OPTIMAL UTILITY-BASED TRAFFIC CONTROL FOR DATACENTER NETWORKS

by

AKSHIT SINGHAL

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON
May 2022

OPTIMAL UTILITY-BASED TRAFFIC CONTROL FOR DATACENTER NETWORKS

The members of the Committee approve the doctoral dissertation of AKSHIT SINGHAL

Dr. Hao Che
Supervising Professor _____

Dr. Hong Jiang _____

Dr. Bahram Khalili _____

Dr. Song Jiang _____

Dean of the Graduate School _____

To my parents, Pramila Gupta and Dr. Lokesh Gupta, and my sister, Nupur Aggarwal, who supported me during my studies.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor, Dr. Hao Che, for motivating and inspiring me and for his vital advice during the course of my doctoral studies. It has been a great pleasure to work with him on research projects. I have learned a lot through useful discussions with him during my research.

I would also like to thank Dr. Hong Jiang for his continuous guidance. This work would not have been possible without his wisdom, support and encouragement.

I would also like to express my gratitude to Dr. Bahram Khalili and Dr. Song Jiang for their interest in my research and for taking time to serve on my dissertation committee.

I would also like to extend my appreciation to Dr. Bahram Khalili and Ramez Elmasri for providing me the opportunity of becoming a Graduate Teaching Assitant and Adjunct Faculty which provided the financial support for my doctoral studies.

I wish to thank our entire ACES group including Dr. Zhijun Wang, Dr. Ning Li, Prathyusha Enganti, Todd Rosenkrantz, and Xuan Wang for their interest in my research and the helpful discussions and invaluable comments. I am grateful to all the professors who taught me during the years I spent in school, first in India and in the United States. I would like to thank Ashok Kumar and Rahul Majethia for encouraging and directing me to pursue graduate studies. Finally, I am also extremely grateful to my family for their sacrifice, encouragement, and patience. I also thank my friends who have helped me throughout my career.

March 28, 2022

ABSTRACT

OPTIMAL UTILITY-BASED TRAFFIC CONTROL FOR DATACENTER NETWORKS

AKSHIT SINGHAL, Ph.D.

The University of Texas at Arlington, 2022

Supervising Professor: Dr. Hao Che

As datacenter applications with diverse service requirements proliferate, it becomes imperative to enable datacenter network flow rate allocation that satisfies minimum user-utility requirements, while allowing for user-utility-based fair resource allocation. Multiple Path Transmission Control Protocols (MPTCPs) allow flows to explore path diversity of data center networks and multihoming to improve throughput, reliability, and network resource utilization. The work in this dissertation aims to develop optimal utility-based datacenter traffic control protocols to meet diverse service requirements for datacenter applications.

Specifically, this dissertation makes contributions to four highly related research topics. First, we put forward a HOListic traffic control framework for datacenter NETworks (HOLNET) that reformulates the network utility maximization (NUM) framework into a HOLNET NUM framework that fully harnesses the potential of the existing NUM-based solutions to allow large families of traffic control protocols of various degrees of sophistication to be developed, i.e., host-based, single, or multiple Class-of-Service (CoS) enabled, single or multi-path congestion control, with or with-

out in-network load balancing. Case studies, based on both single and multi-path host-based solutions, demonstrate the viability and flexibility in HOLNET design space exploration. To further test the backward compatibility and performance with respect to some existing lightweight solutions, we develop HOLNET-UTA, an integrated congestion control and load balancing protocol, achieving TCP-fair resource allocation. HOLNET-UTA is found by the simulation to improve the average flow completion time (FCT) by more than 20%, compared to DRILL [1] with DCTCP [2].

Second, we derive and implement in Linux kernels two distinct families of NUM-optimal MPTCP protocols, EUTCP($\boldsymbol{\gamma}$), a function of a rate-scaling vector of the sub-flow rate-scaling coefficients, $\boldsymbol{\gamma}$, and WUTCP($\boldsymbol{\omega}$), a function of a utility weight vector of the sub-flow weights, $\boldsymbol{\omega}$, respectively. While the former allows resource pooling, the latter does not. The performance of the two families with equal weight and equal rate-scaling coefficient for all sub-flows when coexisting with TCP is also analyzed based on experiments in a testbed.

Third, we propose a Hybrid MPTCP (H-MPTCP) with a built-in mechanism to incentivize users to use multiple paths with different pricing structures. In the meantime, H-MPTCP preserves the nice properties enjoyed by the state-of-the-art MPTCP solutions. Extensive real Linux implementation results verify that H-MPTCP can indeed achieve the design objectives.

Finally, we revisit a state-of-the-art datacenter traffic control protocol, known as datacenter TCP (DCTP) [2]. DCTCP is empirically designed and hence, does not have a specific global design objective in mind. In this research, we prove that with a simple modification of the congestion indicator, DCTCP can be turned into a NUM-optimal traffic control protocol.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In this chapter, we first introduce the existing traffic control design space for datacenter networks in section 1.1. Then we highlight the challenges facing the existing solutions in section 1.2. Our research contributions to tackle those challenges are given in section 1.3, followed by section 1.4 which outlines the remaining chapters.

1.1 Datacenter Traffic Control Design Space

Traffic control in today's loosely controlled public Internet has been largely limited to end-to-end TCP congestion control [3] and static, equal-cost-multipath (ECMP) load balancing [4]. In contrast, in a well-controlled environment like a datacenter network, more sophisticated traffic control solutions become viable. This has led to the proliferation of a wide spectrum of congestion control solutions for datacenter networks in recent years, ranging from congestion control protocols with minimum involvement of in-network nodes, like DCTCP [2] and D2TCP [5] based on Explicit Congestion Notification(ECN), all the way to those using the link load information for the control, e.g., FCP [6], or even clean-slate solutions requiring network architectural redesign, e.g., NUMFabric [7]. Meanwhile, various dynamic load balancing solutions have emerged, exploring path diversity to further improve datacenter network resource utilization. Again, the input for the control ranges from purely local, e.g., DRILL [1], all the way to the path load, e.g., CONGA [8].

The existing traffic control solutions for datacenter networks can be broadly classified into two categories, i.e., host-based transport congestion control and host-

based multipath or in-network load balancing. A partial list of the existing solutions (including some earlier ones which are not datacenter specific) is given in Table 1.1. For each solution listed, the performance target(s) and required input information for the control is also given.

First, we note that the listed solutions in Table 1.1 are point by design with respect to the following aspects:

1. Except for some host-based solutions, such as MPTCP [9] and NDP [10], congestion control and load balancing are developed independently.
2. The required input for the control can be quite different from one solution to another, ranging from purely local, all the way to path-utilization-based ones.
3. The performance targets are also quite diverse, e.g., average flow completion time (FCT), throughput (TP), flow deadline (FD), Network Utility Maximization(NUM)[1] and utility min-max.

Moreover, some solutions, e.g., pFabric [11], PIAS [12], and D3[13], are clean-slate and require architectural redesign. Clearly, it is difficult to combine such solutions to fully explore performance, scalability, and design complexity tradeoffs and to allow adaptation to software/hardware capability changes for datacenter networks at large. Moreover, such solutions, when coexisting with one another, may adversely interact with one another, leading to suboptimal, unexpected performance, or even network instability [8, 14, 15, 16, 17, 18].

Second, most solutions listed in 1.1 are empirical by design. Although some solutions are NUM-based, e.g., [6], [7], [19] they are again point solutions with lim-

[1]NUM is a widely recognized network optimization framework that aims at maximizing the sum of user utilities as a function of user flow rates, subject to the network resource constraints, where the user utilities are meant to capture the QoE of the users. Please refer to Eq. (2.1) for the exact definition.

ited scope, applicable to elastic traffic only, and deal with either congestion control or load balancing, but not both.

Table 1.1: Existing Design Space

Solution	Transport	Host-based LB	In-network LB	Input Informtation	Performance Target (QoS)
DCTCP [2]	√	×	×	ECN	FCT&TP
FCP [6]	√	×	×	budget/link price	NUM
DX [20]	√	×	×	latency	queuing Delay
HULL [21]/DCTCP	√	×	×	ECN/QL	mean & tail PL
D^2TCP [5]	√	×	×	ECN/FDL	FD (√)
pFabric [11]	√	×	×	FD	FCT (√)
ExpressPass [22]	√	×	×	credit-based signaling	FCT (√)
pHost [23]	√	×	spraying	token	FCT, TP
NUMFabric[7]	√	×	×	Weights	NUM
RC3[24]	√	×	×	double loop	FCT
XCP[25]	√	×	×	ECN	TP
PIAS[12]	DCTCP/TCP	×	×	AQM	FCT
PASE [26]	DCTCP/D^2TCP	×	×	FD/AQM	FCT (√)
L^2DCT [27]	√	×	×	ECN	FCT
DCQCN [28]/PFC [29, 30]	√	×	×	ECN, Pause	PL
TIMELY [31]/PFC (opt)	√	×	×	RTT, Pause (opt)	packet TL/TP
Karuna [14]/DCTCP	√	×	×	ECN/RTT	FD/FCT (√)
D3 [13]	√	×	ECMP	min-rate	FD (√)
MPTCP [9]	√	√	×	source inferred	TP
NDP[10]	√	√	×	Packet trimming	FCT/TP
PDQ[32]/RCP[33]	√	√	ECMP	FD,Pause	FCT,FD (√)
Presto [34]	×	√	×	path weights	TP
DRB [35]	TCP	√	×	ECN	FCT/TL
FlowBender [36]	TCP	√	ECMP	ECN	FCT/TL
HERMES [37]	×	√	×	ECN/RTT/Timeout	FCT
Clove [38]/ECMP	×	√	√	ECN/path-utilization	FCT
HULA [39]	×	×	√	per-hop utilization	FCT/TP
LocalFlow [19]	TCP	×	√	local	NUM

RBS [40]	×	×	√ path-ECMP	Flow paths	FCT/TP
MATE [41]	×	×	√ edge-based	path utilization/delay	Min-Cost
TeXCP [42]	×	×	√ edge-based	path utilization	utility Min-Max
Flare [43]	×	×	√	local	TP
LetFlow [44]	×	×	√	none	FCT
CONGA [8]	×	×	√ edge-based	path congestion	FCT
Expeditus [45]	×	×	√	local/path combined	FCT/TL
DRILL [1]	×	×	√	local	FCT
ECMP [4]	×	×	√	none	TP
LIA [46]	√	√	×	source inferred	TP
Semicoupled [47]	√	√	×	source inferred	TP
OLIA [48]	√	√	×	source inferred	TP
BALIA [47]	√	√	×	source inferred	TP
EWTCP [49]	√	√	×	source inferred	TP

FCT: Flow Completion Time; TP: Throughput; TL: Tail Latency; PL: Packet Latency; FD: Flow Deadline; RTT: Round Trip Time; NUM: Network Utility Maximization; AQM: Active Queue Management; ECN: Explicit Congestion Notification

1.2 Challenges

In this section, we look at the two major drawbacks of the existing traffic control design space for datacenter networks:

1. The solutions for traffic control in datacenter networks are mostly **point by design**, focusing on one aspect of the traffic control only, e.g., single-path or multi-path congestion control, or in network load balancing, but not both in an integrated fashion with various performance targets and input requirements. More often than not, a point solution has to coexist with some other point solutions in practice, e.g., a newly developed congestion control protocol coexisting with a load balancing solution and/or a legacy TCP congestion control protocol. This, however, may adversely impact the effectiveness of all protocols involved. Indeed, it has been widely recognized [8, 14, 15, 16, 17, 18] that independently developed traffic control solutions, especially at different layers, may adversely

interact with one another, leading to suboptimal, unexpected performance, or even network instability. Moreover, as point solutions, they cannot easily adapt to software/hardware capability changes and hence, cannot fully explore the performance, scalability, and design complexity tradeoffs.

2. Most of the existing solutions for traffic control in datacenter networks are **empirical by design**, without provable convergence, stability, and optimality properties. Although some solutions are NUM-based, e.g., [7, 6, 19], they are again point solutions with limited scope, applicable to elastic traffic only and deal with either congestion control or load balancing, but not both. Notable examples are the two earlier traffic engineering (i.e., load balancing) solutions, i.e., MATE [41] and TeXCP [42], and three more recent datacenter network solutions, i.e., NUMFabric[7] and FCP [6] for congestion control, and LocalFlow[19] for load balancing.

The difficulty lies in the fact that the current NUM framework is incapable of providing performance guarantee for inelastic flows with diverse performance requirements. The root root cause lies in the fact that NUM in the current formulation is not suitable to support multiple classes of services for three main reasons:

1. **NUM cannot provide user utility guarantee:** NUM is purely a "social-welfare" based framework, striving to maximize the sum of user utilities, with no regard to application-specific requirements, nor meaningful fairness among users of distinct CoSes. Under this framework, the achievable individual user utilities, regardless of which CoSes they belong to, are strong functions of traffic load and traffic mix, which, however, may change over time. For example, For for a single-link network with bandwidth, C, the sum of user utilities, $nu(C/n)$, for n non-real-time (NRE) flows with the same concave utility function, $u(x)$, is an ever-increasing function of n [50]. This means that as n increases, the non-

real-time flows with the concave user utility may take over the entire network resources, while starving real-time flows with non-concave user utilities (see chapter 2 for more details).

2. A solution to NUM with more than one distinct user utility or even a single user utility but more than one distinct saturated rate (flow rate that maximizes the user utility. See chapter 2 for more details) **may not provide meaningful fair flow rate allocation**.
3. Generally, it is **difficult, if not impossible, to accurately quantify user utility**, as it is just one of many important factors that determine QoE.

This then begs the following fundamental question that motivates the work in this dissertation: Can NUM be reformulated in such a way to produce rich, optimal distributed solutions to create solutions with a common global objective and fairness criterion?

1.3 Dissertation Contributions

In this section, we highlight the contributions of the work in this dissertation in four highly related research topics, aiming at addressing the above challenges.

First, we reformulate the NUM framework to fully harness its potential in terms of enabling large families of traffic control protocols for datacenter applications. We put forward a HOListic, user-utility-based flow rate allocation framework for datacenter NETworks (HOLNET). It makes the following contributions:

1. HOLNET allows large families of congestion control and load balancing protocols of various degrees of sophistication to be developed, with all the protocols in a family achieving a common global objective and fairness criterion.
2. All the protocols developed under HOLNET enjoy provable convergence, stability, and optimality properties by design.

3. It allows fully integrated host-based (multipath) congestion control and in-network load balancing protocols to be developed, making fair resource allocation in the presence of both congestion control and load balancing a reality.
4. With proper design, HOLNET leads to protocols that are backward compatible with TCP and hence are TCP friendly by design.

Second, based on the HOLNET framework, we derive and implement in Linux two families of MPTCPs, i.e., Equal Utility MPTCP with sub-flow rate-scaling vector, $\boldsymbol{\gamma} = \{\gamma_1, ..., \gamma_m\}$ (EUTCP($\boldsymbol{\gamma}$)) and Weighted Utility MPTCPs with sub-flow weight vector, $\boldsymbol{\omega} = \{\omega_1, ..., \omega_m\}$ (WUTCP($\boldsymbol{\omega}$)), where m is the number of sub-flow paths. This work makes the following contributions:

1. We demonstrate that the family members of EUTCP($\boldsymbol{\gamma}$) with rate-scaling coefficient in the range of [1,1.1] outperforms three well-known MPTCPs with resource pooling capability, including LIA, OLIA, and Balia, in terms of achieving satisfactory tradeoffs among responsiveness, fairness, and throughput.
2. We show that the Semicoupled algorithm [47] and EWTCP [49] are in fact EUTCP($\mathbf{1}$) and WUTCP($1/m^2$), where m is the number of sub-flow paths, and hence, are NUM-optimal.

Third, to address the case where different paths may use a different pricing models, e.g., Wi-Fi versus cellular connections, we propose a hybrid EUTCP-and-WUTCP algorithm to incentify users with multihoming to use multipath. This work helps us tackle the challenge of improving user' QoE by providing better throughput and responsiveness over other MPTCPs as well as single-path protocols with different pricing models.

Finally, based on the HOLNET framework, we revisit the state-of-the-art, empirically designed datacenter protocol, i.e. DCTCP [2] and we propose Optimal-

DCTCP to prove that a simple modification of the congestion indicator, DCTCP can be turned into a NUM-optimal traffic control protocol.

This dissertation aimed at addressing challenges with existing datacenter traffic control design space being point and empirical in design. Our work can provide the framework to achieve these 2 main objectives and convert two of the more related state-of-the-art solutions of MPTCPs and DCTCP into NUM-optimal solutions. We also look at the practical aspect of the design of MPTCP protocols under different pricing models and provide a solution for it.

1.4 Dissertation Organization

The remainder of this dissertation is organized as follows. In Chapter 2, we put forward a traffic control framework for datacenter networks called HOLNET [51] which is principled and systematic in approach. We also introduce a family of protocols called HOLNET with Utility-of-TCP-based flow rate Allocation (HOLNET-UTA) which is TCP friendly by design and backward compatible with TCP Reno. Chapter 3 presents the two families of optimal MPTCPs for datacenter Networks. Chapter 4 proposes the Hybrid MPTCP solution for using multiple paths with different pricing structures. Chapter 5 introduces the NUM-optimal solution for DCTCP. Finally, chapter 6 makes concluding remarks and discusses our future research plan.

CHAPTER 2

HOLNET: A HOLISTIC TRAFFIC CONTROL FRAMEWORK FOR DATACENTER NETWORKS

2.1 Introduction

In this chapter, we put forward a HOListic traffic control framework for datacenter NETworks (HOLNET). The approach taken by HOLNET is principled and systematic. At the core of HOLNET is the introduction of the notion of center-of-utility fairness and the reformulation of the NUM framework into what we call, HOLNET NUM. HOLNET NUM fully harnesses the potential of the existing NUM solutions, allowing a whole new spectrum of traffic control protocols to be developed. The protocols developed under HOLNET can provide (soft) minimum flow rate guarantee and center-of-utility fair sharing of network resources for diverse applications. Hence, HOLNET makes a key contribution to bridge the gap between the existing solutions to NUM and their practical application to the design of traffic control protocols in support of diverse applications. More specifically, HOLNET possesses the following salient features:

1. It allows large families of congestion control and load balancing protocols of various degrees of sophistication to be developed, with all the protocols in a family achieving a common global objective and fairness criterion. Protocols from a given family can be selected to fully explore the performance, scalability, and design complexity tradeoffs and adapt to various possible software/hardware capability changes.

2. All the protocols developed under HOLNET enjoy provable convergence, stability, and optimality properties by design.
3. It allows fully integrated host-based (multipath) congestion control and in-network load balancing protocols to be developed, making fair resource allocation in the presence of both congestion control and load balancing a reality.
4. With proper design, HOLNET leads to protocols that are backward compatible with TCP and hence are TCP friendly by design.

The flexibility for the HOLNET protocol design space exploration is demonstrated by the design of protocols, i.e., an end-to-end multi-Class-of-Service (CoS), single-path congestion control protocol, and an end-to-end single-CoS multipath congestion control. To further test the backward compatibility, scalability, extensibility, and performance of HOLNET-based solutions, we introduce a family of protocols, called HOLNET with Utility-of-TCP-based flow rate Allocation (HOLNET-UTA) and we study one member of this family, a congestion control protocol with in-network load balancing. HOLNET-UTA is TCP-friendly by design and backward compatible with TCP Reno [3]. It is found by the simulation to reduce average FCT by more than 20% compared to DRILL [1] with DCTCP [2].

2.2 BACKGROUND AND MOTIVATION

Since HOLNET is also rooted in NUM, in what follows, we first give a background overview of NUM. As discussed in chapter 1, the three major drawbacks of NUM, which motivates our work and also helps to explain why the existing NUM-based solutions are point by design and have limited scope.

NUM and NUM Solutions: NUM can be formally stated as

$$max\{V(\mathbf{x}) = \sum_{i=1}^{n} w_i u_i(x_i)\}, \tag{2.1}$$

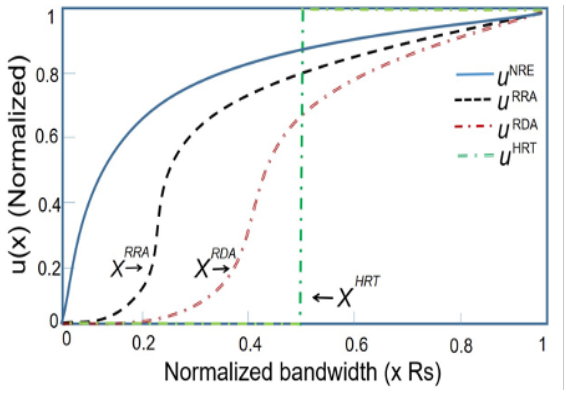


Figure 2.1: Normalized user utilities for four CoSes

subject to link bandwidth constraint (i.e., the total flow rate for flows sharing a link must not exceed the link bandwidth), where n is the number of active flows, $u_i(x_i)$ and w_i are the normalized user utility function of the allocated flow rate x_i and the weight for flow i respectively. $V(\mathbf{x})$ is the sum of the weighted user utilities for all active flows. Here $u_i(x_i)$'s are meant to be used to characterize Quality of Experience (QoE) in terms of bandwidth demand for users of diverse applications. Note that QoE or how a user feels about a service is, in general, a complex function of many factors, e.g., TP, FCT, packet delay and delay jitter, and even price paid for using the service. Meeting the bandwidth demand of a user may be viewed as the first-order approximation of meeting QoE.

For example, in his seminal work [50] back in 1995, Shenker discussed protocol design challenges for the future Internet in the context of NUM and defined four broad CoSes to meet diverse application requirements, in terms of four different types of user utilities, as illustrated in Figure 2.1, i.e., u^{NRE}, u^{RDA}, u^{RRA}, and u^{HRT}, for Non-Realtime Elastic (NRE), Realtime Delay Adaptive (RDA), Realtime Rate Adaptive (RRA) and Hard Realtime (HRT) CoSes, respectively. R_s in Figure 2.1 is defined as the saturated flow rate that maximizes the user utility, e.g., the highest encoding rate for an application of RRA CoS or maximum link bandwidth for an application

of NRE CoS. In general, any flow i with rate x_i, is associated with a given user utility function, $U_i(x_i)$, and the corresponding normalized user utility $u_i(x_i)$, where,

$$u_i(x_i) = U_i(x_i)/U_i(R_{s,i}) \tag{2.2}$$

and $R_{s,i}$ is the saturated rate for flow i. Clearly, $u_i(x_i)$ is a more convenient measure of the degree of user satisfaction than $U_i(x_i)$.

To design the Internet protocols on the basis of the above NUM, the first thing one must do is to find distributed solutions to NUM. The first breakthrough came along in 1998 when Kelly, et. al. [52] showed that the utility function of logarithm form, i.e., $u_i(x_i) = log(x_i) \in u^{NRE}$, for $\forall i$, results in a solution in the form of a distributed flow rate adaptation algorithm that resembles the distributed TCP congestion control, achieving the so-called proportional fairness. This encouraging result has stimulated the research interests in finding more general distributed solutions to NUM. In particular, Low [53, 54] proposes a distributed primal-dual congestion control solution to NUM with arbitrary concave user utilities in u^{NRE} CoS. This solution requires that the flow rate at the source of a flow and a variable, called price (a function of the link load), at each and every link along the flow path are iteratively updated and exchanged. This solution provided the theoretical underpinning for all the aforementioned NUM based datacenter network traffic control protocols of the NRE CoS, including NUMFabric [7] and FCP [6] for flow rate control, and LocalFlow [19] for load balancing.

Another line of research based on Sliding Mode in control theory [55] has culminated in the finding of distributed solutions spanning a large design space, allowing for flow multipath and in-network flow load balancing with minimum information exchange, and admitting flow rate constraints and both concave (e.g., u^{NRE}) [56, 57] and non-concave [58] (e.g., u^{RDA}, u^{RRA}, and u^{HRT}) user utilities.

Although promising, so far, we have seen no application of the above NUM solutions as the theoretical underpinning for the design of traffic control protocols for inelastic datacenter applications. The root cause lies in the fact that NUM in the current formulation is not suitable to support multiple CoSes for three main reasons as discussed in chapter 1.

2.3 HOLNET

In this section, we first introduce HOLNET NUM which overcomes the three drawbacks of NUM. Then we characterize the design space of HOLNET solutions.

2.3.1 HOLNET NUM

As discussed above, NUM fails to provide:

1. User utility guarantee
2. Meaningful fairness in the presence of flows with different user utilities and/or saturated rates
3. Faithful characterization of user utility

HOLNET NUM directly addresses (1) and (2) by providing minimum user utility guarantee and center-of-utility fairness, respectively, which together capture the essential characteristics of user utility, hence, to a great extent, achieving (3).

MINIMUM USER UTILITY GUARANTEE: We note that a user utility is generally composed of two parts, i.e., a non-concave lower part and a concave higher part, e.g., the parts below and above the inflection points X^{RRA} (X^{RDA}) for RRA (RDA) as shown in Figure 2.1. For instance, X^{RRA} may be the lowest level encoding rate for an adaptive video stream. NRE and HRT are two extreme cases, one without a lower part and the other without a higher part. While offering more bandwidth or user utility beyond the lower part may make a user happier, which, however, is not

critically important, satisfying the lower part of the user utility, called the minimum user utility in this paper, is likely to be essential to the user satisfaction of the service and hence, cannot be compromised.

The minimum user utility for a datacenter service is often spelled out by datacenter service providers in terms of a given service level objective (SLO), e.g., a tail-latency SLO for a user-facing interactive service. Since an SLO, in turn, can be translated into given flow rates or flow deadlines [13], many existing traffic control solutions aim at providing minimum flow rate or the flow deadline guarantee [11, 26, 32], without having to explicitly construct the user utility function. Likewise, HOLNET aims at providing minimum flow rate guarantee, assuming that the minimum flow rate required to achieve the minimum user utility is already known.

To this end, HOLNET NUM simply extends NUM to allow the minimum flow rate, e.g., θ, to be enforced as a flow rate constraint, i.e., $x \geq \theta$. Clearly, any feasible solution to HOLNET NUM provides the minimum flow rate, or equivalently, the minimum user utility guarantee, regardless of the traffic load and flow mix pattern.

CENTER-OF-UTILITY FAIRNESS:

HOLNET NUM aims to enable a meaningful user-utility-aware fairness criterion for flow rate allocation among flows with different user utilities, called the center-of-utility fairness. First, we define **center-of-utility**, x_i^c, for flow i, that captures the average user utility of flow i, as follows,

$$x_i^c = \frac{\int_0^{R_{s,i}} x_i \frac{dU_i(x_i)}{dx_i} dx_i}{U_i(R_{s,i})} = \int_0^{R_{s,i}} x_i \frac{du_i(x_i)}{dx_i} dx_i. \tag{2.3}$$

The concept for the center-of-utility is borrowed from mechanics in physics. More specifically, if we view the user utility density, $dU_i(x_i)/dx_i$, as the mass density at x_i for an uneven bar of length, $R_(s, i)$, and total mass, $U_i(R_{s,i})$, then the center of mass [59] of the bar is x_i^c, hence the name. x_i^c is the center, to which the user utility,

$U_i(x_i)$, is concentrated. Hence, x_i^c is the exact measure of the average bandwidth demanded by the users of flows with user utility, $U_i(x_i)$.

We now propose to enable the center-of-utility fair flow rate allocation, i.e.,

$$\frac{x_i}{x_j} = \frac{x_i^c}{x_j^c}, \tag{2.4}$$

for any pair of flows i and j sharing a bottleneck link. As a result, HOLNET aims to achieve **center-of-utility fair flow rate allocation**, provided that the minimum flow rates to sustain the minimum user utilities are satisfied.

The rationale behind the use of the above fairness criterion can be best illustrated by an example. Consider an adaptive audio flow and an adaptive video flow sharing a bottleneck link. It is clear that both the average bandwidth demands (i.e., the center-of-utilities) and the minimum encoding rates (i.e., the flow rates to sustain the minimum user utilities) for the audio and video applications are quite different, say, 50 Kbps versus 5 Mbps, and 10 Kbps versus 1 Mbps, respectively. With the center-of-utility fairness, the video flow will then be allocated 100 times of the additional link bandwidth than the audio flow, provided that the minimum encoding rates for the two flows, i.e., 10 Kbps and 1 Mbps, are satisfied. Clearly, this flow rate allocation solution captures the QoE in terms of user bandwidth demand well, i.e., the guaranteed minimum user bandwidth and additional bandwidth allocation in proportion to the relative bandwidth demands.

However, as we mentioned earlier, it is generally difficult to exactly quantify $U_i(x_i)$, which, in turn, makes it difficult to quantify x_i^c as it is derived from $U_i(x_i)$. Fortunately, however, with the minimum user utility guaranteed, it is no longer essential to allocate the additional resource among flows perfectly in proportion to the relative bandwidth demands, which by itself, is difficult to define. So, in practice, x_i^c may be roughly estimated in terms of the order of the bandwidth demand of the

underlying application without having to know the exact $U_i(x_i)$. For example, assume that an adaptive audio application has several encoding levels in the range of 10Kbps-100Kbps. It suffices to set x_i^c at around 50Kbps to capture the order of the overall bandwidth demand of this application.

Next, we recast NUM in Eq. (2.1) in such a way that it will indeed lead to the fair flow rate allocation defined in Eq. (2.4). To this end, we define HOLNET NUM as follows,

$$max\ \{V(\mathbf{x}) = \sum_{i=1}^{n} w_i U_c(x_i)\}, \tag{2.5}$$

subject to both link bandwidth constraints and flow rate constraints for flows with minimum user utility requirements, where $U_c(x)$ is a concave user utility called the base utility, shared by all the flows[1]. In other words, our design goal of HOLNET NUM is no longer to maximize the sum of the user utilities, but to achieve center-of-utility fair flow rate allocation via careful design of w_i and $U_c(x)$ as follows.

Consider flows sharing a bottleneck link. With the Lagrange multiplier technique [60], it can be easily shown that to maximize the sum of $w_i U_c(x_i)$ for all the flows sharing this link, we must have,

$$\frac{w_i}{w_j} = \frac{dU_c(x_j)/dx_j}{dU_c(x_i)/dx_i}, \qquad \forall i, j, \tag{2.6}$$

for any pair of flows i and j bottlenecked at this link.

For instance, a widely studied family of concave-fairness user utilities [61], is given as,

$$U_\alpha(\alpha, x) = x^{1-\alpha}/(1-\alpha), \quad for \ \ \alpha \in (0, \infty). \tag{2.7}$$

[1]While on the surface, using a single concave utility with weights in NUM is not new [7], HOLNET uses it in a completely different way than the traditional one, i.e., realizing the center-of-utility fair flow rate allocation, rather than achieving some network-centric performance target, e.g., minimizing average FCT [7]

Now let $U_c(x) = U_\alpha(\alpha, x)$ defined in Eq. (2.7), we have,

$$w_i/w_j = (x_i/x_j)^\alpha, \qquad \forall i, j. \tag{2.8}$$

Here, the weight, w_i, for any user utility, $U_i(x_i)$ can be calculated against the weight, $w_0 = 1$, for the base utility, $U_0(x_0) = U_c(x_0)$, with a base center-of-utility, x_0^c, calculated from Eq. (2.3). Here, w_0 is set to 1, without loss of generality. By substituting Eq. (2.4) into Eq. (2.8), we have,

$$w_i = (x_i^c/x_0^c)^\alpha, \qquad \forall i. \tag{2.9}$$

Then, substituting the above w_i and $U_c(x_i) = U_\alpha(\alpha, x_i)$ into Eq. (2.5), we arrive at a new NUM, called HOLNET NUM, achieving center-of-utility fair flow rate allocation with minimum user utility guarantee. It can be easily shown that HOLNET NUM also works for the case where each flow x_i can be split into L subflows, $x_{i,l}$ for $l = 1, ..., L$, with each subflow taking different paths to the same destination. A nice property of HOLNET NUM is that it only deals with a single concave user utility. As such, all the existing solutions to NUM with concave user utilities can be applied to HOLNET NUM, hence harnessing the potential of NUM, to be described in the next section.

Finally, we note that in HOLNET, the user utility information for a given flow i, including the minimum user utility and center-of-utility fairness, is incorporated in HOLNET NUM only through a pair of parameters, θ_i and w_i. This means that HOLNET can also be used to enable user-utility-agnostic traffic control solutions, as long as this pair is properly defined. For example, this pair may be tied to a pricing model to achieve price-proportional flow rate allocation and fairness.

2.3.2 HOLNET Design Space

In this section, we first formally state the problem and its solutions. Then we outline the design space of the solutions.

Consider a network with a set of nodes (i.e., switches) B. Let L_b be the set of all links l connected to node b. $L_{b_{s_i}}$ be the set of all outgoing links l of a source node S_i and $x_{i,l}$ be the subflow rate of flow i through link l. Namely, flow i is split into n_i subflows, which are mapped to different first-hop nodes and we have,

$$x_i = \sum_{l \in L_{b_{si}}}^{n_i} x_{i,l}, \quad \forall i. \tag{2.10}$$

This setup allows for host-based multipath or multi-next-hop congestion control and load balancing. Now HOLNET NUM is to achieve the global objective given in Eq. (2.5), subject to the network link bandwidth constraints and the following flow rate constraints,

$$\theta_i \leq x_i \leq \Theta_i, \quad \forall i. \tag{2.11}$$

Different values of lower bound θ_i and upper bound Θ_i can be used to provide minimum user utility guarantee for different CoSes. For example, for NRE, $\theta_i = 0$ and $\Theta_i = \infty$; for HRT, $\theta_i = \Theta_i > 0$; for RRA and RDA, $\theta_i > 0$ and $\Theta_i = \infty$.

HOLNET is underpinned by the family of optimal, distributed traffic control solutions to NUM with concave user utilities given by Su, et. al. [57], rather than the family of primal-dual solutions [53, 54], which underpin all the aforementioned datacenter NUM-based protocols. The former one is adopted because it is most sophisticated one that

1. accommodates flow rate constraints
2. covers a large design space, including integrated (multi-path) congestion control and load balancing

3. allows for the exploration of performance, scalability, and design complexity tradeoffs and adaption to hardware/software changes.

In other words, HOLNET NUM fully harnesses the potential of the most sophisticated NUM solutions for the design of a whole new set of families of traffic control protocols.

In its most general form, these families of solutions are applicable to a multi-domain environment. The families of solutions are generally composed of a set of user-utility-based, multi-next-hop-enabled congestion controllers running at hosts (i.e., servers) and a set of in-network load balancers running at network nodes.

Multi-next-hop-enabled congestion controller: The families of optimal congestion controllers for flow i at source host S_i can be generally written as [57],

$$\dot{x}_{i,l} = z_i(t, x_i, cg_l, r_{i,l}^{out})[f_i(x_i) - (1 - \overline{cg}_l r_i r_{i,l}^{out})], \tag{2.12}$$

where

$$f_i(x_i) = 1 - e^{-\partial U_i(x_i)/\partial x_i}, \tag{2.13}$$

where $U_i(x_i)$ can be any concave function and for HOLNET NUM, $U_i(x_i) = w_i U_c(x_i)$ where each $U_c(x_i)$ defines a family of congestion controllers, cg_l is a local congestion indicator, taking value 1 if the outgoing link l is congested and 0 otherwise. $\overline{cg}_l$ is the logic negation of $cg_l$and $z_i(.)$ is a user-defined piecewise continuous positive scalar function. r_i is determined by the Cos as follows,

$$r_i \quad = \quad \begin{cases} r_{max}^{CoS} & if \quad x_i < \theta_i \\ 1 & if \quad \theta_i \leq x_i \leq \Theta_i \\ r_{min}^{CoS} & if \quad x_i \geq \Theta_i, \end{cases} \tag{2.14}$$

an extension to the results in [57]. Here $r_{max}^{CoS} > 1$ and $r_{min}^{CoS} < 1$ are tunable constants and $r_{i,j}^{out}$ is determined by the congestion information fed back from the first hop node (i.e., a top-of-rack (ToR) switch in the context of a datacenter network),

a per-flow-based single-hop signaling. It is calculated based on Table (2.1), where $r_{max} > 1$ and $r_{min} < 1$ are design parameters.

Table 2.1: $r^{in}_{i,b}$ and $r^{out}_{i,(b),l}$ (it represents both $r^{out}_{i,l}$ and $r^{out}_{i,b,l}$) vs congestion information

Congested			
next hop	**local**	$r^{in}_{i,b}$	$r^{out}_{i,(b),l}$
yes	yes	r_{max}	r_{min}
yes	no	r_{max}	r_{min}
no	yes	r_{max}	1
no	no	1	1

Here are some salient features of the above families of congestion controllers:

1. A congestion controller for any given flow i is dependent on the user utility function, $U_i(x_i)$, or in the case of HOLNET, θ_i and $w_i U_c(x_i)$, for flow i only (see Eq. (2.13)). As we shall see shortly, the in-network load balancers are user-utility agnostic. This means that flows belonging to a new CoS with a new user utility function can be added by simply activating a new congestion controller with its minimum flow rate and weight value properly set at runtime.
2. The only nonlocal information is the congestion feedback, $r^{out}_{i,l}$, from the first-hop nodes, making the congestion control highly scalable.
3. With multi-next-hop flow load balancing capability, these families of congestion controllers enable host-based load balancing features, as well as both host and in-network load balancing features by working seamlessly with in-network load balancers.
4. Each base utility, $U_c(x)$, defines a family of congestion controllers. Each family is composed of congestion controllers of various degrees of sophistication. Different congestion controllers may take as input different information for the

control, by engineering $z_i(.)$ as a function of various additional information, ranging from the local information, all the way to the entire path information for performance enhancement, without changing the global design objective and fairness criterion.

5. The above families of congestion controllers degenerate into families of transport congestion controllers [56] by simply setting $r_{i,l}^{out} = 1$, if the in-network nodes are not involved. cg_l now represents the congestion indicator for path l from the source to the destination host, which may be
   (a) inferred by the source host, resulting in highly scalable end-to-end solutions, similar to the end-to-end TCP congestion control
   (b) acquired explicitly using ECN [62], leading to lightweight ECN-based solutions, similar to DCTCP and D2TCP
   (c) explicit link load or even path information

In-network load balancer: At each node b, the outgoing data rate $x_{i,b,l}^{out}$ from node b through link l ($l = 1, 2, ..., n_{i,b}$) is given by,

$$\dot{x}_{i,b,l}^{out} = z_{i,b}(t, x_i, cg_l, r_{i,b}^{in}, r_{i,b,l}^{out})[-1 + \bar{cg}_l r_{i,b}^{in} r_{i,b,l}^{out})], \tag{2.15}$$

where $n_(i, b)$ is the number of next-hop nodes for flow i, which may be determined locally by node b. $z_(i, b)(.)$ is again a piecewise continuous positive scalar function. $r_(i, b, l)^out$ and $r_(i, b)^in$ are determined by the local and next-hop congestion information, respectively, given in Table 2.1. Again, $r_(i, b, l)^out$ enables per-flow-based congestion feedback. This family of load balancers can be directly applied to HOL-NET NUM without further modification.

Here are some salient features of the above load balancers:

1. They are independent of user utilities
2. They are flow rate constraint unaware

These two features make the load balancing user-utility agnostic and scalable.

To further improve the scalability, [57] introduces a percentage-based destination node-based load balancer that further reduces the total number of load balancers per node to be the number of edge nodes. Let $p_{i,b,l}$ be the percentage of all incoming traffic ($\sum_j x^{in}_{i,b,j}$) at node b routed to outgoing link l towards destination i. The control law for the outgoing traffic $x^{out}_{i,b,l}$ through l is given by,

$$x^{out}_{i,b,l} = p_{i,b,l} \sum_{j \in L_b} x^{in}_{i,b,j} \tag{2.16}$$

and

$$\dot{p}_{i,b,l} = z_{i,b}(t, x_i, cg_l, r^{in}_{i,b}, r^{out}_{i,b,l})(\dot{x}^{out}_{i,b,l} \sum_{j \in L_b; j \neq l} p_{i,b,j} - p_{i,b,l} \sum_{j \in L_b; j \neq l} \dot{x}^{out}_{i,b,j}), \tag{2.17}$$

with

$$x^{out}_{i,b,l} = -1 + \bar{cg}_l r^{in}_{i,b} r^{out}_{i,b,l} \tag{2.18}$$

HOLENT DESIGN SPACE: The above HOLENET-NUM based families of solutions span a large design space, as shown in Table 2.2, from single-path rate-adaptive congestion control to the most comprehensive ones involving both multi-CoS, multipath congestion control, and in-network load balancing.

The HOLNET-NUM based families of solutions span a large design space, as shown in Table 2.2, from single-path rate adaptive congestion control to the most comprehensive ones involving both multi-CoS, multipath congestion control and in-network load balancing.

1. CC and MCC on the top of the list are the two simplest solutions among all, which are rate-adaptive congestion control solutions without and with multipath, respectively. CC-CoS and MCC-CoS further enable services with minimum user utility guarantee. All four types of solutions are host-based and can be end-to-end (as scalable as end-to-end TCP), ECNbased (as scalable as

DCTCP), link-load-based, or even pathload-based, leading to a wide range of solutions of various degrees of sophistication, allowing for full exploration of the performance, scalability, and design complexity tradeoffs.

2. The next four solutions take advantage of path diversity to further improve the flow performance via in-network load balancing. In addition to the congestion controllers running at hosts, these solutions require that a network node implement a set of load balancers with per-hop signaling for flow aggregates at some given granularities, e.g., ToR-to-ToR or destination-ToR based. These congestion controllers and load balancers use the local and next-hop congestion information as input for the control[2].

Table 2.2: HOLNET Design Space

Solution	Host-based	Host-based LB	In-network LB	Multi-CoS	Comments
CC	yes	no	no	no	Elastic Congestion Control (CC)
MCC	yes	yes	no	no	Multipath CC (MCC)
CC-CoS	yes	no	no	yes	CC and multi-CoS
MCC-CoS	yes	yes	no	yes	multipath CC-CoS
CC-LB	yes	no	yes	no	CC with in-network load balancing (LB)
MCC-LB	yes	yes	yes	no	Multi-next-hop CC-LB
CC-CoS-LB	yes	no	yes	yes	CC-CoS with LB
MCC-CoS-LB	yes	yes	yes	yes	MCC-CoS with LB

Practical Applicability of HOLNET: As stated at the beginning of the introduction section, sophisticated traffic control solutions, such as those with QoS

[2]The proposed four solutions are the most lightweight and hence the most scalable optimal solutions possible. Indeed, it is shown by example [8] that load balancing with pure local information can result in worse performance than a traffic-oblivious solution, like ECMP. This means that to achieve optimal control, the input for the control must be at least per-hop based, which is the case for the current solution.

features derivable from HOLNET (i.e., the types of protocols involving CoS features listed in Table 2.2), clean-slate, and flow-deadline aware protocols listed in Table 1.1 become viable only in a well-controlled environment like a datacenter. This is simply because in a loosely controlled environment, such as the public Internet, selfish users can cheat the system by using protocols with better QoS features or more aggressive user utilities than needed. In contrast, in a well-controlled environment, what user utility, or equivalently, what HOLNET protocol may be used by a host (e.g., a server in a datacenter) in that environment is under the full control of the network operator/service provider in that environment. Moreover, to support users of the applications/services that require strict minimum user utility guarantee, some additional mechanisms, such as a call admission or a network resource monitory mechanism, must be in place, which is feasible only in a well-controlled environment.

2.4 CASE STUDIES

This section aims to demonstrate the viability and flexibility of HOLNET design space exploration. Specifically, we develop a family of CC-CoS congestion controllers and an MCC congestion controller and test them by ns-3 simulation.

2.4.1 A FAMILY OF CC/CC-CoS Controllers

To limit the exposure, we skip the subscription, i, for flow in for the rest of the section. The family of CC-CoS controllers using $U_c(x) = U_\alpha(\alpha, x)$ as the base utility with $z(.) = \gamma x^\alpha$ can be easily derived from Eq. (2.12) with $r_l^{out} = 1$, as follows,

$$\dot{x} = \begin{cases} \gamma x^\alpha (r - e^{-wx^{-\alpha}}) & if \quad cg = 0 \\ -\gamma x^\alpha e^{-wx^{-\alpha}} & if \quad cg = 1. \end{cases} \tag{2.19}$$

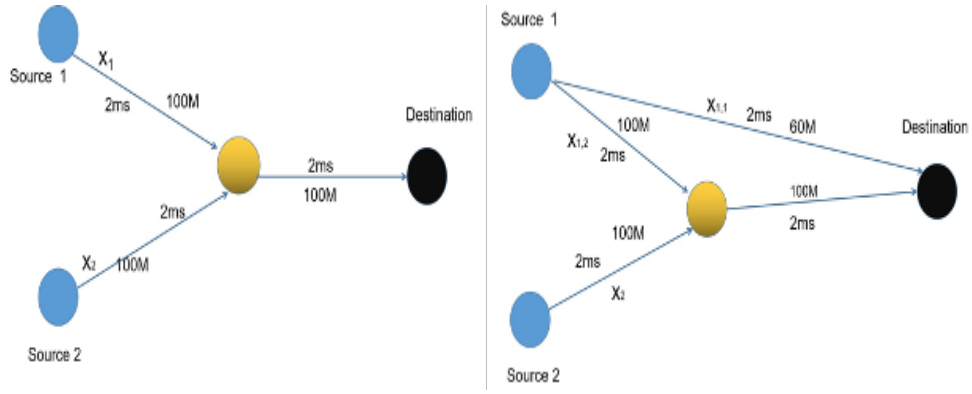
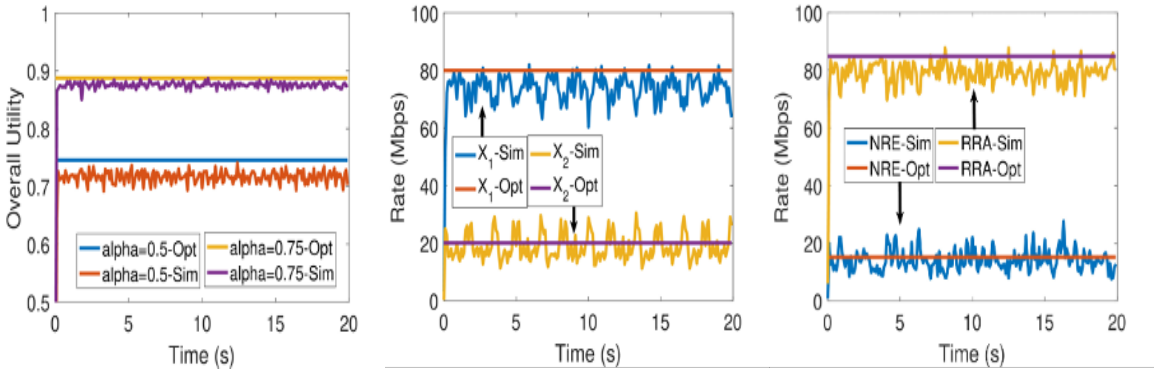


Figure 2.2: Network topologies for (a) MCC controller; (b) CC-CoS controllers.

where r is the same as r_i defined in Eq. (2.14) and we set r_{max}^{CoS} at 3 for all the case studies for this chapter. These controllers are then turned into window-based congestion control protocols. In this case study, we only consider end-to-end control, meaning that the controllers use only source inferable information for the control, i.e., the three replicated acknowledgments (ACKs) and timeout, similar to TCP Reno. The only difference is that for the current controllers, the rate adjustment is the same for both timeout and three duplicated ACKs. These controllers are as scalable as TCP Reno.

We first apply a CC controller in the above families of controllers with base utility, $U_c(x) = U_\alpha(0.5, x)$ or $\alpha = 0.5$, to two NRE flows (i.e., $\theta = 0$ and $r = 1$) with the same user utility, $U_1(x) = U_2(x) = U_\alpha(0.5, x)$ which share a 100 Mbps link, as shown in Figure 2.2 (a). The saturated rates, R_s's, for the two are set at 400 Mbps and 100 Mbps, respectively. The center-of-utility ratio for the two flows can then be calculated (i.e., Eq. (2.3)), which turns out to be 4, resulting in the weight ratio of 2 (i.e., Eq. (2.8)). In other words, the optimal flow rate allocation should be 4:1, or 80 Mbps and 20 Mbps for flow 1 and flow 2, respectively, in order to achieve the center-of-utility fairness.



(a) (b) (c)

Figure 2.3: (a) Overall utility; Rate allocation for (b) $\alpha = 0.5$ and (c) $\alpha = 0.75$.

Figures 2.2 (a) and (b) depict the achieved normalized $V(x)$ in Eq. (2.5) and flow rate allocations against their respectively optimal ones. As one can see, both converge to near optimal values quickly, resulting in the flow rate ratio of 3.61, close to, but lower than the optimal ratio of 4. This is because flow x_1, which has a higher flow rate than flow x_2 generally senses more congestion and hence achieves a lower than expected flow rate. Note that the aggregated flow rate is less than the link bandwidth of 100 Mbps, due to discrete-time window-based control that cannot fully utilize the link bandwidth.

Now we apply a different family of the CC-CoS controllers, i.e., the α-utility with α=0.75 as the base utility. Consider an NRE flow, x_1, with $U_{NRE}(x_1) = log(1 + x_1)$, and a RRA flow, x_2, with $U_{RRA}(x_2) = (x_2 - x_2^{RRA})^{1/3} + (x_2^{RRA})^{1/3}$ and $x_2^{RRA} = \theta = 20$ Mbps sharing a 100 Mbps link, as shown in Figure 2.2 (a), and flows have the same saturated rate $R_{s,1} = R_{s,2} = 100$ Mbps. The center-of-utility ratio for the two flows is then 5.59 (resulting in a weight ratio of 3.63) and hence the optimal flow rates for NRE and RRA flows should be 15.2 Mbps and 84.8 Mbps, respectively.

As one can see from Figures 2.3 (a) and (c), the simulated utility and flow rate allocation well match with the optimal ones. The flow ratio is 5.24, close to the optimal value, 5.59. Since the overall rate is higher than the minimum required rate of 20 Mbps for the RRA flow, the minimum rate does not play a role in the rate allocation.

2.4.2 Multipath Congestion Control

Now we develop a scalable elastic, NRE (i.e., $r = 1$), end-to-end multipath congestion controller. More specifically, we assume that all the flows have the same center-of-utility and saturated rate. This means that all the flows have the same weight, which can then be set to 1, without loss of generality. We let $U_c(x) = \epsilon log(x)$ $= \epsilon log(\sum_{l=1}^{L} x_l)$, where x_l is the flow rate of the subflow l , $x = \sum_{l=1}^{L}$ is the flow rate and ϵ is a constant. We set, $z(.) = \gamma x_l$, where γ is a constant. The values for γ and ϵ are chosen such that the control law degenerates to the TCP AIMD (i.e., additive increase and multiplicative decrease) control in the case of a single-path flow. Then the optimal controller for subflow l can be readily derived from Eq. (2.12) with $r = 1$ and $r_{i,l}^{out} = 1$, as follows,

$$\dot{x_l} = \begin{cases} \gamma x_l(1 - e^{-\epsilon/x}) & if \quad cg = 0 \\ -\gamma x_l e^{-\epsilon/x} & if \quad cg = 1. \end{cases} \tag{2.20}$$

The above controller can be understood as follows. When $x >> \epsilon$, Eq. (2.20) can be approximated as $\dot{x_l} \approx \gamma \epsilon x_l/x$ at $cg = 0$ and $\dot{x_l} \approx -\gamma x_l$ at $cg = 1$. Namely, when congestion is detected, which is source inferred, flow decrease rate is similar to that of TCP congestion control. In the absence of congestion, on the other hand, the flow increase rate is inversely proportional to its overall flow rate x, meaning that a subflow increase rate becomes smaller if the overall flow rate becomes larger. This

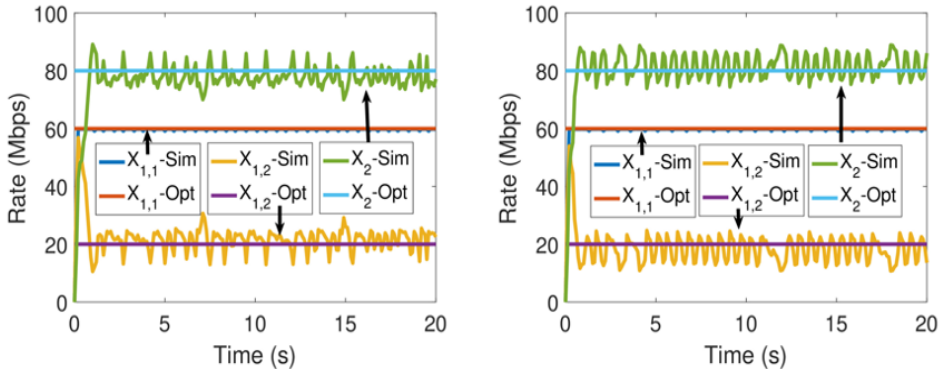


Figure 2.4: Rate Allocation: (a) MCC Controller; (b) MPTCP

controller is similar to and as scalable as MPTCP [9, 46], allowing all flows to evenly share network bandwidths.

Consider a multipath flow with two subflows $x_{1,1}$ and $x_{1,2}$ and a single-path flow x_2 that share a network, shown in Figure 2.2 (b). Subflow, $x_{1,1}$, takes a path with the bandwidth of 60 Mbps, and subflow $x_{1,2}$ and x_2 share a 100 Mbps link. With this network and flow path configurations, it can be easily shown that the optimal flow rate allocation for this HOLNET NUM problem is: $x_{1,1}$ =60 Mbps, $x_{1,2}$ = 20 Mbps, and x_2 = 80 Mbps, i.e., to equalize the flow rates. The performance of the MCC controller is tested against this optimal flow rate allocation (for this and all the rest of the case studies, we only show flow rate allocation, not $V(\mathbf{x})$, as the latter is just a means to achieve the former). It is also compared against MPTCP based on the ns-3 open-source code [63].

Figure 2.4 presents the performance results. As one can see, overall both MCC controller and MPTCP are able to allocate flow rates evenly between the two flows. For the MCC controller, x_2 is slightly lower than the optimal one (73 vs 80 Mbps) while the other two subflows are nearly equal to their respective optimal ones. The reason is the same as the previous case. Namely, a flow with a higher rate (i.e.,

x_2) generally senses more congestion than a flow with a lower rate (i.e., $x_{1,2}$), thus resulting in a more reduced rate, compared to the optimal one. Since MPTCP is empirically designed, its flow rate allocation cannot be easily interpreted with reference to the optimal one. Indeed, for MPTCP, x_2 actually achieves a higher rate than the optimal one, whereas $x_{1,2}$ is lower than the optimal one.

2.5 HOLNET-UTA

In this section, we develop a pragmatic family of lightweight traffic controllers, called HOLNET with Utility-of-TCP-based flow rate Allocation (UTA), or HOLNET-UTA in short. HOLNET-UTA is TCP-friendly and backward compatible with TCP Reno. Since in HOLNET, both minimum user utility and the center-of-utility are factored into the congestion controller in terms of θ and w, in this study, we assume θ and w are given and focus on the testing of traffic controller performance. Since HOLNET-UTA is lightweight, requiring only per-hop feedback from network nodes, it is only compared against some well-known lightweight transport and in-network load balancing solutions. Furthermore, since all the solutions to be compared against aim at achieving network-centric performance targets, such as FCT, we adopt the same performance targets.

First, to be TCP-friendly by design, HOLENET-UTA uses TCP utility as the base utility. We adopt the following TCP utility function and its z-function for TCP Reno, derived in [56]. Let ρx and βx be the multiplicative increase rate and multiplicative decrease rate, respectively. The TCP utility function in the SSP is then given by,

$$U_{tcp}(x) = xlog(1 + \frac{\rho}{\beta}), \tag{2.21}$$

and in the congestion avoidance phase (CAP) is given by,

$$U_{tcp}(x) = (\frac{\mu}{\beta} + x)[log(\mu + \beta x) - 1] - x[log(\beta x) - 1], \tag{2.22}$$

where μ is the additive-increase rate. The z-function of the TCP control law is derived in [56] as,

$$z_{tcp}(t, x, cg, z) \quad = \quad \begin{cases} (\rho + \beta)x & for\ SSP \\ \mu + \beta x & for\ CAP \end{cases} \tag{2.23}$$

With $U_c(x)$ being the TCP utility function in Eqs. (2.21) and (2.22) and $z(.)$ given above. Then HOLNET-UTA family of congestion controller are readily derived from Eq. (2.12) as,

$$\dot{x} \quad = \quad \begin{cases} (rr^{out} - (1 + \frac{\rho}{\beta})^{-w}(\rho + \beta)x & if \quad cg = 0 \\ -2^{1-w}\beta x & if \quad cg = 1, \end{cases} \tag{2.24}$$

for the SSP, and

$$\dot{x} \quad = \quad \begin{cases} [-(\frac{\beta x}{\mu + \beta x})^w + rr^{out}](\mu + \beta x) & if \quad cg = 0 \\ -(\frac{\beta x}{\mu + \beta x})^w(\mu + \beta x) & if \quad cg = 1 \end{cases} \tag{2.25}$$

for the CAP. Here r is given by Eq. (2.14) for flows with minimum user utility requirements otherwise $r = 1$.

HOLNET-UTA covers a large part of the HOLNET design space, i.e., CC/CC-CoS and CC-LB/CC-CoS-LB, in Table 2.2. It is minimalistic, meaning that it uses the minimal information feedback (i.e., source inferred or per-hop) and simplest possible queuing mechanism, i.e., a single FIFO queue per output port and enabling soft CoS features without call admission control. Hence, the HOLNET-UTA family is highly scalable.

To test, we derive an example CC-LB controller from the above family, i.e., by simply setting $r = 1$ (i.e., $\theta = 0$) and $w = 1$ in Eqs. (2.24) and (2.25) and test

its performance against some well-known lightweight transport and in-network load balancing solutions using a widely adopted realistic workload, i.e., data mining [64].

In our design, each source host runs a set of congestion controllers for its flows and each datacenter network node runs a set of load balancers for flow aggregates (i.e. Eq. (2.16)). Now we discuss some important design aspects in more detail:

1. **Congestion Detection** – In our design, the congestion for any given output port is detected, when the queue level for the buffer corresponding to that port reaches a p_q% threshold. In our case $p_q = 80\%$. This does not mean that when congestion is detected, the incoming packets are blocked. In fact, they are still allowed to enter the output buffer until buffer overflow, when the incoming packets are dropped.
2. **Load Balancing** – In a two-tier leaf-spine datacenter network, to be used for our case study, load balancing is done only by the source host side leaf nodes. The load balancing is coarse-grained, with only one load balancer per destination-leaf flow aggregate, regardless of flow types. The flow aggregate rate allocation to the multi-path is based on the percentage-based load balancers given in Eq. (2.17) by setting $z(.) = 1$. Initially, the percentage of each outgoing port in the multi-path is assigned proportionally to their bandwidth capacities. For example, with n outgoing ports, each has bandwidth B_i, for $i = 1, 2, ..., n$ the initial percentage assignments are set at, $p_i = B_i / \sum_{j=1}^{n} B_j$. After that, the percentages are updated periodically at a given time interval, which is set at RTT/4. At each update epoch, a leaf node checks if any outgoing port or its next hop is congested. If yes, then the percentage is updated according to Eqs. (2.16) - (2.18).
3. **Packet loss recovery** - Since HOLNET-UTA can sense congestion before buffer overflow, the packet drop rate is very low, particularly in the spine and

sender-side leaf nodes. So, we turn off the fast retransmission feature of TCP (i.e., retransmission upon receiving three duplicated ACKs) and solely rely on retransmission timeout for packet loss recovery. When a timeout occurs, for simplicity, the source host retransmits all the packets from the timed-out packet (similar to go back N). In our solution, a timeout will not trigger congestion control. Instead, it solely relies on the local and the next-hop feedback information for the control.

We compare the performance of HOLNET-UTA by simulation against DCTCP, combined with one of the two load balancing solutions, i.e., ECMP [4] or DRILL [1]. DRILL is found in [1] to offer better performance than most of the existing schemes, including CONGA [8] that uses global information.

A widely adopted performance metric for load balancing is FCT. So, we use average FCT as the main performance metric in the context of overall flows, small flows (size $<$ 100K), huge flows (size $>$ 10M bytes), and the 99th FCT for overall flows. A 6x6 leaf-spine network topology with 24 hosts per rack is simulated. The bandwidth/propagation delay is set at 10Gbps/10μs between a host and a leaf node, and at 20Gbps/30μs between a leaf and a spine node. The queue size in a leaf/spine node is 150/300 Kbytes. The ECN marking threshold is set at 65% of queue size, the typical value used in DCTCP [2]. The control parameters are set at $r_min = 0.1$ and $r_max = 1.5$. Flows arrive following a Poisson process. The network load is adjusted with the change of flow arrival rate. When a flow arrives, a host is randomly selected as the sender host and then a host in a different rack is randomly selected as the destination host.

Figure 2.5 gives the results (normalized to DRILL) for the data-mining workload case. UTA outperforms both DRILL and ECMP for most cases studied. Particularly, for the heavy load case, UTA outperforms DRILL (ECMP) by more than 20% (60%)

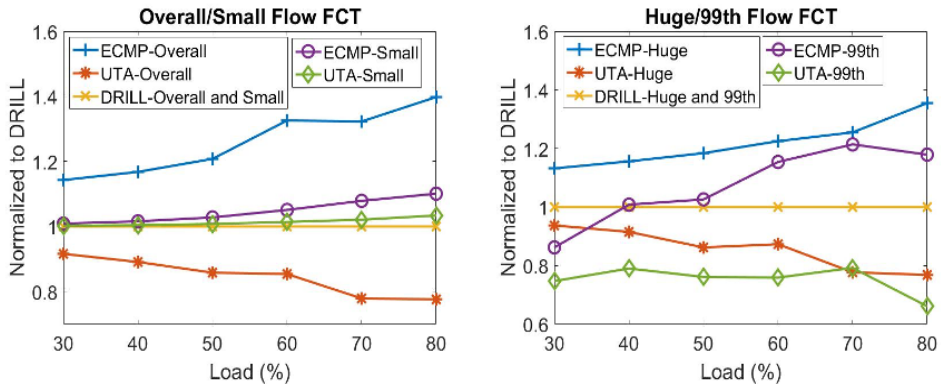


Figure 2.5: Performance Comparison (Data-mining)

in terms of average FCT, average FCT for huge flows, and 99th percentile and it is on par with DRILL for the small flow case. This is because, with per-hop congestion feedbacks and integrated congestion control and load balancing, UTA can respond to network congestions much faster and allow better-balanced load than both ECMP with DCTCP and DRILL with DCTCP, which are not integrated solutions.

2.6 CONCLUSION

This chapter presents HOLNET, a holistic traffic control framework for data-center networks. HOLNET allows large families of traffic control protocols of various degrees of sophistication to be developed. Unlike the existing solutions that are largely empirical by design, HOLNET is a principled, systematic solution. Protocols in each family developed under HOLNET share a common, user-defined global optimization objective. As a result, the protocols in each family can be fairly compared and carefully selected to fully explore the performance, scalability, and design complexity tradeoffs. As an example, we develop HOLNET-UTA, a family of integrated congestion controllers and load balancers, maximizing the sum of weight TCP

utilities. The large-scale simulation demonstrates the backward compatibility and flexibility of HOLNET.

CHAPTER 3

TWO FAMILIES OF UTILITY-BASED OPTIMAL MULTIPATH CONGESTION CONTROL PROTOCOLS FOR DATACENTER NETWORKS

3.1 INTRODUCTION

With the advent of multi-homed networks, mobile devices supporting multiple interfaces, and datacenters allowing multiple paths, the need for a new protocol that makes use of multiple paths to improve application performance was perceived by Raiciu et al. in their paper (Data Center Networking with Multi-path TCP). MPTCP has the potential to enhance application performance by providing better throughput and network resource utilization.

Just like end-to-end TCP, most prevalent MPTCPs, such as LIA [46], OLIA [48], and Balia [47], are end to end, involving two endpoints only. This chapter exclusively focuses on the design of end-to-end MPTCPs. However, despite the significant effort made in the past twenty years in an attempt to develop optimal MPTCPs [17], the existing MPTCPs are largely empirical by design. In their seminal work on LIA [46], which is standardized by IETF [65], Wischik, et. al. acknowledge that due to the lack of theoretical underpinning, LIA can only be designed empirically. Khalili, et. al. [48] further show that LIA is not even Pareto optimal and propose a Pareto-optimal MPTCP, known as OLIA, however, is again designed with no global optimization objective in mind. Peng, et. al. [47] classify and study major MPTCP algorithms with respect to some necessary conditions to achieve optimal traffic control in terms of network utility maximization (NUM), hence, making another step towards the design of an optimal MPTCP. However, the algorithm proposed by the authors, known as

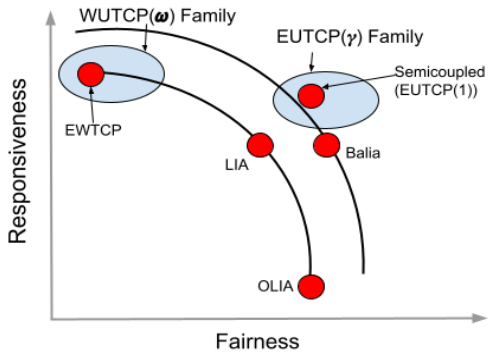


Figure 3.1: TCP Fairness vs Responsiveness

Balia, that seeks to improve responsiveness and TCP friendliness over LIA and OLIA, is again, designed empirically. Interestingly enough, an MPTCP algorithm studied in and cited by [47] as the Semicoupled algorithm given in [46][1] turns out to be NUM-optimal, as we shall show in this chapter. Although EWTCP [49] also turns out to be NUM-optimal, as we shall also prove in this chapter, it is again originally designed empirically without a global objective in mind.

Fig. 3.1 illustrates the whereabouts of some well-known MPTCPs as well as EUTCP($\boldsymbol{\gamma}$) and WUTCP($\boldsymbol{\omega}$) (to be defined shortly) in the responsiveness-and-TCP-fairness design space[2]. Without resource pooling, the WUTCP($\boldsymbol{\omega}$) family including EWTCP (i.e., WUTCP($\{1/m^2\}$), as we shall show later, are highly responsive to network dynamics but cannot guarantee fairness with respect to TCP. Here resource pooling refers to the ability to dynamically allocate sub-flow rates based on the overall resource availability among all sub-flow paths. All the rest of MPTCPs allow resource pooling and hence can achieve better fairness with respect to TCP, at the cost of

[1]To the best of our judgement, the algorithm studied in [47] does not match the Semicoupled algorithm in [46]. In this chapter, the Semicoupled algorithm is specifically referred to as the one presented in [47].

[2]As we shall show, the throughput differences among different MPTCPs are relatively minor and hence is not considered here.

reduced responsiveness in general. As one can see, as a family of NUM-optimal MPTCPs, EUTCP($\boldsymbol{\gamma}$) including Semicoupled (i.e., EUTCP($\mathbf{1}$), as we shall prove later) covers a relatively large design space (i.e., the area covered by the oval) on the upper-right corner and hence, allow better exploration of the design tradeoffs between responsiveness and TCP fairness than the existing ones.

In this chapter, by leveraging the NUM-optimal solution for concave utilities given by Lagoa, et. al., [66] and the concave TCP utility given by Wang, et. al. [67], we derive and implement in Linux two families of MPTCPs, i.e.,

1. **(EUTCP($\boldsymbol{\gamma}$))**

   Equal Utility MPTCP with sub-flow rate-scaling vector, $\boldsymbol{\gamma} = \{\gamma_1, ..., \gamma_m\}$

2. **(WUTCP($\boldsymbol{\omega}$))**

   Weighted Utility MPTCPs with sub-flow weight vector, $\boldsymbol{\omega} = \{\omega_1, ..., \omega_m\}$

, where m is the number of sub-flow paths. Specifically, we consider two different utility functions for multipath flows, i.e., the TCP utility function of a rate-scaled sum of the sub-flow rates and a weighted sum of TCP utility functions of the sub-flow rates. Then by applying the solution given by Lagoa, et. al. [66] to these two utility functions, we arrive at EUTCP($\boldsymbol{\gamma}$) and WUTCP($\boldsymbol{\omega}$). We then show that the Semicoupled algorithm [47] and EWTCP [49] are in fact EUTCP($\mathbf{1}$) and WUTCP($\{1/m^2\}$), respectively, and hence, are NUM-optimal. The performance of the two families with equal rate-scaling coefficient (i.e., $\gamma_l = \gamma$, $\forall l$) and equal weight (i.e., $\omega_l = \omega$, $\forall l$) when coexisting with TCP is also analyzed based on experiments in a testbed.

For EUTCP($\boldsymbol{\gamma}$), comparative performance analyses are carried out between its family members and some well-known resource-pooling-capable MPTCPs, including LIA[46], OLIA[48], and Balia[47]. The test results demonstrate that the family members in EUTCP($\boldsymbol{\gamma}$) with γ in the range of $[1, 1.1]$ outperform the other three MPTCPs in terms of responsiveness and fairness and is on par with the other three in terms

of overall throughput performance. For WUTCP($\boldsymbol{\omega}$), we compare three family members at $\omega = 1/m^2, 1/m$, and $1/\sqrt{m}$. We conclude that for the TCP fairness criterion defined in [49], $\omega = 1/m$ should be used, instead of $\omega = 1/m^2$, as suggested in [49].

3.2 BACKGROUND AND MOTIVATION

The existing MPTCPs can be broadly classified into two categories, with and without resource pooling capability.

For the category without resource pooling capability, a straightforward but naive solution is to simply run subflows as independent TCP flows. This approach, however, is too aggressive and unfair to single-path TCP flows, and without resource pooling, cannot balance the loads among subflow paths. This leads to the design of EWTCP [49], an equally weighted MPTCP. EWTCP attempts to achieve TCP fairness by modifying the previous solution, i.e., reducing the TCP window increase rate by a factor of $1/m^2$ for all m TCP-based subflows. However, besides the lack of load balancing capability inherited from the previous approach, EWTCP may lead to a flow rate lower than the best case single-path TCP[3], discouraging users to use it.

For the other category, i.e., the one with resource pooling capability, there has been a great effort made in the latest twenty some years in an attempt to develop MPTCP algorithms that are proven to be globally optimal in terms of network utility maximization (NUM), which in the form of a fluid-flow model, can be formally stated as follows:

[3]The best-case single-path TCP is defined as the maximum flow rate single-path TCP can achieve on any of the sub-flow paths available to MPTCP.

$$max \sum_{i=1}^{n} U_i(x_{i,1}, x_{i,2}, ..., x_{i,m_i}), \tag{3.1}$$

subject to link bandwidth constraints,

$$\sum_{i,j:l \in L_{i,j}} x_{i,j} - c_l \leq 0; \qquad l \in L, \tag{3.2}$$

where n, m_i, L, and $L_{i,j}$ are the number of active flows, the number of subflows in flow i, the set of links in the network, and the set of links lie in the path of subflow j in flow i, respectively; c_l is the link bandwidth for link $l \in L$; and $U_i(x_{i,1}, x_{i,2}, ..., x_{i,m_i})$ is the user utility for flow i as a function of flow rates, $x_{i,j}$, for subflow j, $j = 1, 2, ..., m_i$.

The design goal is to find distributed solutions to the above NUM problem in the form of distributed flow rate control laws for individual subflows, using only binary congestion information feedback for control, i.e., whether a sub-flow/flow path is congested or not, which is essential to facilitate the development of end-to-end protocols including MPTCPs. Such control laws can then serve as the theoretical underpinning for the design of optimal rate or window-based MPTCP algorithms for any given user utilities that dictate the fairness criterion for resource allocation.

Kelly, et. al. [17] and Han, et. al. [6] convert the NUM problem into its Lagrange dual problem and then solves a relaxation of the dual problem by closely approximating it by incorporating a price function in the utility function. This approach is Pareto optimal, but it is not strictly NUM-optimal and it only selects one subflow path at a time and hence, suffers from flappiness and slow responsiveness [46]. Inspired by this approach, Rauciu, et. al. [46, 9, 65] propose LIA, which modifies the previous approach to allow flow rate load balancing among multiple paths to avoid flappiness and improve response time, at the cost of losing the Pareto optimality. Khalili, et. al. [48] propose OLIA, an improved version of LIA, which possesses the Pareto optimality. Using a duality model, Low, et. al. [53, 54] finds a heuristic model

in an attempt to solve the above NUM, which underlies Balia [47]. On the basis of the work in [53, 54], Peng, et. al. [47] further classify and study major MPTCP algorithms with respect to some necessary conditions to attain the NUM objectives, making another step towards the design of an optimal MPTCP. However, the proposed Balia that seeks to improve responsiveness and TCP friendliness over LIA and OLIA, is again, heuristic by design.

Meanwhile, in a series of publications, Lagoa, et. al. [66, 68, 57, 56] directly solve a generalized version of the above NUM that permits sub-flow rate constraints by means of Sliding Mode Control in control theory [55]. The resulting control laws enable multiple classes of service and require only binary congestion information feedback for control, hence are particularly suitable to serve as the theoretical underpinning for the design of MPTCPs. Wang, et. al. [69, 67] successfully apply this solution to reverse engineer TCP to arrive at the TCP utility function corresponding to the TCP Reno congestion control[4] and subsequently, design a TCP-friendly, end-to-end, single-path soft-minimum-rate-guaranteed congestion control protocol. Wang, et. al. [51] also successfully apply this solution to the development of integrated congestion control and load balancing framework for datacenter networks, including a toy example demonstrating the viability of the solution for the development of MPTCP algorithms. Our work is motivated by these results. In particular, by leveraging the results given by Lagoa, et. al. [66] and Wang, et. al. [69], and with proper selection of user utility functions for multipath flows, we are able to derive two families of NUM-optimal MPTCPs, covering both MPTCP categories.

[4]Note that this TCP utility function is the first one that captures both the slow start and congestion avoidance phases of TCP Reno

3.3 TWO FAMILIES OF NUM-OPTIMAL MPTCPs

In this section, we use the NUM-optimal multipath congestion control solution and the TCP utility function given in chapter 2, and derive EUTCP($\boldsymbol{\gamma}$) and WUTCP($\boldsymbol{\omega}$), in separate subsections.

3.3.1 The EUTCP($\boldsymbol{\gamma}$) Family

Utility function of an MPTCP flow:

Theoretically, to ensure that a NUM-optimal MPTCP is friendly to TCP Reno by design, one can simply apply the TCP Reno utility function to the total flow rate of a multipath flow as follows,

$$U(x_1, x_2, ..., x_m) = U_{tcp}(\sum_{j=1}^{m} x_j), \tag{3.3}$$

In other words, the NUM-optimal solution is to equalize user utilities by balancing the sub-flow rates among sub-flow paths (hence, is resource pooling capable), which equalizes the rate allocation among both MPTCP and TCP flows, and hence, achieve TCP-friendly resource allocation.

However, our experiment results (see Table 2) show that, just like LIA, OLIA, and Balia, the resulting MPTCP corresponding to the above utility leads to skewed flow rate allocation with MPTCP flow rates higher than TCP flow rates. We find that the reason is that, just like TCP, to allow sub-flow windows to increase when network resources become available, an MPTCP flow must maintain a positive minimum rate/window for each of its sub-flow, despite the fact that the optimal control law that underpins the MPTCP algorithm may require that the sub-flow rates be allowed to drop to zero when congestion occurs. Since TCP Reno uses 2×MSS as its minimum window size, most existing MPTCPs set the minimum window size for each sub-flow to be 2×MSS. This effectively makes the minimum window for the en-

tire MPTCP flow to be $2m\times$MSS, leading to skewed flow rate allocation in favor of MPTCP flows over TCP flows in practice, and the higher the number of sub-flows, m, the more skewed the flow rate allocation is.

One possible approach to remedying the above problem is to set the minimum window for each sub-flow to be one MSS instead of $2\times$MSS, as is the case for Balia. While our experiment shows that using one MSS in the MPTCP corresponding to the above utility can lead to almost perfect equal flow rate allocation in most cases, for some corner cases, it may cause unstable flow rate allocation. Namely, a sub-flow competing with TCP flows may not be able to grow its window back once the window reaches its minimum.

In this chapter, we tackle the above challenge by using the following family of rate-scaled user utility functions instead,

$$U(x_1, x_2, ..., x_m) = U_{tcp}(\sum_{j=1}^{m} \gamma_j x_j), \tag{3.4}$$

where γ_j is a rate-scaling coefficient for sub-flow, j, for $j = 1, ..., m$. By setting some or all of the coefficients to be slightly larger than one, the NUM-optimal rate allocation that attempts to equalize user utilities is expected to allocate less rates to MPTCP flows than TCP flows, compensating for the skewed resource allocation.

EUTCP($\boldsymbol{\gamma}$): Let $z_l(t, x_l, cg_l)$ for each subflow l take the same format as its single-path counterpart is given in Eq. (2.23) with rate scaling, i.e.,

$$z_l(t, x_l, cg_l) = \begin{cases} (\alpha + \beta)\gamma_l x_l & \text{in SSP} \\ \mu + \beta\gamma_l x_l & \text{in CAP.} \end{cases} \tag{3.5}$$

Then substitute Eqs. (3.4) and (3.5) into Eqs. (2.12) and (2.13), we arrive at EUTCP($\boldsymbol{\gamma_l}$) as follows:

In SSP:

$$\dot{x_l} = \begin{cases} \alpha\gamma_l x_l & \text{if } cg = 0 \\ -\beta\gamma_l x_l & \text{if } cg = 1. \end{cases} \tag{3.6}$$

In CAP:

$$\dot{x_l} = \begin{cases} \frac{\mu+\beta\boldsymbol{\gamma_l} x_l}{\mu+\beta\gamma_l x}\mu & \text{if } cg = 0 \\ -\frac{\mu+\beta\gamma_l x_l}{\mu+\beta\gamma_l x}\beta\gamma_l x & \text{if } cg = 1. \end{cases} \tag{3.7}$$

Considering the fact that the multiplicative increase rate is much larger than the additive increase rate, i.e., $\beta x_l \gg \mu$, and $\beta x \gg \mu$, Eq.(3.7) can be approximated as

$$\dot{x_l} \approx \begin{cases} \frac{x_l}{x}\mu & \text{if } cg = 0 \\ -\beta\gamma_l x & \text{if } cg = 1. \end{cases} \tag{3.8}$$

EUTCP($\boldsymbol{\gamma_l}$) above simply states that the subflow increase rate is proportional to the ratio of the subflow rate and the overall flow rate, i.e., x_l/x, while its decrease rate is that of TCP Reno scaled by γ_l. It degenerates to TCP Reno at $m = 1$ and $\gamma = 1$.

Now we transform the above EUTCP($\boldsymbol{\gamma}$) to a window-based one.

Let $x = W \times MSS/\tau$ and $x_l = W_l \times MSS/\tau$. Then the congestion window size change for subflow l in each RTT, ΔW_l, according to Eqs. (3.6) and (3.8), are,

In SSP:

$$\Delta W_l \approx \begin{cases} \gamma_l W_l & \text{if } cg_l = 0 \\ -\frac{\gamma_l W_l}{2} & \text{if } cg_1 = 1. \end{cases} \tag{3.9}$$

In CAP:

$$\Delta W_l \approx \begin{cases} \frac{W_l}{\tau_l \sum_{j=1}^{m} W_j/\tau_j} & \text{if } cg = 0 \\ -\frac{\gamma_l W_l}{2} & \text{if } cg = 1. \end{cases} \tag{3.10}$$

3.3.2 THE WUTCP($\boldsymbol{\omega}$) Family

Utility function of an MPTCP flow:: Now we consider the following family of utility functions,

$$U(x_1, x_2, ..., x_m) = \sum_{j=1}^{m} \omega_j U_{tcp}(x_j), \tag{3.11}$$

where ω_j is the utility weight for sub-flow j. Clearly, this family of utility functions will lead to optimal rate allocation without resource pooling. For example, at $\omega_j = 1$ for $\forall j$, each sub-flow in a multipath flow is then treated the same way as a single path TCP flow, the most naive MPTCP solution in the category without resource pooling.

WUTCP($\boldsymbol{\omega}$): Let,

$$z_l(t, x_l, cg_l) = \begin{cases} (\alpha + \beta)x_l & \text{in SSP} \\ \mu_l + \beta x_l & \text{in CAP.} \end{cases} \tag{3.12}$$

Here $\mu_l = MSS/\tau_l$ is the additive increase rate for subflow l. Similarly, we can easily get WUTCP($\boldsymbol{\omega}$) as follows.

In SSP, we have,

$$\dot{x_l} = \begin{cases} \frac{3}{2}(1 - 3^{-\omega_l})\alpha x_l & \text{if } cg_l = 0 \\ -3^{1-\omega_l}\beta x_l & \text{if } cg_l = 1. \end{cases} \tag{3.13}$$

In CAP, we have,

$$\dot{x_l} = \begin{cases} [1 - (\frac{\beta x_l}{\mu_l + \beta x_l})^{\omega_l}](\mu_l + \beta x_l) & \text{if } cg_l = 0 \\ -(\frac{\beta x_l}{\mu_l + \beta x_l})^{\omega_l}(\mu_l + \beta x_l) & \text{if } cg - l = 1. \end{cases} \tag{3.14}$$

Similarly, considering $\beta x_l \gg \mu_l$, then Eq. (3.14) can be approximated as

$$\dot{x}_l \approx \begin{cases} \omega_l \mu_l & \text{if } cg = 0 \\ -\beta x_l & \text{if } cg = 1. \end{cases} \tag{3.15}$$

From Eq. (3.15), we know that the rate increase for a subflow l is proportional to the utility weight ω_l. If $\omega_l < 1$, the subflow increase rate is smaller than that in a single-path TCP flow (i.e.,$\omega_l \mu_l$ vs μ_l). If we set $\omega_l < 1$ for any subflow l, then each subflow obtains no more flow rate than that of a single path TCP in a shared link. WUTCP($\boldsymbol{\omega}$) can allocate more bandwidth to a multipath flow than the best case single-path TCP in some cases, but it may also allocate less bandwidth to a multipath flow than the best case single-path TCP in other cases, as we shall see later.

Now we convert WUTCP($\boldsymbol{\omega}$) above into a window-based one.

In SSP, we have,

$$\Delta W_l = \begin{cases} \frac{3}{2}(1 - 3^{-\omega_l}) W_l & \text{if } cg_l = 0 \\ -\frac{3^{1-\omega_l}}{2} W_l & \text{if } cg_l = 1, \end{cases} \tag{3.16}$$

and in CAP, we have,

$$\Delta W_l \approx \begin{cases} \omega_l & \text{if } cg = 0 \\ -\frac{W_l}{2} & \text{if } cg = 1. \end{cases} \tag{3.17}$$

3.4 PERFORMANCE EVALUATION

In this section, we evaluate the performance of EUTCP($\boldsymbol{\gamma}$) and WUTCP($\boldsymbol{\omega}$) against some well-known MPTCPs, including EWTCP [49], LIA [46], OLIA [48] and Balia [47]. We first present, compare and analyze different congestion control algorithms and then perform experimental testing of those algorithms focusing on the fairness, responsiveness, and throughput performance metrics.

3.4.1 Algorithm Analysis

In the following algorithm analysis, we only analyze CAP for all the MPTCP algorithms for two reasons. First, the existing MPTCPs are mainly focused on the design and analysis of CAP, assuming that SSP for individual sub-flows follows that of TCP. Second, as aforementioned, the retransmission timeout is treated the same way as three duplicated ACKs in our solutions, meaning that once entering CAP after initial SSP, the system will stay in CAP, and hence, the long-run flow rate allocation is determined by CAP only.

For the convenience of comparison, Table 3.1 lists the congestion control algorithms in CAP for all the MPTCPs to be studied in this section.

Table 3.1: Congestion control algorithms in CAP

Solution	**Increase** (per Ack)	**Decrease** (per loss)	**Comments**
With Resource Pooling			
LIA (Coupled) [17, 70, 46]	$\frac{(W_l/\tau_l^2)}{(\Sigma_{i\epsilon R} W_i/\tau_i)^2}$	$\frac{W_l}{2}$	
Semicoupled [47] (optimal)	$\frac{1}{\tau_l(\Sigma_{i\epsilon R} W_i/\tau_i)}$	$\frac{W_l}{2}$	
OLIA [48]	$\frac{(W_l/\tau_l^2)}{(\Sigma_{i\epsilon R} W_i/\tau_i)^2} + \frac{\delta}{W_l}$	$\frac{W_l}{2}$	δ is modulation factor
Balia [47]	$\left\{ \frac{(W_l/\tau_l^2)}{(\Sigma_{i\epsilon R} W_i/\tau_i)^2} \right\} (\frac{1+\alpha_l}{2})(\frac{4+\alpha_l}{5})$	$\frac{W_l}{2} min\{\alpha_l, \frac{3}{2}\}$	$\alpha_l \triangleq \underset{j\epsilon R}{max}\{x_j\}/x_l$
EUTCP($\boldsymbol{\gamma}$) (optimal)	$\frac{1}{\tau_l(\Sigma_{i\epsilon R} W_i/\tau_i)}$	$\frac{\gamma_l W_l}{2}$	γ_l is the scaling coefficient
Without Resource Pooling			
EWTCP [49] (optimal)	$\frac{\alpha}{W_l}$	$\frac{W_l}{2}$	α is weight for each route
WUTCP($\boldsymbol{\omega}$) (optimal)	$\frac{\omega_l}{W_l}$	$\frac{W_l}{2}$	ω_l is the subflow utility weight

The algorithms belonging to the two distinct categories, i.e., with or without resource pooling capability, are listed separately. Note that the increasing part of each algorithm is given on a per ACK basis, not per RTT. As a result, the per-RTT-based window increase formula in Eq. (3.10) and Eq. (3.17) must be divided by W_l to arrive at the corresponding increase parts in Table 3.1.

For the category with resource pooling capability, we first observe that the Semicoupled algorithm is indeed a family member of EUTCP($\boldsymbol{\gamma}$), i.e., EUTCP ($\{\mathbf{1}\}$), meaning that it is NUM-optimal, achieving the global objective given in Eq. (3.4) at $\gamma_l = 1$, $\forall l$. Second, we note that compared with all the other algorithms in the category, the EUTCP($\boldsymbol{\gamma}$) family including the Semicoupled are the least complex and hence, most computationally efficient ones. This also implies that if some other algorithms also turn out to be NUM optimal, the corresponding utility functions are likely to be substantially more complex and hence, harder to interpret, particularly with respect to the fairness to TCP.

For the category without resource pooling, it becomes clear that EWTCP is indeed a family member of WUTCP($\boldsymbol{\omega}$) with $\omega_l = \alpha$, $\forall l$, and hence, is NUM optimal as well. In fact, in the original paper on EWTCP [49], it is suggested that $\alpha = 1/m^2$ should be used. By doing so, the paper shows that each sub-flow of an EWTCP flow sharing a bottleneck link with a TCP flow will then be allocated one mth of the TCP flow rate and hence is TCP fair, in the sense that the overall EWTCP flow rate is equal to the TCP flow rate if each and every sub-flow shares a bottleneck link with a TCP flow. The proof, however, is based on a TCP throughput model given in [71] that assumes that the retransmission timeout is a more frequent event than the three duplicated ACKs and the TCP goes back to the slow start phase after the retransmission timeout happens.

In fact, since we now know that EWTCP is NUM-optimal and its utility function is given by Eq. (3.11) with $\omega_l = \alpha$ for $l = 1, ..., m$, we can prove that to satisfy the above TCP fairness criterion, α should be set at $1/m$, instead of $1/m^2$. First, we note that since $\beta x \gg \mu$, U_{tcp} in Eq. (3.11) can be approximately written as $U_{tcp} \approx \frac{\mu}{\beta} log(\beta x)$, i.e., a log function of x. Then it can be easily shown [51] that for any given number of sub-flows, each having the weighted utility, αU_{tcp}, which share a bottleneck link with any given number of TCP flows, the sum of the utilities for all the flows/sub-flows sharing this link is maximized if each sub-flow is allocated α times of the rate allocated to each TCP flow. This means that to meet the above TCP fairness criterion for EWTCP, α should be $1/m$, not $1/m^2$. This is also confirmed by the experiment results presented in the following section.

Nevertheless, the work in this chapter is not meant to give a definitive answer as to what fairness criteria and hence, what parameters, $\boldsymbol{\omega}$ and $\boldsymbol{\gamma}$, or equivalent, which family members of the two families, should be adopted in practice. Instead, the objective of this work is to reveal the performance tradeoffs among the members of the two families as well as the other MPTCPs in the list so that users can make an informed decision as to which MPTCP should be adopted to best serve their application needs. For example, in a multi-homing scenario, different paths may charge different usage fees and/or offer different service qualities. In this case, a user may want to assign different ω_l's or γ_l's for different sub-flows to fully explore the tradeoffs between the performance and cost.

3.4.2 Experimental Analysis

For the ease of comparison with Balia [47], the state-of-the-art solution, we adopt the same network topology as the one used in [47]. Namely, all the test cases are performed based on the topology shown in Fig. 3.2. It involves two source hosts,

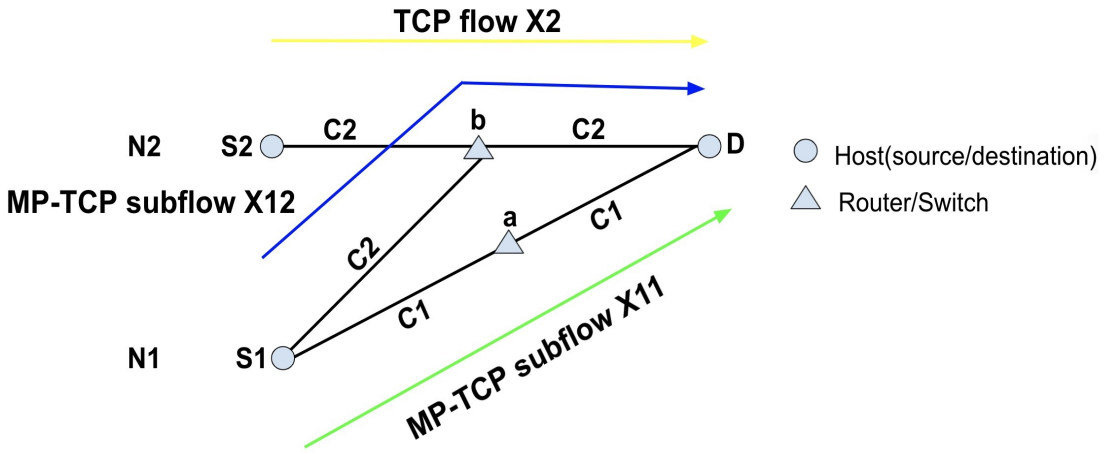


Figure 3.2: Network topology for MPTCP Performance Evaluation

S_1, sending N_1 MPTCP flows with two sub-flows (i.e., $m = 2$), x_{11} and x_{12}, and S_2, sending N_2 TCP flows, to the same destination host D. Nodes, b and a, provide a single path from S_1 to D via b and two sub-flow paths from S_2 to D, via a (i.e., sub-flow x_{12}) and b (i.e., sub-flow x_{11}), respectively.

The hosts (i.e., S_1, S_2, and D) are Dell Poweredge servers, each equipped with 8-core processors with 10GB memory and running Linux 16.04. Nodes a and b are Dell N4032F switches, each with multiple 1 Gbps Ethernet interfaces running Ubuntu 16.04.1 LTS (Linux kernel 4.19.98). The link bandwidth for all the links can be configured at any rate lower than or equal to 1 Gbps through the networking interface traffic control command *tc*, allowing for the testing of the MPTCP responsiveness to sudden link bandwidth changes. Both MPTCP families are implemented by modifying the open-source Linux kernel codes of LIA and Balia [72]. The source code of both families will be made available in the public domain, upon the publication of the work.

For both EUTCP($\boldsymbol{\gamma}$) and WUTCP($\boldsymbol{\omega}$) families, we only test the single parameter cases, i.e., the cases where $\gamma_l = \gamma$ and $\omega_l = \omega$, $\forall l$. Hence, they can be simply written in terms of a single parameter, i.e., EUTCP(γ) and WUTCP(ω). We present the results for the two families and the related MPTCPs in the two categories, separately, focusing on the fairness, throughput, and responsiveness performance metrics.

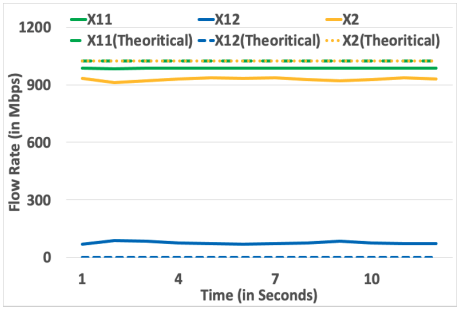
3.4.3 EUTCP(γ)

In this section, we study the performance of three family members, EUTCP(1), EUTCP(1.05), and EUTCP(1.1), together with LIA, OLIA, and Balia.

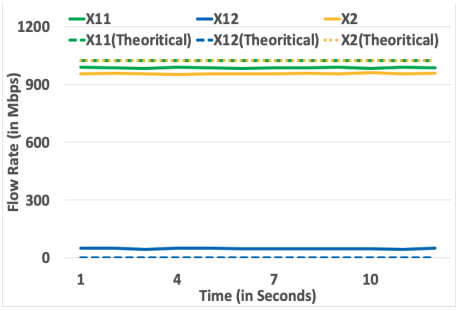
TCP Fairness and Throughput: First, we test the performance of these MPTCPs in a balanced network by setting $C_1 = C_2 = 1024$ Mbps (i.e. 1 Gbps link) and $N_1 = N_2 = 1$, i.e., one multipath flow and one TCP flow (the TCP Reno in the Linux kernel is applied without modification). With this setup, all the MPTCPs will strive to equalize and maximize the flow rates for the two flows, i.e., all targeting at the optimal flow rate allocation: $x_1 = x_2 = 1024$ Mbps, $x_{11} = 1024$ Mbps and $x_{12} = 0$ Mbps.

In this experiment, we consider steady state performance. The two flows are long-lived, meaning that each flow has an unlimited amount of data to send and hence, lasts throughout the entire measurement window. Each flow reaches its stable rate quickly and in Figs. 3.3 (a)-(f) we present the results in steady state for LIA, OLIA, Balia, EUTCP(1), EUTCP(1.05), and EUTCP$(1, 1)$, respectively.

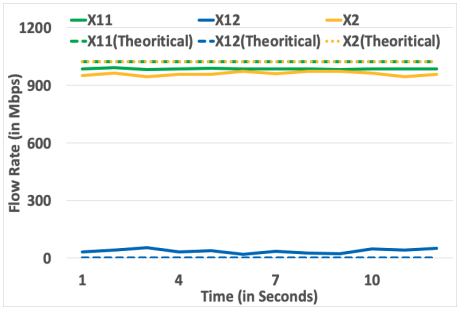
First, we note that for all MPTCPs, x_{11} (green) and x_2 (yellow) are below their respective optimal flow rate targets, i.e., the link bandwidths, whereas x_{12} (blue) are above its optimal flow rate target that is zero. This is inevitable. The former is due to the discrete-time (once every RTT) window-based adaptive flow control with delay, which guarantees that the achievable flow rate cannot saturate the link bandwidth.



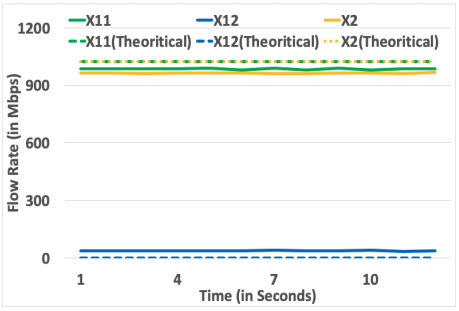
(a) LIA



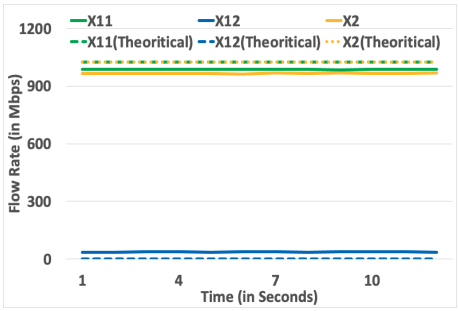
(b) OLIA



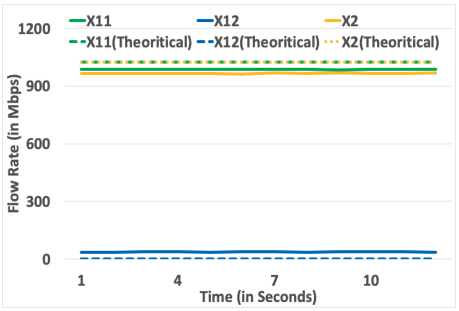
(c) Balia



(d) EUTCP(1)



(e) EUTCP(1.05)



(f) EUTCP(1.1)

Figure 3.3: Steady State Performance for MPTCPs in the resource-pool-capable category

The latter is caused by the need to set a non-zero minimum window for a sub-flow as explained earlier, which prevents it from being allocated zero bandwidth. This also contributes to the reduction of the flow rate, x_2. Moreover, all the MPTCPs are able to achieve almost the same flow rate allocation for x_{11}. This is easy to understand as the entire sub-flow path is dedicated to this sub-flow without congestion.

Second, we note that in terms of flow rate allocation for x_2 and x_{12}, LIA offers the lowest performance among all, OLIA and Balia perform almost equally well, but not as good as the three family members in the EUTCP(γ) family, all of which perform equally well. A point to make here is that EUTCP(1) or the Semicoupled algorithm given in [47] turns out to perform better than Balia, which does not agree with the finding in [47]. This, however, may be because in [47], the Semicoupled algorithm tested is the one given in [46], not the one listed in [47] and tested in our chapter (see footnote one for more explanation).

Table 3.2: Average flow throughput (Mbps), TCP unfairness, and aggregate throughput (Multiple flows of MPTCP flows and single path TCP flows combined) (Mbps) at $N_1 = N_2 = 5$.

	MPTCP Throughput	TCP Throughput	unfairness (%)	Aggregate Throughput
LIA	1061.92	926.99	14.56	1988.91
OLIA	1032.78	955.33	8.11	1988.11
Balia	1022.01	961.48	6.29	1983.49
EUTCP(1)	1023.54	963.92	6.18	1987.46
EUTCP(1.05)	1022.07	965.69	5.83	1987.75
EUTCP(1.1)	1034.19	953.87	8.42	1988.06

In our next experiment, we study the average throughput and TCP fairness performance for long-lived MPTCP flows in steady state. The experiment setup is the same as the previous one, except now we have, $N_1 = N_2 = 5$. To quantify the TCP fairness, we define a TCP unfairness measure in Eq. (3.18). The smaller the TCP unfairness is, the fairer the MPTCP is with respect to TCP.

$$TCP\ Unfairness = \frac{per\ MPTCP\ flow\ throughput - per\ TCP\ flow\ throughput}{per\ TCP\ flow\ throughput} \times 100\ (\%). \tag{3.18}$$

For each MPTCP algorithm, we repeat the experiment three times and take the average throughput. The results are shown in Table 3.2, including average flow throughput for both multipath flows and TCP flows, the TCP unfairness, and the aggregate throughput, i.e., the throughput for TCP and multipath flows combined. As one can see, LIA is the least fair to TCP with TCP unfairness of 14.56%. OLIA and Balia significantly improve the TCP fairness over LIA, with TCP unfairness of 8.11% and 6.29%, respectively, at the cost of almost negligible deduction of the aggregate throughput, consistent with the results given in [47].

Interestingly enough, again we observe that the Semicoupled algorithm or EUTCP(1) offers better performance than Balia in both TCP fairness and aggregate throughput. Moreover, EUTCP(γ) reaches the smallest TCP unfairness at about $\gamma = 1.05$ and then grows back up as γ further increases, as evidenced by the results for EUTCP(1.05) and EUTCP(1.1). The reason is that increasing γ from one makes the sub-flow window for x_{12} drop faster as the congestion occurs (see the decreasing part of the algorithm in Table 3.1), reducing the skewed flow rate allocation and hence, improving TCP fairness. However, as the sub-flow window decreases faster, it also increases faster in the absence of congestion, simply because the sub-flow window increase rate is inversely proportional to the sum of sub-flow window sizes (see the increasing part of the algorithm in Table 3.1). These two competing effects result in the existence of a γ value around 1.05, where the TCP unfairness is minimized, i.e., about 5.83%, the smallest among all MPTCPs studied, without scarifying the throughput performance.

In fact, the differences of the aggregate throughput among all the MPTCP algorithms in Table 3.2 are extremely small and can be largely neglected. Furthermore, we note that EUTCP(1.05) improves EUTCP(1) over the TCP fairness performance

by about 6%, implying that EUTCP(γ) is not too sensitive to the setting of γ and setting γ anywhere close to 1.05 should be good.

The above test results clearly demonstrate that EUTCP(γ) with γ taking a value in the range of $[1, 1.1]$ offers the best performance among all the MPTCPs in the resource-pool-capable category.

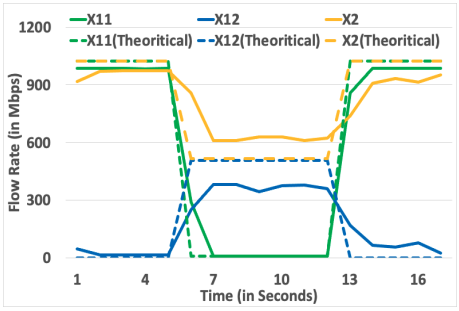
Responsiveness: Next, we test the performance of these MPTCPs in terms of responsiveness in a dynamically changing environment. To this end, we consider $N_1 = N_2 = 1$ and set the link bandwidth $C_1 = C_2 = 1024$ Mbps in the initial first 5 seconds; then suddenly change C_2 from 1024 Mbps to $C_2 = 8$ Mbps for the next 7 seconds (i.e., from second 6 to second 12); and finally switch it back to $C_2 = 1024$ Mbps. With this setup, all the MPTCPs will strive to arrive at the following flow rate allocation that equalizes and maximizes the flow rates for the two flows: $x_{11} = x_2 = 1024$ Mbps and $x_{12} = 0$ Mbps, before the 6th second and after 12th second, and $x_{11} = 8$ Mbps, $x_{12} = 508$ Mbps and $x_2 = 516$ Mbps from the 6th second to 12th second.

The throughput of the flows is given in Fig. 3.4 and the flow rate convergence times for all the MPTCPs except OLIA upon the bandwidth changes are given in Table 3.3.

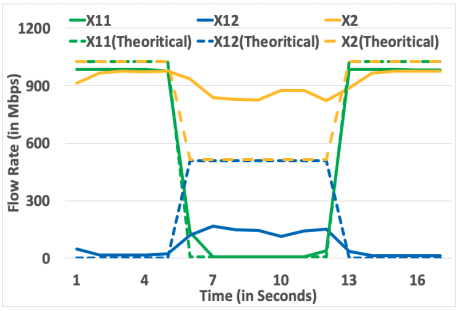
Table 3.3: Convergence times (seconds) of MPTCP flows during bandwidth changes

	LIA	Balia	EUTCP(1)	EUTCP(1.05)	EUTCP(1.1)
x_{11}	2	2	2	2	2
x_{12}	2.5	2.5	2	1.5	2

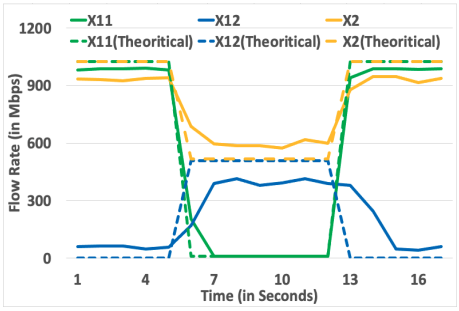
First, we note that among all the MPTCPs, OLIA is the least responsive, even though it outperforms LIA in terms of TCP fairness. It does not even come close to the new optimal flow rate allocation, compared with all the other MPTCPs, which



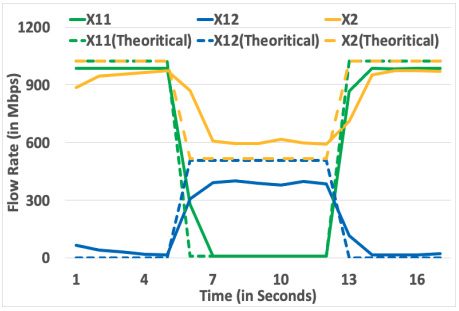
(a) LIA



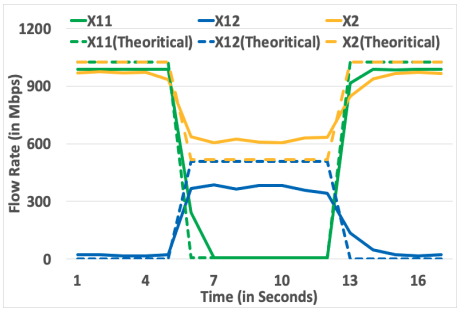
(b) OLIA



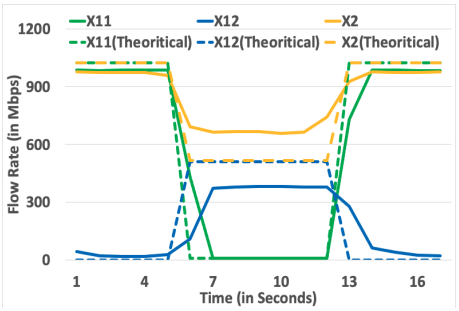
(c) Balia



(d) EUTCP(1)



(e) EUTCP(1.05)



(f) EUTCP(1.1)

Figure 3.4: Responsiveness under sudden link bandwidth changes for MPTCPs in the resource-pooling-capable category

is the reason why it is excluded from Table 3.3 for convergence comparison. LIA, Balia, EUTCP(1.1), and EUTCP(1) (or semicoupled algorithm) are on par with one another. Clearly, EUTCP(1.05) performs the best among all MPTCPs with the smallest convergence time.

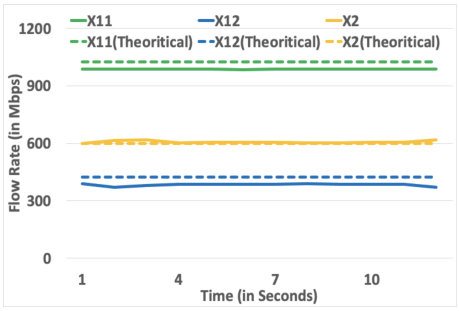
However, one may notice that for all the MPTCPs, x_{12} (x_2) converges to a flow rate lower (higher) than the optimal one, similar to the experimental results given in [47]. We find that this skewed flow rate allocation in favor of the single-path TCP is largely caused by the way such protocols are normally implemented. A TCP flow, whether it is single-path or multipath is normally run by a single thread. As a result, the processing resource allocated to the window control for each sub-flow in a multipath flow reduces, and hence, the processing delay for each sub-flow increases, as the number of sub-flow paths increases. For a sub-flow shares its sub-flow path with a TCP flow (e.g., x_{12} and x_2 in our case), it sees a larger RTT than the TCP flow as it incurs a larger processing delay, hence, receiving lower flow rate allocation. This is particularly problematic in an experimental environment of ours, where the entire testbed is hosted in a single rack, where the end-to-end propagation delay is negligible, making the RTT and hence, the performance highly sensitive to the processing delay. The following experiment results will further confirm this is indeed the case. A possible way to fix this problem is to assign more processing resources to MPTCP flows than TCP flows. How much more should be assigned to MPTCP flows, however, is an implementation issue to be addressed in the future.

Based on the above performance analysis, we conclude that the family members in EUTCP(γ) with γ taking values in the range of $[1, 1.1]$ offer the best tradeoffs among TCP fairness, responsiveness, and throughput in the resource-pooling-capable category.

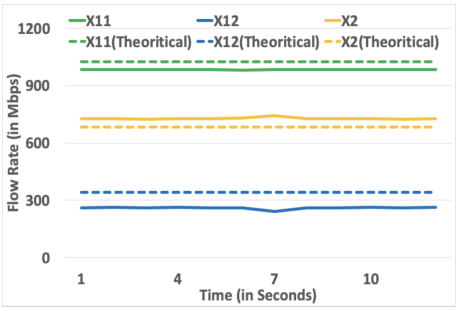
3.4.4 WUTCP($\boldsymbol{\omega}$)

Now we present the test results for the WUTCP(ω) family including EWTCP or WUTCP($\frac{1}{m^2}$) in terms of TCP fairness, throughput, and responsiveness.

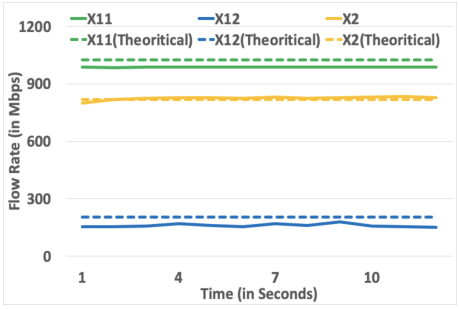
TCP fairness and Throughput: Again, we first consider the following network setup: $C_1 = C_2 = 1024$ Mbps and $N_1 = N_2 = 1$. As aforementioned, the optimal rate allocation for a WUTCP($\boldsymbol{\omega}$) sub-flow is approximately ω times of a TCP flow sharing the same bottleneck link. Hence with this setup, the optimal flow rate allocation is: $x_{11} = 1024$ Mbps, $x_{12} = 1024 \times \frac{\omega}{1+\omega}$ Mbps and $x_2 = 1024 \times \frac{1}{1+\omega}$ Mbps.



(a) WUTCP($1/\sqrt{m}$)



(b) WUTCP($1/m$)



(c) WUTCP($1/m^2$)

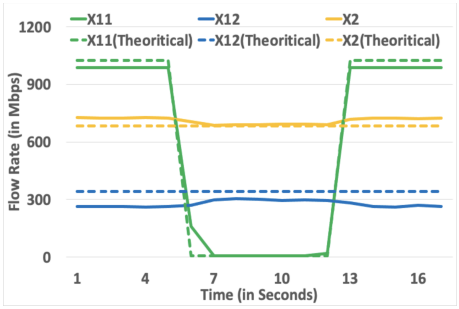
Figure 3.5: Steady State Performance for WUTCP(ω) Family

Fig. 3.5 (a), (b), and (c) plot the results for the EUTCP(ω) members at $\omega = \frac{1}{\sqrt{m}}$, $\omega = \frac{1}{m}$ and $\omega = \frac{1}{m^2}$, respectively. Again, without experiencing any congestion, the sub-flow rate, x_{11} (green), is almost the same for all three cases. Also for all the cases, the flow rate allocated to, x_2 (x_{12}), is always slightly higher (lower) than the corresponding optimal one, for the same reason discussed in the previous case.

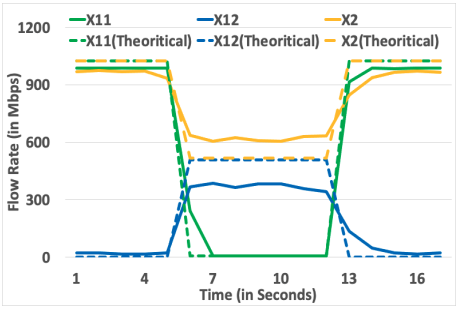
The results clearly demonstrate that the case with $\omega = 1/m = 1/2$ indeed gives flow rate allocation much closer to the desired ratio, $\frac{x_{12}}{x_2} = \frac{1}{2}$ than the case with $\omega = 1/m^2 = 1/4$, which instead gives the ratio, $\frac{x_{12}}{x_2} = \frac{1}{4}$ with pretty high accuracy as our model predicts. These results further support our earlier claim that to satisfy the TCP fairness criterion given in the chapter on EWTCP [49], the weight α should be set at $\alpha = \frac{1}{m}$, not $\frac{1}{m^2}$.

Responsiveness: As shown in Fig.3.1 and also discussed in Section 3.2, MPTCPs without resource pooling capability, including WUTCP(ω), cannot respond to network dynamics effectively, especially in terms of maintaining TCP fairness. In what follows, we only give the experiment results on WUTCP($\frac{1}{m}$) to demonstrate this.

Consider the same experimental setup as the previous responsiveness testing case for EUTCP(γ). The only difference is that now the MPTCP flow, x_1, is run by WUTCP($\frac{1}{m}$).



(a) WUTCP(1/m)



(b) EUTCP(1.05)

Figure 3.6: Responsiveness of WUTCP($\frac{1}{m}$) vs EUTCP(1.05) to sudden changes of a link bandwidth

As shown in Fig. 3.6 (a), as C_1 drops from 1024 Mbps to 8 Mbps, x_{12} and x_2 change slightly. In fact, in theory, they should not change at all because without

resource pooling capability, EUTCP(γ) controls the send windows for individual sub-flows independently, as evidenced by the EUTCP(γ) algorithm given in Table 1. The fact that both x_{12} and x_2 moved closer to their respective optimal values during the time period where $C_1 = 8$ Mbps further confirms that the actual processing resources allocated to individual sub-flows have an impact on the flow rate allocation. When C_1 drops from 1024 Mbps to 8 Mbps, the transmission delay for packets in the sub-flow x_{11} increases substantially, resulting in more than 10$\times$ increase of its RTT, as the measured statistics show. This means that the processing resource demand for sub-flow, x_{11}, is also reduced by that many times, yielding much of the processing resource to sub-flow, x_{12}. This effectively reduces the RTT for x_{12} and hence, the competitiveness of x_{12} with respect to TCP flow, x_2, resulting in reduced skew for flow rate allocation during the time period when $C_1 = 8$ Mbps. Similar behaviors are also observed for other family members in EUTCP(γ). This case study also confirms the earlier claim that MPTCP without resource pooling may perform worse than the best case single-path TCP.

In contrast, as plotted in Fig. 3.6 (b) again for the case of EUTCP(1.05), the resource pooling capability of EUTCP(1.05) helps to balance the flow rate allocation to maintain an equal share of the flow rate allocation, or to meet the same fairness criterion, in response to C_1 changes.

Based on the above performance analyses, we summarize the performance in terms of fairness and responsiveness in Table 3.4. Note that the aggregate throughputs for different MPTCPs are close to one another as shown in Table 3.2 and hence the throughput is not included as a performance metric in the table. From the summary given in this table, we conclude that EUTCP(γ) with γ in the range of $[1, 1.1]$ offer the best overall performance among all the MPTCPs tested in this chapter.

Table 3.4: Performance comparison of MPTCPs

Solutions	**TCP fairness**	**Responsiveness**
With Resource Pooling		
LIA (Coupled)	poor	medium
Semicoupled or EUTCP(1)	good	fast
OLIA	fair	poor
Balia	good	medium
EUTCP(1.05)	very good	very fast
EUTCP(1.1)	fair	fast
Without Resource Pooling		
EWTCP or WUTCP($\frac{1}{m^2}$)	poor	fast
WUTCP($\frac{1}{m}$)	fair	fast

3.5 Conclusions

In this chapter, we derive and implement in Linux two distinct families of Network Utility Maximization (NUM)-optimal Multiple Path Transmission Control Protocol (MPTCP) protocols, EUTCP($\boldsymbol{\gamma}$) and WUTCP($\boldsymbol{\omega}$), by leveraging the TCP utility function and the NUM solution for concave utilities, respectively. EUTCP($\boldsymbol{\gamma}$) is a function of a rate-scaling vector of the sub-flow rate-scaling coefficients, $\boldsymbol{\gamma}$ and WUTCP($\boldsymbol{\omega}$) is a function of a utility weight vector of the sub-flow weights, $\boldsymbol{\omega}$. We also show that the Semicoupled algorithm is a protocol in the family of EUTCP($\boldsymbol{\gamma}$) with $\boldsymbol{\gamma} = 1$ and EWTCP is a protocol in the family of WUTCP($\boldsymbol{\omega}$) with $\boldsymbol{\omega} = 1/m^2$, where

m is the number of sub-flow paths, and hence, are NUM-optimal. The performance of the two families when coexisting with TCP is also analyzed based on experiment in a testbed. The test results demonstrate that the family members with equal sub-flow rate-scaling coefficient setting in the range of $[1, 1.1]$ in EUTCP($\boldsymbol{\gamma}$) outperform three well-known MPTCPs with resource pooling capability, including LIA, OLIA and Balia, in terms of responsiveness and fairness and are on par with the three MPTCPs in terms of the throughput performance.

CHAPTER 4

HYBRID MULTIPATH CONGESTION CONTROL

4.1 INTRODUCTION

A widely accepted design objective for MPTCP is three-pronged[46]: (i) the overall flow rate for an MPTCP flow should be at least as high as the highest flow rate a single-path TCP can achieve using any of the sub-flow paths of the MPTCP flow, and the single-path TCP flow that achieves this flow rate is called the best single-path TCP flow; (ii) to be fair to a single-path TCP, the total flow rate for any number of subflows of an MPTCP flow sharing a bottleneck link with a single-path TCP flow should not exceed the flow rate of the single-path TCP flow; and (iii) an MPTCP flow must be able to balance the load among subflow paths [46]. Notable MPTCPs that meet the three-pronged objective include Semi-coupled, LIA, OLIA, and Balia [46, 49, 47, 48]. In this chapter, we generally call these MPTCPs Equal bandwidth shared MPTCP (EMPTCP) because they tend to distribute the flow rates evenly among all flows, regardless of how many subflows there are in a flow.

In this chapter, we argue that the above three-pronged design objective should be augmented with yet another aspect, making it a four-pronged design objective, i.e. to provide a mechanism to incentivize users to use the protocol. Although the first of the three prongs of the above objective does provide some incentive for a user to use EMPTCP over single-path TCP, an EMPTCP may end up discouraging users from using it. This is simply because different paths used by an EMPTCP may be owned by different Internet service providers based on different pricing structures. For example, mobile devices, such as cell phones, are equipped with both WiFi connectivity and

cellular data service (e.g., 3G/4G/5G) most of the time. Usually, the WiFi connection is without usage fee and the mobile data service may charge a usage fee based on the amount of data sent/received. Considering the scenario of a social gathering, e.g., a family party, where many guests may want to use their mobile phones to browse the Internet, watch online videos, sending/receiving messages, and so on, via the host's WiFi network. This may result in low flow rates seen by and hence, poor Internet experiences for individual guests. When this happens, a guest with a cellular data service may be tempted to turn on an EMPTCP that meets the three-pronged design objective, thinking that by doing so, his/her cellular data service may help prop up the bandwidth needed to gain good experience at the cost of paying a small amount of cellular data service fee. In reality, however, he/she may end up using the cellular data service almost entirely and receiving a hefty bill later. This may well be the case because an EMPTCP may attain the desired flow rate using the cellular connection only, hence giving up much of the free/low-cost bandwidth on the WiFi side. Obviously, this resource allocation is unfair to the guest who uses EMPTCP, and hence, would discourage him/her from using EMPTCP again in the future. This example clearly indicates that to incentivize users to use MPTCP in the face of multiple paths with different pricing structures, the three-pronged design objective should be augmented to a four-pronged one, with the fourth one being subflow path pricing structure aware.

A naive solution is to simply use a weighted single-path TCP for each subflow and assign a heavier weight to a subflow with a lower price. In this chapter, we call this kind of MPTCP weighted MPTCP (WMPTCP). Note that an earlier MPTCP protocol, known as equal-weight MPTCP [49] is a special case of WMPTCP, which assigns the same weight to all the subflow paths. Again, using the above scenario as an example, one may assign, e.g., 75% and 25% weights to subflows using WiFi

and cellular, respectively. By doing so, the guest who uses WMPTCP can get 75% of the single-path TCP flow rate from the WiFi side for sure. In the meantime, it can still compete for the bandwidth on the cellular side, but much less aggressively. Unfortunately, however, WMPTCP does not meet the first and the third prongs of the three-pronged design objective, meaning that it may lead to the overall flow rate lower than that of the best single-path TCP flow and cannot balance the load.

In this chapter, we propose a new MPTCP that meets the four-pronged design objective. The idea is to combine an EMPTCP with a WMPTCP to come up with a hybrid MPTCP (H-MPTCP). H-MPTCP leverages the ability of WMPTCP to allow price-aware subflow path rate allocation and the ability of EMPTCP to meet the three-pronged objective, hence resulting in its ability to achieve the four-pronged design objective. We implement the proposed protocol in Linux Kernel based on the open-source MPTCP source codes [72]. Extensive test results demonstrate that H-MPTCP can indeed achieve the four-pronged design objective. In the meantime, it outperforms EMPTCP, WMPTCP, and some well-known MPTCP protocols (i.e., LIA[46] and Balia[47]) in terms of throughput and responsiveness.

4.2 Background and Motivations

A straightforward but naive approach to end-to-end MPTCP design is to simply run independent end-to-end TCP flows as subflows on different paths. This approach, however, is too aggressive and unfair to single-path TCP flows, and cannot balance the loads among subflow paths. This leads to the design of EWTCP [49], a member of the WMPTCP family. EWTCP attempts to achieve TCP fairness by modifying the previous approach, i.e., reducing the TCP window increase rate by a factor of $w_l = 1/S^2$ for all S TCP-based subflows. However, besides the lack of load balancing capability inherited from the previous approach, EWTCP may lead to inefficient use

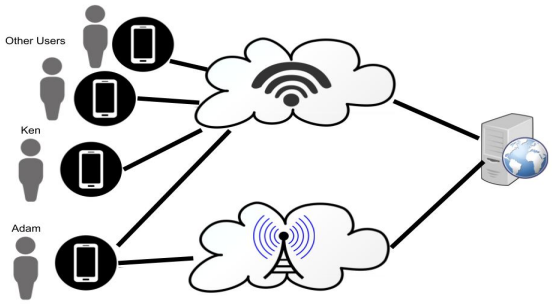


Figure 4.1: A family party example network

of networks and flow rates lower than the best single-path TCP. The shortcomings of EWTCP further lead to the design of a coupled congestion control algorithms, also known as the Linked Increases Algorithm (LIA) [65], [46]. LIA is purposely designed to meet the three-pronged design objective and standardized by IETF. OLIA [48] improves over LIA in terms of the Pareto optimality. More recently, semi-coupled and Balanced linked adaptation (Balia) algorithm [47] were proposed to strike a good balance between TCP-friendliness, responsiveness, and window oscillation, especially to further improve responsiveness when network condition changes. All these solutions (i.e., LIA, OLIA, semi-coupled, and Balia) meet the three-pronged design objective and hence, are members of the EMPTCP family.

However, neither the WMPTCP family nor the EMPTCP family is capable of achieving the four-pronged design objective. To demonstrate this, we take a closer look at the family party example. Consider the network topology shown in Fig.4.1. Adam is a MPTCP user who can use both WiFi free of charge and cellular with 10 cents per unit bandwidth per hour charge. Ken is a single-path TCP user who only uses WiFi for free. So, we have two users where Adam and Ken can both use WiFi service and Adam can also use cellular service. Assume that the maximum data rates for both Adam's and Ken's applications are 100 bandwidth units. The link band-

Table 4.1: Flow rate allocations for the example

Scenario	**WiFi=Cellular=100**				**Cellular=5, WiFi=100**			
	Adam			Ken	Adam			Ken
Subflows	Cellular	WiFi	Total	WiFi	Cellular	WiFi	Total	WiFi
Best-case TCP	100($10)	N.A.	100	100	N.A.	50	50	50
Equal MPTCP	100($10)	0	100	100	5($0.5)	47.5	52.5	52.5
Weighted MPTCP	57 ($5.7)	43	100	57	5($0.5)	43	48	57

widths for both WiFi and cellular connections represent the capacity of the bottleneck link bandwidths of the connections. Now, we consider two different scenarios, where in the first scenario, both WiFi and cellular connections can support up to 100 units of bandwidth, and in the second scenario where WiFi has 100 units of bandwidth but cellular has only 5 units of bandwidth. For both scenarios, EMPTCP will attempt to equalize the flow rate allocation between both users, whereas WMPTCP will attempt to allocate flow rates in proportion to their weights. For WMPTCP, we further assume that the weights assignments are 0.75 and 0.25 for WiFi and cellular, respectively. This will allow the subflow using the WiFi network to be allocated roughly 75% of the flow rate of a single-path TCP flow on the same network, resulting in a potentially much reduced flow rate needed for the subflow on the cellular network side.

Table 4.1 gives the flow rate allocation and costs for both Adam and Ken as well as the best single-path TCP flow when Adam uses either EMPTCP or WMPTCP for both scenarios. For the first scenario on the left, EMPTCP for Adam equalizes the flow rate allocation by fully utilizing the cellular connection, yielding to the single-path TCP flow for Ken completely on the WiFi connection. This results in Adam paying for the full usage fee of the cellular link of $10 per hour, whereas Ken enjoys the same flow rate performance for free. In contrast, WMPTCP that has a weight of 0.75 for WiFi and 0.25 for cellular is able to rip 43% of the WiFi link bandwidth and

57% of cellular link bandwidth, resulting in the same total flow rate of 100 units, as the cost of $5.7 per hour, more than 40% lower than the case of EMPTCP. In other words, in this scenario, WMPTCP offers a better incentive to users to use MPTCP than EMPTCP by reducing the usage costs. These weights may be set as a function of the pricing models of individual subflow paths to further incentivize users to use them.

For the second scenario on the right in Table 4.1, to equalize the flow rate allocation, EMPTCP has to take up a much bigger chunk of the bandwidth from WiFi connection, i.e. 47.5, to be exact. So, EMPTCP is able to achieve a total bandwidth of 52.5 units. In contrast, with the weight of 0.75, WMPTCP still can only grab 43% of the bandwidth from a WiFi connection, leaving it with a total flow rate, 48 units, lower than both EMPTCP and the flow rate of 50 for the best single-path TCP.

The above example clearly demonstrates the need for a new MPTCP, which motivates us to propose H-MPTCP. As we shall demonstrate, in scenario 1, H-MPTCP will automatically select WMPTCP over EMPTCP, whereas in scenario 2, it will automatically select EMPTCP. In either case, H-MPTCP results in better than the best single-path TCP performance at a lower cost than EMPTCP, hence, meeting the four-pronged design objective.

Based on the discussion so far and the performance data to be presented in the later sections, Table 4.2 provides a summary of the features for some notable end-to-end MPTCP protocols as well as H-MPTCP. As one can see, H-MPTCP possesses the most desirable features among all MPTCPs.

Finally, we note that there are other works that focus on specific aspects of the MPTCP protocol design challenges, e.g., bottleneck detection [73]−[74] and packet

Table 4.2: MPTCP Design objectives and performance parameters

	Design Objectives				**Performance Evaluation**
Solutions	Best-case TCP	TCP Friendliness	Load Balancing	Adoption Incentive	Responsiveness
EWTCP[49]	No	Yes	No	Yes	High
LIA [46]	Yes	Yes	Yes	No	Low
Balia[47]	Yes	Yes	Yes	No	Medium
H-MPTCP	Yes	Yes	Yes	Yes	High

scheduling [75]−[76]. However, they are not concerned with new end-to-end MPTCP protocol design, which is the main focus of this chapter.

4.3 Hybrid Multipath Congestion Control

In this section, we first briefly describe EMPTCP and WMPTCP. Then we introduce H-MPTCP.

4.3.1 EMPTCP

For EMPTCP, the congestion window size (W_l) for subflow (l) in each RTT in the slow start phase is the same as that in TCP Reno. The congestion window change for subflow l in each RTT in the congestion avoidance phase is,

$$\Delta W_l = \begin{cases} \frac{W_l}{\tau_l \sum_{j=1}^{S} W_j/\tau_j} & \text{if } cg = 0 \\ -\frac{W_l}{2} & \text{if } cg = 1. \end{cases} \tag{4.1}$$

where W_j and τ_j are the congestion window size and RRT for subflow j ($j = 1, ..., S$). In this chapter, we choose semicoupled mentioned in Balia[47] for EMPTCP, but in general, we can use any members in the EMPTCP family.

From Equation (4.1), we know that the subflow increase rate is proportional to the ratio of the subflow rate with the overall flow rate, i.e., x_l/x while its decrease

rate is the same as that in TCP Reno. It means that the subflow rate increase is slower for a multipath flow with a higher flow rate, or the network tries to evenly allocate the flow rate to all flows. For single-path TCP, $W_l = W$, then the congestion window change degenerates to single-path TCP Reno.

4.3.2 WMPTCP

For WMPTCP, the change in congestion window size (W_l) for subflow (l) in each RTT in the slow start phase is the same as that in TCP Reno in the slow start phase. The congestion window change for subflow l with weight (ω_l) in each RTT in the congestion avoidance phase is,

$$\Delta W_l = \begin{cases} \omega_l & \text{if } cg = 0 \\ -\frac{W_l}{2} & \text{if } cg = 1. \end{cases} \tag{4.2}$$

From Equation (4.2), we know that the rate increase for a subflow l is proportional to the weight ω_l. If $\omega_l < 1$, the subflow increase rate is smaller than that in a single path TCP flow. If we set $\omega_l < 1$ for any subflow l, then each subflow obtains no more flow rate than that of a single path TCP in a shared link. WMPTCP can allocate more bandwidth than the best single-path TCP in some cases, but it may not guarantee the rate of WMPTCP flow always be no less than the best single TCP flow rate as shown in Table 4.1. If $\omega_l = 1$ for any subflow l, the congestion window change of each subflow is the same as in TCP Reno. In this case, a multipath flow is just the combination of S individual single TCP flows.

4.3.3 H-MPTCP

To meet the four-pronged design objective, we now design H-MPTCP. We set the weight $1/S \leq 1$ for each subflow of an MPTCP flow with S subflows. With

such weight assignment, each subflow can get no more flow rate than that of a single TCP flow rate for a shared link. But the overall flow rate of an MPTCP may have a chance to get more rate than its best single-path TCP flow. In the slow start phase, H-MPTCP behaves the same as TCP-Reno. In the congestion avoidance phase, H-MPTCP selects the larger subflow increase rate from the increased rates of EMPTCP and WMPTCP, i.e.,

$$\Delta W_l = max(\Delta W_l^{Equal}, \Delta W_l^{Weighted}), \tag{4.3}$$

where ΔW_l^{Equal} and $\Delta W_l^{Weighted}$ are the congestion window increases in each RTT defined in Equation. (4.1) and Equation. (4.2) in case of no congestion, respectively.

Under "normal" situation, e.g., scenario one in the example given in Section 4.2, WMPTCP is likely to be automatically selected because it is more responsive/aggressive than EMPTCP [47]. EMPTCP will take over only under "abnormal" situations, e.g., scenario two in Section 4.2, when WMPTCP fails to reach flow rate equal to or higher than the best single-path TCP. In this case, EMPTCP will ensure that the flow rate will be balanced to further improve the flow rate performance.

In summary, in the slow start phase, we have,

$$\Delta W_l = \begin{cases} W_l & \text{if } cg_l = 0 \\ -\frac{W_l}{2} & \text{if } cg_1 = 1. \end{cases} \tag{4.4}$$

and in the congestion avoidance phase, we have,

$$\Delta W_l = \begin{cases} max(\omega_l, \frac{W_l}{\tau_l \sum_{j=1}^{S} W_j/\tau_j}) & \text{if } cg = 0 \\ -\frac{W_l}{2} & \text{if } cg = 1. \end{cases} \tag{4.5}$$

4.4 Performance Evaluation

In this section, we evaluate the proposed H-MPTCP compared to the EMPTCP and WMPTCP and the existing MPTCP solutions LIA [46] and Balia [47] in Linux

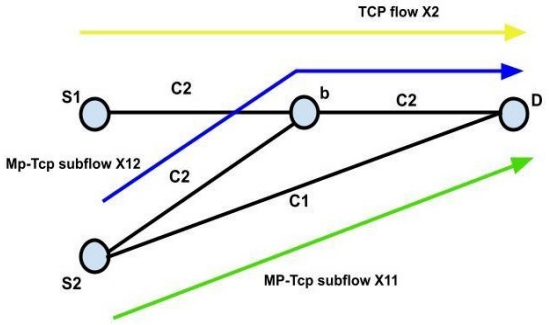


Figure 4.2: Network Topology for H-MPTCP Performance Evaluation

kernel implementation on a testbed. The hosts (i.e., S_1, S_2, and D) are Dell Poweredge servers, each equipped with 8-core processors with 10 GB memory and running Linux 16.04. Node b is a Dell N4032F switch with multiple 1 Gbps Ethernet interfaces running Ubuntu 16.04.1 LTS (Linux kernel 4.19.98). The link bandwidth for all the links can be configured at any rate lower than or equal to 1 Gbps through the networking interface traffic control command *tc*, allowing for the testing of the MPTCP responsiveness to sudden link bandwidth changes. H-MPTCP is implemented by modifying the open-source Linux kernel codes of LIA and Balia [72].

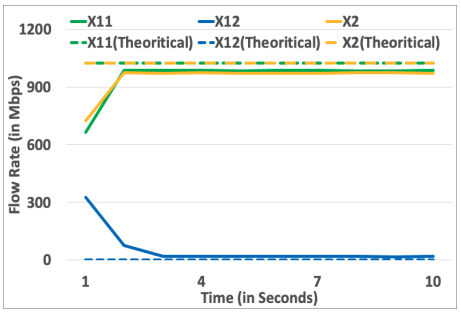
We use the network topology as shown in Fig. 4.2. In the network, source S_1 has a single path TCP flow x_2 running TCP Reno, and source S_2 have an MPTCP flow with two subflows x_{11} and x_{12}. Subflow x_{12} representing the Wi-Fi link shares the link from node b to destination D with flow x_2. We set different bandwidth combinations of C_1 and C_2 to test the flow rate allocation in different MPTCP solutions. This network topology is also adopted in Balia [47] for the testing of their solution.

We test the performance of the proposed protocol in three different cases, each has a different network setup (i.e., different bandwidth C_1 and C_2). For the test, each source has a huge data to send to the destination D, and each flow can be viewed as an infinite data flow during our test (i.e., the test is finished before the

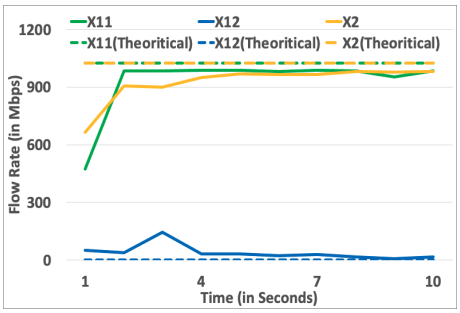
data is fully sent out). In this case, the MPTCP user is trying to get the maximum bandwidth available to improve the flow completion time. Each flow reaches its stable rate quickly and we present the first 10-second results in the first two cases and the first 20-second for the third case. The weight values in WMPTCP is set as 1/2 for both subflows x_{11} and x_{12}. Flow x_2 and the two subflows of x_1 (i.e., x_{11} and x_{12}) start at the same time in each test.

Case 1: First we test the performance of MPTCP solutions in a symmetric network by set $C_1 = C_2 = 1024$ Mbps (i.e. 1 Gbps link). With this setup, the rate allocation for EMPTCP are $x_{11} = 1024$ Mbps, $x_{12} = 0$ Mbps (i.e., the overall flow rate $x_1 = 1024$ Mbps) and $x_2 = 1024$ Mbps and for WMPTCP are $x_{11} = 1024$ Mbps, $x_{12} \approx 341$ Mbps (i.e., the overall flow rate $x_1 \approx 1365$ Mbps) and $x_2 \approx 683$ Mbps, respectively.

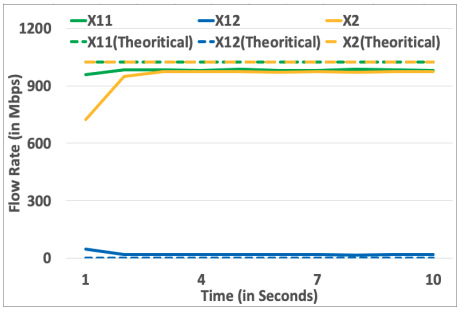
Fig. 4.3 shows the results of rate allocation (the average rate in a second) in the proposed protocol compared with EMPTCP, WMPTCP, LIA, and Balia. The rate of each flow/subflow is also presented using a dashed line with the same color as the real measured rate in an MPTCP solution. As LIA and Balia have equal sharing of the bandwidth, they achieve similar rate allocations as EMPTCP, and hence the theoretical rates in EMPTCP are also listed in LIA and Balia results for comparison. From the results, we know that the rate allocations in EMPTCP and WMPTCP are closely matched to their corresponding theoretical rates with little lower rates. The results indicate that EMPTCP and WMPTCP can really achieve the rate allocation. The lower rate is due to the discrete-time control and network condition feedback delay, these make the protocols unable to achieve full bandwidth usage. LIA and Balia also achieve their design objective, i.e., sharing network bandwidth to all flows as even as possible. In this case, WMPTCP allows the MPTCP flow x_1 to obtain a higher flow rate than the best single-path TCP flow. As WMPTCP has a higher flow



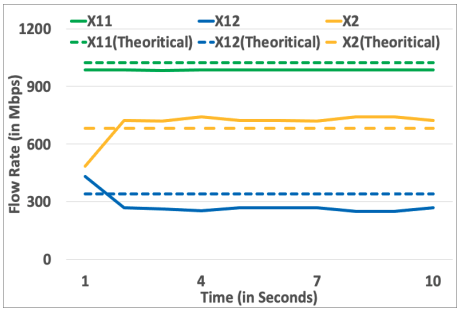
(a) LIA



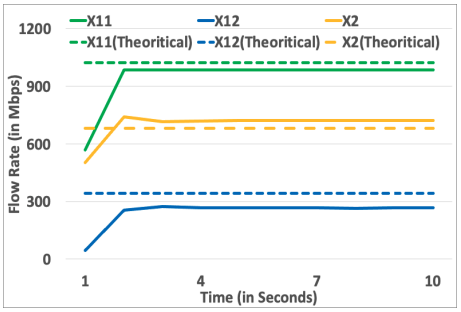
(b) Balia



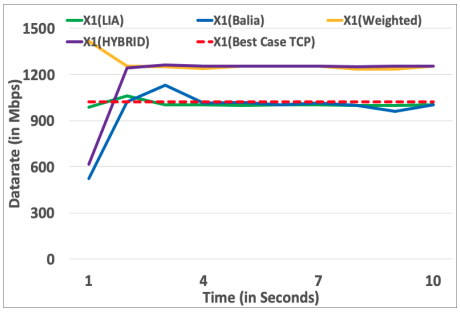
(c) EMPTCP



(d) WMPTCP



(e) H-MPTCP



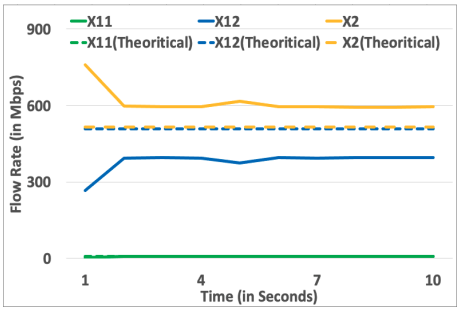
(f) MPTCP flow rate comparison

Figure 4.3: Performance for H-MPTCP in Case 1

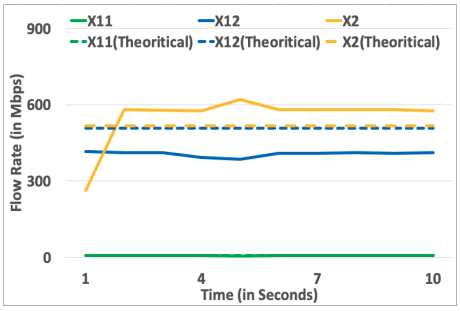
rate than EMPTCP, H-MPTCP selects WMPTCP and hence achieves the same flow rate allocation as that in WMPTCP (see Fig. 4.3 (e)).

The overall flow rate of the MPTCP flow (i.e., x_1) and its theoretical best single-path TCP flow rate (denoted as dashed red line) are shown in Fig. 4.3 (f). From the results, we can see that all EMPTCP, LIA, and Balia are just achieve close to the theoretical best single-path TCP flow rate. Although they make flow x_2 to get a

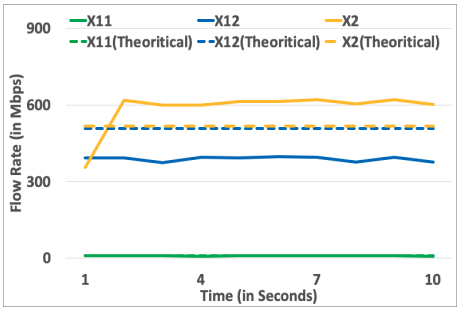
higher flow rate, they may have no incentives for the usage of MPTCP. WMPTCP/H-MPTCP achieves a higher MPTCP flow rate than the best single flow rate. It also benefits flow x_2 in case x_1 was using a single path TCP as congestion control and chooses the path x_{12}. Hence WMPTCP/H-MPTCP gives more incentives to users to apply MPTCP.



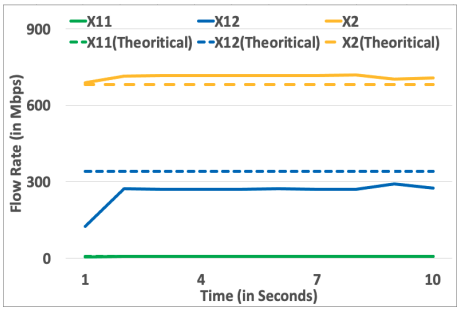
(a) LIA



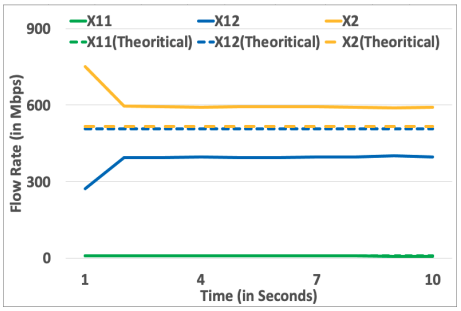
(b) Balia



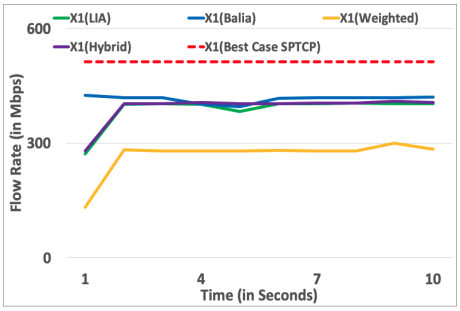
(c) EMPTCP



(d) WMPTCP



(e) H-MPTCP

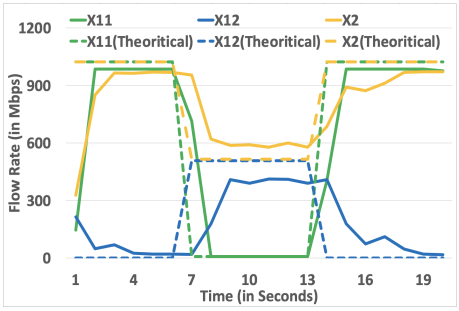


(f) MPTCP flow-rate comparison

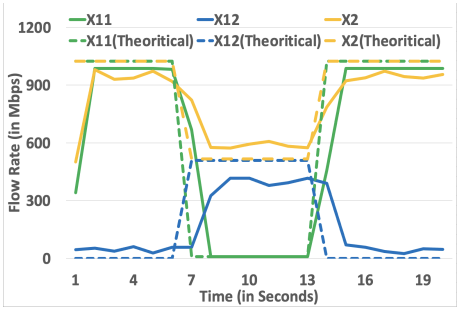
Figure 4.4: Performance for H-MPTCP in Case 2

Case 2: Now we test the performance of MPTCP solutions in an asymmetric network by set $C_1 = 8$ Mpbs and $C_2 = 1024$ Mbps. With this setup, the rate allocation for EMPTCP are $x_{11} = 8$ Mbps, $x_{12} = 508$ Mbps (i.e., the overall flow rate $x_1 = 516$ Mbps) and $x_2 = 516$ Mbps and for WMPTCP are $x_{11} = 8$ Mbps, $x_{12} \approx 341$ Mbps (i.e., the overall flow rate $x_1 \approx 349$ Mbps) and $x_2 \approx 683$ Mbps, respectively. In this case, x_1 in WMPTCP has a lower flow rate than that of the best single path TCP flow. As EMPTCP can achieve a higher flow rate, H-MPTCP selects EMPTCP to increase the rate and hence achieves the same flow rate allocation as that in EMPTCP, as shown in Fig. 4.4 (c) and Fig. 4.4 (e). Fig. 4.4 shows the results of rate allocation in the MPTCP solutions. Again, we see that the rate allocations of EMPTCP/H-MPTCP and WMPTCP closely match their corresponding theoretical rates. From Fig. 4.4 (f), we can see that all the protocols can still get more rates (516 vs. 512 Mbps theoretically) than that of the best single-path TCP. Although the path S_2 to D has very limited bandwidth but EMPTCP, H-MPTCP, LIA, and Balia can still balance the rates and benefit the single path flow x_2. This indicates that MPTCP can be useful to balance the traffic and increase network utilization. From both case 1 and case 2, we know that H-MPTCP can guarantee an MPTCP flow to achieve at least the best single-path TCP flow rate as that in EMPTCP while it can try to obtain higher flow rate as many as possible and hence can encourage users to use it.

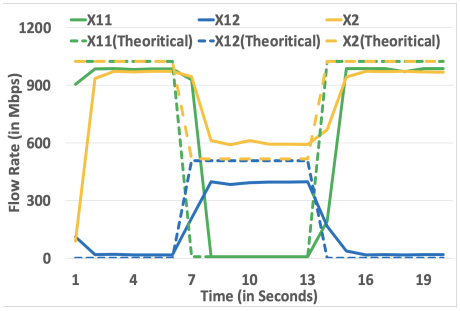
Case 3: Finally, we test the performance of the proposed protocol in a dynamic network environment to see the responsiveness to network changes. In this case, we set the link bandwidth $C_1 = C_2 = 1024$ Mbps in the first 6 seconds and change the bandwidth $C_2 = 8$ Mbps in the next 7 seconds (i.e., from second 7 to second 13), and then C_2 is switched back to 1024 Mbps. As this setup, the rate allocation in the proposed protocol during the first 6 seconds and after the second 13 are the same



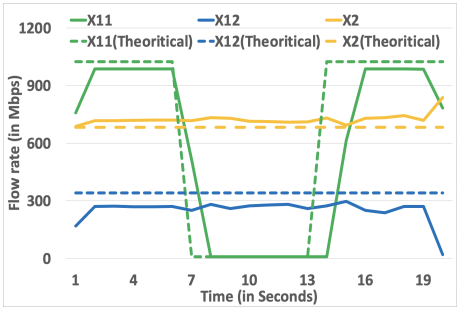
(a) LIA



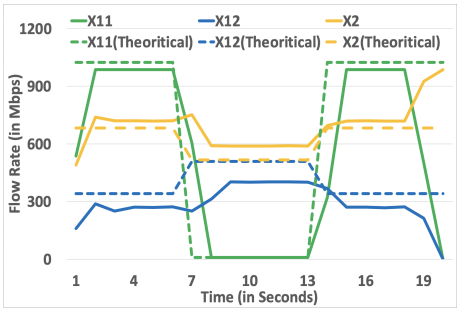
(b) Balia



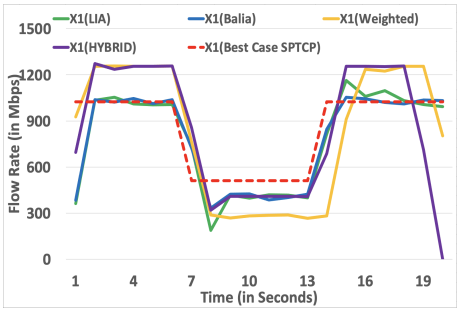
(c) EMPTCP



(d) WMPTCP



(e) H-MPTCP



(f) MPTCP flow-rate comparison

Figure 4.5: Performance for H-MPTCP in Case 3

as that in case 1, and the rate allocation during the time period between the second 6 to second 13 is the same as that in case 2. For H-MPTCP, it always selects the higher flow from EMPTCP and WMPTCP, and hence it selects WMPTCP congestion control in the first 6 seconds and after the second 13 and chooses EMPTCP during the time period from second 6 to second 13.

Fig. 4.5 shows the results of rate allocation in the five MPTCP solutions. From the results, we see that EMPTCP, WMPTCP, and H-MPTCP can closely match their rates in this dynamic network environment. Now let us look at the flow rate change during the network bandwidth changes.

First, we note that the subflow rate x_{12} in WMPTCP has no change, because the weight for the subflow is only dependent on the number of subflows, and hence the flow rate allocation in the shared link does not change.

Second, in EMPTCP, LIA and Balia, we can see that the subflow rate of x_{11} (denoted as a green line) drops to 8 Mbps almost the same time in all the three protocols, this is because subflow x_{11} does not compete with the bandwidth with any other flow/subflow. As the subflow rate x_{11} is reduced, the rate of subflow x_{12} is then increased in all three protocols, and hence the flow rate x_2 should be reduced because of the shared bandwidth with x_{12}. From Fig. 4.5 (a), Fig. (b) 4.5, and Fig. 4.5 (c), we can see that the transition time for flow x_2 (i.e., yellow line) dropped to its new balanced rate has almost the same time in the three protocols.

Third, from Fig. 4.5 (a), Fig. (b) 4.5, and Fig. 4.5 (c), we know that the transition time for subflow x_{12} (i.e., blue line) reaching its new balanced rate is different in the three protocols. In LIA and Balia, the transition time takes about 3 seconds from second 6 to about second 9 while in EMPTCP, the transition time is about 2 seconds from second 6 to about second 8. This indicates that EMPTCP can be more quickly to catch up with the network condition change to reach the new balanced rate than LIA and Balia. This is because Semicoupled has higher responsiveness as compared to LIA and Balia as mentioned in the Balia paper[47].

Fourth, H-MPTCP switches from WMPTCP at second 6 and switches back to Weighted-MPCTP at second 13. The transition time in H-MPTCP is similar to that in EMPTCP as shown in Fig. 4.5 (d).

From the results, we can also see that H-MPTCP always chooses the higher flow rate from EMPTCP and WMPTCP, and hence benefits the user to apply MPTCP. The results show that EMPTCP/H-MPTCP can quickly respond to the network condition change and reaches its new balanced state.

Through the three-case studies, we conclude that the proposed H-MPTCP solution is ready to be applied in today's Internet and can meet the MPTCP design goals and give more incentives for users to apply the MPTCP solutions.

4.5 Conclusion

In this chapter, we propose a Hybrid-MPTCP(H-MPTCP)that always achieves a higher flow rate from EMPTCP and WMPTCP and provides a built-in mechanism that can encourage users to apply it over other MPTCPs as well as single-path protocols with different pricing structures. Extensive real Linux implementation test results verify that the proposed H-MPTCP can indeed achieve the design objectives.

CHAPTER 5

OPTIMAL-DCTCP

5.1 INTRODUCTION

Datacenter networks are significantly different from the public Internet. The round trip time (RTT) for a datacenter network is usually much smaller (less than 200 μs versus tens to hundreds of milliseconds in the public Internet). As a result, a data flow may consume a huge amount of bandwidth at very low packet latency. TCP is particularly ineffective in such a high-bandwidth-low-latency environment, causing excessive packet queuing delay and packet losses, especially in the presence of incast congestion [2]. On one hand, DCTCP [2] manages to address the above shortcomings of TCP by employing the ECN mechanism for earlier congestion detection, allowing switches to maintain short queues and hence, reducing packet latency. However, DCTCP is an empirical solution. Therefore, in this chapter, we revisit this state-of-the-art datacenter protocol and derived the NUM-optimal congestion control protocol for DCTCP using our HOLNET framework. First, we give some background of the existing state-of-the-art DCTCP protocol and the reason for it not being NUM-optimal. Then we propose the new NUM-optimal DCTCP(O-DCTCP) and prove that with a simple modification of the congestion indicator, DCTCP can be turned into a NUM-optimal traffic control protocol. Finally, we discuss the future work and conclusions for this chapter.

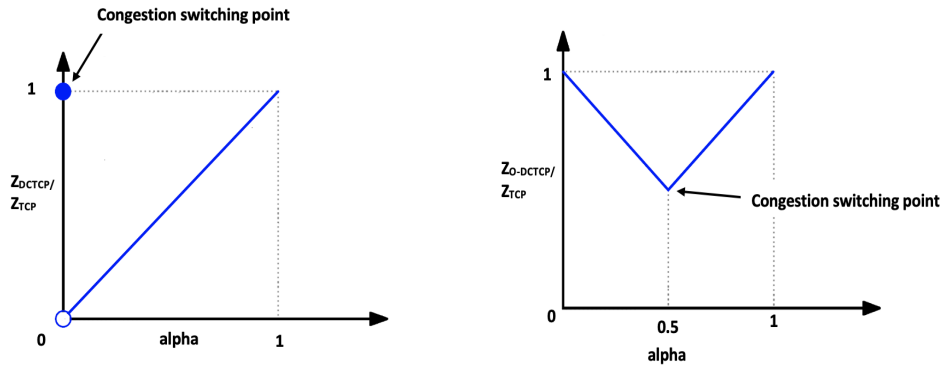


Figure 5.1: z(.) function comparison for (a) Current DCTCP vs (b) Optimal-DCTCP

5.2 Background

Here we analyze the current DCTCP algorithm. DCTCP uses α, i.e., the fraction of packets that are marked in one RTT as a measurement of the degree of congestion. DCTCP relies on α and reacts in proportion to the extent of congestion, not its presence. The current DCTCP traffic control algorithm according to [2] can be defined as

$$\dot{x} = \begin{cases} \mu & if \quad cg = 0 \\ -\alpha\beta x & if \quad cg = 1 \end{cases} \tag{5.1}$$

5.3 Optimal-DCTCP

In this section, we use the HOLNET framework and the TCP utility function given in chapter 2 and derive a new NUM-Optimal Utility function for DCTCP.

Using Eq. (2.22), i.e. TCP utility function as the base utility in the CAP is given by,

$$U(x) = (\frac{\mu}{\beta} + x)[log(\mu + \beta x) - 1] - x[log(\beta x) - 1], \tag{5.2}$$

where μ is the additive increase rate and βx is the multiplicative decrease.

We take the first order derivative of (5.2) as,

$$\frac{dU(x)}{dx} = [[log(\mu+\beta x)-1]*1+(\frac{\mu}{\beta}+x)*(\frac{\beta}{\mu+\beta x})]-[[log(\beta x)-1]*1+x*\frac{\beta}{\beta x}] \quad (5.3)$$

$$\frac{dU(x)}{dx} = [[log(\mu+\beta x)-1]+(\frac{\mu+\beta x}{\beta})*(\frac{\beta}{\mu+\beta x})]-[[log(\beta x)-1]+1] \quad (5.4)$$

$$\frac{dU(x)}{dx} = [[log(\mu+\beta x)-1]+1]-[[log(\beta x)] \quad (5.5)$$

$$\frac{dU(x)}{dx} = [log(\mu+\beta x)-log(\beta x)] \quad (5.6)$$

$$\frac{dU(x)}{dx} = log(\frac{\mu+\beta x}{\beta x}). \quad (5.7)$$

.

Now, using Eq. (5.7) in Eq. (2.13) gives us

$$f(x) = 1-e^{-log(\frac{\mu+\beta x}{\beta x})}, \quad (5.8)$$

$$f(x) = 1-e^{log(\frac{\beta x}{\mu+\beta x})}, \quad (5.9)$$

$$f(x) = 1-\frac{\beta x}{\mu+\beta x}, \quad (5.10)$$

$$f(x) = \frac{\mu}{\mu+\beta x}, \quad (5.11)$$

Let $z_{DCTCP}(t, x, cg)$ be the positive piecewise continuous scalar function for DCTCP. Even though DCTCP is empirical in design, we use the congestion control laws defined by HOLNET NUM and define the $z(.)$ function for the current DCTCP as

$$z_{DCTCP}(t, x, cg) = z_{tcp}(t, x, cg) * (\alpha cg + \bar{cg}). \tag{5.12}$$

For DCTCP, $cg = 0$ when α =0 and $cg = 1$ when $\alpha > 0$. So, the defined $z(.)$ function is discontinuous at instants the congestion status changes (i.e., from congestion to non-congestion and vice versa) as shown in Figure 5.1 (a). Since the $z(.)$ function is discontinuous, the current DCTCP's optimality cannot be proved according to [55].

Next, we show the $z(.)$ function for O-DCTCP.

$$z_{O-DCTCP}(t, x, cg) = z_{tcp}(t, x, cg) * (\alpha cg + (1 - \alpha)\bar{cg}), \tag{5.13}$$

where z_{tcp} is the z-function of the TCP control law in CAP as derived in Eq. (2.23). As shown in Figure 5.1 (b), the point of congestion status change is now for α =0.5. So the z(.) function remains a continuous function and hence its optimality can be proven.

Now by substituting Eq. (5.13) and (5.11) into Eq. (2.12), we derive the new OPTIMAL-DCTCP conegstion control procotol as,

$$\dot{x} \quad = \quad \begin{cases} (1 - \alpha)\mu & if \quad cg = 0(\alpha < 0.5) \\ -\alpha\beta x & if \quad cg = 1(\alpha >= 0.5) \end{cases} \tag{5.14}$$

for the CAP.

The main difference between the existing and the new O-DCTCP lies in the $z(.)$ function. The current DCTCP goes to the decreasing part as ECN bits are marked, i.e., $\alpha > 0$. Changing the point of congestion status change (i.e., from congestion to

non-congestion and vice versa) to $\alpha = 0.5$ makes the $z(.)$ function continuous and makes the DCTCP protocol NUM-optimal.

5.4 Future Work and Conclusion

Based on the HOLNET framework, we revisit the state-of-the-art, but empirically designed datacenter protocol, i.e. DCTCP [2] and propose a new NUM-optimal traffic control protocol called Optimal-DCTCP(O-DCTCP). We plan to focus on further testing of the proposed solution for O-DCTCP in handling different kinds of workloads and different datacenter topologies.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This dissertation aimed at addressing challenges with existing datacenter traffic control design space being point and empirical in design. We reformulated the NUM framework to fully harness its potential in terms of existing, rich distributed solutions, to enable large families of traffic control protocol for datacenter applications. We presented HOLNET, a holistic traffic control framework for datacenter networks. HOLNET allows large families of traffic control protocols of various degrees of sophistication to be developed. Protocols in each family developed under HOLNET share a common, user-defined global optimization objective. As a result, the protocols in each family can be fairly compared and carefully selected to fully explore the performance, scalability, and design complexity tradeoffs. As an example, we develop HOLNET-UTA, a family of integrated congestion controllers and load balancers, maximizing the sum of weight TCP utilities. The large-scale simulation demonstrates the backward compatibility and flexibility of HOLNET.

Using the HOLNET framework, the dissertation also introduces two distinct families of Network Utility Maximization (NUM)-optimal Multiple Path Transmission Control Protocol (MPTCP) protocols, i.e., Equal Utility MPTCP with sub-flow rate-scaling vector, $\boldsymbol{\gamma} = \{\gamma_1, ..., \gamma_m\}$ (EUTCP($\boldsymbol{\gamma}$)) and Weighted Utility MPTCPs with sub-flow weight vector, $\boldsymbol{\omega} = \{\omega_1, ..., \omega_m\}$ (WUTCP($\boldsymbol{\omega}$)), where m is the number of sub-flow paths. The performance of the two families when coexisting with TCP is also analyzed based on an experiment in a testbed.

This dissertation also proposes a Hybrid-MPTCP (H-MPTCP) that always achieves a higher flow rate from EMPTCP and WMPTCP and provides a built-in mechanism that can encourage users to apply it over other MPTCPs as well as single-path protocols with different pricing structures.

In the last part of this dissertation, we revisit another state-of-the-art datacenter protocol that is empirically designed, i.e. DCTCP, and present how to make this protocol NUM-optimal.

In our next steps, we plan to focus on further testing of the proposed solution for NUM-optimal DCTCP. In our future work, we also plan to work on a NUM-optimal Hybrid-MPTCP solution.

REFERENCES

[1] S. Ghorbani, Z. Yang, P. B. Godfrey, Y. Ganjali, and A. Firoozshahian, "DRILL: Micro Load Balancing for Low-latency Data Center Networks," in *Proceedings of ACM SIGCOMM*, 2017.

[2] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *Proceedings of ACM SIGCOMM*, 2010.

[3] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 133–145, 2000.

[4] C. E. Hopps, "Analysis of an Equal-Cost Multi-Path Algorithm," in *RFC 2992*.

[5] B. Vamana, J. Hasan, and T. Vijakumar, "Deadline-Aware Datacenter TCP (D^2TCP)," in *Proceedings f ACM SIGCOMM*, 2012.

[6] D. Han, R. Grandl, A. Akella, and S. Seshan, "Fcp: A flexible transport framework for accommodating diversity," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 135–146, 2013.

[7] K. Nagaraj, D. Bharadia, H. Mao, S. Chinchali, M. Alizadeh, and S. Katti, " NUMFabric: Fast and Flexible Bandwidth Allocation in Datacenters," in *Proceedings of the 14th ACM SIGCOMM*, 2014.

[8] M. Alizadeh, T. Edsall, S. Dharmapurikar, R. Vaidyanathan, K. Chu, A. Fingerhut, V. T. Lam, F. Matus, R. Pan, N. Yadav, and G. Varghese, " CONGA: Distributed Congestion-Aware Load Balancing for Datacenters," in *Proceedings of ACM SIGCOMM*, 2014.

[9] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving Datacenter Performance and Robustness with Multipath TCP," in *Proceedings of ACM SIGCOMM*, 2011.

[10] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Noore, G. Antichi, and M. Wojcik, "Re-architecting datacenter networks and stacks for low latency and high performance," in *Proceedings of ACM SIGCOMM*, 2017.

[11] M. Alizadeh, S. Yang, M. Sharif, and S. Katti, "pFabric: Minimal Near-Optimal Datacenter Transport," in *Proceedings of ACM SIGCOMM*, 2013.

[12] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and W. Sun, "PIAS: Practical Information-Agnostic Flow Scheduling for Data Center Networks," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, 2014.

[13] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron, "Better Never than Late: Meeting Deadlines in Datacenter Networks," in *Proceedings of ACM SIGCOMM*, 2011.

[14] L. Chen, K. Chen, W. Bai, and M. Alizadeh, "Scheduling Mix-flows in Commodity Datacenters with Karuna," in *Proceedings of ACM SIGCOMM*, 2016.

[15] D. Acemoglu, R. Johari, and A. Ozdaglar, "Partially Optimal Routing," *IEEE Journal of Selected Areas of Communications*, vol. 25, pp. 1148–1160, 2007.

[16] L. Qui, Y. R. Yang, Y. Zhang, and S. Shenker, "On Selfish Routing in Internet-like Environments," in *Proceedings of ACM SIGCOMM*, 2003.

[17] F. Kelly and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 2, pp. 5–12, 2005.

[18] Y. Liu, H. Zhang, W. Gong, and D. F. Towsley, "On the interaction between overlay routing and underlay routing," in *Proceedings of IEEE INFOCOM*, 2005.

[19] S. Sen, D. Shue, S. Ihm, and M. J. Freedman, "Scalable, optimal flow routing in datacenters via local link balancing," in *Proceedings of ACM CoNEXT*, 2013.

[20] C. Lee, C. Park, K. Jang, S. Moon, and D. Han, "Accurate latency-based congestion feedback for datacenters," in *Proceedings of USENIX ATC*, 2015.

[21] M. Alizadeh, A. Kabbani, T. Edsall, and B. Prabhakar, "Less is More: Trading a little Bandwidth for Ultra-Low Latency in the Data Center," in *Proceedings of USENIX NSDI*, 2012.

[22] I. Cho, K. Jang, and D. Han, "Credit-Scheduled Delay-Bounded Congestion Control for Datacenters," in *Proceedings of ACM SIGCOMM*, 2017.

[23] P. X. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, and S. Shenker, " pHost: Distributed near-optimal datacenter transport over commodity network fabric," in *Proceedings of ACM CoNEXT*, 2015.

[24] R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "Recursively Cautious Congestion Control," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, 2014.

[25] D. Katabi, M. Handkwy, and C. Rohrs, " Congestion control for high bandwidth-delay product networks," in *Proceedings of ACM SIGCOMM*, 2002.

[26] A. Munir, G. Baig, S. M. Irteza, I. A. Qazi, A. X. Liu, and F. a. Dogar, "Friends, Not Foes: Synthesizing Existing Transport Strategies for Data Center Networks," in *Proceedings of ACM SIGCOMM.*, 2014.

[27] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. Ismail, M. S. Iqbal, and B. Khan, "Minimizing Flow Completion Times in Data Centers," in *Proceedings of IEEE INFOCOM*, 2013.

[28] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. Haj, and Y. Ming, "Congestion Control for Large-Scale RDMA Deployments," in *Proceedings of ACM SIGCOMM*, 2015.

[29] IEEE, "DCB. 802.1Qbb - Priority-based Flow Control," 2011. [Online]. Available: http://www.ieee802.org/1/pages/802.1bb.html

[30] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "RDMA over Commodity Ethernet at Scale," in *Proceedings of ACM SIGCOMM*, 2016.

[31] R. Mittal, U. C. Berkeley, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. G. Microsoft, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "TIMELY : RTT-based Congestion Control for the Datacenter," in *Proceedings of ACM SIGCOMM*, 2015.

[32] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," in *Proceedings of ACM SIGCOMM*, 2012.

[33] N. Dukkipati and N. Mckeown, "Why Flow-Completion Time is the Right metric for Congestion Control and why this means we need new algorithms," *ACM SIGCOMM Computer Communication Review*, vol. 36, pp. 59–62, 2006.

[34] K. He, E. Rozner, K. Agarwal, W. Felter, J. Carter, and A. Akella, "Presto: Edge-based Load Balancing for Fast Datacenter Networks," in *Proceedings of ACM SIGCOMM*, 2015.

[35] J. Cao, R. Xia, P. Yang, C. Guo, G. Lu, L. Yuan, Y. Zheng, H. Wu, Y. Xiong, and D. Maltz, "Per-packet Load-balanced , Low-Latency Routing for Clos-based Data Center Networks Categories and Subject Descriptors," in *Proceedings of ACM CoNEXT*, 2013.

[36] A. Kabbani, B. Vamanan, J. Hasan, and F. Duchene, "FlowBender: Flow-level Adaptive Routing for Improved Latency and Throughput in Datacenter Networks," in *Proceedings of ACM CoNEXT*, 2014.

[37] H. Zhang, J. Zhang, W. Bai, K. Chen, and M. Chowdhury, "Resilient Datacenter Load Balancing in the Wild," in *Proceedings of ACM SIGCOMM*, 2017.

[38] N. Katta, A. Ghag, M. Hira, I. Keslassy, A. Bergman, C. Kim, and J. Rexford, "Clove: Congestion-Aware Load Balancing at the Virtual Edge," in *Proceedings of ACM CoNEXT*, 2017.

[39] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "HULA: Scalable Load Balancing Using Programmable Data Planes," in *Proceedings of ACM SOSR*, 2016.

[40] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the Impact of Packet Spraying in Data Center Networks," in *Proceedings of ACM INFOCOMM*, 2013.

[41] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS Adaptive Traffic Engineering," in *Proceedings of IEEE INFOCOM*, 2001.

[42] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the Tightrope : Responsive Yet Stable Traffic Engineering," in *Proceedings of ACM SIGCOMM*, 2005.

[43] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic load balancing without packet reordering," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 2, p. 51, 2007.

[44] E. Vanini, R. Pan, M. Alizadehand, P. Taheri, and T. Edsall, "Let It Flow : Resilient Asymmetric Load Balancing with Flowlet Switching," in *Proceedings of ACM NSDI*, 2017.

[45] P. Wang, H. Xu, Z. Niu, D. Han, and Y. Xiong, "Expeditus : Congestion-aware Load Balancing in Clos Data Center Networks," in *Proceedings of ACM SoCC*, 2016.

[46] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath tcp," in *Proceedings of NSDI*, vol. 11, 2011, pp. 8–8.

[47] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath tcp: analysis, design, and implementation," *IEEE/ACM Transactions on Networking (TON)*, vol. 24, no. 1, pp. 596–609, 2016.

[48] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "Mptcp is not pareto-optimal: Performance issues and a possible solution," *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1651–1665, 2013.

[49] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath congestion control for shared bottleneck," in *Proceedings of PFLDNeT workshop*, vol. 357. Citeseer, 2009, p. 378.

[50] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal of Selected Areas in Communications*, vol. 13, pp. 1176–1188, 1995.

[51] Z. Wang, A. Singhal, Y. Wu, C. Zhang, H. Che, H. Jiang, B. Liu, and C. Lagoa, "Holnet: A holistic traffic control framework for datacenter networks," in *2020 IEEE 28th International Conference on Network Protocols (ICNP)*. IEEE, 2020, pp. 1–12.

[52] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 1, pp. 237–252, 1998.

[53] S. H. Low and D. E. Lapsley, "Optimization Flow Control I: Basic Algorithm and Convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 861–874, 1999.

[54] S. H. Low, "A Duality Model of TCP and Queue Management Algorithms," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 525–536, 2003.

[55] S. Korovin and V. Utkin, "Using sliding modes in static optimization and nonlinear programming," *Automatica*, vol. 10, no. 5, pp. 5250–532, 1974.

[56] B. A. Movsichoff, C. Lagoa, and H. Che, "End-to-End Optimal Algorithm for Integrated QoS, Traffic Engineering, and Failure Recovery," *ACM/IEEE Transactions on Networking*, vol. 15, no. 4, pp. 813–823, 2007.

[57] W. Su, C. Liu, C. Lagoa, H. Che, K. Xu, and Y. Cui, "A Family of Optimal, Distributed Taffic Control Laws in a Multidomain Environment," *IEEE Transactions on Control System Technology*, vol. 23, no. 4, pp. 1373–1386, 2015.

[58] M. Ashour, J. Wang, C. M. Lagoa, N. Aybat, and H. Che, "Non-Concave network utility maximization: A distributed optimization approach," in *Proceedings of IEEE INFOCOM*, 2017.

[59] "Center of mass," https://en.wikipedia.orgwikiCenter_of_mass.

[60] "Lagrange multiplier," https://en.wikipedia.orgwikiLagrange_multiplier.

[61] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," pp. 556–567, 2000.

[62] W. Bai, L. Chen, K. Chen, and H. Wu, "Enabling ECN in Multi-Service Multi-Queue Data Centers," in *Proceedings of ACM NSDI*, 2016.

[63] M. Kheirkhah, I. Wakeman, and G. Parisis, "Multipath-TCP in ns3," in *Proceedings of ACM Workshop on ns-3*, 2014.

[64] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A Scalable and Flexible Data Center Network," in *Proceedings of ACM SIGCOMM*, 2009.

[65] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," in *RFC6356*, 2011.

[66] C. Lagoa and H. Che, "Decentralized Optimal Traffic Engineering in the Internet," *ACM SIGCOMM Communication Review*, vol. 3, pp. 1007–1018, 2000.

[67] L. Ye, Z. Wang, H. Che, and C. M. Lagoa, "TERSE:A Unified End-to-End Traffic Control Mechanism to Enable Elastic, Delay Adaptive, and Rate Adaptive

Services," *IEEE Journal on Selectied Areas in Communications*, vol. 29, no. 5, pp. 938–950, 2011.

[68] C. M. Lagoa, H. Che, and B. A. Movsichoff, "Adaptive control algorithms for decentralized optimal traffic engineering in the Internet," *ACM/IEEE Transactions on Networking*, vol. 12, no. 3, pp. 415–428, 2004.

[69] L. Ye, Z. Wang, H. Che, H. B. Chan, and C. M. Lagoa, "Utility function of tcp," *Computer communications*, vol. 32, no. 5, pp. 800–805, 2009.

[70] H. Han, S. Shakkottai, C. Hollot, R. Srikant, and D. Towsley, "Overlay tcp for multi-path routing and congestion control," in *IMA Workshop on Measurements and Modeling of the Internet*, 2004.

[71] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling tcp throughput: A simple model and its empirical validation," in *Proceedings of the ACM SIGCOMM'98 conference on Applications, technologies, architectures, and protocols for computer communication*, 1998, pp. 303–314.

[72] S. B. C. Paasch, "Multipath TCP in the Linux Kernel," http://www.multipath-tcp.org/, 2019, [Online].

[73] S. Hassayoun, J. Iyengar, and D. Ros, "Dynamic window coupling for multipath congestion control," in *2011 19th IEEE International Conference on Network Protocols*. IEEE, 2011, pp. 341–352.

[74] W. Wei, Y. Wang, K. Xue, D. S. Wei, J. Han, and P. Hong, "Shared bottleneck detection based on congestion interval variance measurement," *IEEE Communications Letters*, vol. 22, no. 12, pp. 2467–2470, 2018.

[75] F. H. Mirani, N. Boukhatem, and M. A. Tran, "A data-scheduling mechanism for multi-homed mobile terminals with disparate link latencies," in *2010 IEEE 72nd Vehicular Technology Conference-Fall*. IEEE, 2010, pp. 1–5.

[76] K. Xue, J. Han, D. Ni, W. Wei, Y. Cai, Q. Xu, and P. Hong, "Dpsaf: Forward prediction based dynamic packet scheduling and adjusting with feedback for multipath tcp in lossy heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 2, pp. 1521–1534, 2017.