Measurement, Analysis, and Enhancement of Multipath TCP Energy Efficiency for Datacenters

Jia Zhao, *Student Member, IEEE*, Jiangchuan Liu, *Fellow, IEEE*, Haiyang Wang, *Member, IEEE*, Chi Xu, *Student Member, IEEE*, Wei Gong, *Member, IEEE*, and Changqiao Xu, *Senior Member, IEEE*

***Abstract*— Multipath TCP (MPTCP) has recently been suggested as a promising transport protocol to boost the utilization of underlaying datacenter networks, yet it also increases the host CPU power consumption. It remains unclear whether datacenters can indeed benefit from using MPTCP from the perspective of energy efficiency. Through realworld measurement of MPTCP, we show that the energy efficiency of MPTCP is largely related to the flow completion time and the existence of link-sharing subflows. In particular, we find that the link-sharing subflows in MPTCP will significantly elevate the CPUs' power consumption on hosts. To make the matter worse, it will also reduce the transmission efficiency for both throughput-sensitive long flows and latency-sensitive short flows. To address such a problem, we present MPTCP-D, an energy-efficient enhancement of MPTCP in datacenter networks. MPTCP-D incorporates a novel congestion control algorithm that improves energy efficiency by minimizing the flow completion time. It also has a build-in subflow elimination mechanism that precludes link-sharing subflows from increasing the host CPU power consumption. We implement MPTCP-D in the Linux kernel, analyze the parameter selection in the algorithm and study its performance through packet-level simulation and on Amazon EC2. Our results show that, without degrading the performance of the long flow throughput and the short flow completion time, MPTCP-D reduces the long flow energy consumption by up to 72% compared to DCTCP for data transfers, and reduces the short flow power consumption by up to 46% compared to MPTCP with link-sharing subflows.**

***Index Terms*— Multipath TCP, energy efficiency, datacenters, link-sharing subflows, congestion control.**

Manuscript received May 29, 2017; revised January 19, 2018, September 30, 2018, January 26, 2019, and July 22, 2019; accepted October 8, 2019; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Bremler-Barr. Date of publication December 13, 2019; date of current version February 14, 2020. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant, in part by the Industrial Canada Technology Demonstration Program (TDP) Grant, and in part by the EVCAA R&S Grant from the University of Minnesota at Duluth, Duluth, MN, USA. The work of Chi Xu was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61602214. The work of Changqiao Xu was supported by the National Natural Science Foundation of China (NSFC) under Grant 61871048 and Grant 61872253. An earlier version of this work appeared in IEEE INFOCOM'17. *(Corresponding author: Jiangchuan Liu.)*

J. Zhao, J. Liu, and C. Xu are with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: zhaojiaz@sfu.ca; jcliu@cs.sfu.ca; xuchi.int@gmail.com).

H. Wang is with the Department of Computer Science, University of Minnesota at Duluth, Duluth, MN 55812 USA (e-mail: haiyang@d.umn.edu).

W. Gong is with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China (e-mail: weigong@ustc.edu.cn).

C. Xu is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: cqxu@bupt.edu.cn).

Digital Object Identifier 10.1109/TNET.2019.2950908

I. INTRODUCTION

NOWADAYS, the world's datacenters consume over 3 percent of the global electricity production, producing tremendous carbon footprint [31]. In particular, the datacenters of the United States consumed 91 billion kilowatt-hours of electricity in 2013, which is predicted to increase to 140 billion kilowatt-hours by 2020, equivalent to 51 power plants (at a scale of 500 megawatts) [1]. TABLE I shows that data transfers significantly increase the power consumption of end-hosts. There is a need to improve energy consumption at the end-hosts from the perspective of network resource utilization.

It is known that the energy consumption of data communication depends on both the host CPU power[1] and the flow completion time. A host's instantaneous CPU power during data transfer noticeably increases compared to that in the idle state, and it is affected by such factors as the sending rate, the number of network interfaces, and the number of simultaneously-used TCP sockets. The flow completion time is closely related to the traffic pattern. In today's datacenters, most (90%) of the data are delivered by long flows (of sizes from 1MB to about 1GB); yet a majority (90%) of the flows are indeed short flows (of size smaller than 1MB) [3], [4]. The completion time of the long flows decreases with throughput, and that of the short flows, however, largely depends on the instant path quality. These short flows often have completion deadlines, too, ranging from tens to hundreds of milliseconds. As such, energy optimization for data transfer has to jointly consider all these factors. Simply upgrading the inside network may not necessarily improve the energy efficiency. In fact, even the common 1Gbps/10Gbps networks are still not well utilized, and TCP remains the bottleneck there for throughput and for energy efficiency [8]. Although such customized protocols as DCTCP [5] and TIMELY [6] have attempted to better serve the latency-sensitive short flows, they are not optimized for the long flows and the overall energy.

Different from the traditional TCP, multipath transport protocols (e.g., Multipath TCP (MPTCP) [7]) explore multiple routes between each pair of hosts to increase throughput, thereby reducing the flow completion time, particularly for long flows [8]. There has been significant research on improving datacenter's performance with MPTCP [8]–[14], mostly from throughput and latency perspectives. Throughput improvement brought by using multiple interfaces however would also increase the instantaneous host CPU power.

[1]In this paper, the term *power* (in Watt) denotes the electrical energy consumed per second, and the term *energy* (in Joule) denotes the integral of power over time. Energy depends on both power and time.

TABLE I
CPU POWER CONSUMPTION: IDLE STATE V.S. MPTCP DATA TRANSFER

	Idle	Data transfer
Intel Core i7	1.375 Watts	5.437 Watts
Intel Core Haswell EX processor	26.678 Watts	68.108 Watts

It remains unclear whether datacenter can indeed benefit from using MPTCP from the energy efficiency perspective. As a matter of fact, it has been shown that short MPTCP flows' completion time may increase when a large number of subflows are used [13]. This would severely delay latency-sensitive short flows, and the consequent expirations and retransmissions would increase power consumption.

In this paper, we aim to take an initial step toward understanding MPTCP's energy efficiency in datacenter networks. By analyzing the performance of the standard MPTCP on a realworld testbed, we show that, compared to the single path TCP, the reduced completion time (and hence energy) of long flows by using MPTCP is enough to overshadow the increased host CPU power. When multiple subflows are generated on overlapped paths,[2] MPTCP however can hurt both the long flows' throughput and the short flows' completion time, and consequently consumes significantly more energy.

Based on these findings, we design MPTCP-D, an energy-efficient Multipath TCP for Datacenters. MPTCP-D can save energy for both long and short flows without sacrificing the long flows' throughput and the short flows' completion time. A fluid model is developed for multipath congestion control in MPTCP-D. The model ensures energy efficiency by minimizing the flow completion time for long flows and by Round Trip Time (RTT) based traffic shifting for short flows. To reduce the power consumption of link-sharing sublfows, an Extra Subflow Elimination (ESE) mechanism is developed to close the congestion windows (`cwnds`) of extra subflows, ensuring that only one subflow exists on overlapped paths.

To evaluate the performance of our enhancement, we have implemented MPTCP-D on Linux and evaluated it on Amazon's Elastic Compute Cloud (EC2). We show that MPTCP-D reduces the long flow energy consumption by up to 72% compared to DCTCP, and achieves better latency for short flows. It is able to maintain as good throughput as the basic MPTCP for long flows; yet for multiple subflows on overlapped paths, the ESE mechanism in MPTCP-D is highly effective, reducing up to 20% and 46% of power consumption for long and short flows, respectively. To evaluate the performance of MPTCP-D over larger-scale datacenter topologies with realistic traffic, we also perform packet-level simulations with the discrete-event network simulator ns-3. In the simulations, we study the key parameter configurations for the ESE mechanism and demonstrate that the MPTCP-D's congestion control algorithm achieves similar throughput and RTT as compared with MPTCP in a large topology under different traffic patterns.

The remainder of this paper is organized as follows. Section II gives an overview of MPTCP and related work. Section III analyzes the energy efficiency of long and short MPTCP flows. We present the design of MPTCP-D in section IV, and evaluate the performance of MPTCP-D in section V. Section VI concludes the paper.

[2]In this paper, *overlapped paths* denote the routes that have common links for a host-to-host MPTCP connection. We do not use the term *bottleneck*, because conventionally a bottleneck in [7], [15], [16] is defined to study the fairness between competing flows, e.g., an MPTCP flow and a TCP flow coexisting on a shared link. Instead, we focus on overlapped paths to study the mutual interference between link-sharing subflows of MPTCP and how it impacts MPTCP's energy overhead, rather than repeat the fairness issues.

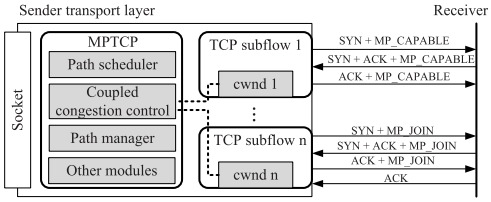


Fig. 1. The MPTCP Linux kernel and end-to-end connection set up.

II. BACKGROUND AND RELATED WORK

A. Overview of MPTCP

MPTCP is a multipath extension of TCP that can simultaneously use multiple paths and support the setup of backup paths. To this end, it adds an extra layer which provides scheduling and congestion control on the top of multiple TCP subflows and has good compatibility to the current TCP. As shown in Fig. 1, an MPTCP connection is initialized with a MP-CAPABLE handshake, in which two ends will exchange a key and establish the first subflow. Then other subflows can be associated with this MPTCP connection by using the MP-JOIN option after verifying the key.

The MPTCP architecture was introduced in RFC 6182 [34], MPTCP congestion control was described in RFC 6356 [35], and MPTCP was published as an experimental standard in RFC 6824 [36] in 2013. Apple has supported MPTCP for its application Siri since iOS 7, and iOS 11 provides a new API for MPTCP [37], [38]. Korean Telecom has provided service for MPTCP applications of Samsung and LG mobile devices [41]. Citrix's Netscaler and F5 BIG-IP also support MPTCP [39], [40]. Fig. 1 shows the transport layer architecture of the MPTCP Linux kernel and how the connection is set up. The scheduler decides which path a packet will be sent over. The path manager decides how to create subflows among pairs of IP-addresses. Since the MPTCP Linux kernel version 0.90, multiple subflows can be created on the path between two IP-addresses. The version 0.93 supports the path manager to re-create subflows after a timeout. The multipath congestion control couples the subflows window adaptation. To serve different design goals, various multipath congestion control algorithms have been proposed [7], [9], [23]–[26], [32].

B. Related Work

There have been significant studies on investigating the transport layer design in datacenter networks. TCP shows inefficiency for both long and short flows in datacenters [8], [13]. Advanced single-path transport protocols, such as DCTCP [5] and TIMELY [6], have been developed to improve the performance of short flows in datacenter networks. Multipath

transmission has also been suggested, and Raiciu *et al*. [8] validated that MPTCP can improve datacenter network utilization on different network scales with different topologies. Khalili *et al*. [9] further investigated the optimized resource allocation among a large number of simultaneous MPTCP flows. Following these pioneer studies, various traffic scheduling methods are proposed to explore disjoint multiple paths in datacenter networks [11], [12], aiming to make full utilization of their high aggregation bandwidth.

Besides bandwidth efficiency, latency has long been a critical issue for the datacenter networks [5], [6], [13]. Optimization of latency-sensitive short flows is attracting increased attention in recent years. Cao *et al*. [10] proposed a multipath congestion control scheme for datacenters to explore path diversity for better link utilization. It uses link queue buffer control and traffic shifting to balance throughput and latency. Kheirkhah *et al*. [13] proposed a random packet scheduler to exploit good-quality paths for short multipath flows, so as to reduce their completion time. Chen *et al*. [14] further designed a fast loss recovery approach for multipath transmission. This design uses good-quality paths to retransmit the loss packets rather than wait for timeout on a lossy path.

The energy efficiency of MPTCP for datacenters, however, remains largely unclear to the research community. In this paper we show that, when subflows are sharing one/many common links, MPTCP has poor energy efficiency, especially for short flows. There have been methods for bottleneck link detection with multipath congestion control [15], [16]. They however are mostly designed to provide fairness among competing flows (e.g., MPTCP and TCP flows) on their coexisting link, while we focus on the mutual interference of MPTCP subflows on their shared link. Such a mutual interference increases the overhead of flow management, elevating the energy consumption. These methods also need to wait for packet losses [15] or take 10-20 seconds to make a decision [16], which is inefficient for the latency-sensitive short flows. To deal with these issues, we introduce an Extra Subflow Elimination module in MPTCP-D that first detects whether multiple subflows are sharing a common link and then closes the `cwnds` of the extra subflows. Some preliminary results have been presented in [43] with the evaluation based mainly on a multipath transport testbed. In this paper, we perform comprehensive measurement and evaluation across not only realworld datacenters but also realistic traffic workloads, as well as deep analysis on different components and key parameters of the proposed algorithm.

Datacenter energy proportional management has been studied in the literature [2], [17] to match the energy consumption with server workload. The targets of these works are: (i) datacenters can ideally consume little power in idle state; and (ii) datacenters can consume the amount of power proportional to the level of workload in active state. Differing from them, our work focuses on the transport layer energy efficiency. Our measurement results in TABLE I also show that a server's CPU power consumption increases significantly from idle state to data transfer state. Our design not only reduces the energy consumption in active states for data transfers but also potentially increases the duration of idle states, serving as a complement of the energy proportional management.

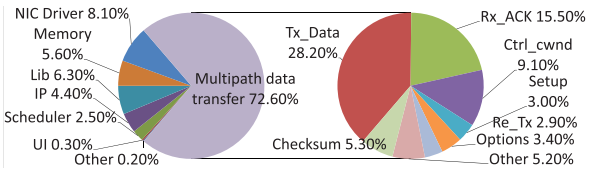


Fig. 2. Energy consumption of CPU during a multipath data transfer, and the refined results of softwares including tools, Linux kernel modules and MPTCP-related operations according to their share of CPU usage (Tx_Data: transmitting data segments; Rx_ACK: receiving ACK; Ctl_cwnd: controlling congestion window; etc).

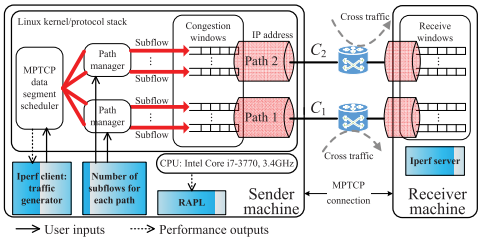


Fig. 3. Testbed I setups for the experiments that emulate the path overlap scenarios: MPTCP connection is set up between sender and receiver machines, each one is configured with two interfaces (path 1 has capacity C_1 and path 2 has capacity C_2), Intel Core i7-3770 CPU, and the MPTCP Linux kernel v0.90. `iperf` is used to generate flows. CPU power is read from RAPL. The 'num_subflows' parameter in MPTCP path manager module is used to control the number of subflows for each path. The setups can have disjoint paths ('num_subflows' $= 1$) or n subflows for each path ('num_subflows' $= n$).

III. Measurement of MPTCP Energy Consumption

In this section, we will examine the energy consumption of MPTCP. We first investigate the long flow energy consumption in our local cluster and EC2, and then dive into a more realistic case in datacenter networks with shared links.

We record instant host CPU power from the Intel's Running Average Power Limit (RAPL) driver [18]–[20]. RAPL works as a power management software platform for x86 architecture CPUs. It uses a software-model to estimate, monitor, control and obtain information about the system-on-chip power consumptions. Accordingly, it is also a built-in power meter that can record the instantaneous power consumption of the supported CPUs. We use the `rapl-read` tools [21] and access the power information from Model-Specific Registers (MSRs). TABLE I shows that multipath data transfers largely increase the CPU power compared to the idle state. Fig. 2 shows a record of the energy consumed by an MPTCP sender according to its share of CPU usage. We also include the refined results of the multipath data transfer operations, including functions related to data transmission and reception, congestion window control, and other operations.

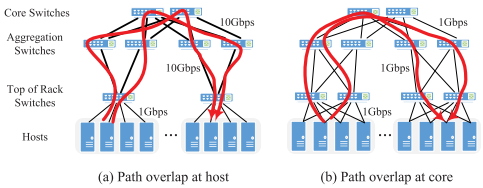
A. Testbed Overview

We measure the host CPU power during host-to-host data transfers in both our server cluster and Amazon EC2. Our cluster has 10 machines, each with double NICs and a Quad-core Intel Core i7-3770 CPU. We rent server instances on EC2 and configure them with different types of CPUs. These include Quad-core Intel Xeon E5-2680 v2, Octa-core cores Intel Xeon E5-2670 v2, and Octa-core Intel Xeon E5-2680 v2. We install the MPTCP Linux kernel of version 0.90 [22] on our local servers and the EC2 instances. We use `iperf` to generate flows. The setup of testbed I is shown in Fig. 3. In this

TABLE II

MPTCP VS. TCP: THROUGHPUT'S IMPACT ON BOTH HOST CPU POWER AND LONG FLOW COMPLETION TIME

CPU type	Config.	Mean rate: Mbps	CPU power increase	Transfer time decrease	Overall energy saving
Core i7-3770, 3.4GHz, 4 cores	TCP	19	baseline	baseline	baseline
	MPTCP	37	6.4%	50.1%	46.9%
		58	14.7%	68.3%	63.6%
		77	26.2%	75.8%	69.5%
		96	33.5%	80.6%	74.1%
XeonE5-2680 v2, 2.8GHz, 4 cores	TCP	205	baseline	baseline	baseline
	MPTCP	408	6.8%	49.8%	46.4%
		610	8.4%	66.4%	63.6%
Xeon E5-2670 v2, 2.5GHz, 8 cores	TCP	205	baseline	baseline	baseline
	MPTCP	408	5.8%	49.8%	46.9%
		611	8.1%	66.4%	63.7%
		819	10.9%	75.0%	72.3%
		1020	12.6%	79.9%	77.4%
Xeon E5-2680 v2, 2.8GHz, 8 cores	TCP	205	baseline	baseline	baseline
	MPTCP	408	8.3%	49.8%	45.6%
		611	11.3%	66.4%	62.6%
		819	14.1%	75.0%	71.5%
		1016	14.2%	79.8%	76.9%



(a) Path overlap at host (b) Path overlap at core

Fig. 4. MPTCP with multiple subflows on overlapped paths: (a) VL2 topology, path overlap at host; (b) multihomed topology, path overlap at core.

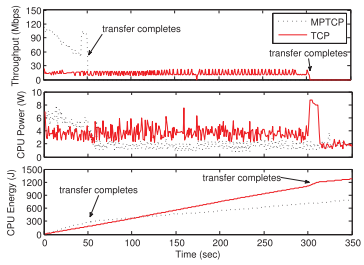


Fig. 5. Performance of long flows (500 MB data transfer). Compared with TCP, MPTCP largely increases the aggregation throughput and decreases the transfer completion time, thus reducing the overall energy consumption.

section, we measure the energy efficiency of MPTCP in two scenarios: (i) long flow energy consumption of host-to-host connection with MPTCP and TCP; (ii) mutual interference between subflows on shared links. For long flow energy consumption in subsection III.C, we use the testbed I's setup and set Path 1 and Path 2's bandwidth using traffic shaping tools. For the CPU types other than Intel Core i7-3770, the measurements are conducted on EC2. For the energy consumption of link-sharing subflows in subsection III.D, we use the testbed I's setup and the same path configuration in subsection III.C.

B. Long Flow Energy Consumption

We first study the long flow energy consumption of host-to-host connection with MPTCP and TCP, respectively. The long flow energy consumption depends on both the CPU power and the transfer completion time.

We use the same testbed setup as in Fig. 3, in which we configure the CPU type and path capacity in different measurements and obtain the results in Fig. 5 and TABLE II. For the Quad-core Intel Core i7-3770 CPU, we use the testbed I (Fig. 3), where TCP uses Path 1 and MPTCP uses both paths. The capacity of the NIC in a host is 100Mbps, which is the maximum througput that can be reached over a single path, so can the aggregated throughput of mulitple paths. We use traffic shaping tools to limit Path 1's capacity to 20Mbps and vary Path 2's from 0 Mbps to 80Mbps (progressively increasing by a step of 20Mbps), which offers a wide range (5x) to examine the impact of throughput on energy consumption. For Quad-core Intel Xeon E5-2680 v2, Octa-core cores Intel Xeon E5-2670 v2, and Octa-core Intel Xeon E5-2680 v2 CPUs, the data transfer is between a pair of virtual machines in EC2. Each virtual machine has two NICs with an aggregated capacity limit of 1024Mbps. We set the minimum NIC capacity to be 210Mbps, and the maximum NIC capacity to be 512Mbps. TCP uses a single path with capacity 210Mbps, and MPTCP uses two paths with aggregated capacity progressively increased to 1024Mbps (corresponding to different tested mean rates in TABLE II).

Fig. 5 shows the results from a server with Quad-core CPU Intel Core i7-3770. It is easy to see that MPTCP is more energy efficient because it can significantly reduce the flow completion time (FCT). The FCT of classic TCP flow is around 200s. MPTCP, on the other hand, only uses 50s to complete the downloading. Although MPTCP has higher instantaneous power, its short FCT increases the CPU's idle duration and hence successfully reduces the total energy consumption. As can be seen in the subfigure for CPU energy consumption, such a gap will also increase when we have longer flows or larger files. To validate that this result is not valid for the Intel Core i7 CPU only, we also conduct measurements under different types of CPUs. TABLE II validates that MPTCP consumes less energy than TCP.

C. Mutual Interference Between Subflows on Shared Links

As shown in Fig. 4, typical datacenter networks can hardly avoid path overlap for MPTCP's subflows. Such path overlap can be either the links between hosts and Top of Rack (ToR) switches, or the core switches with limited capacity [8]. To understand the mutual interference between subflows of MPTCP on a shared link, we deploy MPTCP on our local testbed (Fig. 3). We use the MPTCP Linux kernel of version 0.90, in which an MPTCP connection consists of multiple paths. Each path can have one or more subflows with independent congestion windows.

Fig. 6 compares the results of two cases: (i) two subflows over two disjoint paths; and (ii) four subflows with two of them sharing each link. We measure the path conditions (throughput and RTTs) and CPU energy consumption for both heterogeneous and homogeneous path configurations. First, we use a stationary path configuration $C_1 = 100$Mbps and $C_2 = 20$Mbps for the testbed in Fig. 3. This setup corresponds to two heterogeneous paths. In subsection V.C, We also show the experimental results in the scenario of homogeneous paths. For both heterogeneous and homogeneous path configurations,

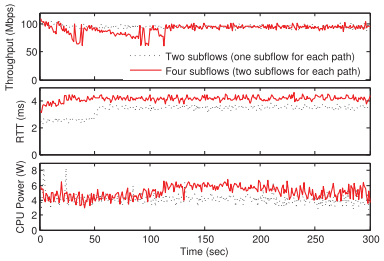


Fig. 6. Performance comparison of the two cases: the MPTCP connection with four subflows sharing two paths consumes more CPU power than the connection using two disjoint paths, and it also degrades throughput and has higher latency.

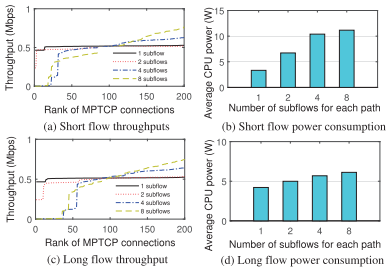


Fig. 7. Performance of 200 MPTCP flows: each flow uses different number of subflows sharing each path. Each long MPTCP flow transmits 100MB data, and each short MPTCP flow transmits 0.5MB data.

the measurement results demonstrate the same phenomenon of mutual interference between subflows on the shared link. Fig. 6 shows that link-sharing subflows not only degrades RTT and throughput but also increases the energy consumption.

To pinpoint the root cause, we take a closer look at these link-sharing subflows. We use `iperf` to generate 200 parallel MPTCP connections between the two servers (still using the path configuration $C_1 = 100$Mbps and $C_2 = 20$Mbps). We use multiple connections because it is necessary to consider many coexisting flows in datacenter networks, e.g., a large number (from hundreds to thousands) of flows may arrive at a switch simultaneously [4]. We measure the power consumption of both long flows (100 MB data transfer for each connection) and short flows (0.5 MB data transfer for each connection). These long flows and short flows are isolated to separate tests, because: (i) we want to examine the mutual interference between subflows for long flows against that for short flows; (2) we need to calculate the accumulated energy consumption over long flow duration. The short flow power consumption is the instantaneous power record (Joules per second) from the CPU's RAPL driver, and the long flow power consumption is calculated by using the flow completion time to divide the integral of power over time. Each connection can use the two paths simultaneously to transmit data. On each path, we change the number of subflows from 1 to 8 (increased by powers of 2). This is done by modify Linux kernel's MPTCP path-manager module in '/sys/module/mptcp_fullmesh/parameters/num_subflows' (note that this option is not included in the MPTCP Linux kernel of version lower than 0.90).

Fig. 7(a) shows the short flow throughput of individual MPTCP connections, ranked in ascending order of their throughputs, with different number of subflows for each path of MPTCP. Clearly, using multiple subflows for each path fails to achieve higher average throughput. Although approximately half of the connections with 4 or 8 subflows for each path get higher throughputs, such a link sharing affects the throughputs of the other half of connections, and thus nearly 50% of the short flows are severely delayed. Fig. 7(b) shows how the sender's average CPU power consumption from the short flows changes with different number of subflows for each path. We see that using multiple subflows consumes far more power than using only one subflow. For the long flow throughput of individual MPTCP connections, as shown in Fig. 7(c), using multiple subflows for each path cannot improve throughput, either, and some connections with 4 and 8 subflows experience severe throughput degradation. Fig. 7(d) shows that the sender's average CPU power consumption from the long flows also increases with the number of subflows. The power consumption of using 8 subflows is nearly 50% more than that of using one subflow. Fig. 6 and Fig. 7 indicate that the existence of link-sharing subflows in MPTCP hurts the throughput and hence the energy consumption for both long and short flows. This can be ascribed to the sharply increased operations to control too many subflow congestion windows (each corresponds to a TCP socket) simultaneously, the increase of loss and retransmission on each path, and the highly fluctuating windows causing slow convergence to a stable equilibrium.

IV. The Design of MPTCP-D

A. Overview of MPTCP-D Design

In this section, we present the design of MPTCP-D for energy-efficient data transfer in datacenter networks. It seeks to minimize the flow completion time and to preclude link-sharing subflows through detecting overlapped paths, thereby reducing the power consumption for both long and short flows. Fig. 8 shows an overview of our design. For the congestion control algorithm, we set the design goals including full utilization of the low-delay path, TCP-friendliness and Pareto-optimality. We design the key parameters in extra subflow elimination and a basic fluid model for congestion control to meet the three design goals.

B. Congestion Control

MPTCP-D does not directly use the existing multipath congestion control algorithms, e.g., LIA [7] and OLIA [9]. It has been shown in [9] that a large number of coexisting MPTCP flows using LIA may downgrade the throughput of each other. MPTCP-D however needs as high throughput as possible to achieve good energy efficiency for long flows. While OLIA can avoid the mentioned problem of LIA, it does not have any delay-based traffic shifting strategy and hence can be inefficient for latency-sensitive short flows.

As in [9], we consider a network that consists of a link set L. A link $l \in L$ has finite capacity c_l. There is a set S of MPTCP connections in the network. Each connection $s \in S$ has multiple subflows. Each subflow $r \in s$ traverses multiple links. If subflow r traverses a link l, then we denote $l \in r$. For each subflow r, let $RTT_r(t)$ and $w_r(t)$ denote the round trip time and congestion window, respectively, and let $x_r(t) = w_r(t)/RTT_r(t)$ represent the send rate at time t. For each connection $s \in S$, let $\mathbf{x}_s(t) = (x_r(t), r \in s)$.

Let $[R^l_{sr}]$ be the routing matrix where $R^l_{sr} = 1$ if subflow r of connection s traverses link l, and $R^l_{sr} = 0$ otherwise.

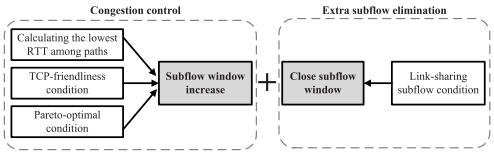


Fig. 8. Overview of MPTCP-D design.

Let $y_l = \sum_{s\in S}\sum_{r\in s} R^l_{sr}x_r$ be the aggregate traffic on link l. Let $p_l(y_l)$ be the packet loss probability at link l, which is an increasing function of y_l with constraint $y_l \le c_l$. The packet loss probability of subflow r can be expressed as $\lambda_r = 1-\prod_{l\in r}(1-p_l) \approx \sum_{l\in r} p_l$. For simplicity, we omit the time t in the functions w_r, RTT_r, x_r and $\mathbf{x}_s$.

Our design is based on a general fluid model for multipath congestion control as follows.

$$\frac{dx_r}{dt} = \frac{\psi_r x_r^2}{RTT_r^2(\sum_{k\in s} x_k)^2} - \frac{1}{2}\lambda_r x_r^2 \tag{1}$$

Similar models have been used in previous work [9], [23], [24], [32] to design multipath congestion control algorithms.

To provide high throughput for long flows, MPTCP-D should have aggressive but fair enough send rate increase. To fit latency-sensitive short flows, MPTCP-D should be able to use low-delay good-quality paths for transmission.

1) Full Utilization of Low-Delay Paths: For mutlipath TCP flows in datacenters, we should choose proper variables to estimate the path quality and shift the traffic to good-quality paths. This avoids frequent window decrease due to severe packet loss, thereby reducing energy consumed by retransmission or recovery operations.

It has been studied in [9], [23], [24], [32] that parameter ψ_r decides the aggressiveness of congestion window (or send rate) increase. The aggressiveness decides how much traffic to be shifted to a subflow. To fully utilize the low-delay paths, we design the traffic-shifting parameter ψ_r as follows

$$\psi_r = \frac{\min_{k\in s} RTT_k}{RTT_r}. \tag{2}$$

Among all the available paths, only the path with the best RTT maintains high aggressiveness of send rate increase. The send rate increase of other paths decreases with their RTT values. This will shift traffic to the path with the best RTT. Short flows are usually latency-sensitive, and thus it is very intuitive to use RTT to estimate the path delay. As shown in [6], in datacenter networks, RTT is an effective congestion signal without the need for switch feedback, and it can accurately reflect the host-to-host path delay. Moreover, a datacenter network usually consists of highly homogeneous hosts, links and intermediate devices. Hence, for intra-datacenter traffic, path delays depend mainly on two factors: the link congestion level and the number of hops between source and destination hosts. Large datacenters (scale of tens of thousands of servers) typically have round trip delays around 200-500μs, where congestion can cause delay spikes up to tens of milliseconds, and switches can increase delay by tens of microseconds for each hop [27]. This means that the path delays can sensitively reflect the quality difference among multiple available paths.

2) TCP-Friendliness: Given the amount of data in a transfer, high throughput brings short transfer time and we have shown in section III that high throughput can save energy for long flows. Yet the send rate increase of MPTCP-D cannot be too aggressive. According to the design goal of TCP-friendliness in [7] and RFC 6356,[3] multipath TCP should not take up more capacity than if it was regular TCP using the best path. The equilibrium of the fluid model as Equation (1) is a vector of send rates $\mathbf{x}^*_s(t) = (x^*_r(t), r \in s)$ that makes the right side of Equation (1) equal to zero for any route r. Our analysis, with the theoretical methodology similar to [9], also focuses only on the properties of the equilibrium.

Theorem 1: The congestion control algorithm of MPTCP-D, derived from Equations (1) and (2), satisfies the fairness design goal suggested by the RFC 6356, and the aggregate send rate of an MPTCP-D connection s at the equilibrium of Equation (1) is no more than the send rate that a regular TCP connection would achieve on the best path of s:

$$\sum_{k\in s} x^*_k \le \sqrt{2/\lambda_h}/RTT_h \tag{3}$$

where $h = \operatorname{argmax}_{k\in s} x^*_k$.

Proof: The traffic parameter ψ_r in Equation (2) satisfies $\psi_r \le 1$ for any path $r \in s$. Hence we have $\psi_h \le 1$. At the equilibrium of Equation (1), we have $\frac{dx_r}{dt} = 0$ and on the best path h we have $\frac{\psi_h x_h^2}{RTT_h^2(\sum_{k\in s} x^*_k)^2} = \frac{1}{2}(\mathbf{x}^*_s)\lambda_h x_h^2$. MPTCP-D has an aggregate throughput of $\sqrt{\frac{2\psi_h}{\lambda_h}}/RTT_h$, which is no more than the throughput $\sqrt{\frac{2}{\lambda_h}}/RTT_h$ achieved on the best path if it is a regular TCP. □

3) Pareto-Optimality: Besides TCP-friendliness, MPTCP-D also has to accommodate the many coexisting flows in datacenter networks, where the number of active flows at a switch in any given second could be as high as 10,000 [3]. That said, Pareto-optimality should be maintained here. In resource allocation problems, Pareto optimality is the equilibrium state at which one individual cannot gain more profit without damaging the profits of other individuals. For resource pooling of transport protocols in a network, Pareto optimality is the state at which it is impossible for an end-to-end connection to increase its throughput without decreasing the throughput of other coexisting end-to-end connections or increasing congestion.

Suppose there are $|S|$ connections using the same multipath congestion control algorithm in the network. Each connection $s \in S$ has a utility $U_s(\mathbf{x}_s)$, which is an increasing function of x_k for all $k \in s$. The aggregate utility of all the connections is as follows

$$\sum_{s\in S} U_s(\mathbf{x}_s) - \frac{1}{2}\sum_{l\in L}\int_0^{\sum_{k\in l} x_k} p_l(y)dy \tag{4}$$

where $p_l(y)$ is the link price or congestion cost of link l, and it is an increasing function with $p_l(0) = 0$ for all links $l \in L$. We have the following theorem.

Theorem 2: A multipath congestion control algorithm is Pareto-optimal if for all routes $r \in s$ it has a concave utility function $U_s(\mathbf{x}_s)$ that satisfies $\theta_r(\mathbf{x'}_s)\left.\frac{\partial U_s(\mathbf{x}_s)}{\partial x_r}\right|_{\mathbf{x}_s=\mathbf{x'}_s} =$

[3]C. Raiciu, M. Handly, and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols," IETF RFC 6356, 2011. https://tools.ietf.org/html/rfc6356

$I_r(\mathbf{x'}_s)$, where $\theta_r(\mathbf{x}_s)$ is a positive function related to step size of differential equation for the algorithm, $\mathbf{x'}_s = (x'_r(t), r \in s)$ is the maximizer of Equation (4) and $I_r(\mathbf{x}_s)$ is the function that decides the send rate increase of the algorithm.

Proof: In Equation (4), the utility $U_s(\mathbf{x}_s)$ is an increasing function of x_k for all $k \in s$, and the path price (congestion) term $\frac{1}{2}\sum_{l\in L}\int_0^{\sum_{k\in l} x_k} p_l(y)dy$ also increases with $\mathbf{x}_s$. At the maximizer $\mathbf{x'}_s$ of Equation (4), it is impossible to increase $\mathbf{x}_s$ for connection s without decreasing the throughput of other connections or increasing congestion. Therefore, the maximizer $\mathbf{x'}_s$ is at Pareto-optimality, and if its utility increasing rate $\theta_r(\mathbf{x'}_s)\left.\frac{\partial U_s(\mathbf{x}_s)}{\partial x_r}\right|_{\mathbf{x}_s=\mathbf{x'}_s}$ is equal to the the send rate increase $I(\mathbf{x}_s)$ of the multipath congestion control algorithm, then the algorithm has the best aggressiveness of throughput increase. It is known that Coupled [23], [24], EWTCP [25], DWC [15] and OLIA [9] all have $\theta_r(\mathbf{x}_s) = x_r^2$, and wVegas [26] has $\theta_r(\mathbf{x}_s) = x_r/\lambda_r$. Therefore, they all satisfy Pareto-optimality in Theorem 2. □

Theorem 3: The congestion control algorithm of MPTCP-D satisfies Pareto-optimality.

Proof: According to Equations (1) and (2), $\theta_r(\mathbf{x}_s) = x_r^2$, the send rate increase of MPTCP-D is $I_r(\mathbf{x}_s) = \frac{\psi_r x_r^2}{RTT_r^2(\sum_{k\in s} x_k)^2}$, and $\frac{d\psi_r}{dx_r} = 0$ for any $r \in s$. Let the utility function be $U_s(\mathbf{x}_s) = -\frac{(\sum_{r\in s}\psi_r x'_r/RTT_r^2)^2}{(\sum_{r\in s} x'_r)^2(\sum_{r\in s}\psi_r x_r/RTT_r^2)}$, where $\mathbf{x'}_s = (x'_r(t), r \in s)$ is the maximizer of Equation (4). We have $\theta_r(\mathbf{x'}_s)\left.\frac{\partial U_s(\mathbf{x}_s)}{\partial x_r}\right|_{\mathbf{x}_s=\mathbf{x'}_s} = I_r(\mathbf{x'}_s)$. According to Theorem 2, $I_r(\mathbf{x}_s)$ is Pareto-optimal based on $U_s(\mathbf{x}_s)$. □

Theorems 1 and 3 guarantee that MPTCP-D uses a fair and Pareto-optimal window evolution to achieve good energy efficiency for long flows.

C. Extra Subflow Elimination

As shown in Fig. 7, the link-sharing subflows consume more CPU power and cause increased latency. Link-sharing subflows degrade the performance of MPTCP for both long and short flows, especially for short flows which usually have strict completion deadlines ranging from tens of milliseconds to hundreds of milliseconds. We design an Extra Subflow Elimination (ESE), which can identify and eliminate the extra link-sharing subflows, so that only one subflow stays on a shared link and the other subflows on the link set their congestion windows to be zero.

Datacenters have many latency-sensitive short flows. Hence ESE must detect overlapped paths and close extra `cwnds` very quickly. As mentioned, RTT is an effective signal and we use it for ESE to detect link-sharing subflows. To make full utilization of the first several RTTs within the flow completion time, ESE uses the first significantly changed RTT to trigger the detection, and then uses both the RTT change and the RTT difference between subflows to decide whether a subflow shares a link with the subflow that triggers the detection. In our implementation, the number of candidate RTTs for the detection can be set to adapt to the variation of flow durations in different application scenarios. We set it to 80 in our current experiments. In realworld datacenters, it is very possible that there are a large number of very short flows with the number of packets in each subflow being less than the threshold (i.e., the number of candidate RTTs). The detection algorithm can hardly identify such flow patterns. To deal with the issue, we use an adaptive update mechanism for the number of candidate RTTs. This mechanism can be described as a conditional statement: If $NUM_{pkt} < NUM_{candidate}$, then $NUM_{candidate} = NUM_{pkt}$, where NUM_{pkt} is the number of packets in a subflow, and $NUM_{candidate}$ is the current number of candidate RTTs. As we have discussed in Section III, the link-sharing subflows degrade the RTT performance for both long flows and short flows. The design of ESE is based on the observations in section III. When MPTCP has multiple subflows sharing a common link, its instant RTTs always start from the first several low values and then gradually increase to the high values. After that, the following RTTs are maintained at the high values, without large difference between each other. Yet there would be large difference between a high value and the lowest value. For example, in the experiments of link-sharing subflows over our testbed, the RTTs of two subflows on a link at the stable state can be 30% higher than that at the starting state, due to the interference between link-sharing subflows. We use Fig. 19 to illustrate how the interference increases the congestion of paths. Our experiments are conducted over a cloud testbed, whose setup details will be introduced in subsection V.C. We test the results in both Linux bridge and Open vSwitch configurations. The data transfer lasts for 300 seconds and uses two paths with 300Mbps path capacity. Fig. 19 shows that MPTCP retransmissions increases with the number of subflows per path. Since retransmissions can be regarded as the level of congestion, the link-sharing subflows exaggerates the congestion on the paths. The path congestion also increases the variation of RTTs. We can explain this as follows: one subflow starts transmission over a less congested path and has a relatively stable RTT; another subflow on the same path starts to compete the bandwidth with the first subflow. The path becomes congested and the average RTT increases significantly (average RTT of two subflows per path is about 30% more than that of one subflow per path in our measurement). Such a difference can be used to trigger the detection of overlapped paths. In addition, the link-sharing subflows experience the same link condition, and hence they have the similar RTT values. This can be used to exclude the interference from cross traffic. Suppose that, a link l is used by a subflow of MPTCP without link-sharing subflows, and there is cross traffic on l. Although the cross traffic can increase the RTT on l, it does not affect the other subflows, which do not use the link l. That said, the other subflows do not have the RTTs similar to the RTT on l, and their `cwnds` should not be closed in this case.

We use flag $TRIG^{(s)}$ to indicate whether a subflow of connection s with significant RTT change has triggered the detection. If $TRIG^{(s)}$ is off, the subflow whose instant RTT is larger than a threshold ($\alpha \cdot baseRTT$, where $baseRTT$ is the minimum RTT experienced by the subflow) will turn on $TRIG^{(s)}$ and trigger the detection, and its RTT value will be used as a reference $rtt^{(s)}$ to be compared with the changed RTTs of other subflows. If $TRIG^{(s)}$ is on, the subflow, who has a significantly changed RTT and the current RTT similar to the reference $rtt^{(s)}$, will be identified as the extra subflows on the shared link. The identified subflows

Algorithm 1: Extra Subflow Elimination

Input: RTT_r, r corresponds to a subflow, and $r \in s$
Output: w_r
if $RTT_r > \alpha \cdot baseRTT_r$ **then**
  if $TRIG^{(s)} = 0$ **then**
    set $TRIG^{(s)} = 1$;
    set $rtt^{(s)} = RTT_r$;
  if $(TRIG^{(s)} = 1) \wedge (|RTT_r - rtt^{(s)}| < \beta \cdot baseRTT_r)$ **then**
    set $w_r = 0$;
for *each ACK on path* r **do**
  $w_r \leftarrow w_r + \frac{\psi_r w_r / RTT_r^2}{(\sum_{k \in s} w_k / RTT_k)^2}$; //derived from Eq. (1)
for *each loss on path* r **do**
  $w_r \leftarrow \frac{1}{2} w_r$;
return w_r

will close their `cwnds` immediately. Otherwise, the subflow's `cwnds` will adapt following Equations (1) and (2). The link-sharing subflow detection and window evolution algorithm are summarized in Algorithm 1. The selection of parameters α and β will be discussed in Section V. The performance of ESE is not impacted by heterogeneous paths with different path delays, because it uses both large variation of intra-subflow RTTs and small difference of inter-subflow RTTs to confirm that subflows go over the same path. If two subflows are sent separately over two heterogeneous paths, their inter-subflow RTT difference will be large, which does not satisfy the link-sharing subflow condition.

V. Performance Evaluation

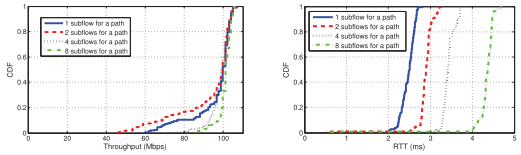
In this section, we evaluate the performance of MPTCP-D through realworld experiments. We measure energy consumption, throughput and latency of long and short flows that use MPTCP-D. Most of our measurements are performed on EC2. We do not know the underlying network topology of VM instances, and we cannot control the background traffic. We also perform testbed experiments and packet-level simulations.

We build up our virtual private cloud and four private subnets on EC2. In the cloud, we create 40 instances as hosts. Each host is attached with four Elastic Network Interfaces (ENIs), and each ENI has the capacity of 256Mbps. Each ENI uses a private IP address to connect to a subnet. Accordingly, there are four available routes between each pair of hosts. We implement MPTCP-D in the MPTCP Linux kernel of version 0.90 [22] and run the kernel on the instances. The instance configuration is shown in TABLE III.

For the testbed experiments and EC2 experiments in this section, we use `iperf` to generate long flows and short flows. We use `iperf` command line option "-P" to generate parallel connections. We read the throughput of each flow from `iperf`'s trace record. We use `tcpdump` and `tcptrace` to analyze the RTTs of a connection. We read host CPU's instant power consumption from Intel's RAPL driver. The experiments take 18 hours in total. The location is EC2's US West datacenter, and the samples in the measurement are cross both time and topology.

TABLE III
EC2 Instance Configuration

Configuration	Parameter
Instance type	c4.xlarge
CPU type	Intel Xeon E5-2666 v3
Number of vCPU cores	4
Memory	7.5GB
Storage	40GB
Availability zone	us-west-2c
Operating system	Ubuntu Server 14.04 LTS
Kernel version	Linux 3.18.34 and MPTCP v0.90



(a) Distribution of throughput (b) Distribution of RTTs

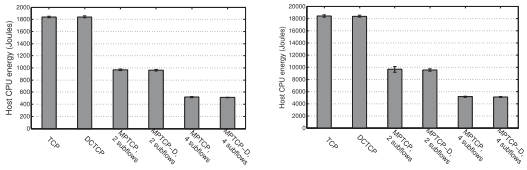
Fig. 9. Comparison of one subflow and multiple subflows for a path: (a) CDF (cumulative distribution function) of throughput; (b) CDF of RTTs.

We perform the measurement and compare the results of MPTCP-D with other transport protocols, including regular TCP, DCTCP [5], MPTCP [7]. We aim to validate that MPTCP-D can achieve energy efficiency for both long and short flows, together with comparable throughput as MPTCP and comparable latency as DCTCP.

A. Energy Efficiency

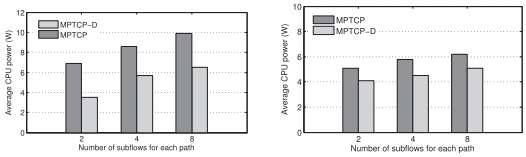
We first analyze the parameter selection in the ESE algorithm. We study how throughput and RTT change with different number of subflows for a path when using MPTCP. To do this, we record the throughput and RTT of the samples in different measurements. We also change the path manager module provide by Linux kernel of version 0.90 for this operation. An MPTCP connection consists of multiple paths and each path can have one or more more subflows with independent congestion windows. Each connection can use the two paths simultaneously to transmit data. On each path, we modify the Linux kernel's MPTCP path-manager module in '/sys/module/mptcp_fullmesh/parameters/num_subflows' and set the num_subflows to be 1, 2, 4 and 8 in different measurements. We use two machines for data transfers. Each machine has a Quad-core Intel Core i7-3770 CPU and two NICs. When the parameter 'num_subflows' is changed (e.g., from 1 to 2), the path delay also changes from low values to high values. We measure the RTT and instantaneous throughput during the data transfers, as shown in Fig. 9.

Fig. 9(a) shows the distribution of the throughput sampled during a multipath data transfer. For each data transfer, we set the number of subflows for a path to be 1, 2, 4 and 8, respectively. For an MPTCP connection, the throughput samples are read from the iperf trace record that outputs an average throughput value once a second during the data transfer. The four cases have a significant portion of samples (around 70%) achieving similar throughput. Fig. 9(b) shows the distribution of RTTs of the samples observed during a multipath data



(a) One connection per host (b) Ten connections per host

Fig. 10. Energy consumption for long flows (10 GB data transfer) on EC2.



(a) Short flow power (b) Long flow power

Fig. 11. Power for both long and short flows over testbed in Fig. 3.

TABLE IV

THROUGHPUT AND RTT WHEN USING DIFFERENT NUMBER OF SUBFLOWS PER PATH

num_subflows=	1	2	4	8
Avg. Throughput (Mbps)	96.1065	93.0940	98.4740	100.7485
Avg. RTT (ms)	2.4649	2.8816	3.4199	4.3138

TABLE V

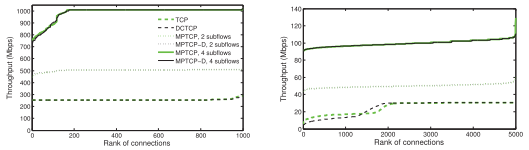
SUCCESS RATE OF CLOSING EXTRA SUBFLOWS `cwnds`

Parameter	$\alpha = 1.1$	$\alpha = 1.3$	$\alpha = 2.3$	$\alpha = 3.4$
$num\_subflows = 2$	0.433	0.904	0.531	0.377
$num\_subflows = 4$	0.352	0.736	0.680	0.621
$num\_subflows = 8$	0.272	0.815	0.693	0.575

transfer. The results indicate that RTT increases significantly as the number of subflows for a path increases from 1 to 8. Let D_i be the average RTT of using i subflows for a path. According to the results in TABLE IV, we can calculate the ratios: $D_2/D_1 \approx 1.2$, $D_4/D_1 \approx 1.4$, $D_8/D_1 \approx 1.8$, $D_4/D_2 \approx 1.2$, $D_8/D_2 \approx 1.5$ and $D_8/D_4 \approx 1.3$. These ratios help us select a proper value of α in Algorithm 1. The value should be large enough to be able to detect the significant change of RTTs, but it should not be too large in order to avoid insensitivity. For example, the above ratios are all less than 2, and if $\alpha > 2$ it is possible that α cannot detect RTT's large change due to the use of multiple subflows. From TABLE V we see that MPTCP-D has high success rate of closing the extra subflow `cwnds` with the parameter $\alpha = 1.3$.

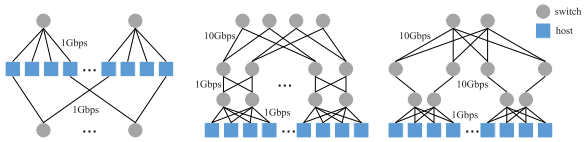
We measure the long flow energy consumption for different transport protocols on EC2. We generate different loads (one connection per host and 10 connections per host) of data transfers for the hosts in the datacenter network. Each connection transmits 10GB data. As shown in Fig. 10, MPTCP-D with two subflows saves up to 50% of aggregated energy of DCTCP, and MPTCP-D with four subflows saves up to 72% of aggregated energy of DCTCP.

To measure the power consumption of both long and short flows using overlapped paths, we still use our testbed in Fig. 3 (note that this experiment cannot be directly conducted over EC2 as its underlying physical network topology is unknown to us). For the testbed in Fig. 3, we set $C_1 = 100$Mbps, $C_2 = 20$Mbps and use `iperf` to generate 200 parallel MPTCP connections between the two machines. We use `iperf` command line option "-t" to generate one second



(a) One connection per host (b) Ten connections per host

Fig. 12. Distribution of throughput in the EC2 experiments. MPTCP-D gets as good utilization as MPTCP (the throughput of MPTCP-D with 2 subflows is similar to that of MPTCP with 2 subflows; the throughput of MPTCP-D with 4 subflows is similar to that of MPTCP with 4 subflows).



(a) BCube (b) FatTree (Multihomed) (c) VL2 (Multihomed)

Fig. 13. The three datacenter topologies with multihomed hosts.

short flows and 170 seconds long flows. This ensures that MPTCP and MPTCP-D have the same flow completion time. Fig. 11 shows the average host CPU power consumption of MPTCP and MPTCP-D with parameters $\alpha = 1.3$ and $\beta = 0.1$. For short flows, MPTCP-D reduces the power consumption by up to 46% compared to MPTCP in the scenario of two subflows per path. For long flows, MPTCP-D reduces power consumption by more than 20%.

B. Throughput and Latency

We next measure the long flow throughput on EC2. Fig. 12 shows the throughput of individual connections for different number of coexisting flows (one connection per host and ten connections per host) and different number of subflows. Our results indicate that, for long flows, MPTCP-D has similar throughput as compared with MPTCP.

To evaluate the long flow throughput in different datacenter topologies, we use `htsim` [28], a packet-level simulator that works well for large scale network traffic and has been used to evaluate the performance of MPTCP in datacenters [8]. Various datacenter topologies have been proposed to address traffic concentration on a small number of bottleneck links in datacenter networks. We study the network utilization of MPTCP-D in three representative topologies, namely, FatTree [29], VL2 [4] and BCube [30]. As shown in Fig. 13, FatTree and VL2 are the hierarchical topologies, which organize switches in access, aggregate, and core layers. VL2 uses 10Gbps links between switches. BCube employs a generalized hypercube topology, making use of some hosts to relay traffic. We set the parameters of the three topologies (FatTree: 128 hosts, 80 switches, 100Mbps 100ms links; VL2: 128 hosts, 80 switches, 1Gbps 100ms links; BCube: 128 host, 64 switches, 100Mbps 100ms links) in our simulations. Each host sends a long-lived MPTCP flow (1000 seconds by referring to long flow duration in [42]) to another host that is chosen at random. In each configuration, we specify the protocol, the topology and the number of subflows. For each configuration, we simulate 10 times and calculate the average of the aggregated throughputs. Fig. 14 and Fig. 15 show that, for the long flows, the throughput of MPTCP-D is similar to that of MPTCP in different datacenter topologies.

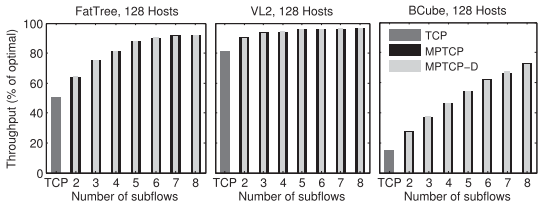


Fig. 14. Aggregated throughput in three datacenter topologies. MPTCP-D gets as good utilization as MPTCP.

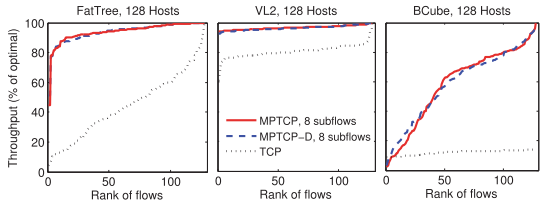


Fig. 15. Distribution of throughput in three types of datacenter topology. MPTCP-D and MPTCP, both with 8 subflows, have high network utilization.

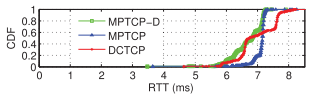


Fig. 16. RTTs measured in the experiments on EC2.

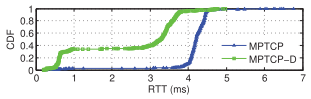
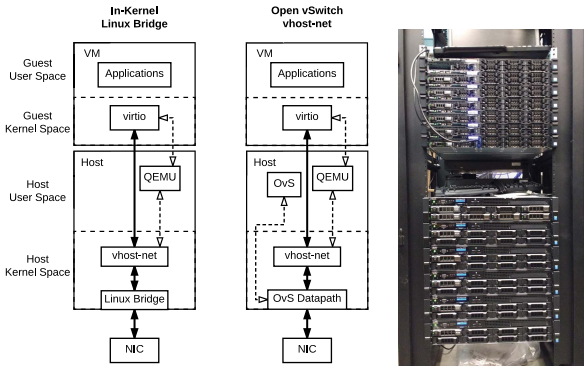


Fig. 17. RTTs measured in the testbed experiments.

For short flows, we examine the latency of using different protocols by both EC2 and testbed experiments. Fig. 16 compares the RTTs for MPTCP-D, MPTCP and DCTCP in our experiments on EC2. The results show that MPTCP-D outperforms the other two protocols. In Equation (2), the parameter ψ_r is used to shift traffic to low latency paths. Experiment results indicate that our design of the RTT-based traffic-shifting parameter works effectively in realworld datacenter environments. Short flows may also be delayed by the overlapped paths in multipath transmission. Fig. 17 shows that, in our testbed experiment for extra subflow elimination, MPTCP-D largely reduces latency compared to MPTCP with multiple subflows sharing a common link. This indicates that the ESE of MPTCP-D can close the `cwnds` of extra subflow on overlapped paths and hence improves the path quality.

C. Extensive Experiments in Virtualized Datacenter

In today's datacenters, networking functions are increasingly performed by software running on physical machines. In this subsection, we investigate the performance of MPTCP and MPTCP-D in virtualized environments. We conduct extensive experiments in different architectures of virtualization, varied network configurations, and different traffic characteristics (e.g., incast and bursty traffic). We first briefly introduce the used network I/O subsystems in the Kernel-based Virtual Machine (KVM) environment and our experimental setup. Fig. 18(a) shows the network I/O subsystems used in our experiments, including Linux bridge and Open vSwitch, together with vhost-net.



(a) KVM (b) Testbed II

Fig. 18. Virtualized environment experiment setup: (a) Network I/O subsystems in the KVM environment: Linux bridge and Open vSwitch with vhostnet; (b) Our testbed II in a cloud.

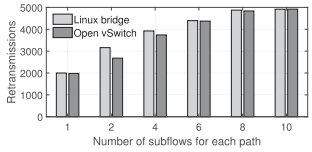


Fig. 19. Retransmission vs. number of subflows per path when using MPTCP in different network configurations.

We conduct the experiments over a cloud testbed[4] as shown in Fig. 18(b). All rack servers in the testbed are Dell PowerEdge R430 servers. Each one is equipped with two Intel Xeon E5-2630 v3 2.4GHz CPU nodes. Supported by Intel hyperthreading technology, a CPU node has 8 physical cores and 16 logical CPUs. The rack server has 256GB RAM with a typical NUMA architecture. It also has two Intel Ethernet Controller 10 Gigabit X540-AT2 NICs, supporting Dual 10 Gbps networking connections. Each server has four additional Broadcom NetXtreme Gigabit Ethernet PCIe NICs. We have two types of ToR switche, including Netgear ProSAFE Smart Managed Switch XS712T and Dell Networking N4032 switch. The Dell Networking N4032 switch supports advanced switching functions, e.g., RED/ECN marking scheme. This scheme ensures that the sources are quickly notified of the queue overshoot before excessive packet loss events happen.

The operating system installed in both hosts and guest VMs is Ubuntu 16.04.2 LTS. We install multipath TCP Release version 0.91 on all virtual machines. The corresponding Linux kernel version is 4.1.39. For the local KVM testbed, we install QEMU version 2.9.0. The virtual switches installed at the hypervisor are Open vSwitch and Linux briding utilities, whose architectures are shown in Fig. 18(a). We compile Open vSwitch 2.7.0 from its source tree. To quantify the virtual switching performance, we use common profiling tools integrated in mainline Linux distribution, including `iperf`, `ping`, and `netstat`. For an in-depth analysis on RTT variations and retransmissions, we use `tcpdump/tcptrace` to extract such information.

Performance in Linux bridge and OvS: In this experiment, we configure a virtual machine on each rack server. Each virtual machine is configured with two virtual interfaces, lever-

[4]https://sfucloud.ca/client/

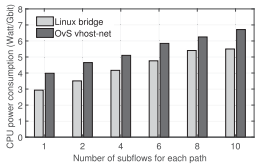
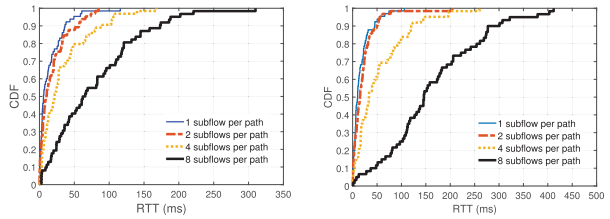
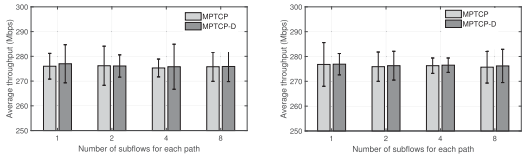


Fig. 20. Power consumption vs. number of subflows per path when using MPTCP in different virtualized network configurations.



(a) Linux bridge (b) OvS host-net

Fig. 21. RTT change with the number of subflows per path for different virtualized network configurations: (a) Linux bridge; (b) OvS host-net.



(a) Linux bridge (b) OvS host-net

Fig. 22. Throughput change with the number of subflows per path in different virtualized network configurations: (a) Linux bridge; (b) OvS host-net.

aging two disjoint paths. We set the bandwidth for each virtual machine to be 300 Mbps[5] and divided equally on the two virtual interfaces. In section III, we have shown that the CPU power consumption increases with the number of subflows per path for heterogeneous path configuration. Fig. 20 shows the same increasing trend of MPTCP energy consumption for homogeneous path configuration. When two subflows share a path, MPTCP increases the energy consumption by up to 20% compared to the case of one subflow per path. When the number of subflows per path increases to 10, the energy consumption of link-sharing subflows is 70% more than that of non-link-sharing subflows. Fig. 21 shows the RTT performance when using different number of subflows per path. These results validate the inefficiency of link-sharing subflows for both Linux bridge and OvS configurations. Fig. 22 shows that the throughput of MPTCP-D is similar to that of MPTCP in this experiment. TABLE VI shows the performance of the ESE algorithm with parameters $\alpha = 1.3$ and $\beta = 0.1$ ($\alpha > \frac{D_2}{D_1} \approx 1.27$ for Linux bridge configuration and $\alpha > \frac{D_2}{D_1} \approx 1.29$ for OvS configuration). The operation of closing extra subflow window in MPTCP-D achieves the true positive rate of 0.833 and the false positive rate of 0.053.

Incast Traffic: Incast is a traffic pattern that usually occurs in many-to-one communication sessions in datacenters. When too many servers are simultaneously sending data via switches, there might be severe packet loss due to switch buffer overflow. Fig. 23 illustrates how incast happens at the virtual switch. Many virtual machines generate the traffic that simultaneously traverse a virtual switch. Since the virtual switch also uses tail drop mechanisms to avoid congestion, incast problem may happen in the bottlenecks of virtual switch. In this experiment, we create 10 VMs on the host with OvS network configuration. On each VM, we use iperf to generate 10 parallel connections (10 MPTCP/MPTCP-D flows). Each VM is configured with two virtual interfaces, each of which has 150Mbps bandwidth. The ESE algorithm uses the parameters $\alpha = 1.3$ and $\beta = 0.1$ ($\alpha > \frac{D_2}{D_1} \approx 1.29$). We examine how the ESE algorithm of MPTCP-D performs with the incast traffic. As shown in Fig. 25, while MPTCP and MPTCP-D achieve similar throughput, MPTCP-D largely reduces the latency if there are link-sharing subflows. TABLE VII shows that the true positive rate decreases and the false positive rate increases compared to the result in TABLE VI. This is due to the interference between the parallel flows, which may cause large variation of RTTs.

[5]Considering the limits of memory and network load in our testbed, we can create at most 30 VMs on a host. The switch port bandwidth is 10Gbps. So, we set a VM's bandwidth to be $300Mbps < \frac{10Gbps}{30}$.

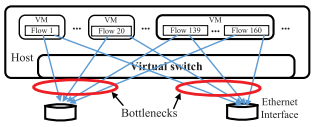


Fig. 23. The scenario of incast at the virtual switch.

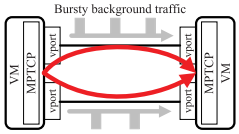
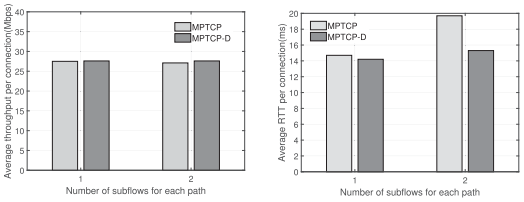


Fig. 24. The scenario of MPTCP flow between VMs with bursty background traffic.



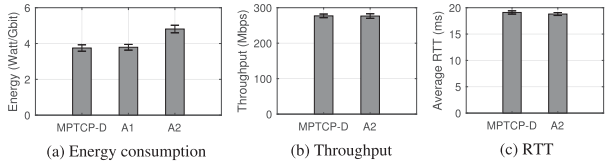
(a) Throughput (b) RTT

Fig. 25. Performance of MPTCP and MPTCP in the incast experiment.

TABLE VI
STATISTICS OF TESTING THE EXTRA SUBFLOW ELIMINATION OVER TESTBED II

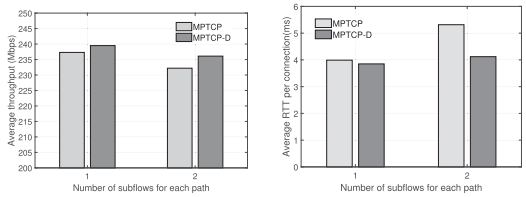
	Has link-sharing subflows?	Detected	Rate
True positive	Yes	Yes	0.833
False negative	Yes	No	0.167
False positive	No	Yes	0.053
True negative	No	No	0.947

Bursty Background Traffic: We also evaluate the performance of MPTCP-D with bursty background traffic. Fig. 24 illustrates the experimental setup. An MPTCP/MPTCP-D connection uses two disjoint paths between two pairs of virtual interfaces, and each path has 150Mbps bandwidth. We use the open source software MGEN [33] to generate on each path a bursty traffic that follows Pareto pattern at rate 50Mbps (more than 30% of the path bandwidth) and occurs at random intervals (average ten seconds) with average bursty duration of two seconds. The ESE algorithm uses parameters $\alpha = 1.3$ and $\beta = 0.1$ ($\alpha > \frac{D_2}{D_1} \approx 1.29$). Fig. 27 shows that MPTCP-D can improve the RTT performance in the case of link-sharing



(a) Energy consumption (b) Throughput (c) RTT

Fig. 26. Contribution of ESE and congestion control.



(a) Throughput (b) RTT

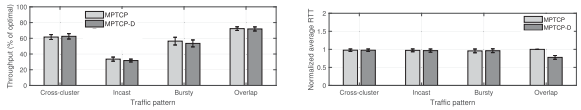
Fig. 27. Performance of MPTCP and MPTCP in the bursty traffic scenario.

subflows without throughput degradation. TABLE VII shows that the performance of ESE is not much impacted by the bursty background traffic, because ESE uses both the variation of intra-subflow RTTs and the comparison of inter-subflow RTTs to decide whether subflows go over the same path. If there are multiple subflows on a path, the intra-subflow RTTs of link-sharing subflows will increase together when bursty background traffic occurs. For the case of non-link-sharing subflows, if a burst occurs on a path, there is large RTT difference between the subflows with and without bursty background traffic. Since two paths are rarely with bursty traffic at the same time, the ESE algorithm can still achieve a low false positive rate.

We also analyze the respective contributions of the two components (ESE and congestion control). To do this, we use the algorithm A1 that incorporates the ESE of MPTCP-D and the congestion control of MPTCP, and the algorithm A2 that removes the ESE but retains the congestion control of MPTCP-D. We perform both link-sharing (two subflows for each path) and non-link-sharing experiments using the same setup as Subsection V.C. For the link-sharing case, Fig. 26(a) shows that A1 and MPTCP-D have similar performance, which means that the ESE mechanism contributes nearly 100% to the reduction of energy consumption. Fig. 26(b) and (c) show the results of the non-link-sharing case. The performance of A2 is similar to MPTCP-D, which indicates that the congestion control of MPTCP-D makes nearly 100% contribution to maintain the same performance with MPTCP. In the practical deployment, a combination of the congestion control of MPTCP and the ESE may achieve the similar performance as MPTCP-D. Yet, as we have mentioned in Subsection IV.B, the MPTCP-D's congestion control satisfies Pareto-optimality and also theoretically ensures fair resource allocation among many coexisting flows in datacenter networks.

D. Packet-Level Simulation and Algorithm Analysis

We perform packet-level simulations with the discrete-event network simulator ns-3 [44] of version 3.27. The datacenter network topology is a FatTree with 128 hosts and 80 switches. The hosts are separated into two clusters. Each host has two routes to another host and uses MPTCP and MPTCP-D to transmit data, respectively. The FatTree topology has three layers. The access layer has 128 hosts, which are separated



(a) Throughput (b) RTT

Fig. 28. Performance in packet-level simulation.

into two clusters. Each cluster has 16 racks. Each rack has 2 access switches and 4 hosts. In a rack, each host is connected to the two switches. Each cluster has 4 aggregation switches, two of which can reach any of 16 access switches, and the other two of which can reach any of the other 16 access switches in the same cluster. The other cluster uses the same configuration. The core layer has 8 switches, each connected to two aggregation switches that belongs to different clusters. On this topology, we generate four traffic patterns. 1) Cross-cluster traffic: each host communicates with another host from a different cluster and the traffic workloads increase from 16 host-to-host transfers to 64 host-to-host transfers. 2) Incast traffic: 32 hosts transmit data simultaneously to one host. 3) Bursty traffic: bursty background traffic exists on the links between ToR and aggregation switches. 4) Path-overlap traffic: two host-to-host paths have a common link between ToR and aggregation. We record the simulation results including throughput and RTT for MPTCP and MPTCP-D, respectively. We simulate 10 times and calculate the average throughput and average RTT (normalized by the maximum value between MPTCP and MPTCP-D, i.e., $RTT' = RTT/max(RTT^{(MPTCP)}, RTT^{(MPTCP-D)})$). Fig. 28(a) shows that MPTCP and MPTCP-D achieve similar throughput for the four different traffic patterns. Fig. 28(b) shows that MPTCP-D outperforms MPTCP in RTT for path-overlap traffic.

As we have discussed in sections III and IV, The energy efficiency of MPTCP-D comes from two aspects: (i) it has higher throughput than single-path TCP; and (ii) it can close extra subflows when subflows share a link. In subsections V.A and V.C, our experiments over a testbed, a campus cloud and EC2 datacenters, demonstrate the improvement of energy efficiency from these two aspects. Our simulation results in this subsection further demonstrate that MPTCP-D can also provide similar throughput and RTT compared with MPTCP in a large topology and different traffic patterns. According to all of our results in section V, MPTCP-D can achieve energy efficiency without other performance degradation.

The sensitivity of the ESE mechanism is examined for different traffic patterns using packet-level simulation. The non-link-sharing case uses the same configuration as Fig. 28. For the link-sharing case, the incast traffic uses 32 hosts to simultaneously transmit data with MPTCP-D to a destination host, and the subflows from the 32 hosts are configured to traverse the same link between ToR and aggregation. For each host-to-host data transfer, the bursty background traffic is generated on a link between ToR and aggregation, and the transfer's two subflows share that link. We calculate the true positive (i.e., there is a link shared by subflows and it is correctly identified) rate and the false positive (i.e., wrongly reporting a shared link that in reality does not exist) rate. TABLE VII compares the simulation result with the testbed result in Subsection V.C. We find that the performance of ESE

TABLE VII
STATISTICS OF TESTING THE EXTRA SUBFLOW ELIMINATION

Traffic	Rate	Testbed II	ns-3 simulation
Incast	True positive	0.778	0.706
	False positive	0.133	0.229
Bursty	True positive	0.818	0.684
	False positive	0.083	0.154

TABLE VIII
ESE ACCURACY FOR DIFFERENT VALUES OF α

	Incast		Bursty	
	True positive	False positive	True positive	False positive
α=1.1	0.733	0.471	0.705	0.398
α=1.3	0.706	0.229	0.684	0.154
α=1.5	0.651	0.207	0.607	0.136
α=1.7	0.595	0.197	0.535	0.132
α=1.9	0.523	0.173	0.510	0.115
α=2.1	0.470	0.164	0.448	0.096

has degradation in the simulation. The main reason is that our current implementation of ESE relies much on the threshold α and has limitations to obtain associated traffic information. TABLE VIII shows the true positive rate and false positive rate of ESE for different α. We can see that $\alpha = 1.3$ achieves a balance between true positive and false positive. When α is smaller than 1.3, ESE is hyper-sensitive with a high false positive. When α is larger than 1.3, the true positive rate decreases. This is also validated under different traffic patterns (e.g., incast and bursty) over both ns-3 simulations and Linux-based experiments, as shown in TABLE VI, TABLE VII, and TABLE VIII. Based on the experimental data, we find that $\alpha = 1.3$ is a typical value of good performance for a wide range of traffic patterns. We select $\alpha = 1.3$ as the default value for MPTCP-D and also allow users to reconfigure it for their practical deployment, which may be complex with varying traffic. A user can fine-tune α to adapt to a specific traffic pattern based on traffic monitoring and analysis. For example, the user can collect network performance metrics using the traffic monitoring tools, e.g., [45], and accordingly use the traces as specific traffic pattern to evaluate MPTCP-D.

VI. CONCLUSION

In this paper, we have presented a systematic study on the energy efficiency of MPTCP for datacenters. Although MPTCP can increase host CPU power by about 14%-36% compared to TCP, it can fully utilize datacenter network bandwidth to achieve high aggregated throughput and thus largely saves energy for long flows. Yet in the scenarios where multiple subflows share a common link, MPTCP has poor performance in power consumption and latency, especially for short flows. We present MPTCP-D to reduce the power consumption for both long and short flows. The ESE of MPTCP-D can detect link-sharing subflows and eliminate the extra subflows on the shared link, thereby reducing the power consumption. Our experimental results have shown that MPTCP-D achieves good energy efficiency with a throughput similar to MTPCP and a latency better than both DCTCP and MPTCP. MPTCP-D has great potential to evolve in different network scenarios and support a variety of high-quality services. Large-scale datacenter topologies with realistic traffic will impact the performance of ESE. For more complex application scenarios (e.g., some hotspot/transient congestion created by bursty traffic, and non-equal-cost paths in a cubic topology), a possible solution is to combine the threshold α with the detection of specific traffic patterns. It involves cross-layer designs, and we will focus on it in future work.

REFERENCES

[1] *America's Data Centers Consuming and Wasting Growing Amounts of Energy*. Accessed: Dec. 4, 2019. [Online]. Available: https://www.nrdc.org/resources/americas-data-centers-consuming-andwastinggrowing-amounts-energy

[2] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," in *Proc. ACM ISCA*, 2010, pp. 338–347.

[3] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. ACM IMC*, 2010, pp. 267–280.

[4] A. Greenberg *et al.*, "VL2: A scalable and flexible data center network," in *Proc. ACM SIGCOMM*, 2009, pp. 51–62.

[5] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," in *Proc. ACM SIGCOMM*, 2010, pp. 63–74.

[6] R. Mittal *et al.*, "TIMELY: RTT-based congestion control for the datacenter," in *Proc. ACM SIGCOMM*, 2015, pp. 537–550.

[7] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. Usenix NSDI*, 2011, pp. 1–14.

[8] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proc. ACM SIGCOMM*, 2011, pp. 1–12.

[9] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J.-Y. Le Boudec, "MPTCP is not Pareto-optimal: Performance issues and a possible solution," in *Proc. ACM CoNEXT*, 2012, pp. 1651–1665.

[10] Y. Cao, M. Xu, X. Fu, and E. Dong, "Explicit multipath congestion control for data center networks," in *Proc. ACM CoNEXT*, 2013, pp. 73–84.

[11] Y. Cao and M. Xu, "Dual-NAT: Dynamic multipath flow scheduling for data center networks," in *Proc. IEEE ICNP*, Oct. 2013, pp. 1–2.

[12] A. Agache, R. Deaconescu, and C. Raiciu, "Increasing datacenter network utilisation with GRIN," in *Proc. Usenix NSDI*, 2015, pp. 1–15.

[13] M. Kheirkhah, I. Wakeman, and G. Parisis, "MMPTCP: A multi-path transport protocol for data centers," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[14] G. Chen *et al.*, "Fast and cautious: Leveraging multi-path diversity for transport loss recovery in data centers," in *Proc. USENIX ATC*, 2016, pp. 1–15.

[15] S. Hassayoun, J. Iyengar, and D. Ros, "Dynamic window coupling for multipath congestion control," in *Proc. IEEE ICNP*, Oct. 2011, pp. 341–352.

[16] S. Ferlin, Ö. Alay, T. Dreibholz, D. A. Hayes, and M. Welzl, "Revisiting congestion control for multipath TCP with shared bottleneck detection," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[17] M. Lin, A. Wierman, L. L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1378–1391, Oct. 2013.

[18] *Intel 64 and IA-32 Architectures Software Developers Manual*, vol. 1, 2A, 2B, 2C, 3A, 3B and 3C, Intel Corp., Santa Clara, CA, USA, 2014.

[19] D. Abdurachmanov *et al.*, "Techniques and tools for measuring energy efficiency of scientific software applications," *J. Phys., Conf. Ser.*, vol. 608, no. 1, 2015, Art. no. 012032.

[20] *Running Average Power Limit*. Accessed: Dec. 4, 2019. [Online]. Available: https://01.org/zh/blogs/2014/running-average-power-limit-%E2%80%93-rapl?langredirect=1

[21] *Tools for Setting and Accessing Advanced Low-Level CPU Features*. Accessed: Dec. 4, 2019. [Online]. Available: https://github.com/deater/uarch-configure

[22] *MultiPath TCP—Linux Kernel Implementation*. Accessed: Dec. 4, 2019. [Online]. Available: http://multipath-tcp.org

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

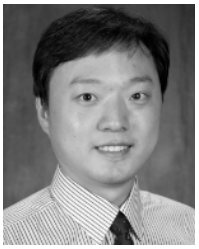
[24] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley, "Multi-path TCP: A joint congestion control and routing scheme to exploit path diversity in the Internet," *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 1260–1271, Dec. 2006.

[25] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath congestion control for shared bottleneck," in *Proc. PFLDNeT Workshop*, 2009, pp. 1–6.
[26] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath TCP," in *Proc. IEEE ICNP*, Oct./Nov. 2012, pp. 1–10.
[27] S. M. Rumble, D. Ongaro, R. Stutsman, M. Rosenblum, and J. K. Ousterhout, "It's time for low latency," in *Proc. HotOS*, vol. 13, 2011, pp. 1–5.
[28] *Multipath TCP Implementations*. Accessed: Dec. 4, 2019. [Online]. Available: http://nrg.cs.ucl.ac.uk/mptcp/implementation.html
[29] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM*, 2008, pp. 63–74.
[30] C. Guo *et al.*, "BCube: A high performance, server-centric network architecture for modular data centers," in *Proc. ACM SIGCOMM*, 2009, pp. 63–74.
[31] *Global Warming: Data Centres to Consume Three Times as Much Energy in Next Decade, Experts Warn*. Accessed: Dec. 4, 2019. [Online]. Available: http://www.independent.co.uk/environment/global-warming-data-centres-to-consume-three-times-as-much-energy-in-next-decade-experts-warn-a6830086.html
[32] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, design, and implementation," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 596–609, Feb. 2016.
[33] *Multi-Generator (MGEN)*. Accessed: Dec. 4, 2019. [Online]. Available: https://www.nrl.navy.mil/itd/ncs/products/mgen
[34] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, *Architectural Guidelines for Multipath TCP Development*. document IETF RFC 6182, 2011.
[35] C. Raiciu, M. Handly, and D. Wischik, *Coupled Congestion Control for Multipath Transport Protocols*, document IETF RFC 6356, 2011.
[36] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, *TCP Extensions for Multipath Operation With Multiple Addresses*, document IETF RFC 6824, 2013.
[37] *Apple iOS 7 Surprises as First With New Multipath TCP Connections*. Accessed: Dec. 4, 2019. [Online]. Available: https://www.networkworld.com/article/2170068/lan-wan/apple-ios-7-surprises-as-first-with-new-multipath-tcp-connections.html
[38] *Apple Opens Multipath TCP in iOS11*. Accessed: Dec. 4, 2019. [Online]. Available: http://www.tessares.net/highlights-from-advances-in-networking-part-1/
[39] *Maximize Mobile User Experience With NetScaler Multipath TCP*. Accessed: Dec. 4, 2019. [Online]. Available: https://www.citrix.com/blogs/2013/05/28/maximize-mobile-user-experience-with-netscaler-multipath-tcp/
[40] *BIG-IP Application Acceleration Manager*. Accessed: Dec. 4, 2019. [Online]. Available: https://f5.com/Portals/1/PDF/JAPAN/PRODUCTS/AAM/BIP_AAM_datasheet.pdf
[41] *Commercial Usage of Multipath TCP*. Accessed: Dec. 4, 2019. [Online]. Available: http://blog.multipath-tcp.org/blog/html/2015/12/25/commercial_usage_of_multipath_tcp.html
[42] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proc. ACM IMC*, 2009, pp. 202–208.
[43] J. Zhao, J. Liu, H. Wang, and C. Xu, "Multipath TCP for datacenters: From energy efficiency perspective," in *Proc. IEEE INFOCOM*, May 2017, pp. 1–9.
[44] *ns-3 Network Simulator*. Accessed: Dec. 4, 2019. [Online]. Available: https://www.nsnam.org
[45] *ns-3 Tutorial*. Accessed: Dec. 4, 2019. [Online]. Available: https://www.nsnam.org/docs/release/3.9/tutorial/tutorial_23.html

Jia Zhao received the M.S. degree in electronic and information engineering from Beijing Jiaotong University, Beijing, China. He is currently pursuing the Ph.D. degree with the School of Computing Science, Simon Fraser University, BC, Canada. His research interests include backscatter communication for low-power sensors, the Internet of Things, embedded and wireless systems, congestion control, and transport protocols.

Jiangchuan Liu (S'01–M'03–SM'08–F'17) received the B.Eng. degree *(cum laude)* from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree from The Hong Kong University of Science and Technology in 2003.

He is currently a Full Professor (with University Professorship) with the School of Computing Science, Simon Fraser University, BC, Canada. He is also a Fellow of the Canadian Academy of Engineering and an NSERC E.W.R. Steacie Memorial Fellow. He is also a Steering Committee Member of the IEEE TRANSACTIONS ON MOBILE COMPUTING and an Associate Editor of the IEEE/ACM TRANSACTIONS ON NETWORKING, the IEEE TRANSACTIONS ON BIG DATA, and the IEEE TRANSACTIONS ON MULTIMEDIA. He was a co-recipient of the Test of Time Paper Award of IEEE INFOCOM in 2015, the ACM TOMCCAP Nicolas D. Georganas Best Paper Award in 2013, and the ACM Multimedia Best Paper Award in 2012.



Haiyang Wang (S'08–M'13) received the Ph.D. degree in computing science from Simon Fraser University, Burnaby, BC, Canada, in 2013. He is currently an Associate Professor with the Department of Computer Science, University of Minnesota at Duluth, Duluth, MN, USA. His research interests include cloud computing, peer-to-peer networking, social networking, big data, and multimedia communications.

Chi Xu (S'14) received the B.Sc. degree in software engineering from Xidian University, Xi'an, China, in 2013, and the M.Sc. degree in computing science from Simon Fraser University in 2016. He is currently a Research Assistant with the School of Computing Science, Simon Fraser University. His current research interests include computer and network virtualization, performance issues in cloud computing, and hardware support for big data applications.

Wei Gong (M'14) received the B.S. degree from the Department of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, in 2003, and the M.S. and Ph.D. degrees from the Department of Computer Science and Technology, School of Software, Tsinghua University, Beijing, China, in 2007 and 2012, respectively. He was a Research Fellow with the School of Computing Science, Simon Fraser University, Canada. He is currently a Professor with the School of Computer Science and Technology, University of Science and Technology of China. His research interests include backscatter networks, mobile computing, and the Internet of Things.

Changqiao Xu (M'04–SM'15) received the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, Beijing, China, in January 2009. He is currently a Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing. He has authored/coauthored more than 100 technical articles in prestigious international journals and conferences. His research interests include wireless networking and multimedia communications. He was a Co-Chair of the IEEE MMTC Interest Group, Green Multimedia Communications.