which is a more reliable option than HTB). In both cases, we turn off most features that could possibly interfere with the experiments (e.g., jumbo frames, TCP

segmentation offload, interrupt coalescing). To emulate a WAN setup, we inflate RTT delay by a constant amount equal to 30 ms using `netem`. We configure the `tc` queuing discipline to either DropTail (i.e., the default `pfifo_fast`), RED (with state-of-the-art configuration) or SFQ.

We generate backlogged traffic during 60 seconds experiments, using `iperf` for all TCP flavors (both best-effort and low-priority) and simple client/server applications provided by `libUTP` [6] for the application-layer LEDBAT implementation. For best-effort TCP, we report results using NewReno, the protocol of choice at the IETF, and leave the study of Compound (default TCP flavor in Windows) and Cubic (default in Linux) for future work [7].

As for queuing delay measurement, we point out that dark buffers may lay at multiple points in the kernel stack (e.g., TCP buffers, device driver), and that buffering may occur outside the host (e.g., in the ADSL modem in Internet experiments). Hence, we opt for a simple methodology that mimics the way in which TCP-Nice measures the queuing delay: specifically, we monitor the RTT through a low-frequency ICMP ping command and estimate the queuing delay samples as $Q_i = \mathrm{RTT}_i - \min_{j \le i} \mathrm{RTT}_j$. Notice that, as the reverse direction is carrying only ACK data and is thus not congested, we are safe in assuming that the RTT variation is due to queuing at the sender side.

A further example of the temporal evolution of the utilization breakdown of TCP vs. LEDBAT flows is depicted in Fig. 3.10 for different AQM techniques (DropTail, RED, SFQ), capacities (500 Kbps, 10 Mbps) and emulation technique (HTB, PHY). The phenomenon early shown in Fig. 2.1 is thus confirmed for different AQMs and testbed settings: shortly, AQMs induce reprioritization of heterogeneous CC flows. We also see that, while when the capacity is abundant the breakdown is smooth, the opposite happens when capacity is scarce (which additionally leads to unfairness at short timescales).

We conduct a more systematic experimental testbed campaign that is summarized in Tab. 3.3, reporting the TCP breakdown and average queuing delay for an emulated HTB capacity of $C = 500$ Kbps. The total number of flows varies in $N \in \{2, 10\}$, with flows equally split in the best-effort TCP and LPCC families. Experiments report the average over 3 independent runs. As we may now expect, DropTail leads to bufferbloat of multiple seconds [8], which is instead solved by both RED and SFQ. The picture may change completely

6. libUTP is the uTorrent Transport Protocol (uTP) library implementation of LEDBAT, released under the MIT license at [55]. Version d4685a3 (May 2012) was used in all experiments, though we point out that libUTP has been very stable since, with only 4 commits, all of which were not related to congestion control issues.

7. While Windows, and thus Compound, constitutes the largest portion of hosts, it would be difficult to replicate the same methodology. Preliminary tests with Cubic suggest the phenomenon to hold; at the same time, we incur in problems with Cubic vs. TCP-LP since the latter appears to be *more aggressive* than Cubic under DropTail– so we prefer to examine the issue at a later time.

8. In our settings, this is due to our emulated capacity limitations, coupled to the default Network Interface Card (NIC) queue length, which for Ethernet NICs is set to 1000 packets in Linux.

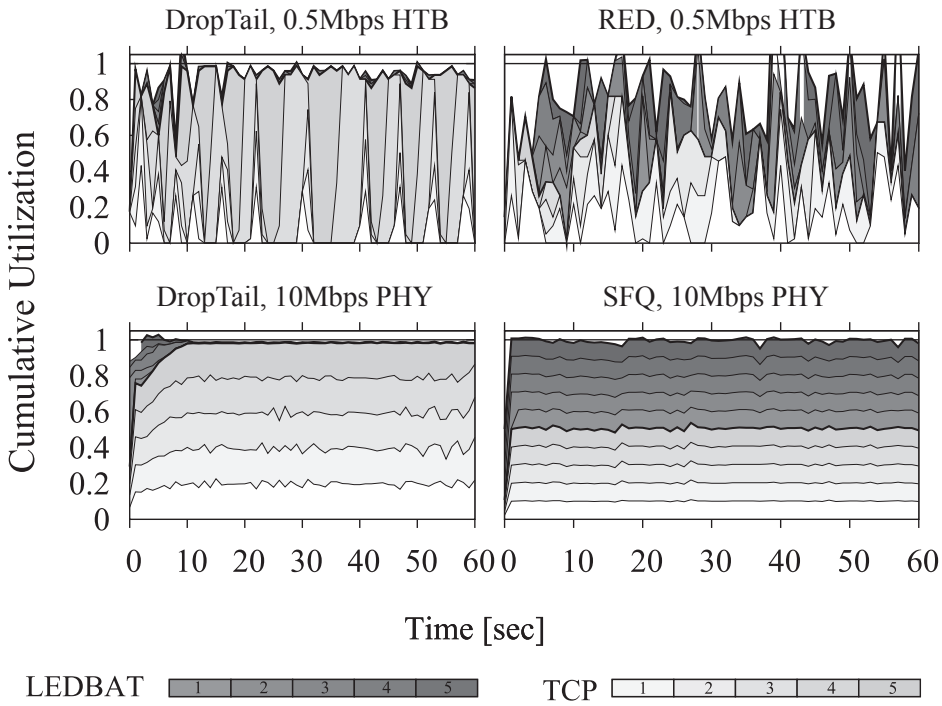


Figure 3.10 Testbed experiments: Utilization breakdown among TCP and LEDBAT, for different AQM techniques (DropTail, RED, SFQ), capacities (500 Kbps, 10 Mbps) and emulation technique (HTB, PHY).

under RED, depending on the LPCC flavor and the number of flows: indeed, while for $N = 10$ the reprioritization happens for both LEDBAT (TCP%=49.3) and TCP-LP (57%), in the case that $N = 2$ flows compete for the same access, the LEDBAT+NewReno combination, through RED, possibly *results in TCP NewReno starvation* (1.7%) while the opposite happens under TCP-LP+NewReno (97.5%).

With respect to reprioritization, which is a general issue holding for a large number of LPCC vs. AQM combinations and AQM settings, we believe the starvation phenomenon to be of more limited interest (as it happens for a single AQM, and furthermore non-systematic, thus tied to specific configurations). Yet, the phenomenon is still worth elucidating further, as these differences reflect the LPCC dynamics of LEDBAT and TCP-LP. The latter is AIMD-controlled, and its low priority stems from a slower recovery *after* losses that the AIMD dynamics force. The former is delay-based and PID-controlled: by limiting the queue size, it will seldom be penalized under RED. Whenever the queue grows due to best-effort TCP AIMD, LEDBAT reduces its own window, and so the chances that a packet will get dropped are very low. Whenever TCP experienced a timeout and abruptly shrank its window,

Table 3.3 Testbed: LEDBAT vs. TCP-LP

		5+5 flows		1+1 flows	
		TCP%	E[Q] ms	TCP%	E[Q] ms
LEDBAT	FIFO	94.9	6437.3	99.5	5304.3
	RED	49.3	11.3	1.7	9.5
	SFQ	76.1	106.4	57.7	15.1
TCP-LP	FIFO	50.8	7870.6	65.8	7471.9
	RED	57.0	21.0	97.5	2.7
	SFQ	49.8	144.2	50.0	25.2

LEDBAT instead grows its window again, but in reason of its low target delay, it will limit the amount of queuing and again prevent its packets from being dropped.

Under SFQ, TCP starvation phenomena are avoided. All flows get hashed in different buckets at enqueue time, and since hash buckets are queried in a round-robin fashion, this guarantees that each flow is able to send data in turn – including the best-effort TCP that was heavily penalized under RED, though reprioritization still occurs. However, the queuing delay under SFQ (which has a default buffer size of 127 packets) grows up to about 100-150 ms, thus approaching the limit of what is considered to be harmful for interactive communication [58].

3.3.2 Internet experiments

We then perform additional experiments on the wild Internet. Since we have already shown the reprioritization to hold for different combinations of LPCC and AQM, our aim here is to disproof that these are artifacts that only arise in testbed due to, e.g., the extremely controlled settings or, conversely, due to unexpected interactions inside the emulation layer.

In order to replicate a setup as close as possible to the typical user scenario, the sender is connected with 802.11g WiFi to an ADSL box. The receiver is connected to another ADSL box of another ISP through the Ethernet interface. The minimum RTT delay between the two hosts is approximately 50 ms, and the capacity between the hosts only slightly exceeds 500 Kbps (so that it can be compared with the testbed).

We carry on experiments only for the LEDBAT LPCC, using two configurations [9] of RED with results summarized in Tab. 3.4. It can be seen that TCP starvation persists under

9. Both RED and RED* parameters are set based on recommended settings proposed in [8]. RED* is a configuration inspired by [27, 59] and crafted ad hoc by a standard trial and error procedure. We stress once more that configuration details and scripts used for these experiments are available at [57].

Table 3.4 LEDBAT: Testbed vs. Internet Experiments

		5+5 flows		1+1 flows	
		TCP%	E[Q] ms	TCP%	E[Q] ms
Testbed	FIFO	94.9	6437.8	99.5	5304.3
	RED	49.3	11.3	1.7	9.5
	RED*	71.9	763.9	98.2	333.8
Internet	FIFO	97.0	6551.7	98.9	916.0
	RED	2.6	45.0	5.1	29.1
	RED*	63.8	745.2	86.7	305.4

RED when the number of flows is small: we stress that we observe TCP starvation only under the RED+LEDBAT combination, since as previously explained RED tries to penalize flows proportionally to the queue they create while LEDBAT is designed to precisely avoid queuing.

This unexpected phenomenon is worth pointing out, as it may add other reasons not to deploy RED other than those listed in [34]. At the same time, it is also worth highlighting that the phenomenon appears to be non-systematic and possibly arises from specific configuration and network environment (that are possibly hard to reproduce [60, 61]). Additionally, starvation is not observed under scheduling disciplines as SFQ, which may play a more important role than RED in the near future. It follows that the practical relevance of this starvation phenomenon is expected to be significantly smaller with respect to the reprioritization phenomenon.

Finally, we experiment with RED*, a configuration inspired by the fluid model in [59], which reinstates the relative CC priorities, but at the cost of an already sizeable bufferbloat (in the order of several hundred milliseconds, which makes it thus an impractical solution). The main difference between RED* vs. RED configurations is that RED* sets a larger min_{th} (16500B) compared to that in RED (1500B). As the queuing delay equivalent to min_{th} exceeds the default LEDBAT queuing delay target value (respectively, 264ms vs 100ms), this partly allows relative prioritization of end-to-end TCP vs. LEDBAT protocols. Intuitively, in this case our configuration starts dropping packets after the LEDBAT queuing delay target, so that TCP has the chance to grow its congestion window at the expense of LEDBAT (that by its LPCC nature will back off in the presence of best-effort TCP) before one of its packets get dropped by AQM; however, TCP recovery is in this case fast enough, and the additional buffer space beyond the LEDBAT target is large enough, to let TCP prevails over LEDBAT.

3.4 Summary

In this chapter, we studied the interaction between Active Queue Management (AQM) and Low-priority Congestion Control (LPCC). We considered a fairly large number of AQM techniques (i.e., RED, CHOKe, SFQ, DRR and CoDel) and LPCC protocols (i.e., LEDBAT, TCP-Nice, and TCP-LP), studying system performance (mainly expressed in terms of TCP% breakdown, average queue size $E[Q]$, and link utilization η) with both simulation and experimental methodologies.

Summarizing our main findings, we observe that *AQM resets the relative level of priority between best-effort TCP and LPCC*. That is to say, the TCP share of the bottleneck capacity drops dramatically, becoming close to the LPCC share. Additionally, while reprioritization generally equalizes the priority of LPCC and TCP, we also find that some AQM settings may actually lead best-effort TCP to starvation – as these are however non-systematic, we believe the starvation issue to be of less practical relevance than reprioritization.

Reprioritization is a fairly general phenomenon, as it holds for any combination of AQM technique, LPCC protocol and network scenario. This is testified by our thorough sensitivity analysis, where we confirmed the phenomenon for varying network parameters such as buffer size Q_{max}, link capacity C, heterogeneous RTT delay, flow numbers and flow duty cycle for over 3,000 simulations.

Reprioritization is a real world phenomenon, easily replicable on testbed and the Internet, which cannot be easily solved via AQM tuning. Hence, we advocate that explicit collaboration is needed from LPCC (e.g., openly advertise low priority) to assist AQM in taking decisions that maintain the desired level of priority.

Chapter 4

Control theoretic analysis

In the previous chapter, we show that "reprioritization" phenomenon is general and can arise from the interaction of any scheduling/AQM discipline and LPCC protocol shown in Fig. 2.2, using a twofold methodology including `ns2` simulation and experiments from both controlled testbed and wild Internet. In this chapter, we further investigate this phenomenon differently in both its depth and methodology: indeed, we adopt a narrower but profound scope, selecting LEDBAT and RED as representative examples of the LPCC and scheduling/AQM design space that we then analytically model. As our main innovation is not on the technique per se, but on its application to the study of a particular problem, we resort to classic models for TCP [49] and RED [47], that we extend to incorporate novel popular protocols such as LEDBAT.

First, to convince the reasonability of our choice of one AQM+LPCC combination to model, we depict in Fig. 4.1 an original representation of simulations performed in [62, 63]: results are arranged as a bubble chart, where the center of the bubble represents the average (TCP share%, queue occupancy) for a specific AQM+LPCC pair, and the radius of the bubble represents the standard deviation over the different network parameters considered. It can be seen that, while under DropTail, TCP% monopolizes the bottleneck causing bufferbloat ("bufferbloat" zone in the y-axis), any combination of AQM+LPCC protocol results in a reduction of the queue length below the threshold considered to be harmful to interactive communications (100ms or 33 packets for the capacity considered in this example), but also on a dramatic low TCP share ("reprioritization" zone in the x-axis). We find no combination in the "ideal" zone, which roughly represents a short queue size with acceptable delay and a high TCP% (according to LPCC's design goal, a 100% TCP% is ideal). It is worth stressing that whereas the actual values TCP share or queue size vary *quantitatively* across combinations, reprioritization is a *qualitatively* general phenomenon. It is thus reasonable

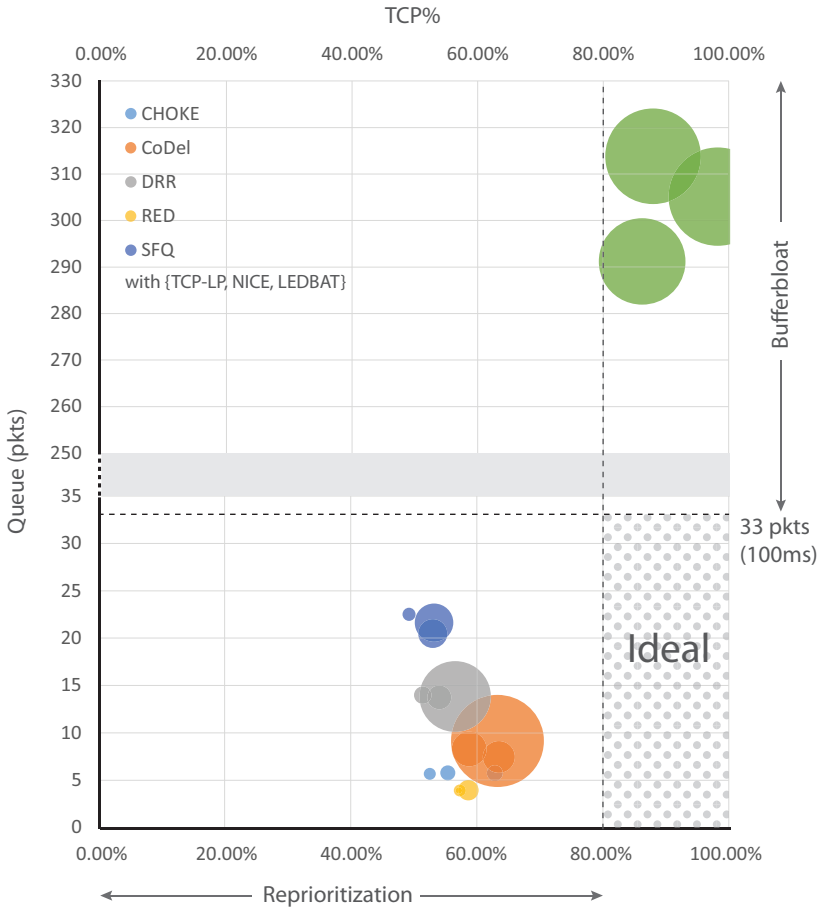


Figure 4.1 Interaction of scheduling/AQM disciplines and LPCC protocols.

to develop a model for one such AQM+LPCC combination, to give both solid theoretic explanation of the observed phenomenon, and intuition and insights behind its root causes.

Second, we stress that our objective is far from being a sterile academic exercise, as mixtures of AQM+LPCC are already commonplace in the current Internet landscape. On the one hand, it has to be observed that LEDBAT is used by BitTorrent, the most prominent P2P applications: whereas downlink traffic is dominated by Video streaming applications and portals (e.g., Netflix and YouTube), BitTorrent is the top-1 application on uplink traffic, and can be credited for over one-third of all upload traffic in North America, Latin America, and Asia Pacific, as the latest report [15] from the Canadian broadband management company Sandvine testifies. As claimed by Brahm Cohen, "LEDBAT is now the bulk of all BitTorrent traffic, [...] most consumer ISPs have seen the majority of their upload traffic switching to a UDP-based protocol" [22], it is also confirmed by our own independent measurement in customer ISPs [21]. On the other hand, it has also to be stressed that AQM adoption

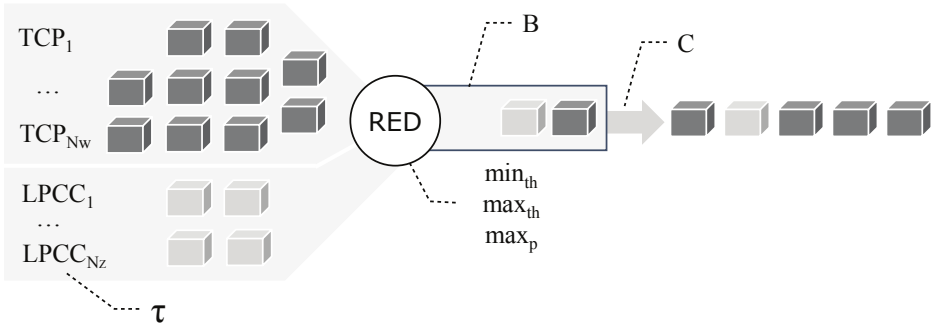


Figure 4.2 Network scenario.

worldwide has started to change, with operators implementing scheduling policies in the upstream of the ADSL modem to improve the quality of user experience. For instance, in France, Free implements Stochastic Fair Queuing (SFQ) since 2005 [9] and Orange has started the deployment of Shortest Queue First (SQF) in 2010 [28]). Similarly, US follow suit with new promising AQM techniques, such as CoDel [4], under active development in DOCSIS modems [10].

Based on the above observations, we argue the RED+LEDBAT combination to be a reasonable modeling target. Indeed, though there is no agreement on a single specific AQM technique, due to the similar behavior of any AQM+LPCC combinations in Fig. 4.1, results will be still qualitatively relevant across combination. To simplify the analysis, we therefore decide to propose a new model for LEDBAT (which is by far the most used LPCC protocol in the Internet nowadays) and resort to known models of RED (which is by far the most popular AQM technique in the literature).

This chapter is organized as following: in Sec. 4.1 we provide closed-form solution for the equilibrium in open-loop and in Sec. 4.2 we provide a stability analysis and a characterization of the reprioritization phenomenon when closing the loop with RED. The validation between the model and corresponding simulation results are reported in Sec. 4.3, followed by a simple yet practical system-level solution proposed in Sec. 4.4. We finally conclude the control theoretic analysis in Sec. 4.5.

4.1 Open-loop model

The goal of this section is to provide an open-loop model of the system composed of the LEDBAT and TCP sources that access a bottleneck whose queue is governed by a generic AQM controller.

As a reminder, the network scenario analyzed in this chapter is shown in Fig. 4.2. N_W TCP long-lived flows and N_Z LEDBAT flows share a bottleneck with a fixed capacity C, whose access queue is of size B and implements a generic AQM scheme. In Sec. 4.1.1 we derive a mathematical model of the *open-loop* system, meaning that we consider the packet dropping probability $p(t)$ set by the AQM controller as given, i.e. it is assumed to be an input parameter of the model. This assumption is not fully realistic, since, in practice, packet loss probability $p(t)$ arises from queuing dynamics, which in turn are induced by the behavior of the sources. Nevertheless, the analysis of this simplified system allows us to gather important preliminary information on the behavior of the system. In particular, our study of the open-loop system focuses mainly on its equilibrium points and the analysis of their properties (see Sec. 4.1.2). Furthermore, we complement our analysis in section Sec. 4.1.3 by providing a clear physical interpretation of our findings. Finally, in Sec. 4.2 we will close the loop considering the effect of queuing dynamics on the loss probability for the specific case of RED.

4.1.1 The mathematical model

We denote the average TCP and LEDBAT window at time t as $W(t)$ and $Z(t)$ respectively. For TCP, we neglect the slow-start phase, which is instead only optional in LEDBAT. As such, we limitedly model the TCP congestion window dynamics in the congestion avoidance phase. At the reception of the $(n+1)$-th ack at time t_{n+1}, the TCP congestion window is updated as follows:

$$W(t_{n+1}) = \begin{cases} \frac{1}{2}W(t_n) & \text{on packet loss,} \\ W(t_n) + \frac{1}{W(t_n)} & \text{otherwise} \end{cases} \tag{4.1}$$

Similarly to TCP, LEDBAT reacts to losses by halving the congestion window, but it increases its congestion window at a rate that is proportional to the distance of the queuing delay $q(t)$ from the delay target τ and is always bounded by the TCP ramp-up in congestion avoidance:

$$Z(t_{n+1}) = \begin{cases} \frac{1}{2}Z(t_n) & \text{on packet loss,} \\ Z(t_n) + \frac{1}{Z(t_n)}\frac{\tau - q_d(t_n)}{\tau} & \text{otherwise} \end{cases} \tag{4.2}$$

with the current queuing delay $q_d(t)$ measured as:

$$q_d(t_n) = D(t_n) - D_{min} \tag{4.3}$$

$$D_{min} = \min_n D(t_n) \tag{4.4}$$

where $D(n)$ represents the instantaneous one-way delay (OWD) estimate, while the base delay D_{min} is the minimum observed OWD. The rationale is that, over a sufficiently large number of observations D_{min} accurately represents the fixed component of the delay (i.e., propagation delay plus negligible transmission delay, which should be the one found when queues are empty) so that the $D(t_n) - D_{min}$ difference represents the variable component of the delay (i.e., queuing delay plus negligible processing delay).

Notice that host synchronization over the Internet is known to be hard. As such, it is worth stressing that the OWD estimate $D(t_n)$ is affected by an unknown clock offset between the two endpoints, and is thus of no practical use. Conversely, the offset cancels in the difference operation in (4.3), which is only affected by clock drift – that is of much smaller magnitude and furthermore easier to correct [64].

From (4.2), we gather that ramp-up is as fast as TCP only whenever the queue is empty (4.3), i.e., $\lim_{q_d \to 0} \frac{\tau - q_d}{\tau} = 1$. Furthermore, whenever the queuing delay hits the target τ, the congestion window settles since – when $q_d = \tau$ holds – it results $Z(t_{n+1}) = Z(t_n)$ for all n.

To analyze the interactions between sources and queue dynamics, we adopt a fluid flow modeling approach [47–50] in which the average dynamics of both sources and queues are described by nonlinear time-delay differential equations. In the following, we denote by $W_i(t)$ the instantaneous congestion window at time t for connection i in the fluid system, by $R_i(t)$ the Round Trip Time (RTT) and by $p(t)$ the packet dropping probability set by the AQM algorithm. We consider the case of N_W TCP and N_Z LEDBAT connections sharing the same bottleneck, where flow-level congestion window evolution the TCP case is adopted from [47]:

$$\frac{dW_i(t)}{dt} = \frac{1}{R_i(t)} - \frac{W_i(t)W(t - R_i(t))}{2R_i(t - R_i(t))} p(t - R_i(t)) \tag{4.5}$$

$$\frac{dZ_i(t)}{dt} = \frac{\tau - q(t - R_i(t))/C}{\tau} \frac{1}{R_i(t)} - \frac{Z_i(t)Z(t - R_i(t))}{2R_i(t - R_i(t))} p(t - R_i(t)) \tag{4.6}$$

$$\frac{dq(t)}{dt} = \sum_{i=1}^{N_W} \frac{W_i(t)}{R_i(t)} + \sum_{i=1}^{N_Z} \frac{Z_i(t)}{R_i(t)} - C\mathbb{1}_{q(t)\geq 0} \tag{4.7}$$

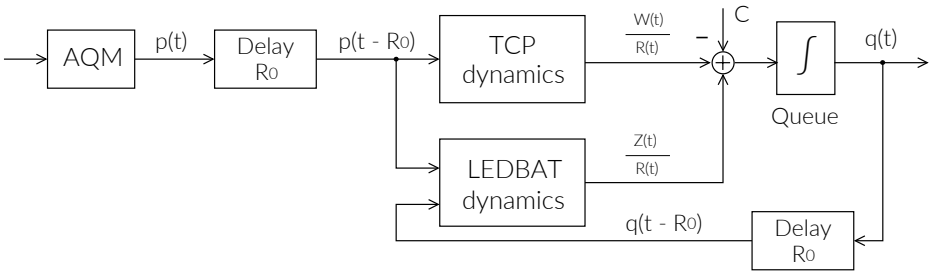


Figure 4.3 Block diagram of the proposed model.

where the queuing delay is modeled as $q_d(t) = q(t)/C$ as suggested in [48]. The RTT $R_i(t)$ is the superposition of two components: one constant component, i.e. the propagation delay T_i, and one time-varying, i.e. the queuing delay $q_d(t)$, thus giving $R_i(t) = T_i + q(t)/C$. For this reason, the RTT cannot be considered to be constant since the component due to queuing delay can be predominant over the propagation delay – which is especially true in the case of bufferbloat due to FIFO buffering. Conversely, in case AQM is used to avoid bufferbloat, it could be reasonable to assume the reverse $q(t)/C \ll T_i$ to hold.

We conclude this section by analyzing Fig. 4.3 that provides a representation of the dynamical system (4.5)-(4.6)-(4.7) in the form of a block diagram. The figure clearly shows that two control components interact to form the overall control system: (i) end-to-end control algorithms, namely TCP and LEDBAT, and (ii) the AQM controller.

The AQM control component sets a packet drop probability $p(t)$ that is first delayed and then fed in parallel to the TCP and LEDBAT blocks. Thus, we can argue that this control action acts equivalently on both the dynamics of the TCP and LEDBAT flows.

Concerning the end-to-end control algorithms, it is easy to check that the only difference stems from the fact that LEDBAT dynamics is explicitly affected by the instantaneous queue length. With this regard it is interesting to notice that, from a control-theoretic point of view, the term $-q(t - R_i(t))/(\tau C)$ in (4.6) plays the role of a "stabilizing" control signal since its sign is always negative or zero ($q(t) \geq 0$). Moreover, it is intuitive to anticipate that the value of the constant positive gain $1/(\tau C)$ plays an important role: the larger this constant, the larger will be the effect of the control signal $-q(t - R_i(t))/(\tau C)$ on the dynamics of the LEDBAT flow. The next sections will give a theoretical ground to this statement and show some important properties of the system.

4.1.2 Equilibrium and properties

In this section, we compute the equilibrium of the proposed mathematical model, which allows us to derive some fundamental properties of the system. For the sake of simplicity, we consider a bottleneck shared by an equal number of TCP and LEDBAT flows, i.e. $N_W = N_Z = N$. Moreover, we consider a homogeneous case, in which all the connections exhibit the same RTT (i.e., for both LEDBAT and TCP sources $T_i = T\ \forall i$). It is also worth noticing that the results presented below are valid regardless of the AQM control algorithm.

Proposition 1. *The unique equilibrium (w,z,q) of (4.5)-(4.6)-(4.7) when $N = N_W = N_Z$, in the case of a generic AQM algorithm setting a steady-state packet drop probability $p \in [0,1]$, is given by:*

$$w = \sqrt{\frac{2}{p}} \tag{4.8}$$

$$z = \begin{cases} w & \sqrt{2} < w \leq \frac{C}{2N}T \\ \frac{w}{2}\left[\sqrt{\left(\frac{Nw}{\tau C}-2\right)^2+4\frac{T}{\tau}}-\frac{Nw}{\tau C}\right] & \frac{C}{2N}T < w \leq (\tau+T)\frac{C}{N} \\ 0 & w > (\tau+T)\frac{C}{N} \end{cases} \tag{4.9}$$

$$q = \begin{cases} 0 & \sqrt{2} < w \leq \frac{C}{2N}T \\ \tau C\left[1-\frac{1}{4}\left(\sqrt{\left(\frac{Nw}{\tau C}-2\right)^2+4\frac{T}{\tau}}-\frac{Nw}{\tau C}\right)^2\right] & \frac{C}{2N}T < w \leq (\tau+T)\frac{C}{N} \\ Nw - TC & w > (\tau+T)\frac{C}{N} \end{cases} \tag{4.10}$$

Proof. We denote with (w,z,q) the equilibrium components of the steady state. By imposing $dW/dt = 0$ in (4.5) we readily obtain:

$$w = \sqrt{\frac{2}{p}}.$$

It has to be noted that since $p \in [0,1]$ the steady-state value of the TCP window is lower bounded by $\sqrt{2}$. By imposing $dZ/dt = 0$ in (4.6) we get:

$$\frac{1}{\tau R}(\tau - q/C) - \frac{z^2}{2R}p = 0,$$

which gives:

$$z = \begin{cases} w\sqrt{1-\frac{q}{\tau C}} & 0 \leq q \leq \tau C \\ 0 & q > \tau C \end{cases} \tag{4.11}$$

Let us begin by focusing on the case $0 \leq q \leq \tau c$. By letting $\dot{q} = 0$ in (4.10) we obtain:

$$Nw + Nw\sqrt{1-\frac{q}{\tau C}} = TC + q \tag{4.12}$$

We now make the change of variables:

$$x(q) = \sqrt{1-\frac{q}{\tau C}} \tag{4.13}$$

that is a monotonically decreasing mapping from the set $[0, \tau C]$ to the set $[0,1]$. By plugging (4.13) into (4.12) we get:

$$x^2 + \frac{Nw}{\tau C}x + \frac{Nw}{\tau C} - \frac{T}{\tau} - 1 = 0 \tag{4.14}$$

By solving (4.14) we get a unique positive solution:

$$x = \frac{1}{2}\left(\sqrt{\left(\frac{Nw}{\tau C} - 2\right)^2 + 4\frac{T}{\tau}} - \frac{Nw}{\tau C}\right) \tag{4.15}$$

The solution (4.15) is meaningful only if it belongs to the domain $[0,1]$. Let us start by looking at the upper bound of the solution by plugging $x = 1$ into (4.15), which corresponds to the case $q = 0$:

$$\sqrt{\left(\frac{Nw}{\tau C} - 2\right)^2 + 4\frac{T}{\tau}} - \frac{Nw}{\tau C} = 2. \tag{4.16}$$

After a little algebra we obtain:

$$w_{min} = \frac{TC}{2N} \tag{4.17}$$

To get a physical interpretation of w_{min} we have to consider that for this value of w the queue is zero and $z(w_{min}) = w$, i.e. the N TCP and N LEDBAT flow obtain exactly the same per flow rate equal to w_{min}/T. It has also be noticed that the sum of the rates of the LEDBAT flows and TCP flows is exactly equal to the capacity of the link:

$$N\frac{w_{min}}{T + \frac{q}{C}} + N\frac{w_{min}}{T + \frac{q}{C}} = \frac{2N}{T}w_{min} = C \tag{4.18}$$

Let us now consider the other extreme case $x = 0$ corresponding to $q = \tau c$. In this case by substituting $x = 0$ in (4.15) we get:

$$w_{max} = \frac{C}{N}(T + \tau) \tag{4.19}$$

Thus, we conclude that the interval $0 < q < \tau C$ maps to the interval $TC/(2N) < w < C(T+\tau)/N$. By considering (4.15) and (4.13) we obtain the equilibrium for the queue when $TC/(2N) < w < C(T+\tau)/N$:

$$q = \tau C \left[1 - \frac{1}{4} \left(\sqrt{\left(\frac{Nw}{\tau C} - 2 \right)^2 + 4\frac{T}{\tau}} - \frac{Nw}{\tau C} \right)^2 \right] \tag{4.20}$$

It is important to notice that $z = 0$ when $w = w_{max}$, meaning that in this case the whole link capacity is allocated to the N TCP flows. Finally, by plugging (4.20) in (4.11) we get the LEDBAT window size at steady-state when $TC/(2N) < w < C(T+\tau)/N$:

$$z = \frac{w}{2} \left[\sqrt{\left(\frac{Nw}{\tau C} - 2 \right)^2 + 4\frac{T}{\tau}} - \frac{Nw}{\tau C} \right]. \tag{4.21}$$

Let us now consider the case $q \geq \tau C$ that occurs when $w \geq C(T+\tau)/N$. In this case $z = 0$ and, as a consequence, $q = Nw - TC > 0$.

To conclude the proof we have to consider the case $w < w_{min} = TC/(2N)$. It is easy to check that $q = 0$ and thus by plugging this value in (4.11) we obtain $z = w$. □

Remark 1. *At steady state w, z, and q vary as a function of p according to the three different operating zones* [1] *that are shown in Fig. 4.4:*

1. No reprioritization $0 \leq p < 2N^2/(C^2(\tau+T)^2)$*: in this zone* $z = 0$ *and thus the dynamics of N TCP flows accessing a bottleneck described in [48] is recovered.*
2. Reprioritization $2N^2/(C^2(\tau+T)^2) \leq p < 8N^2/(CT)^2$*: in this zone* $0 < z < w$ *and* $q > 0$*, meaning that reprioritization is occurring since the LEDBAT flows are getting a non-negligible bandwidth share;*
3. Underutilization $8N^2/(CT)^2 \leq p \leq 1$*: in this zone* $w = z = \sqrt{2/p}$ *and* $q = 0$ *and in this case the link is not fully utilized since* $\sum w/R + \sum z/R = 2Nw/R < C$*;*

Proposition 2. *A sufficient condition to avoid reprioritization in the case an equal number N of TCP and LEDBAT flows access a bottleneck link of capacity C, round-trip propagation*

1. In the following we consider that $CT/(2N) > \sqrt{2}$ otherwise the first zone would collapse to a point.

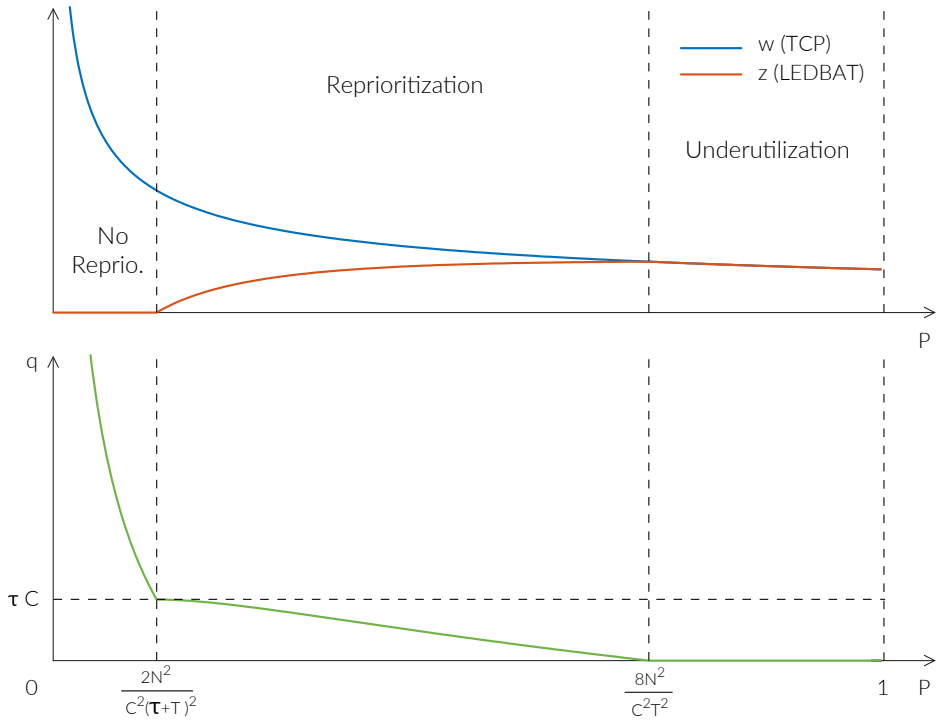


Figure 4.4 LEDBAT window size z, TCP window size w, and queue size q at the equilibrium as a function of $p \in [0,1]$.

delay T, with a queue governed by a generic AQM algorithm is that:

$$\frac{C}{N}\frac{T+\tau}{\sqrt{2}} < 1. \tag{4.22}$$

Proof. To prove this proposition we need to show that for any $p \in [0,1]$ set by a generic AQM algorithm the LEDBAT congestion window z is zero at the equilibrium. From Remark 1 we have that $z = 0$, i.e. no reprioritization occurs, when the following condition holds:

$$0 < p < \frac{2N^2}{(\tau+T)^2C^2}. \tag{4.23}$$

To conclude the proof we observe that when (4.22) holds the right extreme $2N^2/((\tau+T)^2C^2)$ of (4.23) is always greater than 1, meaning that the only possible equilibrium is $z = 0$ regardless of the loss probability p set by the employed AQM control algorithm. □

Remark 2. *Since (4.22) is a sufficient condition, when (4.22) does not hold, reprioritization could be still avoided by a particular AQM control law that sets a steady-state p such that (4.23) holds.*

4.1.3 Discussion

In this section, we analyze the tension that exists between the two main components of the overall system: (i) *end-to-end* control algorithms, implemented at the transport or application layer in the hosts; (ii) *AQM* controllers which are executed in the routers, i.e. in the network. Ideally one could aim at independently tuning the two control algorithms while obtaining the following two properties at steady-state: (i) the LPCC flows only get a negligible share of the bottleneck when competing with TCP flows (reprioritization is avoided); (ii) the queuing delays are minimized (bufferbloat is avoided).

Thus, in this section we investigate whether, by only tuning the LEDBAT delay target τ, it is possible to satisfy the two properties mentioned above. Towards this end, we consider Proposition 2 which provides a sufficient condition to avoid reprioritization. Thus, if we can show that condition (4.22) is general enough in practical scenarios and when it holds, is able to also avoid bufferbloat, we would have satisfied both the properties by only tuning τ.

With this purpose in mind we rewrite (4.22) as follows:

$$0 < \tau < \sqrt{2}\frac{N}{C} - T \tag{4.24}$$

and we observe that if the inequality (4.24) is verified for some positive τ then reprioritization is avoided regardless of the specific AQM employed. Of course, satisfying (4.24) would

need the apriori knowledge, or a worst case estimate, of the number of concurrent TCP and LEDBAT flows N, the capacity of the bottleneck C, and the round-trip propagation delay of connections T. By considering a typical example, we show that using (4.24) might be unfeasible because it could require setting a delay target τ that is either negative or too small.

To begin with, notice that for a typical ADSL whose upload data rate is 500Kbps, the transmission of a full MTU packet takes about 24 ms: initial versions of LEDBAT used to set τ = 25 ms, i.e., a packet worth of queuing. However, due to practical limitations (including timestamp precision in Windows OS, clock drift of several ppm in off-the-shelf PCs, etc.) this setting did not allow to fully exploit the link capacity, the reason why the target was later increased to τ =100 ms. Thus, even though in principle one should set τ as low as possible, for practical issues τ has to be set to a value that is lower bounded at least by τ_{min} =25 ms (for which we already know that underutilization of the bottleneck occurs).

Therefore, if we require now that $\tau > \tau_{min}$, for (4.24) to be feasible we have to satisfy the following condition:

$$N > \frac{C}{\sqrt{2}}(T + \tau_{min}). \tag{4.25}$$

Now, we turn our attention to the case in which an ADSL whose uplink data-rate is 1 Mbps, which corresponds to $C = 100$pkt/s for packets of fixed size $P = 1250$B, is shared by flows whose round-trip propagation delay T is 50ms. In this case (4.25) would require at least $N = N_Z = N_W = 6$, i.e. a total of 12 flows sharing the bottleneck, a condition which might not hold in general. From these arguments, we can say that it is in general not possible to tune τ independently from the AQM control law in order to both avoid reprioritization and bufferbloat.

We now take a different point of view and we analyze the formula of the TCP bandwidth share ρ at steady-state that is able to both quantitatively explain the interplay between AQM and LEDBAT algorithms and the reprioritization issue shown in Fig. 4.1. In particular we have:

$$\rho = \frac{w}{w+z} \tag{4.26}$$

By plugging the steady state w and z in the form of (4.11) we readily obtain:

$$\rho = \begin{cases} \frac{1}{1+\sqrt{1-\frac{q/C}{\tau}}} & 0 \leq q/C \leq \tau \\ 1 & q/C > \tau \end{cases} \tag{4.27}$$

It is straightforward to notice that $\rho \in [1/2, 1]$: $\rho = 1$ corresponds to the desirable case in which reprioritization is avoided and that is obtained when the steady state queuing delay q/C is greater than the LEDBAT target delay τ; $\rho = 1/2$ represents the non-desirable situation where LEDBAT and TCP get the same bandwidth share which is obtained when $q/C \to 0$. This discussion confirms what we have anticipated at the end of Sec. 4.1.1, i.e. that the control term $-q/(\tau C)$ that appears in the LEDBAT differential equation (4.6) is the main component driving the interplay between the two control algorithms acting on the system, i.e. AQM and end-to-end.

Interestingly, (4.27) is the only function of the two parameters used by the AQM and the LEDBAT algorithms. The main parameter that can be used to tune LEDBAT is τ, whereas AQM algorithms trying to avoid bufferbloat strive to make the queuing delay $q_d = q/C$ small.

While a 100 ms target may be reasonable for an end-to-end protocol, such as in the case of LEDBAT [14], an AQM may be more precise in measuring the queue size and in adopting more aggressive dropping policy. This is, in fact, the case for two recently proposed AQM algorithms, namely CoDel and PIE: CoDel employs a target sojourn time of only 5 ms, whereas the default queuing delay target used by PIE is 20 ms. It is then reasonable to assume that in practical scenarios $q/C < \tau$ holds and consequently $\rho < 1$. This explains why the ideal interplay of AQM and LPCC that avoids bufferbloat and reprioritization shown in Fig. 4.1 is unfeasible.

Another important observation can be drawn by analyzing the absolute sensitivity of ρ with respect to τ or $q_d = q/C$, i.e. $\partial\rho/\partial q_d$ and $\partial\rho/\partial\tau$, in the region $q_d < \tau$ where reprioritization occurs:

$$\frac{\partial\rho}{\partial q_d}(\tau, q_d) = \frac{1}{2\tau(1+\sqrt{1-q_d/\tau})^2\sqrt{1-q_d/\tau}} \tag{4.28}$$

$$\frac{\partial\rho}{\partial\tau}(\tau, q_d) = \frac{q_d/\tau}{2\tau(1+\sqrt{1-q_d/\tau})^2\sqrt{1-q_d/\tau}} \tag{4.29}$$

Both the sensitivity functions show an asymptote at $q_d = \tau$ meaning that a sharp transition phase occurs when moving from the non-reprioritization ($q_d > \tau$) to the reprioritization zone ($q_d < \tau$) – yet another reason to make it impractical to rely on a single parameter selection τ to drive the overall system performance.

4.2 Closed-loop model

In this section, we focus on the properties of the closed-loop system that is obtained when a particular AQM controller, namely RED, is used. As already mentioned, RED has been chosen as a representative AQM algorithm mainly due to its popularity in the literature. It is worth noting that, even though the *quantitative* results derived in the following are specific to RED, the *qualitative* behavior of the system obtained by closing the loop with another AQM algorithm would be similar (see Fig. 4.1).

We start our analysis by deriving the model of the closed-loop system. To the purpose we take the model described by (4.5)-(4.6)-(4.7) and we plug the RED control law that sets the packet dropping probability $p(t)$ according to a static function $f : \mathbb{R}_+ \to [0,1] \subset \mathbb{R}$ of the queue size $q(t)$ defined as follows:

$$f(q) = \begin{cases} 0 & q < q_{min} \\ \max_p \frac{q - q_{min}}{q_{max} - q_{min}} & q_{min} \leq q \leq q_{max} \\ 1 & q > q_{max} \end{cases} \tag{4.30}$$

where $\max_p \in [0,1]$ is the maximum dropping probability when $q \in [q_{min}, q_{max}]$. In Sec. 4.1 we have shown that reprioritization is driven by two parameters, i.e. the steady state queuing delay q/C and the LEDBAT target τ (see (4.27)). For this reason, we consider the system parameters space to be q_{min}, which relates to q/C, and τ. In particular, we analyze the closed-loop system and we explore the system parameters space to characterize: (i) the region of the system parameters where the reprioritization phenomenon is avoided, (ii) the stability region of the equilibrium.

4.2.1 Characterizing reprioritization

In order to characterize the reprioritization phenomenon in the parameters space, we have to compute the window size of the LEDBAT and the TCP flows at the equilibrium. Towards this end, we use Proposition 1 (Sec. 4.1) that characterizes the equilibrium of the open-loop system in closed form as a function of the packet drop probability p set by a generic AQM controller.

We start by observing that the cases $q < q_{min}$ and $q > q_{max}$ are trivial and can be treated as follows. In the case $q < q_{min}$ RED would set a packet dropping probability equal to 0 that would make $z = 0$, i.e. LEDBAT is starved, and $w \to \infty$. However, since the congestion window w is always limited by the advertised window set by the flow control, in practice we would have $w = w_{max}$. The other limit case occurs when $q > q_{max}$ and RED would compute a

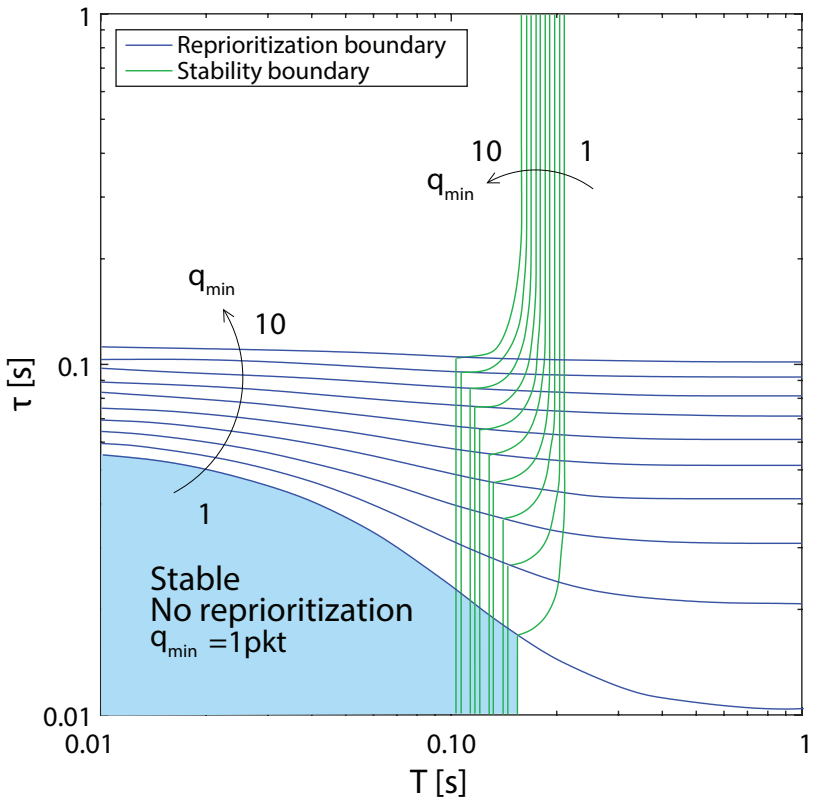


Figure 4.5 Reprioritization and stability regions in the case of $N = 1$, $C = 1$Mbps, $q_{max} = 100$ pkt.

packet dropping probability of 1, meaning that all packets are dropped, a situation of limited practical interest.

From the consideration made above, we limitedly focus on the case $q \in [q_{min}, q_{max}]$ where p is set proportional to the error $q - q_{min}$:

$$p(q) = k(q - q_{min}) \tag{4.31}$$

with the gain k defined as $k = \max_p / (q_{max} - q_{min})$. Thus, substitution of (4.31) in (4.10) yields the following equation has to be solved to get the steady-state queue length $q \in [q_{min}, q_{max}]$:

$$\sqrt{1 - \frac{q}{\tau C}} = \frac{TC + q}{N} \sqrt{\frac{k}{2}(q - q_{min})} - 1 \tag{4.32}$$

Then, in order to get p we substitute the steady state queue length q solution of (4.32) in (4.30). Finally, plugging p in (4.8) and (4.9) we get the steady state window size of the TCP and LEDBAT flows respectively.

In order to characterize the local stability of the equilibrium we linearize the system around the equilibrium (w, z, q) to get the following linear time-delay system:

$$\dot{x}(t) = A_0 x(t) + A_R x(t - R) \tag{4.33}$$

where the state of the linearized system is $x(t) = [W(t) - w, Z(t) - z, q(t) - q]^T$, $R = T + q/C$ and the matrices A_0, and A_R are defined as follows:

$$A_0 = \begin{bmatrix} -\frac{\sqrt{p}}{\sqrt{2}R} & 0 & -\frac{1}{R^2 C} \\ 0 & -\frac{\sqrt{p(1-\frac{q}{\tau C})}}{\sqrt{2}R} & -\frac{\tau + T}{\tau R^2 C} \\ \frac{N_W}{R} & \frac{N_Z}{R} & -\frac{N_W w + N_Z z}{R^2 C} \end{bmatrix}; A_R = \begin{bmatrix} -\frac{\sqrt{p}}{\sqrt{2}R} & 0 & \frac{1}{R^2 C} - \frac{k}{pR} \\ 0 & -\frac{\sqrt{p(1-\frac{q}{\tau C})}}{\sqrt{2}R} & \frac{\tau - q/C}{\tau R^2 C} - k\frac{1-\frac{q}{\tau C}}{pR} \\ 0 & 0 & 0 \end{bmatrix} \tag{4.34}$$

It is well-known that the equilibrium of a non-linear time-delay system is locally stable if all the eigenvalues of the characteristic equation associated to (4.33) are in the left half-plane [65]. Since the equilibrium (w, z, q) cannot be found in closed form, we resort to numerically find the rightmost eigenvalue of (4.33) by using the tools described in [66].

Fig. 4.5 shows both reprioritization and stability regions in the parameters space (T, τ) in the case of a bottleneck with a capacity $C = 100$ pkt/s, propagation round trip time $T = 50$ ms maximum queue length $q_{max} = 100$ pkt which corresponds to a maximum queuing delay of 1 second. It is worth noting that such values are commonplace today, with modem buffers able to hold up to 4 seconds worth of data [5]. We have considered $q_{min} \in \{1, 2, \ldots, 10\}$ pkt

to cover both AQM algorithms specifically designed to contain the queuing delay (CoDeL and PIE) and AQMs designed to contain queue length (f.i. RED, Choke [26]). Finally, for simplicity and without loss of generality, Fig. 4.5 shows the case of one LEDBAT with one concurrent TCP flow, i.e. $N = 1$.

Let us consider a pair (T^*, τ^*) and $q_{min} = q^*_{min}$: the equilibrium is stable if (T^*, τ^*) is at the left of stability boundary (dashed line) associated to q^*_{min}; similarly, it is simple to show that no reprioritization occurs, i.e. the LEDBAT bandwidth share at steady state is zero ($z = 0$) if (T^*, τ^*) is below the *reprioritization boundary* (solid line) associated to q^*_{min}. To give an example, Fig. 4.5 shows in gray the region where both stability of the equilibrium and reprioritization avoidance are obtained in the case of $q_{min} = 1$ pkt.

The figure shows that, as previously observed, the reprioritization phenomenon vanishes when $\tau < q/C$: in other words, when this condition holds all LEDBAT flows will by design yield to TCP and the system will behave as [47, 48]. At the same time, this scenario is unlikely to hold in practice. In fact, as previously observed, end-to-end congestion control protocols such as LEDBAT rely on noisy measures of queuing delay, so that they will not be able to guarantee protocol efficiency when $\tau \to 0$.

In the following, we shall characterize reprioritization using a more fine-grained approach as opposed the binary information that Fig. 4.5 conveys. In particular, we employ the TCP bandwidth share ρ given by (4.27) to measure the level of reprioritization, recalling that when ρ is close to 1 reprioritization is avoided.

We depict in Fig. 4.6 the TCP share ratio ρ at the equilibrium for varying user scenarios (i.e., number of TCP and LEDBAT flows N, LEDBAT target settings τ, and RED settings q_{min}).

Fig. 4.6 shows that under AQM the TCP share exhibits a sharp transition phase as soon as τ exceeds q_{min}, quickly dropping with a hyperbolic slope from a monopoly situation ($\rho \to 1$) to a fair share ($\rho^* \approx 0.58$ for $\tau = 2q$). Interestingly, [42] shows that in the DropTail case, which can essentially be recovered from ours by letting $q_{min} \to q_{max} = B$, a sharp transition phase from TCP monopoly to a fair share happens whenever $\tau \to B$. This difference is rooted on the fact that RED dropping rates are strictly positive as soon as the queue size exceeds q_{min}, whereas DropTail decisions have to wait until the queue exceeds B.

From Fig. 4.6 we also gather that different RED settings only minimally affect the reprioritization phenomenon. Trivially, since no dropping happens for $q < q_{min}$, this parameter plays the biggest role in determining the queue size at the equilibrium. Next comes the load factor, i.e., the number of flows insisting on the bottleneck.

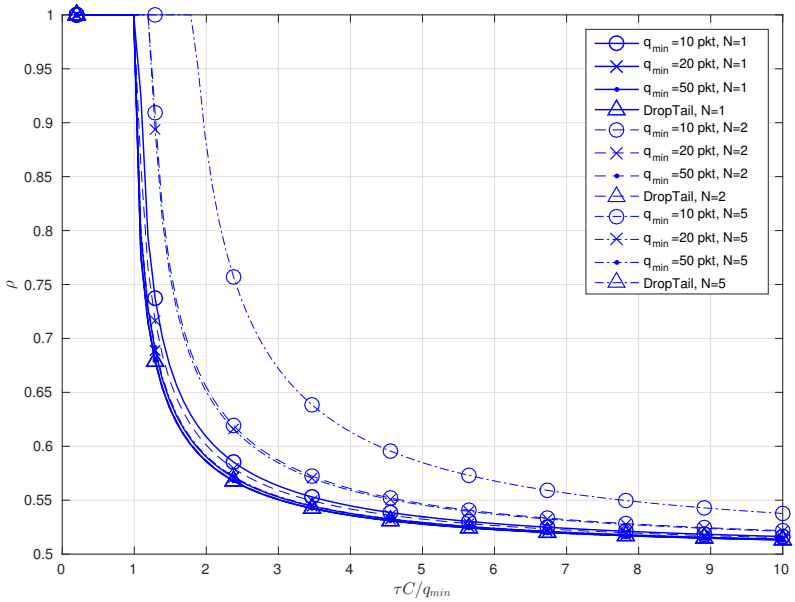


Figure 4.6 TCP share ratio ρ at the equilibrium as a function of $\tau C/q_{min}$ for various flow number N, LEDBAT τ, and RED q_{min} settings.

The impact of the LEDBAT target τ on the queue size has almost a step-like behavior, that can be explained taking into account the discussion about the absolute sensitivity of ρ with respect to τ (see (4.29)) or $q_d = q/C$ (see (4.28)) provided in Sec. 4.1.3.

4.2.2 Characterizing system dynamics

In this section, we explore the parameter space with the aim of characterizing the dynamical behavior of the system. To the purpose, let us consider again Fig. 4.5: the figure shows that the main parameter affecting the stability of the equilibrium is T, whereas the influence of q_{min} and τ is less important. Indeed, the adverse effect of time-delays in control loops is well-known and it affects a wide class of dynamical systems [65]. In particular, many time-delay systems display Hopf bifurcations when the time-delay T is varied: the linearized system is stable, i.e. all the eigenvalues are in the left-half plane, until T is equal to a critical value T_c for which a couple of purely imaginary eigenvalues appear; then, for $T > T_c$ the real-part of those eigenvalues increases. When a Hopf bifurcation takes place, a limit cycle, i.e. an isolated periodic orbit, branches from the fixed point and a self-sustained oscillatory dynamics is established for $T \geq T_c$ [67].

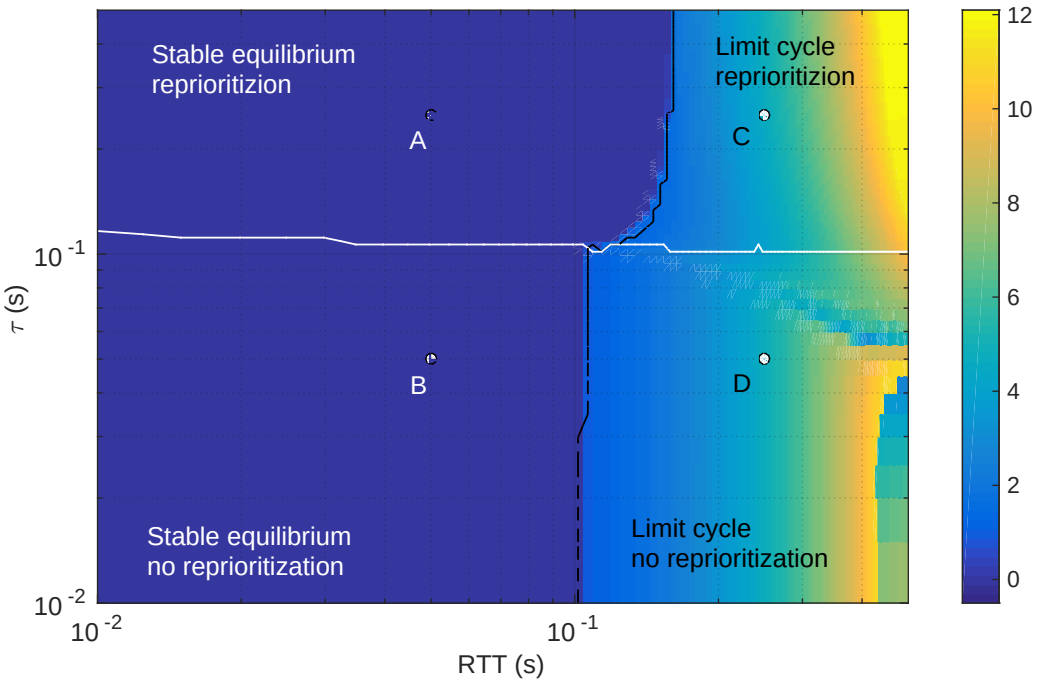


Figure 4.7 Bifurcation analysis. $q_{min} = 10$ pkt, $q_{max} = 100$ pkt, $C = 1$Mbps, $N = 1$

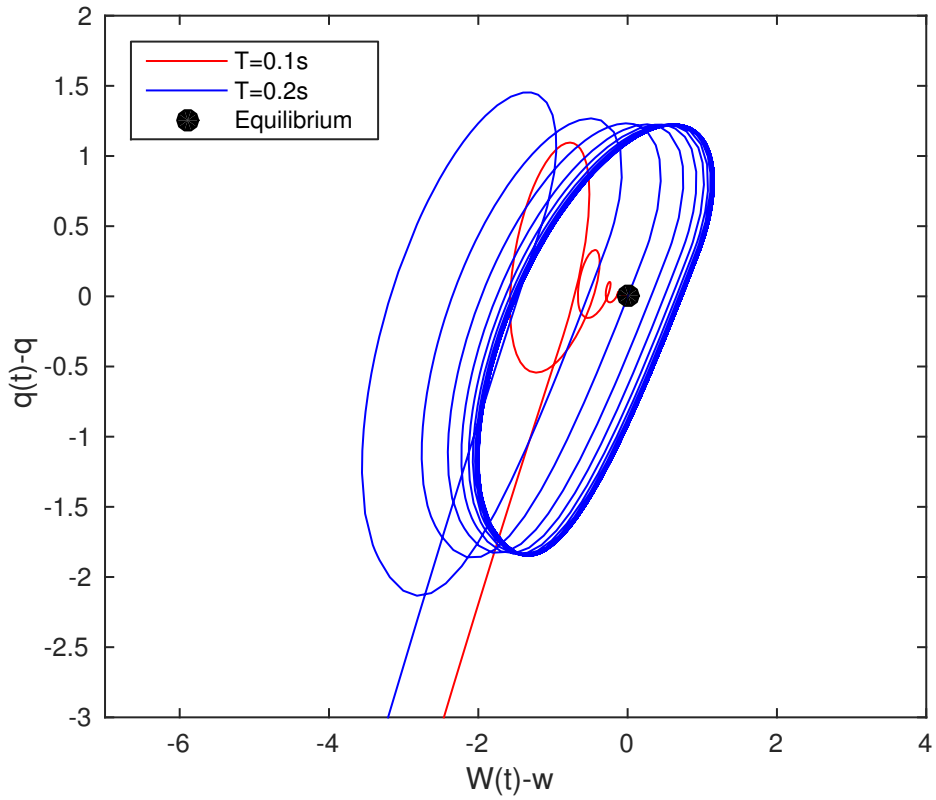


Figure 4.8 Phase plots showing a stationary dynamics ($T = 0.1$ s) and a periodic dynamics ($T = 0.2$ s). $\tau = 0.2$ s, $q_{min} = 10$ pkt, $q_{max} = 100$ pkt, $C = 1$Mbps, $N = 1$

Even though it is not the main focus of this work, in the following we shall characterize the properties of the dynamics of the system when T and τ are varied. In particular, we will show that, when the local stability of the equilibrium is lost, i.e. when passing from the stable to the unstable region shown in Fig. 4.5, a Hopf bifurcation occurs and the behavior of the solutions at steady-state will change from being stationary to being periodic. With this purpose, we make T and τ vary and we numerically solve the time-delay differential equations to measure the amplitude of the oscillations as a function of the two parameters. Fig. 4.7 shows a contour plot of the oscillation amplitude function of T and τ: as expected, in the stable region – the one at the left of the stability boundary show in black – the amplitude of the oscillations is zero, meaning that a stationary behavior is obtained; as soon as we leave the stable region self-sustained oscillations appear whose amplitude gets larger and larger as we move away from the stable region. Fig. 4.8 shows the phase plots in the case of $T = 0.1$ s and $T = 0.2$ s when τ is fixed to 0.2 s: in the case of $T = 0.1$ s, since we are in the stable region, the equilibrium is reached asymptotically; on the other hand when $T = 0.2$ s, the trajectory describes a closed orbit and it does not converge to the equilibrium.

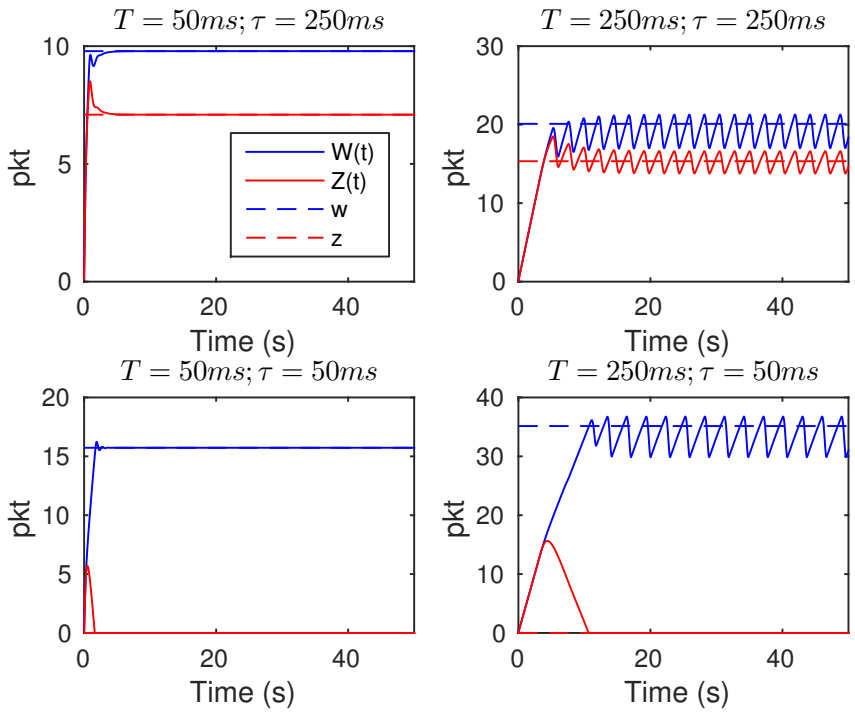


Figure 4.9 LEDBAT and TCP dynamics window comparison in the four regions ($q_{min} = 10$pkt, $q_{max} = 100$pkt, $C = 1$Mbps, $N = 1$)

Let us now consider Fig. 4.9 that shows the dynamics of the system in each of the four regions shown in Fig. 4.5.

The two figures at the left are representative of the region where the system is stable and exhibits a stable response: the figure at the top-left corner shows the case in which reprioritization is obtained and the LEDBAT flow gets a non-negligible share of the bottleneck bandwidth; on the other hand, the figure at the bottom-left corner shows the desirable case where the LEDBAT flow is starved by the TCP flow. The two figures at the right of Fig. 4.9 show the corresponding behavior in the case the propagation delay T is larger than the critical value and, as expected considering the above analysis, a periodic response is obtained.

We conclude this section by showing the time-evolution of the system equations when $N_W = N_Z = 1$ in Fig. 4.10 under either DropTail (a) or RED (b,c,d) disciplines. Top plot shows the LEDBAT and TCP window congestion windows evolution, while queue $q(t)$ is reported in bottom plots.

In the DropTail case, we set $\tau = 0.1$ s and observe the same behavior shown via `ns2` simulation in [23]: i.e., LEDBAT yields to TCP as expected under DropTail. In the RED case, we set $q_{max} = B = 100, q_{min} = 10$ for the sake of illustration and let τ grow from 0.1 s (b) to

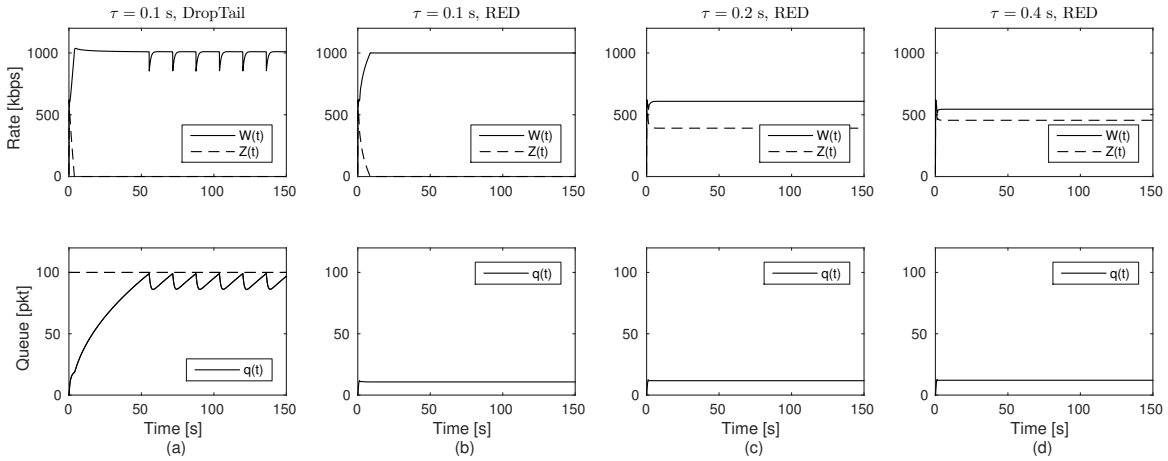


Figure 4.10 Reprioritization phenomenon: Time evolution of $W(t)$, $Z(t)$ and $Q(t)$ under DropTail (a) and RED (b,c,d) for different values of τ.

0.2 s (c) and 0.4 s (d). Notice that in case (b), RED drastically reduces the queue size and let the TCP rate to match the capacity after a short transient. Yet, when the target τ increases in (c) and (d), LEDBAT becomes increasingly aggressive under RED, and competes more fairly against TCP.

To avoid cluttering the pictures, we do not report here the behavior of LEDBAT for increasing target τ under DropTail: from the simulation-based sensitivity analysis reported in [42], it emerges that LEDBAT yields to TCP for a large range of $\tau < B/C$ values, and only whenever τ approaches (or exceeds) B/C LEDBAT behavior becomes loss-based as TCP.

4.3 Validation

In this section, we validate a subset of the numerical results obtained by integrating the time-delay differential equations modeling the system against those obtained from `ns2` simulator using our own implementation of LEDBAT, which is available as open source [2]. We additionally point out that we released all scripts and scenarios [3] to reproduce the simulation (as well as the experimental) results of [63]. In what follows, validation is performed on the most challenging (in terms of matching the simulation vs. fluid model results, to get a conservative estimate of model accuracy) and relevant scenarios (in terms of practical relevance).

2. http://www.enst.fr/~drossi/ledbat
3. http://www.enst.fr/~drossi/dataset/ledbat+aqm

4.3.1 Scenario

We argue that the most challenging scenario, in terms of matching results gathered via simulation and fluid model, is the one with few numbers of flows. This is intuitive since in the case of multiple backlogged connections, statistical multiplexing will smooth out the impact of events, such as TCP retransmission timeout, that would otherwise cause discontinuities in the case of few connections. At the same time, we also argue that the most practically relevant scenario is precisely one with a relatively small number of flows. Indeed, since the bottleneck is the user access, the number of concurrent connections will be likely small, even considering multiple applications/users in the household.

We consider both the Cloud and the P2P cases. In the Cloud case, it is easy to see that a small number of connections will be opened, at any given time, for a specific service. While considering a single user, even the server contacted will evolve over time (e.g., due to load balancing), this likely happens over time-scales that are much larger with respect to the short time-frames that we consider as "backlogged" data transfers (i.e., from tens of seconds to minutes) in this thesis. Hence, the number of backlogged connections is upper-bounded by the number of Cloud services the user subscribes to, such as DropBox for data, GoogleMusic for music and Picasa for pictures/videos. Additionally, the number of simultaneous connections also depends on the on/off synchronization pattern toward the Cloud. As users are not continuously generating all kind of data at the same time, it thus reasonable to envision only a moderate number of concurrent backlogged connections per household, some of which may be lower priorities (e.g., pictures) over others (e.g., critical data, backup).

Consider next the P2P case, where it makes sense to consider file-sharing applications such as BitTorrent due to its popularity, and since it introduced LEDBAT in the first place precisely due to the bufferbloat problem. In BitTorrent, pipelining of piece requests at the application level can cause multiple chunks to be transmitted consecutively over the same connection at transport-level. Since BitTorrent limits the number of concurrent slots to about [4] 4 per torrent, the number of concurrent connections will be again small. Moreover, BitTorrent peers periodically evaluate the throughput toward other peers every tens of seconds, and connections are maintained in case of good end-to-end throughput: coupled to pipelining, this entails that over the tens of seconds to minute timescale, connections can be considered backlogged.

4. The limit actually increases with the square root of the uplink capacity

4.3.2 Model validation against `ns2` simulations

From the above discussion, in the following we will limitedly consider an equal number $N = N_W = N_Z$ of flows, and let the total number of flows vary in $2N \in [2,10]$ range. Unless otherwise stated, we consider homogeneous RTT delay settings with propagation delay $T = 50$ ms (to which we add a jittering component of 1 ms to avoid synchronization of the congestion window dynamics). To precisely characterize system equilibrium properties, we will let the LEDBAT target τ vary – that in the uTorrent implementation of LEDBAT, this can be easily done by adjusting the `net.utp_target_delay` settings.

Without loss of generality, we consider a single access bottleneck and fix the capacity to C=1Mbps, a typical range for ADSL/Cable access. The bottleneck buffer can accommodate up to $B = q_{max} = 100$ packets that, considering P=1250 Bytes sized packets for simplicity, corresponds to a maximum queuing delay of 1000 ms.

We point out that our goal is not to provide an exhaustive coverage of all scenarios considered in [63] (as scripts to reproduce experiments and simulations are available as pointed out before). Rather, our main aim here is to validate the most representative instance of our results – which is clearly represented by the TCP share ratio ρ that precisely quantifies the reprioritization.

As we have previously seen, q_{min} and τ have by far the biggest role in determining the TCP share curve, followed by the number N of flows in the bottleneck. Hence, in Fig. 4.11 we compare the TCP share ratio $\rho = w/(z+w)$ as a function of $\tau C \in [5,100]$pkt and the RED minimum threshold $q_{mim} \in [5,40]$pkt obtained either numerically (top figures) or via `ns2` simulations (bottom figures) in the case of $N = 1$ (left) or $N = 2$ (right). The figure shows that the proposed model is able to qualitatively capture the reprioritization phenomenon that is observed with `ns2` simulations (and experiments) across the considered parameter space $(\tau C, q_{min})$. In particular, by fixing $q_{min} = q^*_{min}$ and moving from the left to the right on the line parallel to the x-axis connecting $(0, q^*_{min})$ to $(\tau_{max} C, q^*_{min})$ an abrupt decrease of the TCP bandwidth share ρ occurs for any q^*_{min}. However, Fig. 4.11 also shows that the model overestimates the reprioritization: in fact, `ns2` simulations provide a zone where $\rho \simeq 1$ (no reprioritization) that is larger than the one obtained by using the model. We shall return shortly on this matter and propose a refinement of the model to address this issue. Finally, Fig. 4.11 shows that the effect of increasing the number N of concurrent flows on reprioritization is qualitatively the same, i.e. with an increased number of flows, reprioritization is mitigated. This result also confirms the theoretical findings conveyed by Proposition 2.

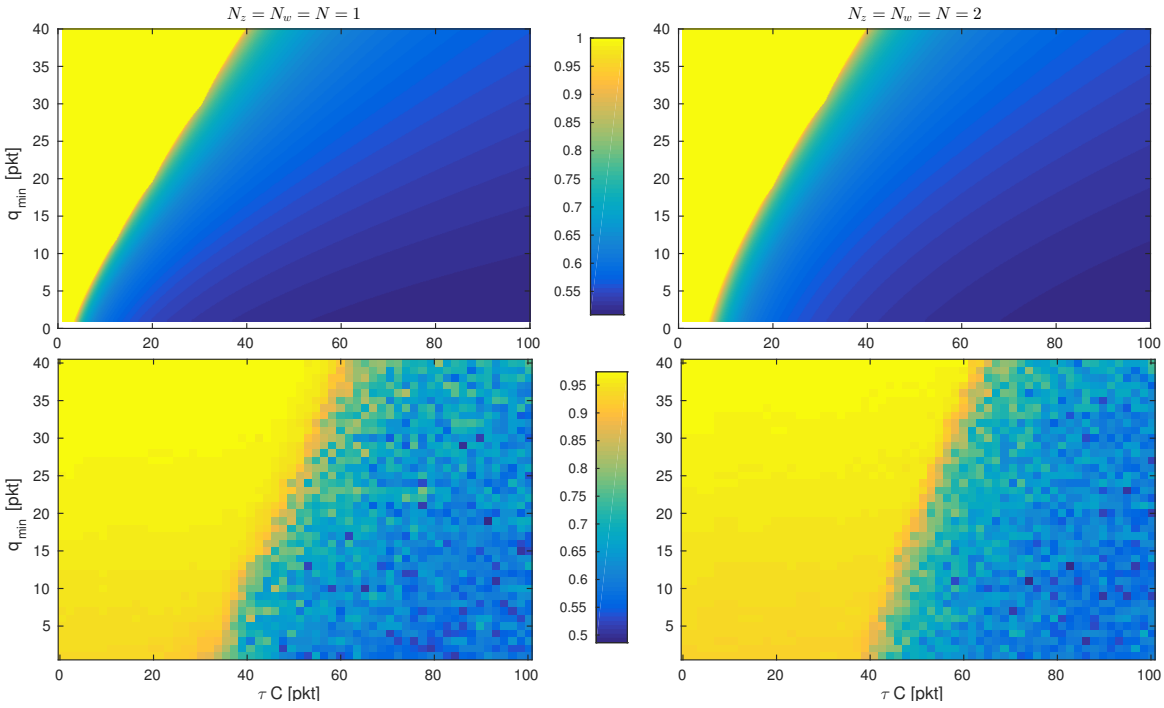


Figure 4.11 Heatmap plots of $\rho(\tau, q_{min})$ in the case of either $N = 1$ (left) or $N = 2$ (right) obtained using the proposed model (top) or `ns2` simulations (bottom).

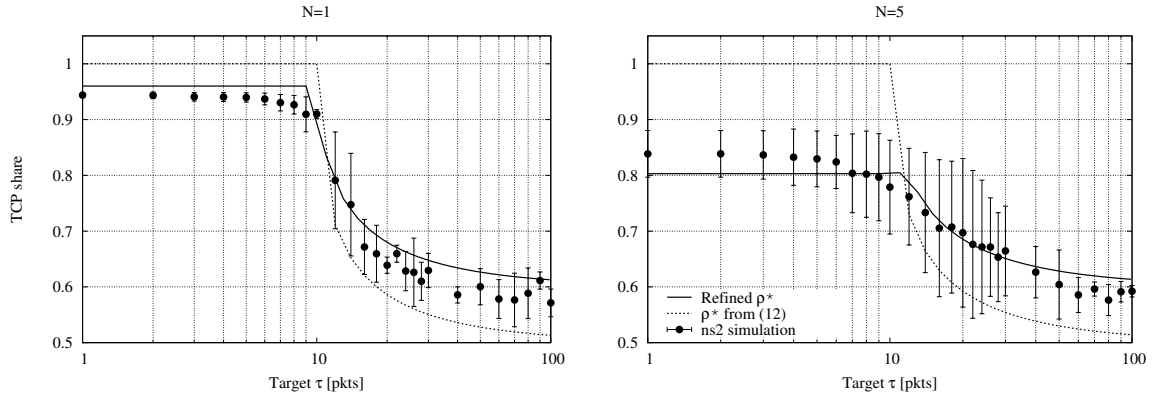


Figure 4.12 Simulation validation of TCP share ratio ρ at steady-state as a function of τC for various traffic scenarios $N_W = N_Z = \{1,5\}$.

4.3.3 Model refinement

We now return on the issue of the model underestimation of the TCP share that has been observed in Fig. 4.11 to provide a refined model. Towards this end, we fix $q_{min} = 10$pkt, $q_{max} = B = 100$, $\max_p = 0.1$ and consider two traffic scenarios $N_W = N_Z = \{1,5\}$. Fig. 4.12 compares average simulation results (solid point, with standard deviation bars over multiple runs) against the equilibrium given by (4.27) previously discussed (dotted line) and a slightly more accurate version (solid black line) that compensates two simplifications of the fluid model that we discuss next. Notice indeed that (4.27) captures reasonably well the essence of the reprioritization phenomenon. Still, two quantitative discrepancies arise.

First, it can be seen that for values of $\tau C > q_{min}$ the model underestimates the TCP share. This results from a known problem of the TCP model presented in [49] that this work extends: i.e., [49] is known to underestimate TCP congestion window with respect to simulation, which can be easily compensated by taking into account a multiplicative decrease factor of 1.5 (instead of 2) as in [49]. The refined equilibrium takes into account this correction and is significantly more accurate when $\tau C > q_{min}$.

Second, recall that when $\tau C < q$, the model degenerates into a simpler one in which only TCP flows compete on the bottleneck, hence $\rho = 1$. In practice, however, we know that LEDBAT will keep sending a minimum of 1 packet per RTT: this is done to continuously measure the queuing delay, at very low frequency and intrusiveness. LEDBAT does this in order to promptly react to queuing delay reduction and effectively utilize the spare capacity as soon as the link becomes free again. Hence, in case $\tau C < q$, our model of LEBDAT window dynamics can be, in principle, easily refined to account for this effect by introducing a non-linearity (i.e, capping from below window) at $z = 1$. As a result, the capacity available

for TCP is reduced proportionally to the number of LEDBAT flows, i.e.,

$$\rho < 1 - \frac{N}{\frac{CT_p}{P} + q}$$

Observe that, the new equilibrium point taking into account this second correction is significantly more accurate with respect to (4.27) when $\tau C < q_{min}$. However, this is obtained at the price of having to handle the additional complexity brought by the additional non-linearity. Yet, we argue that such level of detail can be better captured with `ns2` simulations, and that quantifying the *exact* level of reprioritization is less relevant for practical purposes – i.e., as users will likely be interested in knowing whether their non-critical bulk transfers are indeed lower-priority with respect to critical continuous backups, or if they compete on a roughly equal basis.

4.4 System-level solution

Recent evolution on the upper species (i.e., Internet applications) and lower species (i.e., Internet infrastructure) seems to suggest that AQM and Low priority congestion control (LPCC) protocols will have to coexist: indeed, popular applications are developing delay-based congestion control protocols such as BitTorrent/LEDBAT on the one hand, operators are starting to deploy AQM/scheduling on the user access uplink on the other hand. As such, it is imperative to find solutions to the negative AQM/LPCC interplay we have shown in this thesis. While a general solution is hard to find, as testified by the current standpoint after over 20 years of research, a patch to this specific problem may be within reach.

Some might argue that small buffers would be enough to solve bufferbloat altogether. Yet, there are several reasons why this simple solution is not sufficient. First, in the presence of too small buffers, it would be difficult for TCP and other congestion control to fully saturate the capacity, causing an undesirable efficiency loss. Second, deciding a buffer size is a matter of concern per se: consider indeed WiFi links, where the capacity may fluctuate widely over time, so that no single buffer size can at the same time (i) be large enough to support TCP congestion control and (ii) rule out bufferbloat in a fast-to-slow transition from 54Mbps to 2Mbps. Finally, jeopardizing of relative priorities are not solved by small buffers as we show in [62].

An ideal solution should achieve two goals: (i) meet quality of service constraints while (ii) respecting relative levels of priorities among protocols. Quality of service constraints clearly translate into upper-bounding the queuing delay, that we know is used by protocols to enforce their relative priorities. Since even a single TCP flow may bufferbloat the others, the

solution *needs AQM*, as otherwise the quality of service constraints would be violated. At the same time, to avoid the LPCC reprioritization phenomenon, we argue that classification capabilities will be needed in AQM to account for flows' *explicitly advertised* level of priority.

Although in the more general case classification has failed to be adopted (IP TOS field, DiffServ, etc.), and the ability to claim *higher* priority could be easily gamed, in a hybrid AQM vs. LPCC world it makes sense for flows to claim a *lower* priority: indeed, generally malicious sources would tend to exploit resources by claiming high priority, so if the source is marking its own packets as low priority, for the scheduler there is no reason not to follow the application desire by prioritizing dropping of low-priority packets. We believe that this subtle difference can make an important practical difference in terms of deployability.

A simple way would be to let application exploit IP TOS field. While the overloading of the IP TOS can be troublesome within an operator network, this is not an issue in the home. Indeed, the usefulness of the IP TOS is not end-to-end but merely meant as a low-priority signal to the box in the user home, where the bottleneck and contention arise. Hence, IP TOS could be leveraged by the ISP Customer Premise Equipment (CPE) in the user home to apply differential treatment to best-effort and low-priority traffic (e.g., different AQM loss profiles, different scheduling weights), after which the end-user IP TOS value is no longer useful and can be rewritten by the CPE (or at the DSLAM, or BRAS, etc.) in the network of an operator using DiffServ if needed.

We further stress that the firmware governing home routers and WiFi APs is generally based on some variants of the Linux kernel, possibly open-source as in the OpenWrt or CeroWrt cases. We point out that the above solution is therefore already implementable without any additional development effort – e.g., using strict priority queuing or shaping. In the Linux traffic control (`tc`) suite, this can be achieved with the `PRIO` queuing discipline (`qdisc`) that implements non-shaping container for a configurable number of classes which are dequeued in order. This first solution allows for easy prioritization of traffic, where lower classes are only able to send if higher ones have no packets available. A second solution offered by Linux `tc` is represented by the `CBQ qdisc` that offers shaping and finer-grained prioritization capabilities.

This solution is simple, as it requires a minimum amount of information exchange precisely at the interface between layers, with the upper layers requiring a marking in the lower-layer packet header. This also means that evolution of species can still happen independently, without requiring heavy cross-layer or hybrid solution – which would require much more involved exchanges between species that are not necessarily akin. The simplicity is, we believe, the root of its appeal, as this also increases the chances of its deployment, making it of high practical relevance for the problem under consideration.

4.5 Summary

In this chapter, we analyzed the interdependency phenomenon that occurs when heterogeneous protocols, namely best-effort and low priority congestion control (LPCC) protocols, share a bottleneck governed by an Active Queue Management (AQM) algorithm. In Chap. 3 we have shown by means of simulations and Internet experiments that a negative interplay exists: in particular, the relative level of priority of the congestion control protocols is reset, a phenomenon that we call *reprioritization*, meaning that the protocols compete on a roughly equal basis. With the purpose of analyzing this phenomenon we have proposed a DDE model that captures the dynamics of an overall system composed of: (i) TCP flows, which are representative of best-effort congestion control algorithm; (ii) LEDBAT flows, which is a prominent example of LPCC algorithms; (iii) an AQM control algorithm which is executed at the bottleneck queue.

By analyzing dynamical properties of the system around its equilibrium points, we have been able to provide an explanation to the reprioritization phenomenon and confirmed the generality of this issue. In fact, even though we provide a sufficient condition that allows to independently tune the LEDBAT parameter τ to avoid reprioritization regardless of the AQM employed, we show that such a condition is of scarce practical interest due to the difficulty of accurately measuring the delays at the end systems. Moreover, model predictions have been validated against `ns2` simulations when RED is used as AQM algorithm.

Due to the increasing deployment of both low-priority congestion control and AQM techniques that today are employed to fight bufferbloat, the problem discussed in this thesis may be of significant practical relevance. As we believe that it may be desirable for end-users (or end-user applications) to autonomously and coarsely set their relative level of priorities, we have proposed simple yet an effective system-level design and practices that give a solution to the issue that we have analyzed in this thesis.

Part II

Data center network (DCN)

Chapter 5

Background

In this chapter, we first motivate in Sec. 5.1 our study focusing on the scheduling mechanism deployed in a data center network, with the multi-tenant system assumption. Sec. 5.2 provides a comprehensive taxonomy of existing proposals of low-latency data center network design. We take a broad view identifying the main trends over the decade, as well as an in-depth view on representative proposals focusing on different network layers.

5.1 Motivation

In the last decade, data center networks (DCNs) have increasingly been built with relatively inexpensive off-the-shelf devices. DCNs are frequently assumed to be highly specialized environments, owned by a single entity that has full control of both the network architecture and the workload. DCN transport research has consequently explored a larger design space than that of the Internet leading to designs that generally do not focus on a single layer of the protocol stack but are more typically cross-layer. The proposed transport solutions may for instance integrate application information, use explicit or implicit rate control, incorporate routing and load balancing or rely on an omniscient oracle.

While most work [68, 18, 69–73, 17, 74–76, 19, 77] broadly agrees in identifying application requirements as *throughput* for fat data transfers and *latency* for short transactions, the precise definition of these observables varies across proposal: for instance, special attention is given to either average [68, 69, 18], tail [73] short flows [70, 71], task [76] or co-flows [19, 77] completion times.

Freedom in the DCN design space translates into a relatively crowded landscape of proposals, each of which is typically designed and tweaked with a specialized application scenario in mind. DCN proposals are further tangled as their design is tailored for very specific workloads, with diverse application patterns including for instance, query-response [16, 17],

map-reduce jobs [18, 19], or packet-size access to the DRAM of other machines [20]. Rarely, if ever, is a DCN design tested with a workload other than that for which the system was explicitly designed.

Beyond any reasonable doubt, the single-tenant assumption will be severely challenged in the near future. With increasing user reliance on cloud applications and public cloud providers like Amazon AWS, it will require DCNs to evolve towards multi-tenant system handling a significantly more heterogeneous set of applications and workloads. The range of applications will inevitably increase either because a single-tenant data center is used for new lines of business or simply because the data center is used by an increasing number of independent tenants. DCN workload will thus evolve beyond the typical mixture of short transactions and fat elastic transfers considered nowadays and will notably include a significant fraction of rate-limited flows with strict latency requirements. Such flows are produced by gaming applications [78], instant voice translation [79] and augmented reality services, for example. The precise mix will also be highly variable, depending on both the degree to which networked applications rely on offloading to the Cloud (e.g., streaming gaming meta-data for local rendering or streaming Cloud-rendered videos) and on the expected benefits of offloading that vary depending on device capabilities, battery life, and connectivity opportunities, etc. [80].

As DCN resources are increasingly shared among multiple stakeholders, we must question the appropriateness of some frequently made assumptions. How can one rely on all end-systems implementing a common, tailor-made transport protocol like DCTCP [16], when end-systems are virtual machines under tenant control? How can one rely on applications truthfully indicating the size of their flows to enable the shortest flow first scheduling as in pFabric [17], when a tenant can gain better throughput by simply slicing a long flow into many small pieces? Expected future DCN usage clearly puts these assumptions in doubt and leads us to question the expected benefits of fragile DCN designs that rely on them.

Contrary to the specialization trend, we therefore argue that DCN design would benefit from an increased generalization: by this, we mean reliance on well-understood mechanisms, provided as network services, that require as few assumptions as possible on the DCN workload.

As we would further discuss in Sec. 5.2, we observe an obvious yet surprising omission in the explored DCN landscape, namely a scheduling mechanism providing fairness among flows, coupled with Active Queue Management (AQM) to upper-bound delay. Such a scheduler is FQ-CoDel that has recently gained prominence in another network application, fighting bufferbloat notably in home routers [4, 81]. Its aim of keeping throughput high while controlling queueing delay and its generality with respect to the traffic mix clearly identify it

as an excellent candidate for DCN operation. Our objective in this thesis is to compare its performance to that realized by alternative state-of-the-art DCN proposals. We find notably that, under FQ-CoDel:

- the throughput of plain TCP is better than that of a tailored end-to-end solution like DCTCP [16]
- short flows incur a delay comparable to what they incur under pFabric [17]

The future-proof performance advantages of FQ-CoDel coupled with remarkably simple deployment, since it is already implemented in the Linux kernel, make the scheduler an indispensable component of future multi-tenant DCNs.

5.2 Related work

We review the effort has been made during the last decade on DCN design with both a broad view Sec. 5.2.1 and an in-depth view Sec. 5.2.2 of representative proposals which we believe would briefly illustrate the design space involving almost all network layers.

5.2.1 Broad view

In Tab. 5.1, we report an admittedly incomplete view of DCN research that illustrates several important trends.

The first wave of DCN research begins with studies focused on **network topology**, in the quest of larger throughput and non-blocking transfer with commodity devices. Some well-known examples include FatTree [82], DCell [83], Portland [84], BCube [85] and VL2 [86] and, more recently, we observe a general agreement on the use of the leaf-spine topology [17, 77, 75, 93]. In recent years, to address the lossless assumption of data center networking, researchers focus either on layer-2 fabrics, or Overlay Virtual Network [90] introducing new protocols spanning layer-2 to 4.

A second wave is marked with improved **network fabric** include Hedera [88] and Orchestra [18] that rely on a centralized oracle to deal with routing, load balancing, congestion avoidance and fault tolerance. Starting with pioneering work of DCTCP [16], the community recognized that specific designs were needed for TCP to cope with the high bandwidth and low delay properties of DCNs, sprouting the design of new **end-to-end transmission protocols** including [87], HULL [20] and L2DCT [70] that, to the best of our knowledge have not yet been deployed. It is noteworthy that optimal parameter tuning for deployed TCP variants like DCTCP is still the subject of research [94]. Orchestra [18] goes further by coupling control of both network fabric and end-to-end congestion control.

Table 5.1 Taxonomy of recent data center network design.

Proposal	Yr	Primary metrics	Information	Rate Control†
FatTree[82]	08	Bandwidth		
DCell[83]	08	Bandwidth		
Portland[84]	09	Scalability		
BCube[85]	09	Bandwidth		
VL2[86]	09	Goodput		
DCTCP[16]	10	Latency		Imp+ECN
ICTCP[87]	10	Goodput		
Hedera[88]	10	Bisection bandwidth	switch buffer	
D3[72]	11	Throughput	deadline, size	Exp (D)
MPTCP[89]	11	Throughput		MP-enabled
Orchestra[18]	11	Transfer-CT	everything	Exp (C)
HULL[20]	12	Latency, throughput		DCTCP+ECN
DeTail[73]	12	Tail FCT	priority	TCP+ECN
PDQ[69]	12	FCT	deadline, size	Exp (D)
L2DCT[70]	13	Short flow FCT		Imp+ECN
pFabric[17]	13	FCT	priority	Imp
zOVN[90]	13	FCT		
RepFlow[71]	14	Short flow FCT	size	
Baraat[76]	14	Task-CT	priority	Exp (D)
PASE[74]	14	FCT	size, max rate	Exp+ECN (D)
Varys[19]	14	Coflow-CT	size	Exp (C)
CONGA[77]	14	FCT		
Fastpass[91]	14	Fairness/FCT	size	Exp (C)
FlowBender[92]	14	Latency		
PIAS[75]	14	FCT		DCTCP+ECN
RAPIER[93]	15	Coflow-CT	size	Exp (C)

† Rate/congestion control: (Exp)licit vs (Imp)licit; (D)istributed vs (C)entralized

Table 5.2 Taxonomy of recent data center network design (continued).

Proposal	Yr	Routing	Scheduling	Layers‡				
				L	N	T	A	O
FatTree[82]	08	Yes	Yes	✓				✓
DCell[83]	08	DFR		✓				
Portland[84]	09	Yes		✓	✓			✓
BCube[85]	09	BSR		✓				
VL2[86]	09	ECMP VLB		✓	✓			
DCTCP[16]	10		FS			✓		
ICTCP[87]	10	Imp	FS			✓		
Hedera[88]	10	Yes						✓
D3[72]	11	ECMP VLB	greedy			✓	✓	
MPTCP[89]	11	ECMP				✓		
Orchestra[18]	11	Yes	FIFO, FS, Priority					✓
HULL[20]	12	ECMP				✓		
DeTail[73]	12	packet-based		✓	✓	✓	✓	
PDQ[69]	12	ECMP	EDF/SJF			✓	✓	
L2DCT[70]	13	Imp+ECN	LAS			✓		
pFabric[17]	13	RPS	Priority			✓	✓	
zOVN[90]	13			✓	✓	✓		
RepFlow[71]	14	ECMP				✓	✓	
Baraat[76]	14		FIFO-LM			✓	✓	
PASE[74]	14		SJF			✓	✓	
Varys[19]	14		SEBF + MADD			✓	✓	✓
CONGA[77]	14	flowlet-based			✓			
Fastpass[91]	14	packet-based	maximum matching			✓	✓	✓
FlowBender[92]	14	flow-based			✓	✓		
PIAS[75]	14		SJF			✓		
RAPIER[93]	15	coflow-based	MRTF		✓	✓	✓	

‡ Layers: (L)ink, (N)etwork, (T)ransport, (A)pplication, (O)racle

During the third wave, a number of proposals have promoted a **cross-layer design**. Such DCN architectures impact multiple aspects including explicit [72, 69, 74, 93] or implicit [73, 17] congestion and flow control at end-systems, flow scheduling [72, 69, 17, 76, 74, 19, 91, 75, 93], routing [73, 17, 77, 91–93] and oracles [19, 91]. Among these proposals, pFabric [17] has near ideal flow-level performance and has become a de facto standard against which alternative DCN designs must be compared. Application-layer information has been progressively integrated into decision logic (as either a priority index [73, 17], or a flow deadline [72, 69] or size [72, 69, 71]).

The last wave witnessed a move from network-oriented to **service-oriented metrics**, similarly to the Internet QoS/QoE. Varys [19] and RAPIER [93] use co-flow completion time, and Baraat [76] uses task completion time. Contrarily to implicit rate control of pFabric, [76, 74, 19, 91, 93] all employ explicit rate control (with either distributed or centralized arbitrator).

Besides arbitrator and end-to-end transmission protocol, integration (or more precisely, customization) of *routing and scheduling policies* is gaining its popularity. They are usually carefully tweaked to meet its objective and arbitrators' working mechanism.

Approaches become more diverse. For example, in contrast to mainstream complicated cross-layer design, *deploy-friendly solutions* get their place: RepFlow [71] replicates short flow through different path to reduce short flow's completion time in only application layer (or a module in network stack); CONGA [77] touches only network layer to design a distributed in-network congestion-aware flowlet-based load balancing; and PASE [74] tries to combine the benefit of transport strategies, in-network prioritization and arbitration in a single transport framework. PIAS [75], introduces a practical flow scheduling approach miniziming FCT with no prior application-layer knowledge. Some orthogonal attempts have also been made, such as the integration of multipath capability in the network or at application level.

Important observations

One of the most important designs in the network development history is the layered structure, which decouples different functionalities, enabling good scalability and flexibility. However, with the understanding and analysis of previously mentioned proposals, we can conclude that, researchers worked hard to optimize data center network with regard to limited service traffic pattern (e.g. web search, data mining, etc) and limited network architecture variation (e.g. FatTree, spine tree, non-blocking fabric, etc). The principle philosophy applied (until now) to achieve this is the assumption of single-tenant with a complete control of hardware, software and traffic inside a data center.

Researchers broke the decoupling principle by gathering and communicating information among several layers to help decision-making process (e.g. forwarding scheduling, congestion control, rate control, etc), which could also be implemented in different layers and entities. Besides the precise information collection of network fabric, endpoints, or arbitrators, what makes these proposals different in the performance as well as feasibility is:

- the amount of information collected, which largely decides the performance.
- the interpretation of the metric, which is critical when this represents a translation from an observed metric to the real situation. e.g., whether an increasing delay indicates a network congestion due to concurrent flows through a shared bottleneck, or a too high sending rate saturating network devices' buffer.
- the decision-making process, in which people combine and utilize all available information in an innovative way to make decision on rate, path, time, etc. The choice of the decision also matters, e.g., whether we send a congestion indicator only to reduce rate or follow a strict scheduling policy to pause flow.
- the information flow pattern, which involves the time, carrier, sender and receiver of each piece of information. This usually influences the difficulty in deployment, e.g., complicated decision logic on the packet's header fields, memorizing flow history, and adopting new protocol or arbitrator are generally adverse factors for deployment.

5.2.2 Representative proposals

We briefly introduce some representative proposals. Each exhibits an important aspect of design choice, especially the opportunity and constraints the working layer(s) may enforce. DCTCP and pFabric are selected as our comparison candidates in our work. FQ-CoDel, which we propose to use, was previously covered in Sec. 2.2.

zOVN (zero-loss Overlay Virtual Networking)

Zero-loss Overlay Virtual Networking (zOVN) [90] is a representative proposal closely related to data link layer (Layer-2) network design for data center. With the prospect of increasing adoption of virtualization in datacenters, the underlying assumption is that virtual networks (VN) will be deployed in practically most multi-tenant datacenters, providing a fully virtualized Cloud platform by default. zOVN is designed as the first zero-loss OVN to reduce the query and flow completion time of latency-sensitive datacenter applications.

Recent development in layer-2 data center network design has two trends. The advance of lossless, flat layer-2 fabrics based on commodity-ready Converged Enhanced Ethernet or InfiniBand brings benefits in efficiency and performance. On the other hand, Software

Defined Networking (SDN) based Overlay Virtual Networking (OVN) provides flexibility on management, migration, etc. However, existing OVNs are all lossy, which proved by the authors to be critical to performance. zOVN tries to remedy this situation through lossless virtual switch zVALE, which is designed to operate in a dual push/pull mode. [1] By eliminating the costly packet loss, it [90] claims to "achieve an up to 15-fold reduction of the mean completion time with three widespread TCP versions." Furthermore, zOVN can be orthogonally composed with other schemes for functional or performance enhancements on other layers.

CONGA

CONGA [77] is a network-based distributed congestion-aware load balancing mechanism for datacenters. The main strength is the capability of efficient load balancing and seamlessly asymmetry handling, without requiring any TCP modification. CONGA realizes this goal by taking distributed congestion-aware load balancing inside network fabric, thus nearly achieving the performance of a centralized scheduler (which instead requires transport layer protocols modification) without suffering from slow reaction in centralized scheduling schemes. At the same time, it outperforms widely used ECMP based on local decision without knowledge of potential downstream congestion on each path.

CONGA is designed and evaluated in a Leaf-Spine topologies. The leaf switches take the majority functionality. The source leaf makes load balancing decisions based on per uplink congestion metrics, derived by taking the maximum of the local congestion at the uplink and the remote congestion for the path to the destination leaf that originate at the uplink. The remote metrics are obtained via opportunistic piggybacks from the destination leaf switch. It is worth mentioning that CONGA considers TCP flows as series of flowlets, each consists a sequence of packets using the same uplink without a sufficiently long gap. Load balancing decisions are made on the first packet of each flowlet, and remain valid for subsequent packets in the same flowlet.

In testbed experiments [77], "CONGA has 5× better flow completion times than ECMP even with a single link failure and achieves 2–8× better throughput than MPTCP in Incast scenarios."

DCTCP (Data Center TCP)

Data Center TCP (DCTCP) [16] is the first attempt to enhance the TCP congestion control algorithm (Layer-4 transport layer) for data center networks, with the goal of enabling small

1. pull: output queue pull packets from input queue; push: input queue push packets to output queue.

predictable latency and large sustained throughput for mixing workloads. It leverages Explicit Congestion Notification (ECN), a feature which is increasingly becoming available in modern data center switches. DCTCP sources extract multi-bit feedback on congestion from the single-bit stream of ECN marks by estimating the fraction of marked packets. In doing so, DCTCP sources react to the extent of congestion, not just the presence of congestion as in TCP. In other words, DCTCP starts marking congestion early and aggressively, thus enable the sender to react early.

The DCTCP algorithm has three main components:

- Simple marking at the switch: based on instantaneous queue length, that switch compares with a pre-defined parameter marking threshold, K. It marks the arriving packet with the CE codepoint if the queue occupancy is greater than K.
- ECN-Echo at the receiver: a DCTCP receiver sets the ECN-Echo flag if and only if the packet has a marked CE codepoint.
- Controller at the sender: a DCTCP sender maintains an estimate of the fraction of packets that are marked, called α, which is updated once for every RTT. It further uses α to regulate congestion window size. When the queue starts exceeding K, it gently reduces the congestion window. When congestion persists, α will increase until 1, which turns DCTCP's behavior into the same as TCP.

The importance of this protocol lies in several aspects. One the one hand, it raises the community's awareness that the state-of-the-art TCP has a limitation in real data center environment, as proven by "detailed traffic measurements from a 6000 server data center cluster, running production soft real-time applications". On the other hand, the solution not only provides an impressive enhancement over TCP, as "DCTCP delivers the same or better throughput than TCP, while using 90% less buffer space. ... DCTCP enables the applications to handle 10X the current background traffic, without impacting foreground traffic. Further, a 10X increase in foreground traffic does not cause any timeouts, thus largely eliminating incast problems." , but also with a deploy-friendly design with mechanism already available in commodity hardware. With its implementation in simulator and Linux kernel, it becomes the essential benchmark for all later research.

pFabric

pFabric [17] is the most well-known cross-layer design (Layer 2-4), but its design is beyond simply "patching" different mechanisms, but realise a clean-slate datacenter transport design that provides near theoretically optimal flow completion times even at the 99th percentile for short flows, while still minimizing average flow completion time for long flows.

pFabric's entire design consists of the following:

- End-hosts: put a single number in the header of every packet that encodes its priority based on independent flow information from application layer. All flows start at line-rate and throttle their sending rate only if they see high and persistent loss.
- Switches: have very small buffers and decide which packets to accept into the buffer and which ones to schedule strictly according to the packet's priority number. They perform a strict priority-based packet scheduling and dropping mechanisms. Thus, each switch operates independently in a greedy and local fashion, which lead to an approximately optimal flow scheduling decision across the entire fabric.

The minimalistic design is the prominent characteristic, which requires no flow state or complex rate calculations at the switches, no large switch buffers, no explicit network feedback, and no sophisticated congestion control mechanisms at the end host. Such design permits pFabric to be the state-of-the-art solution and, together with DCTCP, turns it into another benchmark. As claimed by the authors, "pFabric achieves near-optimal flow completion times. ... pFabric reduces the FCT for short flows compared to DCTCP by more than 2.5–4× respectively at the mean, and more than and 3–4× respectively at the 99th percentile." However, it requires modifications both at the switches and the end-hosts, which impedes the deployment in commodity devices. Furthermore, it would be rather brittle and sensitive to set flow priority due to flow and user dynamics.

The fundamental conceptual insight behind the design is the observation that rate control is a poor and ineffective technique for flow scheduling and the mechanisms for the two should be decoupled and designed independently. pFabric shows that large buffers and complex rate control could be unnecessary in datacenters. We strongly believe in the same principle, and we consider it crucial in a multi-tenant datacenter scenario to ensure fairness among heterogeneous users, flows and transport protocols.

RepFlow

RepFlow [71] is a recently proposed solution based purely on application layer (Layer-7) mechanism. Compared to other schemes, RepFlow offers competitive though not optimal performance, but rather innovation that sheds some light on a simple yet practically effective approach. RepFlow aims at providing low latency for short flows both average and in the 99-th percentile, and can be readily deployed in current infrastructures. It can also be used with other data center transport protocols such as DCTCP to further improve performance.

RepFlow uses flow replication to exploit multi-path diversity. It works on top of layer-4 transport protocols, replicating short flows to reduce the completion times, without any change to switches or host kernels. With ECMP, the original and replicated flows traverse

distinct paths with different congestion levels, thereby reducing the probability of long queueing delay. The deployment involves two parameters, i) the threshold to differentiate a flow from long to short; 2) the condition under which to replicate the flows. Both decisions are critical to the overall performance, but also grant great flexibility to each specific case. The low-level implementation is equally flexible, one can either modify application behavior, provided RepFlow as a general library, or further be incorporated into transport protocols.

According to the result under simulation and emulation, "RepFlow improves TCP by around 30%–50% in most cases. RepFlow over DCTCP improves DCTCP further by 30%, providing very close-to-optimal FCT compared to state-of-the-art pFabric." [71].

Chapter 6

Fairness in data center network

Bearing in mind the observation we made, we believe that it is not likely that one single *perfect* solution exists for most scenarios in data center, instead a *good* solution making reasonable compromise is our goal. In Sec. 6.1, we argue the rationality of adopting fair scheduling in DCN. We then describe the methodology we used and recommendation of future research in Sec. 6.2. The result of our simulation is reported in Sec. 6.3, followed by a summary in Sec. 6.4.

6.1 Fairness

We argue that the advantages of decoupling rate control from scheduling, put forward in the pFabric proposal [17], would similarly be obtained by implementing per-flow fair scheduling on bottleneck links. We also argue this design not only to be simple enough, but also to be good enough in terms of performance, as we justify in Sec. 6.3.

6.1.1 Fair scheduling

Starting with the original proposal of Nagle [95], per-flow fair scheduling has often been advocated as a means to make Internet bandwidth sharing robust and more efficient. However, the need has never been considered sufficiently compelling to warrant widespread implementation. An exception is a recent work on fighting bufferbloat, notably on home routers [10] where the preferred solution has been to perform fair queuing on the user access line, in association with the CoDel queue management algorithm [4, 81].

Per-flow scheduling on the user access line ensures low latency for packets of delay-sensitive flows like VoIP while allowing bulk data transfers to fully utilize residual bandwidth. Such scheduling is generally unnecessary on Internet links beyond the access line since these

are rarely a bottleneck: the combined rate of flows in progress, accounting for the access bottleneck, is generally much less than link capacity. This is not the case in data centers, however, where an upstream or downstream link between a server and its top-of-rack switch can be saturated by a single flow.

Using a Fair queuing (FQ) scheduler such as Deficit Round Robin (DRR), the latency for single packet flows is bounded by the duration of a cycle, in turn lasting for the service duration of n flows in the queue. This is of course a heavy penalty for short flows, which can be bypassed by Priority Fair Queuing (PFQ): PFQ adapts DRR to give priority to "new flows", which are served in FIFO fashion in a DRR cycle. New flows include those emitting packets spaced by a time gap greater than a DRR cycle (or, equivalently, at a rate lower than current fair rate). This constraint is easily satisfied for transactions-style communications, as well as for bandwidth-limited transfers such as voice/video/gaming streams.

Priority fair queuing

The DRR-based "FlowQueue Controlled Delay" (FQ-CoDel) scheduler [81], originally proposed for fighting bufferbloat, does in fact more than just fair scheduling: it incorporates a priority mechanism to further reduce the packet latency of low rate flows, which would also be very useful in DCNs. A fair queuing scheduler maintains a list of flows that currently have backlogged packets. It can, therefore, recognize packets that come from flows that are not currently backlogged. Such flows will include single packet queries as well as any flows emitting packets at a rate less than the current fair rate. In FQ-CoDel, these packets are dequeued with priority, minimizing their latency without undue impact on the throughput of backlogged flows.

This mechanism was first proposed as a means to realize implicit service differentiation (e.g., [96]) and has been rediscovered in the fight against bufferbloat and implemented in FQ-CoDel [81]. FQ-CoDel drops packets as necessary using Codel while the proposal in [96] was to use longest queue drop [97].

Controlling queue delay

In FQ-CoDel, packets can be dropped on applying the CoDel [4] AQM algorithm to each flow queue. Packets are also dropped from the flow with the biggest backlog when the shared buffer would otherwise overflow. The latter, longest queue drop policy [97] is another possibility that would, as in [17], avoid the coupling between rate control and scheduling that is implicit in CoDel. However, in this thesis, we prefer to use techniques that are available out-of-the-box and that would arguably speed up deployment in DCN operation. As we do

not fine tune performance in the scenario under investigation, our results may be considered conservative. A detailed study of the impact of implementation options and the precise assessment of the achievable performance, are left for future work.

6.1.2 Suitability for DCN

We argue that per-flow fair scheduling on DCN links would enable low packet latency for delay-sensitive flows, while allowing bulk data transfers to attain maximum throughput at the same time, making it an interesting design for DCN.

Flow identity

Without loss of generality, we identify flows by the usual IPv4 5-tuple. However, it may be more appropriate in a DCN to use other criteria like the origin and destination servers or virtual machines, or to identify co-flows belonging to a single task. Note that the proposal is to apply lightweight, egalitarian fair sharing as opposed to weighted fair sharing. The latter would incur additional complexity to determine the weight from local state or a header field. Yet, whether this additional complexity is truly necessary for performance reasons demands future investigation.

Quantum size

The default DRR quantum size in FQ-CoDel is one 1500 byte MTU. This implies the packets of any flow emitting less than 1500 bytes in a DRR cycle are handled with priority. In the data center environment, it may make more sense to increase the quantum size, to ensure all packets of a short query are handled with priority, for instance.

It is interesting to note that the Baraat [76] scheduler has similar behavior to FQ-CoDel with a particular choice of quantum. Baraat handles flows in FIFO order until they are observed to have emitted more than a given number of bytes θ. They are then required to fairly share the link bandwidth under the end-to-end control of RCP [68]. The result of applying FQ-CoDel with a quantum equal to θ would be broadly similar.

Rate-limited flows

The workloads considered in previous proposals, including DCTCP [16] and pFabric [17], is simulated according to statistics extracted from two traffic patterns: the query-response (search) and data analytics (map/reduce) applications. Although we doubt the validity of such generalization from two single observation in a specific time and place, researchers agree to

one important characteristic observed: flow sizes follow a heavy-tailed distribution. Generally, a data analytics application generates more skewed flows in size than query-response ones.

Although we cannot obtain the exact flow information of these universally used workloads, we believe that they do not include flows whose rate is intrinsically limited. However, such flows exist in any data centers supporting streaming applications, and would benefit from the low latency provided naturally by FQ-CoDel.

Incast

If many flows converge on the same interface of a switch over a short period time, the packets may exhaust either the switch memory or the maximum permitted buffer for that interface, resulting in packet losses. This happens usually at the at the queue of the switch port connected to the aggregator. Even if the flow sizes are small, this partition/aggregator traffic pattern synchronizes the request for data from workers' responses and thus creates incast in a short time but induces severe consequence.

Imposing fair sharing has a similar impact on incast as DCTCP [16] since fair sharing between the fanned-in flows is guaranteed. The fair rate during incast would likely still be high enough that packets of any streaming flow sharing the bottleneck link would be handled with priority.

Stability

As in the evaluation of pFabric [17], we envisage a simple stochastic demand model where flows arrive according to a Poisson process, in which case fair sharing is known to be stable as long as the overall load is less than 1 on every link. Conversely, as overtly recognized in [17], "*size-based traffic prioritization may reduce the stability region of the network*" with respect to the maximum capacity attained by fair sharing. This occurs because the progress of a low priority flow may be delayed by high priority flows momentarily saturating any link on its path. This leads to situations where network capacity is effectively wasted leading to unbounded completion times for the low priority flows, even when the load on every link is less than 1 [98].

Given that analytical results are hard to obtain in case of size-based scheduling [98], a simplistic model of strict priority scheduling may be preferable for illustration purposes. Let us consider a star network with a large number of branches having unit uplink/downlink capacity, and where all flows use one uplink and one downlink, which represents a DCN where only the edge links between the server and the top-of-rack switch are capacity limiting. The network handles two classes of traffic with loads ρ_1 and ρ_2, with ρ_i given by the product

of the average arrival rate and flow size for class i . Flows of class-1 have preemptive priority over class-2, i.e., to make progress, a flow of class-2 should see no flow of type-1 on either of its two hops. Since there is a large number of links, their occupancy is practically independent and the probability both links are available to class-2 is $(1-\rho_1)^2$. Intuitively, to ensure stability, the class-2 load must be less than $\rho_2 < (1-\rho_1)^2$. This inequality defines the traffic capacity, which is minimized when $\rho_2 = \rho_1/2$ and is then equal to 0.75. In other words, to ensure stability we must have $\rho_1 + \rho_2 < 3/4$, implying that 1/4 of the capacity is wasted.

This phenomenon impacts both pFabric and FQ-CoDel. It remains to more thoroughly evaluate this impact under a relevant range of workloads. Of course, neither is pFabric size-based scheduling entirely captured by a strict priority model, nor FQ-CoDel priority to packets of "new" flows is accurately described via perfect fairness. Nevertheless, these models warn for potential inefficiency for pFabric in the high-load regime, and especially in cases where the portion of high-priority traffic is sizable. We recognize this potential limitation while the relevant work is left to future work.

6.2 Methodology

In this section, we rationalize our methodology used, as well as the parameters of network and experiment. As observed in previous research works, due to the fact that each proposal targets (and is possibly tuned to) a specific data center environment and aims at different design objectives, the comparison between them is seldom done in a fair way. Although a perfect comparison is hard to achieve, we argue that certain principles considering network, workload, and measured metric should be tuned. Thus, we carried out a careful calibration before the simulation.

Terms of comparison

We consider criteria in choosing our candidates for comparison. We first narrow down our candidates to those under the same design space as FQ-CoDel, that is a transport layer scheme. Solutions focusing on different layers such as zOVN [90] and RepFlow [71] are not suitable in our context. Indeed, they are orthogonal designs which can be deployed together with FQ-CoDel, thus future work can be planned. Since FQ-CoDel works purely in the network, it would be unfair to compare with those having different design assumptions and requiring application layer involvement. We then exclude solutions requiring additional deadline information, such as D3 [72] and D2TCP [99]. Finally, concerning the recognition in the community and the availablility of open-source code, we compare FQ-CoDel against two representative alternative DCN design: (i) DCTCP [16], a distributed end-to-end design

that exposes a general transport service through an L4 abstraction and (ii) pFabric [17], a clean-slate cross-layer design that aims to optimize flow completion time performance.

The first one has become the benchmark since its proposal in almost all designs, including pFabric [17]. It is actually implemented in production data centers [94], while the second represents an ideal that may only be attained in a particular protected data center environment where end-systems are compliant. However, pFabric [17] is recognized as the state-of-art design in terms of flow completion time reduction. Recent solutions such as RepFlow [71], Baraat [76] and PIAS [75] include it in comparison as well. The code of both designs is available in *ns2*.

We argue that this subset is representative considering the recognition in the community as well as the similar design goal among them. We admit that comparison among designs spanning more layers would be useful and necessary for future work, what we achieve in this work is a good first step towards such objective.

Network and workload

We adopt a downscaled version of the pFabric network scenario (32 vs 144 hosts [17], interconnected by a Leaf-Spine topology with the same delay characteristics). We adopt the pFabric "data mining" workload, presented in Fig. 4b in [17]. Flows arrive according to a Poisson process and have a size drawn independently from an empiric distribution, where half of the flows are single packet while 95% of bytes are in flows larger than 35 MB. No matter how much realistic, a single scenario is only representative of a rather arbitrary evaluation point. Especially, real flow size distributions represent a snapshot of traffic, but are not very telling on how the traffic mixture will evolve.

Note that this workload does not include any rate-limited flows as arise in streaming and gaming applications, for instance. To account for growing impact of rate-limited flows in DCN, we tweak the data mining scenario by controlling the percentage of 1-packet long flows, which in DCN settings are an accurate representation of bandwidth-limited flows (voice, music, games; and in the case that the DRR quantum is extended to a fixed but small number, also of video streams). Indeed, as long as the individual rate of such flows is a small fraction of link capacity (which is clearly the case since the bottleneck link capacities are O(10Gbps) while stream rates are O(10Mbps) even for 4K videos) their packet latency is broadly the same as that of the single packet flows. Specifically, we let the overall volume of 1-packet flow vary between 0.01% and 10% of the overall offered traffic, and proportionally reduce the occurrence of all the other flows sizes.

Table 6.1 pFabric TCP tuning

	param	pFabric	(default)	note
Retransmission	minrto_	45μs	(200ms)	minimum rto
	maxrto_	2s	(60s)	maximum rto
	rtxcur_init_	45μs	(3s)	initial rto
	tcpTick_	1μs	(10ms)	clock granularity
Flow control	interval_	6μs	(100ms)	delayed ack
	window_	10^6	(20)	rx window
Congestion control	windowInit_	12	(2)	initial cwnd
	maxcwnd_	queue-1	(∞)	max cwnd
	windowOption_	0: basic	(1: std)	congestion avoid mode

Performance measures

We compare the performance of the considered DCN designs through two measures widely used in the literature: the flow completion time (FCT) of single packet flows and the mean flow throughput defined as the ratio of size in bits to FCT in seconds. The single packet flow completion time is a measure of latency that, as discussed above, is also a relevant performance measure for rate-limited flows.

Flow completion time (FCT) has become the unanimously accepted performance metric since DCTCP[16], with different focuses, such as, the average FCT (generally), the tail FCT [73] or FCT of short flows [70, 71]. We argue that *throughput* for long data transfers and *latency* for 1-packet transactions can provide a simple yet complete picture of DCN performance. Notice that both metrics directly relate to FCT, since FCT of 1-packet flows practically expresses the DCN latency, whereas throughput of larger flows is inversely proportional to the FCT.

6.2.1 Calibration

Given the fact shown from all the related works that each data center is very different in terms of size, architecture, devices, and more important, services run inside, we point out that few fair comparisons among them have been done. In fact, many of them even bear different design objectives, which makes it quite hard to select suitable comparing candidates (and of course, limitation of simulation and experiment technology stack is not ignorable). We believe that comparison should be done in a way in which scenario and objective together

to be the first-class citizens. As a result, comparison conclusion should be addressed more carefully with the condition of scenario and reasoning of choice.

Since we consider the pFabric scenario (i.e., workload, topology, capacities, etc.), we retain the transport protocol settings of [17] and make necessary adjustments for DCTCP and FQ-CoDel to make the performance comparison as fair as possible.

Local AQM tuning

For DCTCP, we resort to standard DropTail FIFO, and experiment with two buffer sizes (namely, 100 and 1000 packets). It is not reasonable to adopt pFabric's buffer size setting of 24 packets, which is designed with the assumption of very low packet drop possibility. Due to the working mechanism of DCTCP, it does not impose strict queue length management, as a result, these buffer sizes could have a slightly different impact on short flows latency and long flow throughput performance as we will discuss later.

Our fair scheduling candidate is represented by FQ-CoDel, which uses a stochastic fair queuing implementation of DRR. Our preliminary experiments show that buffer size variation does not have a significant impact on our metrics, we then keep it as 100 packets for simplicity. FQ-CoDel relies essentially on three parameters: the number of FQ hash buckets (default 1024), the CoDel target delay (default 5ms) and the inference interval (default 100ms). The default CoDel settings were meant to counter bufferbloat in the access network, with timescales correlated to human perception that are thus orders of magnitude larger than what is reasonable for the DCN environment. Considering that the RTT propagation delay between any two hosts in our DCN scenario is $12\mu s$ and a full-packet packet transmission delay is $1.2\mu s$, we downscale by a factor of 1000x both the target delay (about 4 packets per bucket) and the inference interval. While these settings work well in practice (after verifying with a large set of combination), a more careful tuning along the line of [27] could possibly improve performance further. In terms of hash buckets amount, our experiment result shows that the default buckets amount is large enough without incurring performance degradation.

Different from pFabric, we do not use packet spraying in FQ-CoDel, as we prefer to gather results that are conservative and valid in order sense, as opposite to gather finely-tuned but scenario-specific results.

End-to-end TCP tuning

To perform a fair comparison is necessary to specialize transport protocol parameters to the DCN environment [94]. Tab. 6.1 contrasts the pFabric settings with default TCP values. The DCN environment clearly requires an increase of timestamp precision [100],

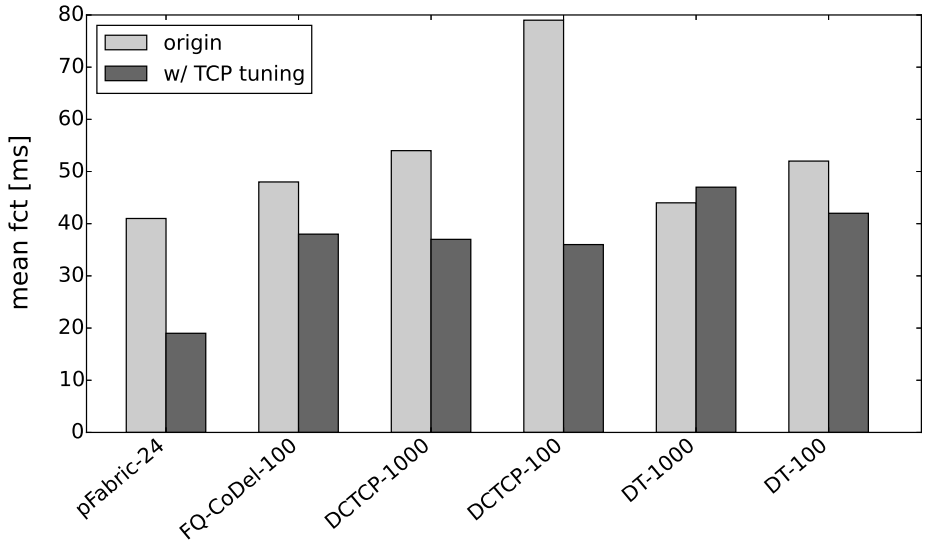


Figure 6.1 pFabric TCP parameters impact overview. (Each combination denoted as "AQM/scheduling - Buffer size")

a significant reduction of time-related parameters (such as delayed ack and retransmission timer [100]), and an increase of window-related parameters. The tuning of TCP minimum RTO (*minrto*) can have significant impact on network performance. The imbalance between *minrto* and data center latencies can result in poor performance for applications sensitive to millisecond delays in query response time [100]. Under severe packet loss, it is unacceptable to experience a timeout that lasts *minrto*. As long as we keep the minimum RTO larger than timer precision and delayed ack, reducing *minrto* can benefit both latency-sensitive flows and overall goodput of network.

To verify that pFabric settings do not play against DCTCP and FQ-CoDel, we compare the mean FCT difference before and after applying pFabric TCP parameters. In the following discussion, we use "AQM/scheduling - buffer size" format to represent the combinations considered. Fig. 6.1 clearly shows the reduce of the mean FCT for candidates (with combination of buffer size variation) considered. Moreover, the pFabric TCP settings reduce the mean FCT under pFabric by *a factor of more than 2* compared to that provided by pFabric used with vanilla TCP settings. We can conclude that applying pFabric TCP parameters is feasible, and benefit all candidates.

To further investigate the impact of TCP tuning in a more detailed manner, we activate features progressively and measure differences in latency (i.e., single packet FCT) and throughput. Results are shown in Fig. 6.2 showing that activating all parameters yields to about a 30% reduction of the mean FCT for DCTCP and 20% for FQ-CoDel. The breakdown

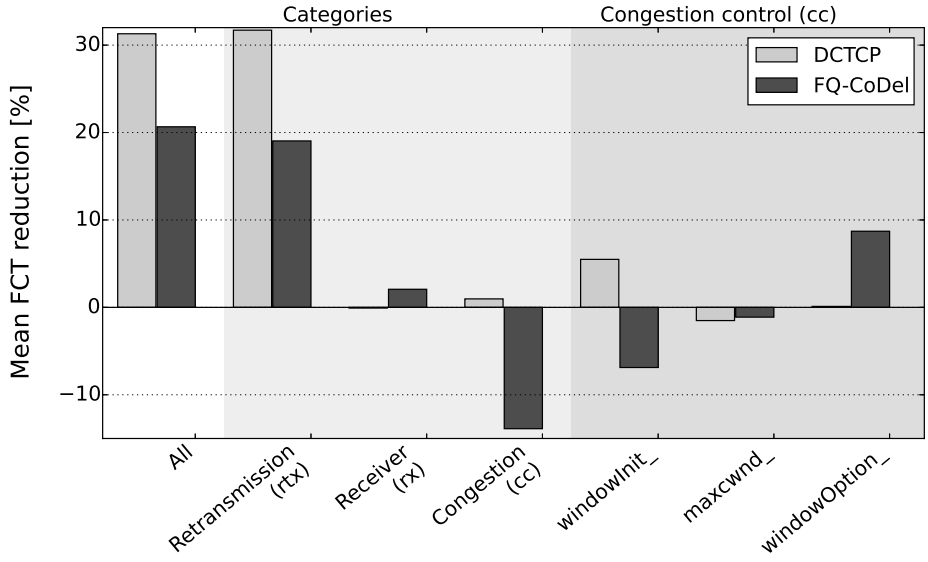


Figure 6.2 Applying pFabric TCP tuning to DCTCP and FQ-CoDel: overall, per category, and breakdown for the cc category impact w.r.t. default TCP settings.

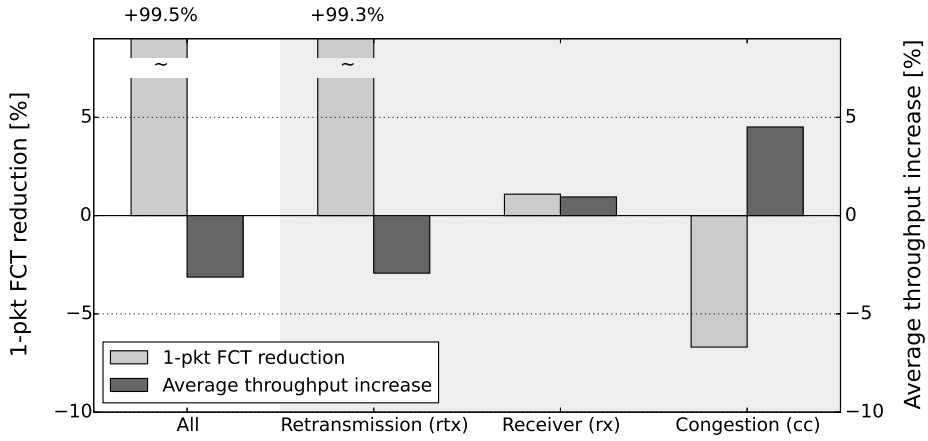


Figure 6.3 Breakdown of pFabric TCP parameters impact on DCTCP: per-category and per-flow size account.

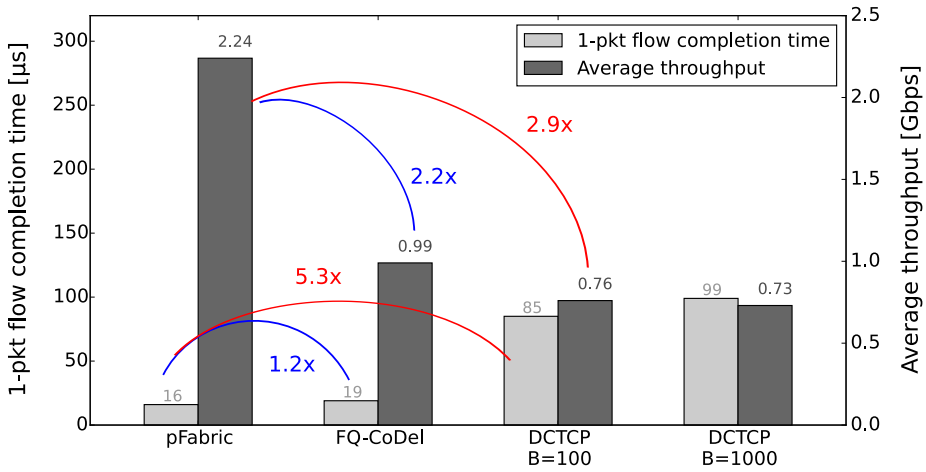


Figure 6.4 Performance at a glance. Original pFabric “data mining" scenario, load 0.6.

per retransmission(rtx)/receiver(tx)/congestion(cc) categories of Tab. 6.1 shows that time-related parameters play a paramount role, as expected. On the other hand, we see that congestion-related parameter have a limited impact on DCTCP and even a negative impact on FQ-CoDel, additionally it can be seen that the combined impact of the cc parameters is more than additive. The bottom line is that each candidate gains the most reduction in mean FCT when all parameters tuning is on. It follows that, by enabling pFabric parameters without any further tuning, our results are slightly more favorable for pFabric, and provide a conservative estimate of FQ-CoDel performance.

By further breaks down the rtx/tx/cc parameter impacts on the FCT vs throughput metrics for DCTCP in Fig. 6.3, we obtain a more precise understanding of its impact on different categories of flows. Here again it can be seen that whereas FCT of 1-packet flows (or equivalently latency of rate-limited flows) benefits of the combined settings, the throughput of long DCTCP suffers slightly due to a bulk adoption of pFabric settings adoption. Yet, selective parameter inclusion is hard, since performance metrics are differently affected from specific settings (e.g., cc parameters have a negative impact for short flows and positive impact for long flows.)

6.3 Simulation

Original data-mining scenario

To show differences at a glance, we start from the original pFabric scenario where 1-packet long flows represent a significant fraction (50%) of the flow volume but just account

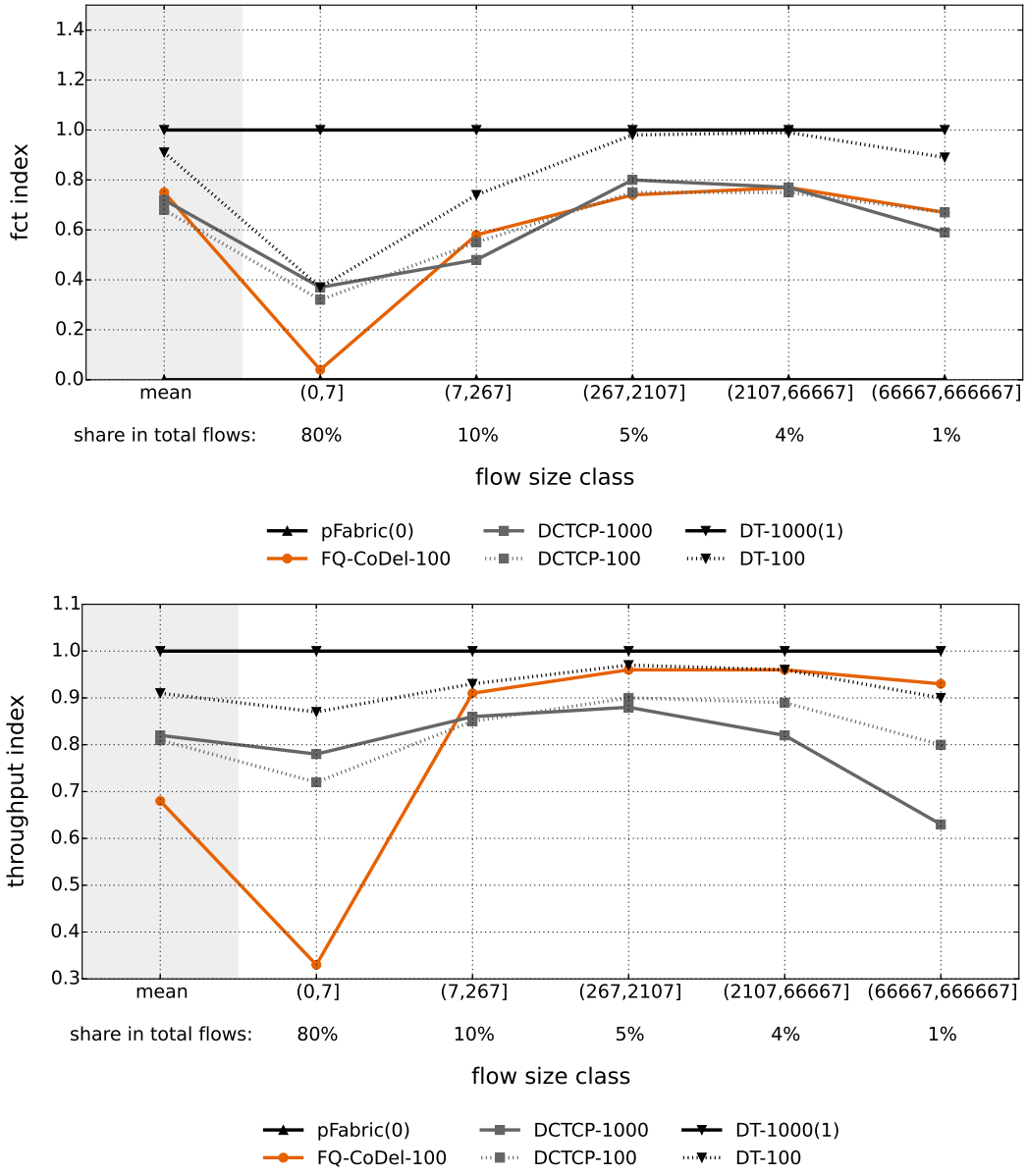


Figure 6.5 Breakdown of FCT and throughput by flow size categories. Original pFabric “data mining” scenario, load 0.6.

for a negligible portion of the byte-wise traffic volume (0.01%). Average throughput and completion time of 1-packet flows are reported in Fig. 6.4 showing that pFabric indeed exhibits outstanding performance for both metrics. Yet, it also shows that FQ-CoDel comes second: specifically, FCT for FQ-CoDel is very close to that of pFabric, while throughput for long flows is significantly smaller than pFabric but higher than that of DCTCP. Already in this scenario, fairness appears to be an interesting alternative, appealing due to both its simplicity, as well as its sufficient performance.

Fig. 6.5 shows a breakdown of flow size categories on both metrics. We apply feature scaling to bring all values into the range [0,1] for an intuitive comparison. Since pFabric and DropTail(with buffer size 1000 packets) represents the best and worst cases respectively in both metrics, in Fig. 6.5, a point closer to bottom (0, pFabric standard) represents a better performance, namely, lower FCT or higher throughput. In terms of FCT, FQ-CoDel claims a similar performance of DCTCP for most flow sizes, but largely improves that of the shortest flows in range $(0, 7]$ packets. This range indeed represents the rate-limited traffic, and confirms our analysis that fair scheduling schemes can benefit latency-sensitive flows a lot. From the observation of DropTail and DCTCP, we again see the impact of the short buffer has on latency improvement of short flows. And we can further conclude that either a control of queue length or packet sojourn time in the buffer can make a limited but not significant difference on backlogged flows. Throughput is basically calculated as the ratio of size in bits to FCT in seconds, thus a positive relation between FCT and throughput can be expected. As Fig. 6.5 shows, FQ-CoDel enjoys an impressive performance boost in short flow throughput. We notice that DCTCP-1000 provides extremely large flows better throughput thanks to the large buffer size.

Tweaked data-mining scenario

Robust DCN design should maintain their performance across scenarios: difference early shown in Fig. 6.4 has to be maintained, if not quantitatively, at least in order sense under a large spectrum of situations. We perform a very simple yet insightful sensitivity analysis of the pFabric, DCTCP and FQ-CoDel performance by increasing the volume of byte-wise traffic generated by 1-packet flows. As a reference, the 1-packet flow accounts for 0.01% volume-wise in total traffic of original scenario. For technical reasons tied to the simulation duration, we cap the maximum flow size to 10^5 packets – yet, as it can be seen the tweak is favorable to pFabric as the gap between pFabric and FQ-CoDel for 0.01% in Fig. 6.6 is larger than in Fig. 6.4. Nevertheless, it is easy to gather that the difference reduces significantly for increasing intensity of short or rate-limited flows: while that gap between

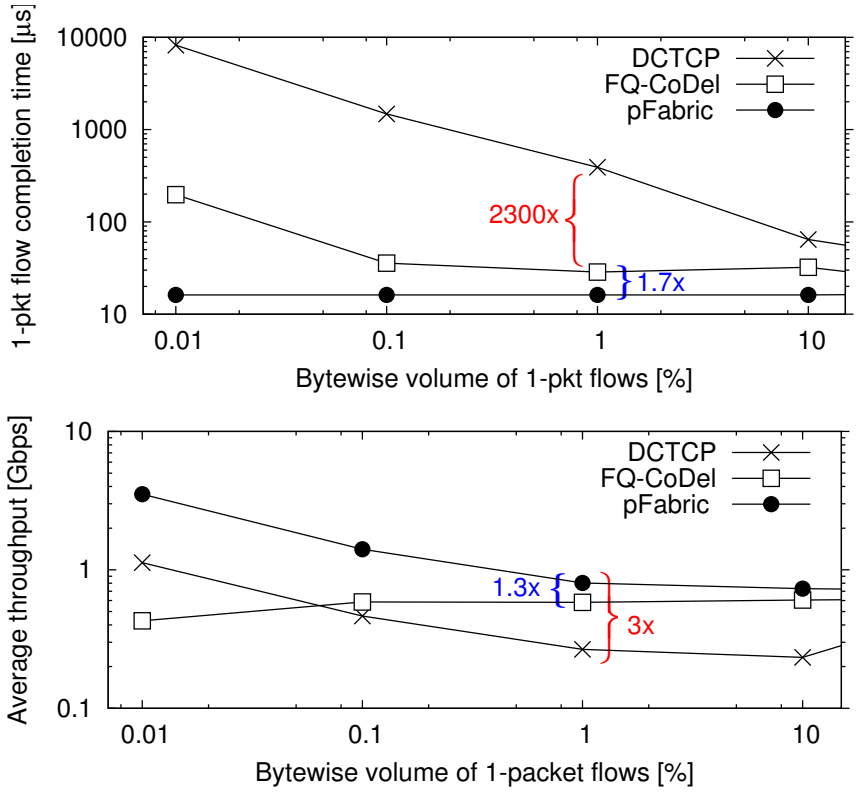


Figure 6.6 Impact of mice flow bytewise intensity: Tweaked pFabric scenario, load 0.6.

DCTCP and pFabric designs always remain remarkable, the performance of FQ-CoDel approaches considerably that of pFabric.

6.4 Summary

In this chapter, we study the impact of fair scheduling scheme in data center network. We pick a recent proposal named FQ-CoDel, which couples fair scheduling and explicit queue size control. Besides, DCTCP and pFabric has been chosen as the comparison benchmark. Both of them have already been widely studied in the community while the former is a layer-4 transport protocol and the latter is a clean slate cross-layer design.

We advise that performing careful calibration before an experiment campaign could ensure fair comparison among candidates. Our work shows that tuning in both AQM/scheduling schemes and end-to-end are necessary in response to specific requirement and network characteristics under study. Due to the high capacity low latency feature of DCN, time-related parameters tweak is of great importance, to name a few, timer precision, retransmission timeout, etc.

In light of our results, FQ-CoDel achieves comparable performance to pFabric in terms of short flow completion time, which exhibits a large improvement over DCTCP. Such feature ensures a relative priority of rate-limited flows (generated by interactive application, etc) over backlogged flows. Our further experiment shows that the performance gap between FQ-CoDel and pFabric becomes smaller when the bytewise volume share of short flows increases, which makes it more interesting for multi-tenant data center with high-level mixture of applications. We believe a fair statement is that FQ-CoDel achieves “good enough” performance while being “simple enough” to be highly appealing in practice. Yet, to date this simple yet effective approach has been neglected in DCN research for no obvious reason.

Chapter 7

Conclusion

7.1 Summary

In response to the fast development of new services over Internet in the latest decade, the quality of user experience can be mapped to network key performance indicators such as throughput and latency. Bearing in mind that the overall service performance depends on the performance of multiple network segments, in this thesis we investigated two network design challenges in two most important segments at the opposite edges of the end-to-end Internet path, namely, the end-user access network vs. the service provider data center network.

Access network

In the first part, we point out possible negative issues "reprioritization" arising from the interaction of AQM and CC techniques. Specifically, under AQM it is likely that LPCC techniques will become as aggressive as best-effort TCP.

We use a simulation campaign to quantify the generality of the reprioritization phenomenon under the largest possible set of scenarios. We include a number of representative AQM and LPCC techniques. Our results show that under all combinations considered this phenomenon occurs. The choice of a specific (LPCC,AQM) combination has only very limited impact on the system performance. A further sensitivity analysis shows that though the specific scenario settings may have an impact on fairness and queue size statistics, they have nevertheless only very limited impact on the reprioritization phenomenon. We conduct a more systematic experimental testbed campaign under controlled environment and wild internet with available AQM and LPCC in Linux kernel. The experiment result confirms the validity of reprioritization in the real world.

With the purpose of analyzing this phenomenon qualitatively, we have proposed a DDE model that captures the dynamics of an overall system composed of TCP flows, LEDBAT flows and an AQM control algorithm that is executed at the bottleneck queue. By analyzing dynamical properties of the system around its equilibrium points, we have been able to provide an explanation of the reprioritization phenomenon and have confirmed the generality of this issue. Moreover, model predictions have been validated against `ns2` simulations when RED is used as AQM algorithm.

Due to the increasing deployment of both low-priority congestion control and AQM techniques that today are employed to fight bufferbloat, the problem discussed in this thesis may be of significant practical relevance.

Data center network

In the second part, we review and compile a comprehensive taxonomy of data center network design, which becomes critical to the service quality and performance. Despite the active community development, we remark the absence of DCN proposals based on simple fair-scheduling strategies. We argue that fair scheduling could bring benefit to general-purpose data center without preassumption of workload mixture.

We pick FQ-CoDel, a deploy-ready scheme originally designed to fight bufferbloat in home routers of the access network, to evaluate the impact of fair scheduling scheme in data center networks. FQ-CoDel combines the scheduling mechanism providing fairness among flows, and Active Queue Management (AQM) to cap upper-bound delay. DCTCP and pFabric have been chosen as the other two comparison benchmarks. Both of them have already been widely studied in the community while the former is a layer-4 transport protocol and the latter is a clean slate cross-layer design.

With careful calibration of end-to-end transport protocol TCP, and parameter tuning of all comparison candidates to adapt them to a high throughput and low latency environment, our evaluation of FQ-CoDel in a DCN environment shows that it achieves better average throughput than end-to-end protocols tailored for DCN such as DCTCP. At the same time, the completion time of short flows approaches that of state-of-the-art DCN proposals such as pFabric: good enough performance and striking simplicity make FQ-CoDel a serious contender in the DCN arena.

We further adjust the bytewise volume share of short flows, the result shows a smaller performance gap between FQ-CoDel and pFabric. FQ-CoDel successfully exhibits the feature of ensuring a relative priority for rate-limited flows over backlogged flows. Therefore, we advocate a fair scheduling scheme coupled with AQM could benefit multi-tenant data center with high-level mixure of applications.

7.2 Future work

In this section, we present the open questions that remain unsolved in our research field, to be explored in future works.

Hybrid AQM techniques

In the first part, the AQMs evaluated either implement drop (RED/CoDel/CHOKe) or scheduling (SFQ) strategies. At the same time, hybrid AQM techniques that jointly exploit fair queuing with early drop are appearing, for instance, the FQ-CoDel [4] has already made its way into Linux kernels starting from 3.5 [101]. We didn't have the opportunity to include it in our analysis on access network due to the unavailable of FQ-CoDel at that time. Including this technique into evaluation could be interesting to complete our study of AQM vs. LPCC interaction, though we argue that the reprioritization problem will remain: intrinsically, AQM and scheduling aim at fairness, whereas LPCC aims at the contrasting objective of unfairness with respect to TCP.

Finer-grain priorities within LPCC

Another sensible question is whether it would be possible to differentiate priorities at a finer grain within the LPCC class: for instance, it is known that different targets in LEDBAT could yield to starvation in case of backlogged flows under a DropTail discipline [42]; at the same time, it is not clear what the behavior would be under AQM.

In-depth evaluation of FQ-CoDel

Although FQ-CoDel has been intensively tested before the deployment in home routers, the study of its application in DCN environment is rather limited. Our work serves a preliminary step into such field, and we believe that more in-depth evaluation is necessary in the future. As in the calibration, we choose the same parameters for all schemes considered. They indeed give an advantage to pFabric in our case. It would be interesting to apply heterogeneous parameter calibration to evaluate the full potential benefits FQ-CoDel could bring to our scenario. Furthermore, we observe from Fig. 6.6 a reversed trend of FQ-CoDel compared to pFabric and DCTCP. It could be important to carry out the additional study to reveal the potential difference behind these schemes.

DCN design study on real deployment

In the second part, our evaluation essentially raises awareness in the community about the suitability of adopting fair scheduling scheme, however much remain to be done. To start with, FQ-CoDel should be compared in a real deployment against DCTCP to prove that benefits are immediately and painlessly achievable in today DCNs. A broader understanding of further key ingredients (e.g., packet spraying; co-flow identification; etc.) that play nicely along FQ-CoDel in a simple yet successful DCN design constitute a logical next step of our agenda.

Flow correlation impact

The workload we used in our simulation derives from the widely-used workload originated from [16] and [86], with simple flow/packet arrival process and size distribution. Considering the fact that they are statistics of two traces from a specific data center at a specific time, there remains two major concerns 1) the appropriateness of using them in diverse designs is questionable; 2) the lack of both time and space correlation information among flows. While the community is still far from being able to model realistic traffic in a practical way, we can indeed add a part of missing information, reflecting the flow correlation to a reasonable extent. Recent proposals like Varys [19], RAPIER [93] and Baraat [76] already consider completion time in terms of co-flow and task. We believe a more systematic design of workload should benefit the entire community.

Bibliography

[1] Kalevi Kilkki. Quality of experience in communications ecosystem. *j-jucs*, 14(5):615–624, mar 2008. http://www.jucs.org/jucs_14_5/quality_of_experience_in.

[2] CACM Staff. BufferBloat: What's Wrong with the Internet? *Communications of the ACM*, 55(2):40–47, 2012.

[3] H. Jiang, Y. Wang, K. Lee, and I. Rhee. Tackling bufferbloat in 3G/4G networks. In *ACM IMC*, 2012.

[4] Kathleen Nichols and Van Jacobson. Controlling Queue Delay. *Communications of the ACM*, pages 42–50, 2012.

[5] Christian Kreibich, Nicholas Weaver, Boris Nechaev, and Vern Paxson. Netalyzr: Illuminating the Edge Network. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, pages 246–259, 2010.

[6] Paul E. Mckenney. Stochastic Fairness Queueing. In *IEEE INFOCOM*, pages 733–740, 1990.

[7] M. Shreedhar and George Varghese. Efficient Fair Queueing Using Deficit Round Robin. *ACM SIGCOMM Computer Communication Review (CCR)*, 25:231–242, 1995.

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

[9] freemat. ADUF - Historique firmware. http://88.191.250.12/viewtopic.php?t=164746&view=previous. 2008-01-20.

[10] G. White. Active queue management in docsis 3.x cable modems. Technical report, CableLabs, may 2014.

[11] Peter B. Key, Laurent Massoulié, and Bing Wang. Emulating Low-priority Transport at the Application Layer: a Background Transfer Service. In *ACM SIGMETRICS/Performance*, pages 118–129, 2004.

[12] Aleksandar Kuzmanovic and Edward W. Knightly. TCP-LP: A Distributed Algorithm for Low Priority Data Transfer. In *IEEE INFOCOM*, volume 3, pages 1691–1701, 2003.

[13] Arun Venkataramani, Ravi Kokku, and Michael Dahlin. TCP Nice: A Mechanism for Background Transfers. *ACM SIGOPS Operating Systems Review*, pages 329–343, 2002.

[14] Mirja Kuehlewind, Greg Hazel, Stanislav Shalunov, and Janardhan Iyengar. *Low Extra Delay Background Transport (LEDBAT)*. Internet Engineering Task Force (IETF), 2012. RFC6817.

[15] Sandvine. Global Internet Phenomena Report (1H 2013). https://www.sandvine.com/downloads/general/global-internet-phenomena/2013/sandvine-global-internet-phenomena-report-1h-2013.pdf, 2013. 2013-12-05.

[16] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center tcp (dctcp). In *ACM SIGCOMM*, 2010.

[17] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. pfabric: Minimal near-optimal datacenter transport. In *ACM SIGCOMM*, 2013.

[18] Mosharaf Chowdhury, Matei Zaharia, Justin Ma, Michael I. Jordan, and Ion Stoica. Managing data transfers in computer clusters with orchestra. In *ACM SIGCOMM*, 2011.

[19] Mosharaf Chowdhury, Yuan Zhong, and Ion Stoica. Efficient coflow scheduling with varys. In *ACM SIGCOMM*, 2014.

[20] Mohammad Alizadeh, Abdul Kabbani, Tom Edsall, Balaji Prabhakar, Amin Vahdat, and Masato Yasuda. Less is more: Trading a little bandwidth for ultra-low latency in the data center. In *USENIX NSDI*, 2012.

[21] Giovanna Carofiglio, Luca Muscariello, Dario Rossi, Claudio Testa, and Silvio Valenti. Rethinking Low Extra Delay Background Transport Protocols. *Elsevier Computer Networks*, 57:1838–1852, 2013.

[22] Bram Cohen. How has BitTorrent as a protocol evolved over time. http://www.quora.com/BitTorrent-protocol-company/How-has-BitTorrent-as-a-protocol-evolved-over-time, 2011. 2013-12-05.

[23] Dario Rossi, Claudio Testa, Silvio Valenti, and Luca Muscariello. LEDBAT: the New BitTorrent Congestion Control Protocol. In *International Conference on Computer Communications and Networks (ICCCN)*, pages 1–6, 2010.

[24] Joscha Schneider, Joerg Wagner, Rolf Winter, and Hans-Joerg J. Kolbe. Out of My Way-Evaluating Low Extra Delay Background Transport in an ADSL Access Network. In *International Teletraffic Congress (ITC)*, pages 1–8, 2010.

[25] Jim Gettys and Kathleen Nichols. Bufferbloat: Dark buffers in the Internet. *Communications of the ACM*, 55(1):57–65, 2012.

[26] Rong Pan, Balaji Prabhakar, and Konstantinos Psounis. CHOKe, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation. In *IEEE INFOCOM*, pages 942–951, 2000.

[27] Mikkel Christiansen, Kevin Jeffay, David Ott, and F. Donelson Smith. Tuning RED for Web Traffic. *ACM SIGCOMM Computer Communication Review (CCR)*, 30:139–150, 2000.

[28] Nabil Benameur, Fabrice Guillemin, and Luca Muscariello. Latency reduction in home access gateways with shortest queue first. In *Proc. ISOC Workshop on Reducing Internet Latency*, 2013.

[29] Addisu Eshete and Yuming Jiang. Approximate Fairness through Limited Flow List. In *International Teletraffic Congress (ITC)*, pages 198–205, 2011.

[30] Jong hwan Kim, Hyunsoo Yoon, and Ikjun Yeom. Active Queue Management for Flow Fairness and Stable Queue Length. *IEEE Transactions on Parallel and Distributed Systems*, 22(4):571–579, 2011.

[31] Dinil Mon Divakaran. A Spike-detecting AQM to Deal with Elephants. *Elsevier Computer Networks*, 56(13):3087–3098, 2012.

[32] Addisu Eshete, Yuming Jiang, and Lars Landmark. Fairness among High Speed and Traditional TCP under Different Queue Management Mechanisms. In *Proceedings of the ACM SIGCOMM Asian Internet Engineeering Conference (AINTEC)*, pages 39–46, 2012.

[33] Wesley Eddy. [aqm] floating a draft charter. http://www.ietf.org/mail-archive/web/aqm/current/msg00127.html, 2013. 2013-12-05.

[34] Martin May, Jean Bolot, Christophe Diot, and Bryan Lyles. Reasons Not to Deploy RED. In *International Workshop on Quality of Service (IWQoS)*, pages 260–262, 1999.

[35] Cisco. Class-based weighted fair queueing and weighted random early detection. 2014-06-18.

[36] Sally Floyd, Ramakrishna Gummadi, Scott Shenker, et al. Adaptive red: An algorithm for increasing the robustness of red's active queue management.

[37] Changwang Zhang, Jianping Yin, Zhiping Cai, and Weifeng Chen. Rred: robust red algorithm to counter low-rate denial-of-service attacks. *Communications Letters, IEEE*, 14(5):489–491, 2010.

[38] Dave Taht. CoDel Overview. http://www.bufferbloat.net/projects/codel/wiki. 2015-10-20.

[39] Shao Liu, Milan Vojnovic, and Dinan Gunawardena. Competitive and considerate congestion control for bulk data transfers. In *IEEE International Workshop on Quality of Service (IWQoS)*, pages 1–9, 2007.

[40] David Ros and Michael Welzl. Less-than-Best-Effort Service: A Survey of End-to-End Approaches. *Communications Surveys & Tutorials*, 15(2):898–908, 2013.

[41] Si Quoc Viet Trang, N. Kuhn, E. Lochin, C. Baudoin, E. Dubois, and P. Gelard. On the existence of optimal ledbat parameters. In *IEEE International Conference on Communications (ICC)*, pages 1216–1221, June 2014.

[42] Giovanna Carofiglio, Luca Muscariello, Dario Rossi, and Claudio Testa. A Hands-on Assessment of Transport Protocols with Lower than Best Effort Priority. In *IEEE Conference on Local Computer Networks (LCN)*, pages 8–15, 2010.

[43] Luigi A. Grieco and Saverio Mascolo. Performance Evaluation and Comparison of Westwood+, New Reno, and Vegas TCP Congestion Control. *ACM SIGCOMM Computer Communication Review (CCR)*, 34(2):25–38, April 2004.

[44] Sally Floyd. Connections with Multiple Congested Gateways in Packet-switched Networks Part 1: One-way Traffic. *ACM SIGCOMM Computer Communication Review (CCR)*, 21(5):30–47, 1991.

[45] Giovanna Carofiglio, Luca Muscariello, Dario Rossi, and Silvio Valenti. The Quest for LEDBAT Fairness. In *IEEE GLOBECOM*, pages 1–6, 2010.

[46] Jong Suk Ahn, Peter B. Danzig, Zhen Liu, and Limin Yan. Evaluation of TCP Vegas: Emulation and Experiment. In *ACM SIGCOMM*, pages 185–205, 1995.

[47] Vishal Misra, Wei-Bo Gong, and Don Towsley. Fluid-based analysis of a network of aqm routers supporting tcp flows with an application to red. In *ACM SIGCOMM Computer Communication Review (CCR)*, pages 151–160, 2000.

[48] CV Hollot, Vishal Misra, Don Towsley, and Wei-Bo Gong. A Control Theoretic Analysis of RED. In *IEEE INFOCOM*, volume 3, pages 1510–1519, 2001.

[49] Yong Liu, Francesco Lo Presti, Vishal Misra, Don Towsley, and Yu Gu. Fluid Models and Solutions for Large-scale IP Networks. *ACM SIGMETRICS Performance Evaluation Review*, 31(1):91–101, 2003.

[50] Marco Ajmone Marsan, Michele Garetto, Paolo Giaccone, Emilio Leonardi, Enrico Schiattarella, and Alessandro Tarello. Using Partial Differential Equations to Model TCP Mice and Elephants in Large IP Networks. *IEEE/ACM Transactions on Networking*, 13(6):1289–1301, 2005.

[51] Wim Michiels, D Melchor-Aguilar, and S-I Niculescu. Stability analysis of some classes of TCP/AQM networks. *International Journal of Control*, 79(9):1136–1144, 2006.

[52] S. Mascolo. Congestion control in high-speed communication networks using the Smith principle. *Special Issue on "Control methods for communication networks" Automatica*, 35(12):1921–1935, 1999.

[53] Luca De Cicco, Saverio Mascolo, and Silviu-Iulian Niculescu. Robust stability analysis of smith predictor-based congestion control algorithms for computer networks. *Automatica*, 47(8):1685–1692, 2011.

[54] http://www.enst.fr/~drossi/ledbat.

[55] The uTorrent Transport Protocol library. http://github.com/bittorrent/libutp, 2010.

[56] Dario Rossi, Claudio Testa, and Silvio Valenti. Yes, we ledbat: Playing with the new bittorrent congestion control algorithm. In *Passive and Active Measurements (PAM)*, pages 31–40, 2010.

[57] http://www.infres.enst.fr/~drossi/index.php?n=Dataset.LEDBATAQM.

[58] ITU-T Study Group 12. *One-way transmission time*. International Telecommunication Union, 2003. ITU G.114.

[59] YiXi Gong, Dario Rossi, and Emilio Leonardi. Modeling the Interdependency of Low-priority Congestion Control and Active Queue Management. In *International Teletraffic Congress (ITC)*, pages 1–9, 2013.

[60] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. Reproducible Network Experiments Using Container-based Emulation. In *ACM CoNEXT*, pages 253–264, 2012.

[61] Justin Costa-Roberts and Carl Case. Seeing RED. http://reproducingnetworkresearch.wordpress.com/2012/06/05/seeing-red, 2012. 2013-12-05.

[62] Yixi Gong, Dario Rossi, Claudio Testa, Silvio Valenti, and Dave Taht. Fighting the Bufferbloat: on the Coexistence of AQM and Low Priority Congestion Control. In *IEEE INFOCOM Workshop on Traffic Measurement and Analysis (TMA)*, pages 3291–3296, 2013.

[63] Y. Gong, D. Rossi, C. Testa, S. Valenti, and M.D. Taht. Fighting the bufferbloat: On the coexistence of aqm and low priority congestion control. *Elsevier Computer Networks*, 65:255 – 267, 2014.

[64] Bram Cohen and Arvid Norberg. Correcting for Clock Drift in uTP and LEDBAT. In *International Workshop on Peer-to-Peer Systems (IPTPS), USENIX*, 2010.

[65] S.-I. Niculescu and W. Michiels. *Stability and stabilization of time-delay systems. An eigenvalue-based approach*. SIAM: Philadelphia, USA, 2007.

[66] Wim Michiels. Spectrum-based stability analysis and stabilisation of systems described by delay differential algebraic equations. *IET control theory & applications*, 5(16):1829–1842, 2011.

[67] Brian D Hassard, Nicholas D Kazarinoff, and Yieh-Hei Wan. *Theory and applications of Hopf bifurcation*, volume 41. CUP Archive, 1981.

[68] Nandita Dukkipati. *Rate Control Protocol (Rcp): Congestion Control to Make Flows Complete Quickly*. PhD thesis, Stanford University, 2008.

[69] Chi-Yao Hong, Matthew Caesar, and P. Brighten Godfrey. Finishing flows quickly with preemptive scheduling. In *ACM SIGCOMM*, 2012.

[70] A. Munir, I.A. Qazi, Z.A. Uzmi, A. Mushtaq, S.N. Ismail, M.S. Iqbal, and B. Khan. Minimizing flow completion times in data centers. In *IEEE INFOCOM*, 2013.

[71] Hong Xu and Baochun Li. Repflow: Minimizing flow completion times with replicated flows in data centers. In *IEEE INFOCOM*, 2014.

[72] Christo Wilson, Hitesh Ballani, Thomas Karagiannis, and Ant Rowtron. Better never than late: Meeting deadlines in datacenter networks. In *ACM SIGCOMM*, 2011.

[73] David Zats, Tathagata Das, Prashanth Mohan, Dhruba Borthakur, and Randy Katz. Detail: Reducing the flow completion time tail in datacenter networks. In *ACM SIGCOMM*, 2012.

[74] Ali Munir, Ghufran Baig, Syed M. Irteza, Ihsan A. Qazi, Alex X. Liu, and Fahad R. Dogar. Friends, not foes: Synthesizing existing transport strategies for data center networks. In *ACM SIGCOMM*, 2014.

[75] Wei Bai, Li Chen, Kai Chen, Dongsu Han, Chen Tian, and Weicheng Sun. Pias: Practical information-agnostic flow scheduling for data center networks. In *ACM HotNets*, 2014.

[76] Fahad R. Dogar, Thomas Karagiannis, Hitesh Ballani, and Antony Rowstron. Decentralized task-aware scheduling for data center networks. In *ACM SIGCOMM*, 2014.

[77] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. Conga: Distributed congestion-aware load balancing for datacenters. In *ACM SIGCOMM*, 2014.

[78] Gartner. http://www.gartner.com/newsroom/id/2614915.

[79] D. Patterson. The trouble with multi-core. *Spectrum, IEEE*, 2010.

[80] Victor Bahl. Cloud 2020: The Emergence of Micro Datacenters (cloudlets) for Mobile Computing, 2015.

[81] T. Hoeiland-Joergensen, D. Taht, J. Gettys, and E. Dumazet. Flowqueue-codel. Internet-draft, https://www.ietf.org/archive/id/draft-hoeiland-joergensen-aqm-fq-codel-01.txt, IETF, 2014.

[82] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In *ACM SIGCOMM*, 2008.

[83] Chuanxiong Guo, Haitao Wu, Kun Tan, Lei Shi, Yongguang Zhang, and Songwu Lu. Dcell: A scalable and fault-tolerant network structure for data centers. In *ACM SIGCOMM*, 2008.

[84] Radhika Niranjan Mysore, Andreas Pamboris, Nathan Farrington, Nelson Huang, Pardis Miri, Sivasankar Radhakrishnan, Vikram Subramanya, and Amin Vahdat. Portland: A scalable fault-tolerant layer 2 data center network fabric. In *ACM SIGCOMM*, 2009.

[85] Chuanxiong Guo, Guohan Lu, Dan Li, Haitao Wu, Xuan Zhang, Yunfeng Shi, Chen Tian, Yongguang Zhang, and Songwu Lu. Bcube: A high performance, server-centric network architecture for modular data centers. In *ACM SIGCOMM*, 2009.

[86] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: A scalable and flexible data center network. In *ACM SIGCOMM*, 2009.

[87] Haitao Wu, Zhenqian Feng, Chuanxiong Guo, and Yongguang Zhang. Ictcp: Incast congestion control for tcp in data center networks. In *ACM CoNEXT*, 2010.

[88] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, Nelson Huang, and Amin Vahdat. Hedera: Dynamic flow scheduling for data center networks. In *USENIX NSDI*, 2010.

[89] Costin Raiciu, Sebastien Barre, Christopher Pluntke, Adam Greenhalgh, Damon Wischik, and Mark Handley. Improving datacenter performance and robustness with multipath tcp. In *ACM SIGCOMM*, 2011.

[90] Daniel Crisan, Robert Birke, Gilles Cressier, Cyriel Minkenberg, and Mitch Gusat. Got loss? get zovn! In *ACM SIGCOMM*, 2013.

[91] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, and Hans Fugal. Fastpass: A centralized "zero-queue" datacenter network. In *ACM SIGCOMM*, 2014.

[92] Abdul Kabbani, Balajee Vamanan, Jahangir Hasan, and Fabien Duchene. Flowbender: Flow-level adaptive routing for improved latency and throughput in datacenter networks. In *ACM CoNEXT*, 2014.

[93] Zhao Uestc, Kai Chen, Wei Bai, Minlan Yu Usc, Chen Tian Hust, Yanhui Geng Huawei, Yiming Zhang Nudt, Dan Li Tsinghua, and Sheng Wang Uestc. RAPIER : Integrating Routing and Scheduling for Coflow-aware Data Center Networks. *IEEE INFOCOM*, 2015.

[94] Glenn Judd. Attaining the promise and avoiding the pitfalls of tcp in the datacenter. In *NSDI*, 2015.

[95] John Nagle. Congestion control in ip/tcp internetworks. Rfc, IETF, 1984.

[96] A. Kortebi, L. Muscariello, S. Oueslati, and J. Roberts. Evaluating the number of active flows in a scheduler realizing fair statistical bandwidth sharing. In *ACM SIGMETRICS*, 2005.

[97] B. Suter, T. V. Lakshman, D. Stiliadis, and A. K. Choudhury. Buffer management schemes for supporting tcp in gigabit routers with per-flow queueing. *IEEE JSAC*, 2006.

[98] Maaike Verloop, Sem Borst, and Rudesindo Núñez-Queija. Stability of size-based scheduling disciplines in resource-sharing networks. *Performance Evaluation*, 2005.

[99] Balajee Vamanan, Jahangir Hasan, and T.N. Vijaykumar. Deadline-aware datacenter tcp (d2tcp). In *ACM SIGCOMM*, 2012.

[100] Vijay Vasudevan, Amar Phanishayee, Hiral Shah, Elie Krevat, David G. Andersen, Gregory R. Ganger, Garth A. Gibson, and Brian Mueller. Safe and effective fine-grained tcp retransmissions for datacenter communication. In *ACM SIGCOMM*, 2009.

[101] CoDel AQM discussions. The Codel Archives. https://lists.bufferbloat.net/pipermail/codel. 2013-12-05.

List of Publications

Workshops

[W1] Yixi Gong, Dario Rossi, Claudio Testa, Silvio Valenti, and Dave Taht. Interaction or Interference: can AQM and Low Priority Congestion Control Successfully Collaborate. In *ACM CoNEXT'12 Student Workshop*, 2012.

[W2] Yixi Gong, Dario Rossi, Claudio Testa, Silvio Valenti, and Dave Taht. Fighting the Bufferbloat: on the Coexistence of AQM and Low Priority Congestion Control. In *IEEE INFOCOM Workshop on Traffic Measurement and Analysis (TMA'13)*, pages 3291–3296, 2013.

Journal and International Proceedings

[P1] YiXi Gong, Dario Rossi, and Emilio Leonardi. Modeling the Interdependency of Low-priority Congestion Control and Active Queue Management. In *International Teletraffic Congress (ITC)*, pages 1–9, 2013. **Best paper award runner-up.**

[J1] Yixi Gong, Dario Rossi, Claudio Testa, Silvio Valenti, and Dave Taht. Fighting the bufferbloat: On the Coexistence of AQM and Low Priority Congestion Control. *Elsevier Computer Networks*, 65:255 – 267, 2014.

Submitted

[S1] Luca De Cicco, Yixi Gong, Emilio Leonardi, and Dario Rossi. A Control Theoretic Tale of Wwo Species: Reprioritization of Low-priority Congestion Control under AQM. *SUBMITTED to ACM Transactions on Modeling and Performance Evaluation of Computing Systems (ACM ToMPECS). (April 2015).*

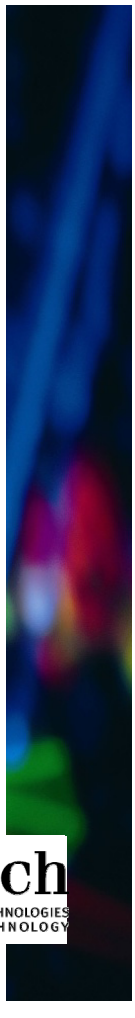
[S2] Yixi Gong, Dario Rossi, and Jame Roberts. None but the Data Centers deserve the Fair. *SUBMITTED to Computer Communication Review (CCR). (February 2016).*

La quête de faible latence sur les deux bords du réseau : conception, d'analyse, de simulation et expériences

Yixi GONG

RESUME : Au cours de ces dernières années, les services Internet croissent considérablement ce qui crée beaucoup de nouveaux défis dans des scénarios variés. La performance globale du service dépend à son tour de la performance des multiples segments de réseau. Nous étudions deux défis représentatifs de conception dans différents segments : les deux les plus importants se trouvent sur les bords opposés la connectivité de bout en bout des chemins d'Internet, notamment, le réseau d'accès pour l' utilisateur et le réseau de centre de données du fournisseur de services.

ABSTRACT : In the recent years, the innovation of new services over Internet is considerably growing at a fast speed, which brings forward lots of new challenges under varied scenarios. The overall service performance depends in turn on the performance of multiple network segments. We investigated two representative design challenges in different segments : the two most important sit at the opposite edges of the end-to-end Internet path, namely, the end-user access network vs. the service provider data center network.



TELECOM ParisTech

EDiTE

ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY