Multipath TCP for Datacenters: From Energy Efficiency Perspective

Jia Zhao†, Jiangchuan Liu†, Haiyang Wang* and Chi Xu†
†School of Computing Science, Simon Fraser University, Canada
*Department of Computer Science, University of Minnesota at Duluth, USA
Email: zhaojiaz@sfu.ca, jcliu@cs.sfu.ca, haiyang@d.umn.edu, chix@sfu.ca

***Abstract*—Nearly 50% of the energy overhead in today's datacenters comes from host-to-host data transfers, which largely depend on the transport layer performance. Multipath TCP (MPTCP) has recently been suggested as a promising transport protocol to improve datacenter network throughput, 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.**

By analyzing the performance of MPTCP, we show that (1) despite consuming higher host CPU power than TCP, MPTCP can largely reduce the long flow completion time and thus save the aggregated energy; (2) link-sharing subflows in MPTCP not only has negative impact on both throughput-sensitive long flows and latency-sensitive short flows, but also noticeably increases the host CPU power, especially for short flows. We present MPTCP-D, an energy-efficient variant of multipath TCP for datacenters. MPTCP-D incorporates a novel congestion control algorithm that can provide energy efficiency by minimizing the flow completion time, and an extra subflow elimination mechanism that can preclude link-sharing subflows from increasing the host CPU power. We implement MPTCP-D in the Linux kernel and study its performance by experiments on Amazon EC2. Our results show that, without degrading the performance of the long flow throughput and 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.

I. Introduction

Tremendous carbon footprint has been produced by the energy-hungry mega datacenters. For example, 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 the scale of 500 megawatts) [1]. As such, the new generation of datacenters is experiencing an evolution towards both high-performance and green computing.

It is known that the servers in today's datacenters consume most of the energy because of their high density in deployment and massive job-processing workload. The servers at full utilization can take up to 90% of a datacenter's power [2], and a great portion of their running time has been for transmitting or receiving data (for instance, data transfers account for more than 50% of job completion time in Facebook's Hadoop cluster [3]). That said, nearly half of the datacenter energy consumption is network communication related.

The energy consumption of host-to-host data flows depends on both host CPU power[1] and flow completion time. A host's instantaneous CPU power during data transfers noticeably increases compared to that in the idle state, and it increases with 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) [4, 5]. 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 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. Although such customized datacenter transport protocols as DCTCP [6] and TIMELY [7] have attempted to serve latency-sensitive short flows better, they are not optimized for long flows and the overall energy.

Multipath transport protocols (e.g., Multipath TCP (MPTCP) [8]) explore multiple routes between each pair of hosts to increase throughput, thereby reducing the flow completion time, particularly for long flows [9]. There has been significant research on improving datacenter's performance with MPTCP [9–15], mostly from throughput and latency perspectives. Throughput improvement brought by using multiple interfaces however would also increase the instantaneous host CPU power. It remains unclear whether datacenter can indeed benefit from using MPTCP from the energy efficiency perspective. In fact, it has been shown that short MPTCP flows' completion time may increase when a large number of subflows are used [14]. This would severely delay latency-sensitive short flows, and the consequent expirations and retransmissions would increase power consumption.

This paper takes a first step towards understanding MPTCP's energy efficiency in datacenter networks. By an-

[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.

alyzing the performance of standard MPTCP over 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 long flows' throughput and short flows' completion time, and consequently consumes significantly more energy.

Motivated by 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 introduces the Round Trip Time (RTT) based traffic shifting for short flows. To reduce power consumption of link-sharing sublfows, an Extra Subflow Elimination (ESE) mechanism is further developed to close the congestion windows (`cwnds`) of extra subflows, ensuring that only one single subflow exists on overlapped paths.

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, which reduces up to 20% and 46% of power consumption for long and short flows, respectively.

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

II. Related Work

There have been significant studies to investigate the transport layer design in datacenter networks. TCP shows inefficiency for both long and short flows in datacenters. Advanced single-path transport protocols, such as DCTCP [6] and TIMELY [7], have been developed to improve the performance of short flows in datacenter networks. Multipath transmission has also been suggested, and Raiciu *et al*. [9] validated that MPTCP can improve datacenter network utilization on different network scales over different topologies. Khalili *et al*. [10] 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 [12, 13], aiming to make full utilization of their high aggregation bandwidth.

Besides bandwidth efficiency, latency has long been a critical issue for the datacenter networks. The optimization of latency-sensitive short flows is attracting increased attention in recent years. Cao *et al*. [11] proposed a multipath congestion control scheme for datacenters to explore path diversity and better link utilization. This scheme uses link queue buffer control and traffic shifting to balance throughput and latency. Kheirkhah *et al*. [14] proposed a random packet scheduler to exploit good-quality paths for short multipath flows and reduce their completion time. Chen *et al*. [15] 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 the lossy path.

The energy efficiency of MPTCP for datacenters, however, remains largely unclear to the research community. We evaluate the performance of MPTCP over realworld experiments and find that MPTCP has severe energy issues. In particular, when the subflows are sharing one/many common links, MPTCP has poor energy consumption, especially for short flows. To deal with the issue, we incorporate the ESE module in MPTCP-D. In order to close the `cwnds` of the extra subflows, ESE first finds whether multiple subflows are sharing a common link. There have been methods for bottleneck link detection with multipath congestion control in the literature [16, 17]. 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 [16] or take 10-20 seconds to make a decision [17], which is inefficient for the latency-sensitive short flows.

III. Measurement of MPTCP Energy Consumption

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

A. 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 examine the CPU power increase and flow completion time decrease of using MPTCP, compared to the baseline of using TCP.

We measure the host CPU power during host-to-host data transfers in our server cluster and over Amazon EC2. Our cluster has 10 machines, each with double NICs and a Quad-core Intel Core i7-3770 CPU. We also rent server instances on EC2 and configure them with different types of CPUs. These include Quad-core Intel Xeon E5-2680 v2, Octa-core

[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 [8, 16, 17] 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.

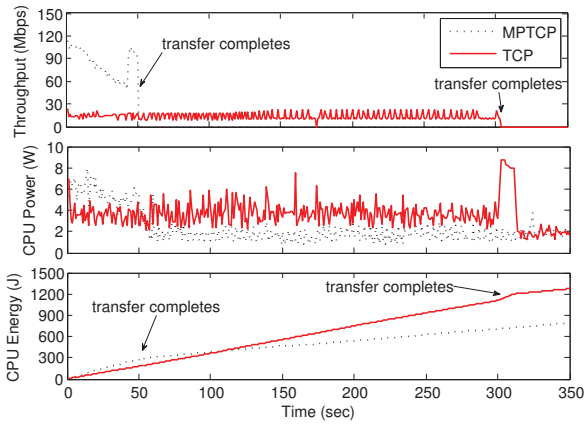
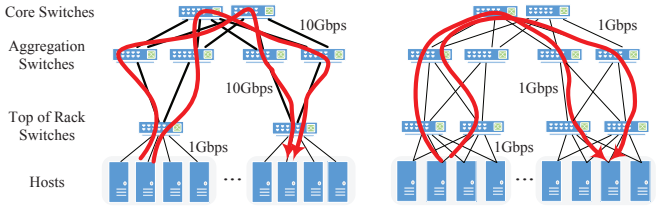


Fig. 1. Performance of long flows (500 MB data transfer).



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

Fig. 2. MPTCP with multiple subflows on overlapped paths: (a) VL2 topology, path overlap at host; (b) multihomed topology, path overlap at 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 [18] on our local servers and EC2 instances. We use `iperf` to generate host-to-host long flows (500MB data). We record instant host CPU power from Intel's Running Average Power Limit (RAPL) driver [19, 20]. Each classic TCP connection uses one interface with the capacity ranging from 20Mbps to 210Mbps. An MPTCP connection uses two interfaces, each of them with the capacity ranging from 20Mbps to 512Mbps. Note that our measurements are in the circumstance under which MPTCP is several times the available bandwidth of TCP. There is no concern with the fairness issue. This is just for the premise that MPTCP can achieve high aggregated bandwidth in multihomed datacenter networks.

Fig. 1 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 avoid possible bias, we also validate this observation under different throughput with different types of CPUs. TABLE I validates that MPTCP consumes less energy in all the test cases.

TABLE I
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	CPU 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%

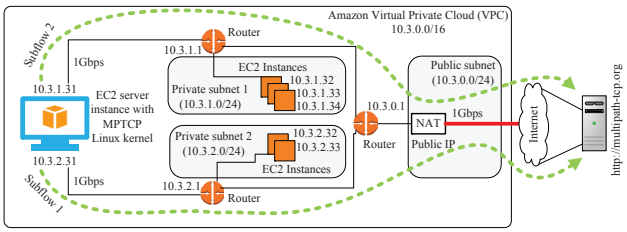


Fig. 3. MPTCP with multiple subflows on overlapped paths in a VPC of Amazon EC2.

B. Mutual Interference between Subflows on shared links

As shown in Fig.2, typical datacenter networks can hardly avoid path overlap for MPTCP's subflows. This path overlap may be either the links between hosts and Top of Rack (ToR) switches, or the core switches with limited capacity [9]. As shown in Fig. 3, on such cloud computing platforms as Amazon EC2, the use of MPTCP may also have overlapped paths for multiple subflows. Fig. 3 shows an example of MPTCP in Amazon's Virtual Private Cloud (VPC). The VPC builds a public subnet and uses NAT to access the Internet. The VPC expands its network by connecting two private subnets to the public subnet. All traffic exchanges between the Internet and the VPC are with a public IP address. When an EC2 instance with MPTCP Linux kernel uses two paths to transmit data towards an MPTCP-enabled destination (e.g., http://multipath-tcp.org) in the Internet, the two subflows of MPTCP share the link between the VPC's public IP address and the Internet gateway.

To understand the mutual interference between subflows of MPTCP on a shared link, we deploy MPTCP on our local testbed (Fig. 4). We use the MPTCP Linux kernel of version 0.90, in which an MPTCP connection consists of

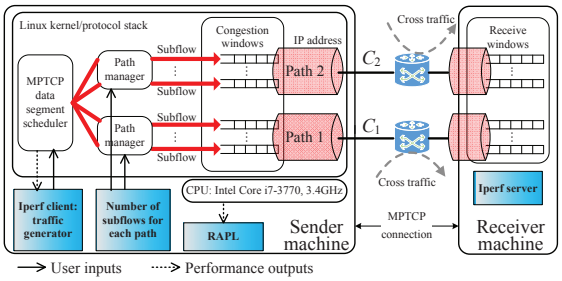


Fig. 4. Testbed setups for the experiments that emulate the path overlap scenarios: MPTCP connection is set up between sender and receiver machines, each one is configured with two interfaces (path 1 has capacity C_1 and path 2 has capacity C_2), Intel Core i7-3770 CPU, and the MPTCP Linux kernel v0.90. `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).

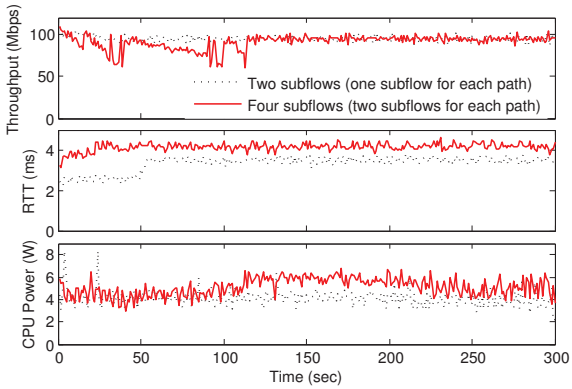
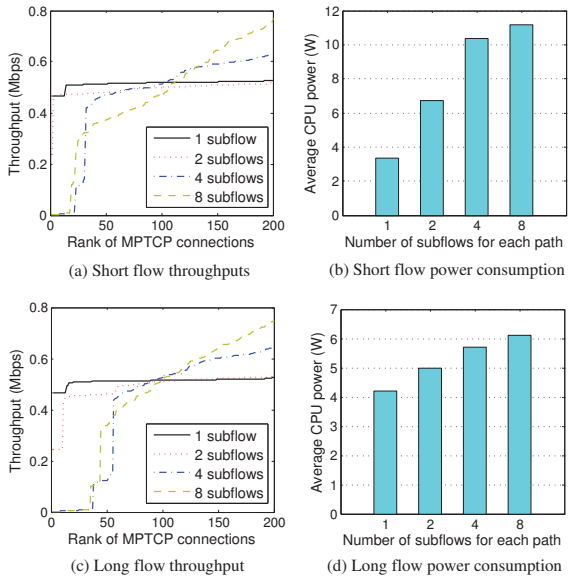


Fig. 5. Performance comparison of the two cases: the MPTCP connection with four subflows sharing two paths consumes more CPU power than the connection using two disjoint paths, and it also degrades throughput and has higher latency.

multiple paths. Each path can have one or more subflows with independent congestion windows.

Fig. 5 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 set $C_1 = 100$Mbps and $C_2 = 20$Mbps, respectively. It is easy to see that the second case consumes more energy, with decreased aggregated throughput and increased latency. 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 (with bandwidth $C_1 = 100$Mbps and $C_2 = 20$Mbps). To avoid possible bias, we also consider both long (100 MB data) and short flows (0.5 MB data). 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 less than 0.90).

Fig. 6(a) shows the short flow throughput of individual



(a) Short flow throughputs (b) Short flow power consumption

(c) Long flow throughput (d) Long flow power consumption

Fig. 6. 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.

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 connections with 4 or 8 subflows for each path get higher throughputs, they affect the throughputs of the other half of connections, and thus nearly 50% of short flows are severely delayed. Fig. 6(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. 6(c), using multiple subflows for each path also cannot improve throughput, either, and some connections with 4 and 8 subflows experience severe throughput degradation. Fig. 6(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 indicates that the existence of link-sharing subflows in MPTCP hurts the throughput and hence the energy consumption. Our results in Fig. 5 and Fig. 6 indicate that the link-sharing subflows can degrade the performance of 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 the stable equilibrium.

C. Discussions

Our experiment results indicate that transport layer energy efficiency is correlated with multiple factors including the

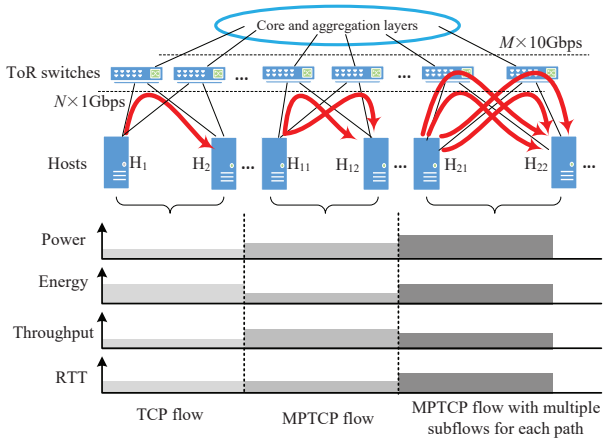


Fig. 7. A summary: transport layer energy efficiency is correlated with the factors including number of paths, number of subflows for a path, throughput and RTT. (1) TCP vs. MPTCP: low energy can be achieved by using more paths and increasing aggregated throughput. (2) MPTCP vs. MPTCP with link-sharing subflows: MPTCP should have one subflow for each path to save energy when it cannot further improve throughput and RTT.

number of concurrently used paths, the number of subflows for each path, aggregated throughput and RTT. Fig. 7 briefly summarizes our observations in the subsections above. It compares three different cases: classic TCP flow, MPTCP flow without link-sharing subflows, and MPTCP flow with link-sharing subflows (these cases are just illustrative examples; as shown in Fig. 2 and Fig. 3, there can be various situations under which MPTCP has link-sharing subflows). Among them, MPTCP without link-sharing subflows has the best overall performance. Compared with TCP, it saves energy by reducing the flow completion time (due to concurrent usage of multiple paths and high aggregated throughput). Such benefits disappear however in the presence of link-sharing subflows. In short, MPTCP should have only one subflow for each path, so that energy efficiency can be achieved by minimizing the flow completion time.

IV. The Design of MPTCP-D

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.

A. Congestion Control

MPTCP-D does not directly use the existing multipath congestion control algorithms, e.g., LIA [8] and OLIA [10]. It has been shown in [10] 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.

We first describe the network model which is similar to [10]. 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 users (i.e. connections) in the network. For any user $s \in S$, let s represent the set of available paths between s and its destination. A route $r \in s$ consists of multiple links. If route r uses a link l, then we denote $l \in r$. For each route r, $RTT_r(t)$ and $w_r(t)$ denote the round trip time and congestion window, respectively, and $x_r(t) = w_r(t)/RTT_r(t)$ represents the send rate at time t. For each user $s \in S$, 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 link l is on the route r of user s and 0 otherwise, and $y_l = \sum_{s \in S} \sum_{r \in s} R^l_{sr} x_r$ be the aggregate traffic on link l. Let $p_l(y_l)$ be the packet loss probability at link l. p_l is an increasing function of y_l with the constraint $y_l \leq c_l$. The packet loss probability on route 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 $\mathbf{x}_s$.

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. These goals lead to the following fluid model design:

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

where the traffic-shifting parameter

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

We now illustrate how the fluid model serves our design goals.

According to the design goal of TCP-friendliness in [8] and RFC 6356, 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 [10], 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 user s at the equilibrium of Equation (1) is no more than the send rate that a regular TCP user would achieve on the best path of s:*

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

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

Proof: The traffic parameter ψ_r in Equation (2) satisfies $\psi_r \leq 1$ for any path $r \in s$. Hence we have $\psi_h \leq 1$. 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. ∎

Besides TCP-friendliness, MPTCP-D also has to accommodate the many coexisting flows in datacenter networks, where the the number of active flows at a switch in any given second could be as high as 10,000 [4]. That said, Pareto-optimality should be maintained here. Suppose there are $|S|$ users using the same multipath congestion control algorithm in the network. Each user $s \in S$ 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 users is as follow

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

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 $\theta_r(\mathbf{x'}_s)\left.\frac{\partial U_s(\mathbf{x}_s)}{\partial x_r}\right|_{\mathbf{x}_s=\mathbf{x'}_s} = I_r(\mathbf{x'}_s)$, 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 user s without decreasing the throughput of other users 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 [21, 22], EWTCP [23], DWC [16] and OLIA [10] all have $\theta_r(\mathbf{x}_s) = x_r^2$, and wVegas [24] 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.

For short 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.

Equations (1) and (2) show that the traffic shifting parameter ψ_r affects the aggressiveness of send rate increase. 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 path delay. As shown in [7], 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: link congestion level and 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 [25]. This means that path delays can sensitively reflect the quality difference among multiple available paths.

B. Extra Subflow Elimination

As shown in Fig. 6, 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. The design of ESE refers to 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

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

link at the stable state can be 30% higher than that at the starting state. 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 RTT values similar to each other. 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 user 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 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.

V. Performance Evaluation

In this section, we evaluate the performance of MPTCP-D through realworld experiments. We rent virtual machine instances on Amazon EC2, which is a large-scale commercial datacenter. We measure energy consumption, throughput and latency of long and short flows that use MPTCP-D. Most of our measurements are performed on EC2. But we do not know the physical network topology under our VM instances, and we cannot control the background traffic. Hence, some measurements and evaluation are performed on our testbed and by packet-level simulations.

TABLE II
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

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 [18] and run the kernel on the instances. The instance configuration is shown in TABLE II.

We use `iperf` to set up connections between each pair of hosts and record their throughputs. 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.

We perform the measurement and compare the results of MPTCP-D with other transport protocols, including regular TCP, DCTCP [6], MPTCP [8]. 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 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. 8, 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. 4 (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. 4, we set $C_1 = 100$Mbps, $C_2 = 20$Mbps and use `iperf` to generate 200 parallel MPTCP connections between the two machines. Each short flow transmits 0.5MB data and each long flow transmits 100MB data. From TABLE III we see that MPTCP-D has

TABLE III
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

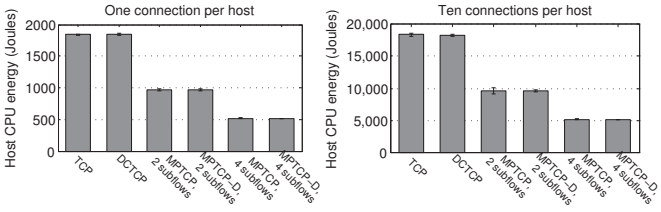
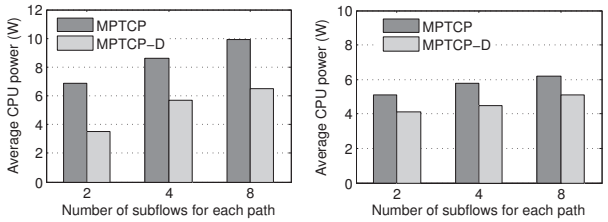


Fig. 8. Energy consumption for long flows (10GB data transfer) on EC2



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

Fig. 9. Power for both long and short flows over testbed in Fig. 4.

high success rate of closing the extra subflow `cwnds` with the parameter $\alpha = 1.3$. Fig. 9 shows the average host CPU power consumption of MPTCP and MPTCP-D with the parameters $\alpha = 1.3$ and $\beta = 0.1$. For short flows, MPTCP-D reduces power consumption by up to 46% compared to MPTCP in the scenario of two subflows per path. For long flows, MPTCP-D saves more than 20% of power consumption of MPTCP.

B. Throughput and Latency

We next measure the long flow throughput on EC2. Fig. 10 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` [26], a packet-level simulator that works well for large scale network traffic and has been used to evaluate the performance of MPTCP in datacenters [9]. 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 [27], VL2 [5] and BCube [28]. FatTree and VL2 follow the hierarchical topology to organize switches in access, aggregate, and core layers. VL2 uses 10Gbps links between switches. BCube employs the generalized hypercube topology rather than hierarchical organization of switches, and it makes use of some hosts to relay traffic. We set the parameters of the

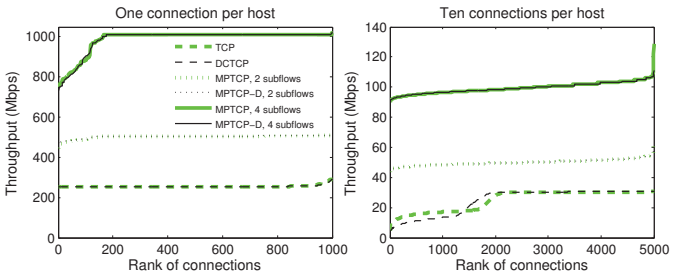


Fig. 10. Distribution of throughput in the EC2 experiments. MPTCP-D gets as good utilization as MPTCP (the throughput of MPTCP-D with 2 subflows is similar to that of MPTCP with 2 subflows; the throughput of MPTCP-D with 4 subflows is similar to that of MPTCP with 4 subflows).

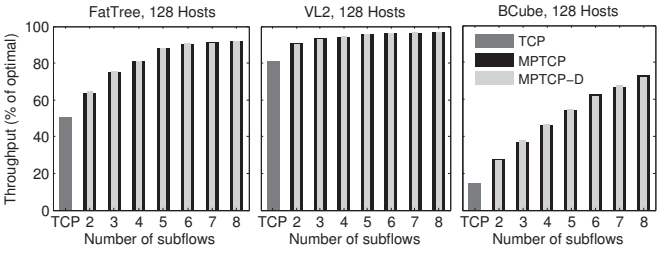


Fig. 11. Aggregated throughput in the three datacenter topologies. MPTCP-D gets as good utilization as MPTCP.

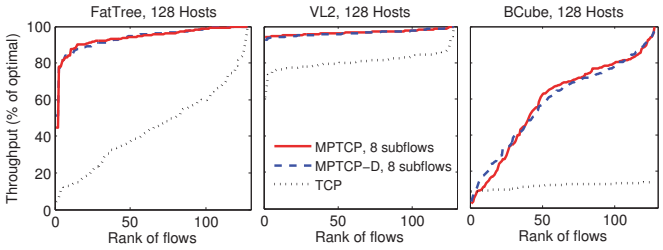


Fig. 12. Distribution of throughput in the three types of datacenter topology. MPTCP-D and MPTCP, both with 8 subflows, have high network utilization.

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 (1000 seconds) MPTCP flow to another host, which 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. 11 and Fig. 12 show that, for the long flows, the throughput of MPTCP-D is similar to that of MPTCP in different datacenter topologies.

For short flows, we examine the latency of using different protocols by both EC2 and testbed experiments. Fig. 13 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 dat-

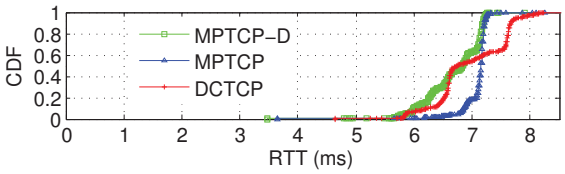


Fig. 13. RTTs measured in the experiments on EC2.

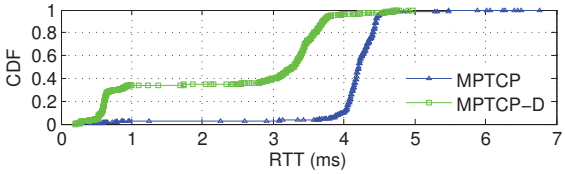


Fig. 14. RTTs measured in the testbed experiments.

acenter environments. Short flows may also be delayed by the overlapped paths in multipath transmission. Fig. 14 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.

VI. Conclusion

In this paper, we have presented a systematic study on energy efficiency of MPTCP for datacenters and proposed a variant of MPTCP to reduce power consumption for both long and short flows. 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. 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 experiment results have shown that MPTCP-D achieves good energy efficiency and has the throughput similar to MTPCP and the latency better than both DCTCP and MPTCP. MPTCP has great potential to evolve in different network scenarios and support a variety of high-quality services. The future work will be concerned with performance evaluation of MPTCP in virtualized cloud environments.

Acknowledgment

We would like to thank the anonymous reviewers for valuable and insightful comments. This research is supported in part by an NSERC Discovery Grant, a Strategic Project Grant, an E.W.R. Steacie Memorial Fellowship, and an Industrial Canada Technology Demonstration Program (TDP) grant, and partly supported by a EVCAA R&S grant from the University of Minnesota, Duluth.

References

[1] https://www.nrdc.org/resources/americas-data-centers-consuming-and-wasting-growing-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.

[3] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, "Managing Data Transfers in Computer Clusters with Orchestra," in *Proc. ACM SIGCOMM*, 2011.

[4] T. Benson, A. Akella, and D. A. Maltz, "Network Traffic Characteristics of Data Centers in the Wild," in *Proc. ACM IMC*, 2010.

[5] 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 *Proc. ACM SIGCOMM*, 2009.

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

[7] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "TIMELY: RTT-based Congestion Control for the Datacenter," in *Proc. ACM SIGCOMM*, 2015.

[8] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, Implementation and Evaluation of Congestion Control for Multipath TCP," in *Proc. Usenix NSDI*, 2011.

[9] 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.

[10] R. Khalili, N. Gast, M. Popovic, U. Upadhyay, and J. L. Boudec, "MPTCP is Not Pareto-optimal: Performance Issues and A Possible Solution," in *Proc. ACM CoNEXT*, 2012.

[11] Y. Cao, M. Xu, X. Fu, and E. Dong, "Explicit Multipath Congestion Control for Data Center Networks," in *Proc. ACM CoNEXT*, 2013.

[12] Y. Cao and M. Xu, "Dual-NAT: Dynamic Multipath Flow Scheduling for Data Center Networks," in *Proc. IEEE ICNP*, 2013.

[13] A. Agache, R. Deaconescu, and C. Raiciu, "Increasing Datacenter Network Utilisation with GRIN," in *Proc. Usenix NSDI*, 2015.

[14] M. Kheirkhah, I. Wakeman, and G. Parisis, "MMPTCP: A Multipath Transport Protocol for Data Centers," in *Proc. IEEE INFOCOM*, 2016.

[15] G. Chen, Y. Lu, Y. Meng, B. Li, K. Tan, D. Pei, P. Cheng, L. L. Luo, Y. Xiong, X. Wang, and Y. Zhao, "Fast and Cautious: Leveraging Multipath Diversity for Transport Loss Recovery in Data Centers," in *Proc. USENIX ATC*, 2016.

[16] S. Hassayoun, J. Iyengar, and D. Ros, "Dynamic Window Coupling for Multipath Congestion Control," in *Proc. IEEE ICNP*, 2011.

[17] S. Ferlin, O. Alay, T. Dreibholz, D. A. Hayes, and M. Welzl, "Revisiting Congestion Control for Multipath TCP with Shared Bottleneck Detection," in *Proc. IEEE INFOCOM*, 2016.

[18] http://multipath-tcp.org

[19] Intel 64 and IA-32 Architectures Software Developers Manual. volumes: 1, 2A, 2B, 2C, 3A, 3B and 3C, 2014.

[20] D. Abdurachmanov, P. Elmer, G. Eulisse, R. Knight, T. Niemi, J. K. Nurminen, F. Nyback, G. Pestana, Z. Ou, and K. Khan, "Techniques and Tools for Measuring Energy Efficiency of Scientific Software Applications," *Journal of Physics: Conference Series*, vol. 608, no. 1, 2015.

[21] 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.

[22] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and Don Towsley, "Multi-path TCP: A Joint Congestion Control and Routing Scheme to Exploit Path Diversity in the Internet," *IEEE/ACM Trans. Networking*, vol. 14, no. 6, 2006.

[23] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath Congestion Control for Shared Bottleneck," in *Proc. PFLDNeT workshop*, 2009.

[24] Y. Cao, M. Xu, and X. Fu, "Delay-based Congestion Control for Multipath TCP," in *Proc. IEEE ICNP*, 2012.

[25] S.M. Rumble, D. Ongaro, R. Stutsman, M. Rosenblum, and J. K. Ousterhout, "It's Time for Low Latency," *HotOS*, vol. 13, 2011.

[26] http://nrg.cs.ucl.ac.uk/mptcp/implementation.html

[27] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *Proc. ACM SIGCOMM*, 2008.

[28] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers," in *Proc. ACM SIGCOMM*, 2009.