On Energy-Efficient Congestion Control for Multipath TCP

Jia Zhao*, Jiangchuan Liu*, and Haiyang Wang†
*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

***Abstract*—Multipath TCP (MPTCP) enables transmission via multiple routes for an end-to-end connection to improve resource usage of regular TCP. Due to the increasing concern in green computing, there has been significant interest in designing energy-efficient multipath transport. For existing MPTCP congestion control algorithms, the research community still lacks a comprehensive understanding of which components in such an algorithm play the fundamental role in energy efficiency, how various algorithms compare against each other from energy-consuming perspective, or whether there exist potentially better solutions for energy saving. In this paper, we take a first step to answer these questions. Based on the MPTCP Linux kernel experiments, we first summarize that the energy consumption is related to three aspects: average throughput, path delay and different network scenarios. In order to bridge congestion control to the three aspects, we analyze the existing algorithms and capture the essential parameters of multipath congestion control model related to MPTCP's energy-efficiency. Then we design a window increase factor to shift traffic to low-delay energy-efficient paths. We further extend this design by using an energy-aware compensative parameter to fit the general hierarchical Internet topology. We evaluate the performance of existing multipath congestion control algorithms and our proposed algorithm in different network scenarios. The results successfully validate the improved energy efficiency of our design.**

I. Introduction

Multipath transport protocols enable data transmission via multiple available paths of an end-to-end connection and match the multihoming network scenarios, e.g., ubiquitous mobile devices equipped with both WiFi and cellular access, and in data centers numerous machines interconnected with multiple routes between each pair. Multipath transport protocol design attracts a lot of interest. IETF started Multipath TCP (MPTCP) and published MPTCP as an experimental standard [1] in 2013. Compared with single-path transport, MPTCP brings the following advantages: 1) it can obtain aggregate bandwidth from concurrent usage of multiple paths, thus increasing throughput [2]; 2) it can have backup paths and enhance fault tolerance; 3) it can be flexible to use good quality paths as much as possible and thus have great potential to support high-quality services.

A major concern raised by MPTCP, however, is the extra energy consumed by using more than one path. The increasing concern in green computing prompts energy-efficient protocol design for the current Internet [3, 4]. Since transport layer largely decides the performance of network communications, it remains unclear whether the general Internet over both wired and wireless networks can indeed benefit from using MPTCP. Although MPTCP has potential to achieve high performance for multihomed networks, it may consume more power[1] (as shown in our experiments) than regular single-path TCP. This problem poses challenges on the design of MPTCP.

There has been significant interest in the design of energy-efficient MPTCP. The methods proposed previously in literature can be categorized into two main classes. The first class focuses on designing efficient path selection schemes on the basis of energy cost prediction [5, 6]. These methods have to sacrifice some user-sensitive service qualities (e.g., bandwidth and file download time) to get energy reduction in return. As shown in [5], the method proposed in [6] chooses only one path as the energy-efficient strategy for all scenarios and fails to obtain more aggregate bandwidth. The state machines in such schemes also bring extra computational overhead and may affect their online performance in practice.

The second class focuses on energy-efficient multipath congestion control [7, 8]. These methods employ energy-efficient designs in congestion window evolution of MPTCP and trade off algorithm responsiveness gently for energy saving. Compared with path selection schemes, energy-efficient congestion control algorithms need quite small modification of MPTCP Linux kernel and naturally fit real-world implementation. However, for the existing multipath congestion control algorithms, the research community still lacks a clear understanding of which components in such an algorithm play the fundamental role in energy efficiency, how various algorithms compare against each other from the respect of energy consumption, or whether there exist potentially better solutions for energy saving.

In this paper, we take a first step towards answering these questions. Based on MPTCP Linux kernel experiments, we first summarize that the energy consumption of MPTCP is related to three aspects: average throughput, path delay and different network scenarios. Then we propose a congestion control model in order to bridge the congestion control parameters to the three aspects. Comparison of the existing algorithms in the experiments leads us to capture the key pa-

[1]In this paper, the term *power* denotes the electrical energy consumed per second. The unit of power is Watt. We use the term *energy* to denote the calculated electrical energy over a period. The unit of energy is Joule. Energy depends on both power and time. The energy consumption of a host-to-host data flow depends on both host CPU power and flow completion time.

rameters that can indeed impact energy efficiency. The traffic-shifting parameter impacts energy efficiency because it decides the throughput and the ability to offload traffic to low-delay paths. The compensative parameter is very useful for designing energy-cost-awareness of MPTCP in different networks, yet it may also bring tradeoff between energy consumption and throughput. The insight we gain has important implications for the design of energy-efficient MPTCP. First, we analyze the conditions under which the traffic-shifting parameter can be designed to increase throughput optimally. Our experiments show that the algorithm satisfying these conditions achieves the best energy performance in the scenarios where multiple MPTCP users share the network bandwidth resource. Second, because shifting traffic to low-delay paths can efficiently reduce energy costs for MPTCP, we analyze the traffic-shifting parameters of the existing algorithms and find that there is still margin for improvement if Round Trip Time (RTT) is used as path quality. Then we design the Delay-based Traffic Shifting (DTS) to utilize a function of RTT to control the aggressiveness of congestion window increase. Furthermore, we utilize the compensative parameter to extend our algorithm by incorporating an energy-related price in the congestion control model, and this price is efficient to save energy for the hierarchy network topologies.

The contributions of this paper are as follows:

- We analyze the energy consumption of MPTCP operations and summarize the main aspects concerned with energy consumption.
- We propose a general model to decompose existing multipath congestion control algorithms, and derive the critical parameters that decide MPTCP energy efficiency in many scenarios.
- We evaluate the performance of the existing algorithms with Linux kernel based experiments in different network scenarios, and we compare the algorithms from energy efficiency perspective.
- We use the critical parameters to design new algorithms for energy-efficient multipath congestion control, and we validate their efficiency in real-world experiments.

The remainder of this paper is organized as follows. Section II introduces related work. MPTCP energy consumption is analyzed in section III. In section IV, we propose the congestion control model and analyze the existing algorithms by adjusting the related parameters. Section V proposes the design of our algorithm. In section VI, we evaluate the performance of our algorithm via real-world experiments. Section VII concludes the paper.

II. Related Work

Multipath TCP (MPTCP) is the most significant extension to TCP in the past decade. Its architecture, congestion control and experimental standard are standardized by IETF as RFC 6182, RFC 6356 and RFC 6824, respectively. To serve different design goals in this new protocol, various multipath congestion control algorithms have been proposed in the literature [7–16]. Among these existing studies, energy-efficient MPTCP becomes increasingly important as energy efficiency continues gaining attention as a key resource for global economics.

Energy-efficient MPTCP designs in the literature can be divided into two categories. The first category includes the work that design energy-efficient path selection mechanisms. In particular, Pluntke *et al*. [6] propose the path usage scheduler that solves Markov Decision Process (MDP) offline to select a low-energy path. After that, Lim *et al*. [5] present the design of energy-aware MPTCP (eMPTCP) that utilizes WiFi and 3G/LTE energy models to estimate and select energy-efficient paths. These approaches will unavoidably degrade MPTCP's QoS (e.g., aggregate throughput or downloading completion time). As shown in [5], the schedulers proposed in [6] only choose WiFi for all scenarios and have the same performance as regular TCP over WiFi, thus losing MPTCP's advantages such as throughput increment. Additionally, the path selection mechanisms (e.g., MDP in [6]) also bring systematic complexity and large computational overhead, and they are difficult to work online and hard to be implemented in practical applications. The second category includes the work on energy-efficient multipath congestion control algorithms. Le *et al*. [7] use the inverse of loss rate as energy model and design congestion control to shift traffic to low-energy paths. Peng *et al*. [8] design a multipath congestion control model by optimizing the energy utility. The congestion control approaches trade off algorithm responsiveness, and they are comparatively easy to be implemented in the MPTCP Linux kernel. But there is still no general framework to describe these approaches in the literature. Differing from the above work, our goal is to propose a multipath congestion control design model through comparative analysis of energy efficiency of MPTCP and design the model's key components.

III. Energy Consumption of MPTCP

In this section, we analyze the energy consumption of MPTCP on our local testbed. We will first demonstrate the existence as well as the importance of MPTCP's energy consumption issue. We then summarize the main aspects that can significantly impact energy consumption of MPTCP.

A. MPTCP Consuming More Power than TCP

Although MPTCP can take advantage of multiple interfaces to improve network utilization of regular TCP, it may also lead to energy-related issue in both wireline and wireless networks. We conduct experiments and measure the performance of MPTCP in different scenarios.

We first measure power consumption of machine-to-machine data transfers over our testbed. There are two types of machines in the testbed. The first type has double NICs (each with capacity 100Mbps) and Quad-core Intel Core i7-3770 CPU. The other type is also configured with double NICs (each with capacity 1000Mbps) and the CPU type of Octa-core Intel Xeon E5-2680 v2. We install the MPTCP Linux kernel of version 0.90 [17] in the machines. We change the number of subflows traversing a NIC by modifying Linux kernel's MPTCP path-manager module

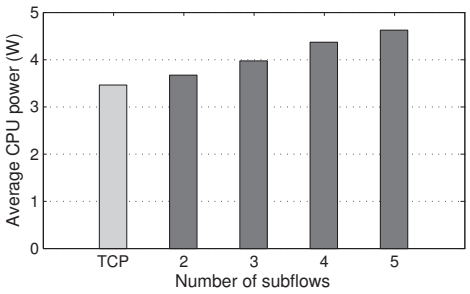


Fig. 1. Power consumed by TCP and MPTCP when using machine CPU Core i7-3770, 3.4GHz, 4 cores.

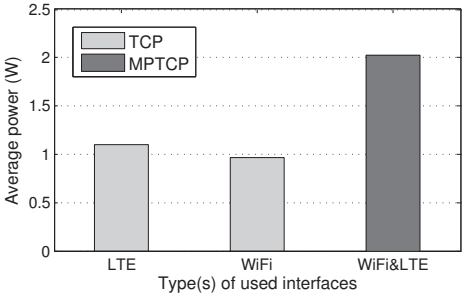


Fig. 2. Nexus 5 power consumption in data transfers.

in '/sys/module/mptcp_fullmesh/parameters/num_subflows'. During data transfers, we record instant host CPU power from Intel's Running Average Power Limit (RAPL) driver [18, 19]. We also capture the power consumption of classic TCP as our baseline for comparison. In this experiment, the classic TCP connection uses one NIC, and the MPTCP connection uses double NICs simultaneously.

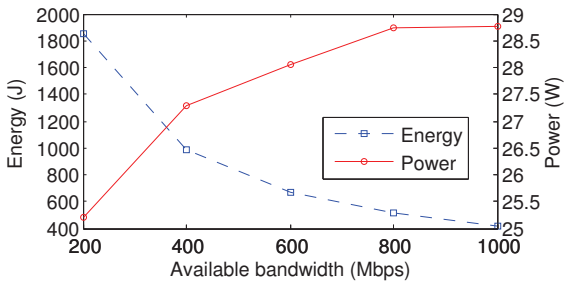
Fig. 1 shows how CPU power changes with the number of subflows for Quad-core Intel Core i7-3770 CPU. It is easy to see that: 1) MPTCP consumes more CPU power than classic TCP; 2) the power consumption of MPTCP increases with the number of subflows.

To better understand this feature in wireless environments, we also investigate MPTCP power consumption on mobile devices. We use Nexus 5 smart phone and install the MPTCP kernel image [20] on it. Both WiFi and LTE (Long-Term Evolution) NICs are enabled for MPTCP. As we can see in Fig. 2, MPTCP largely increases smart phone's power consumption for data transfers.

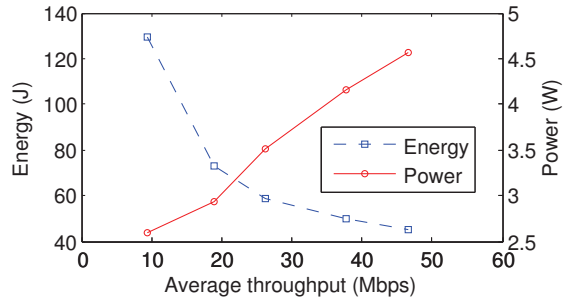
B. Energy-related Factors

We continue to analyze the energy consumption of MPTCP and study the aspects that can significantly impact energy efficiency of MPTCP.

We first study how instant power and total energy change with throughput of MPTCP. We conduct the experiments for the general Internet. The available bandwidth of an MPTCP connection between two machines increases from 200Mbps to 1000Mbps. The connection is used to transmit 10GB data, and during the data transmission we read instant CPU power from RAPL and calculate the total energy consumption.



(a) Data transfer using Ethernet



(b) Data transfer using WiFi

Fig. 3. Energy, power vs throughput of MPTCP: (a) transmitting 10GB data with wired Ethernet; (b) downloading 500MB data with WiFi.

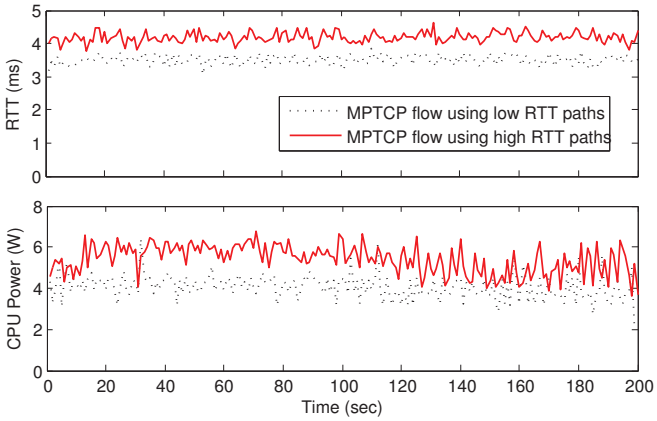


Fig. 4. Power consumption of MPTCP under different path delays: MPTCP flow using high RTT paths consumes more CPU power than the one using low RTT paths.

As shown in Fig. 3, The power consumption of MPTCP is approximately a linear function of throughput by using the WiFi network (single-path case is discussed in [21]), but the power consumption is non-linear related to the throughput by using Ethernet. We can use the following equation to describe the relation between power P^{MPTCP} and throughput τ:

$$P^{MPTCP} \propto f(\tau) \tag{1}$$

where $f(\cdot)$ is an increasing function. $f(\cdot)$ can be regarded as a linear function in wireless networks, but in wired networks it should be a non-linear function.

Fig. 3(a) shows that the total energy decreases with throughput, but the power increases with the throughput. The result

also indicates that the power consumption of MPTCP has gentle increase, which is only about 15% power increase across the bandwidth ranging from 200Mbps to 1000Mbps. We also measure energy and power change with throughput in WiFi networks. Fig. 3(b) shows that the power consumption of MPTCP increases sharply with throughput, up to 90% across the throughput ranging from 10Mbps to 50Mbps.

We also study how power consumption changes with different delays of the used paths. To do this, we need to keep the throughput and change the delay in different measurements. MPTCP Linux kernel of version 0.90 provides a easy way for this operation. In MPTCP Linux kernel of version 0.90, 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 can change the number of subflows by modifying Linux kernel's MPTCP path-manager module in '/sys/module/mptcp_fullmesh/parameters/num_subflows'. The aggregate throughput of multiple subflows on a path is almost the same with the throughput of only one subflow on the path, but it can be observed that path delay increases with the number of subflows of the path. Hence, we can change the path delay by changing the number of subflows of each path. We use two machines for data transfer. Each machine has Quad-core Intel Core i7-3770 CPU and two NICs. When the parameter 'num_subflows' is changed from 1 to 2, the path delay is also changed from low to high. We measure RTT and instance CPU power during the data transfer. Fig. 4 shows that the MPTCP flow using low delay paths consumes less CPU power than that using high delay paths. This indicates that using low delay path can save energy for MPTCP.

Consider an end-to-end MPTCP connection using n paths for a data transfer. Let E_{total} denote the total energy consumption of MPTCP, M denote the data amount and $\overline{\tau}$ denote the average throughput of MPTCP. The data transfer duration is $\frac{M}{\overline{\tau}}$. For path r, let τ_r be the throughput and RTT_r be the round trip time. Based on our results in Fig. 3 and Fig. 4, we describe the total energy consumption of MPTCP as the following equation.

$$E_{total} = \frac{M}{\overline{\tau}} \sum_{r=1}^{n} P_r(\tau_r, RTT_r) \tag{2}$$

where function $P_r(\tau_r, RTT_r)$ is the power for path r and it increases with both τ_r and RTT_r.

From Equation (2), we can summarize the following aspects that can significantly impact the energy consumption of MPTCP.

1) **Throughput** $\overline{\tau}_r$: For each subflow, energy consumption is inversely proportional to average throughput. In MPTCP Linux kernel, window-based congestion control is used to decide the send rate of each subflow. Send rate can indicate current path qualities such as RTT or loss rate, and these qualities are usually used to estimate average throughput on the path.
2) **Ability to shift traffic to low-delay paths**: given n available paths with dynamic changed path delay, MPTCP is supposed to detect good paths with low delay in time, and shift traffic to these paths to save energy. The increase term of congestion control algorithm in MPTCP Linux kernel impacts the ability to shift traffic.
3) **Adaptation to topology and link state**: MPTCP has to adapt to different topologies (e.g. hierarchy) and traffic characteristics (e.g. bursts) on the Internet. MPTCP may also have to deal with different link conditions. The high error rate of wireless link would result in severe packet loss. This will increase MPTCP retransmission operations and significantly increase the energy consumption. In addition, as shown in our experiments the power $P^{(r)}$ increases slowly with throughput in wired networks but increases sharply with throughput in wireless networks.

In section IV and V, we will analyze how to design multipath congestion control to improve energy efficiency of MPTCP from the above three aspects.

IV. Congestion Control Model and Parameter Analysis

Based on the observation in section III, we have proposed Equation (2) to capture the intuitive yet fundamental relationship between energy consumption and properties of MPTCP. However, how to pinpoint the above three features into MPTCP's design parameters (in congestion control) remains challenging.

In this section, we first propose a multipath congestion control model whose parameters decide the window evolution. We can further analyze the existing multipath congestion control algorithms by using these parameters.

Congestion control algorithm plays the fundamental role in MPTCP. We first need a theoretical model to analyze the energy-related design of existing multipath congestion control algorithms. We describe the network model which is similar to [11]. Consider a network that consists of link set L. A link $l \in L$ has finite capacity c_l. There is a set S of MPTCP users in the network. For any user $s \in S$, let s represent the set of available paths between s and its destination. A path $r \in s$ consists of multiple links. If path r uses a link l, then we denote $l \in r$. For each path r, let $x_r(t) = w_r(t)/RTT_r$ be the send rate at time t, RTT_r and $w_r(t)$ denote the round trip time and congestion window respectively. For each user $s \in S$, let $\mathbf{x}_s(t) = (x_r(t), r \in s)$. We propose the following differential equation:

$$\lim_{\Delta t \to \delta} \frac{\Delta x_r}{\Delta t} = \frac{\psi_r(\mathbf{x}_s) x_r^2}{RTT_r^2 (\sum_{k \in s} x_k)^2} - \beta_r(\mathbf{x}_s) \lambda_r x_r^2 - \phi_r(\mathbf{x}_s) \tag{3}$$

where the parameters are described as follows:

- δ: the time step size for the differential equation, which equals to 1 for wVegas algorithm [12] and equals to infinitesimal for the algorithms in [7–16];
- $\psi_r(\mathbf{x}_s)$: the traffic-shifting parameter in the increase term of window evolution, which works as the core of a multipath congestion control algorithm and decides the

important properties (e.g. traffic-shifting ability, TCP-friendliness, Pareto-optimality and responsiveness);
- $\beta_r(\mathbf{x}_s)$: the parameter that decides the decrease term of window-based multipath congestion control;
- λ_r: the signal to trigger window decrease, e.g., a delay condition used for DWC, a delay-based path price used for wVegas, and loss rate used for other algorithms;
- $\phi_r(\mathbf{x}_s)$: the compensative parameter that compensates window increase aggressiveness and impacts responsiveness of a multipath congestion control algorithm.

This model can be used to not only design new multipath congesiton control algorithm, but also generalize the existing algorithms with specified parameters. By using this model, we can decompose each of the algorithms into several parameters. For instance, EWTCP [14] has $\psi_r(\mathbf{x}_s) = \frac{(\sum_k x_k)^2}{x_r^2\sqrt{|s|}}$; Coupled [15][16] has $\psi_r(\mathbf{x}_s) = \frac{RTT_r^2(\sum_k x_k)^2}{(\sum_{k\in s} w_k)^2}$; LIA [9] has $\psi_r(\mathbf{x}_s) = \left(\max_{k\in s}\frac{w_k}{RTT_k^2}\right)\frac{RTT_r^2}{w_r}$; OLIA [11] has $\psi_r(\mathbf{x}_s) = 1$; Balia [8, 13] has $\psi_r(\mathbf{x}_s) = \frac{2}{5} + \frac{1}{2}\frac{\max_k x_k}{x_r}\frac{w_k}{} + \frac{1}{10}(\frac{\max_k x_k}{x_r})^2$; ecMTCP [7] has $\psi_r(\mathbf{x}_s) = \frac{(\max_k \frac{w_k}{RTT_k^2})RTT_r^2 p_r}{w_r p_h}$; DWC [10] has $\psi_r(\mathbf{x}_s) = \frac{RTT_r^3(\sum_k x_k)^2}{|s|\min_k RTT_k w_r \sum_k w_k}$; wVegas [12] has $\psi_r(\mathbf{x}_s) = \frac{RTT_r^2 \min_{k\in s} q_k}{q_r x_r/(\sum_{k\in s} x_k)^2}$ (where $q_r = RTT_r - baseRTT_r$). These specified parameters decide the properties of the algorithms. In order to capture the key parameters that can be used for energy-efficient design, we will next analyze the parameters, evaluate their impact on energy efficiency of the existing algorithms, and then propose our algorithm for improvement.

V. The Design of Energy-Efficient Congestion Control

In this section, we present the design of the parameters $\psi_r(\mathbf{x}_s)$ and $\phi_r(\mathbf{x}_s)$ of Equation (3) in order to achieve energy-efficient congestion control.

A. $\psi_r(\mathbf{x}_s)$ Design for Throughput Increment

Given the amount of data in a transfer, higher throughput brings shorter download time. Short download time can contribute to low energy consumption. From Equation (2), we see that increasing the throughput of multipath TCP is an intuitive way to achieve energy-efficiency. Yet the throughput increase cannot be unbounded. Most data in the current Internet is transmitted by TCP at transport layer. As a transport layer protocol, multipath TCP is supposed to be TCP-friendly to ensure fair bandwidth share with regular TCP in their coexisting network environments. Therefore, increasing the throughput of multipath TCP should be under the constraint of TCP-friendliness, and it is also a reasonable design for energy-efficiency.

Throughput increase depends on the window increase term of multipath TCP congestion control algorithm. The window increase term should be bounded by the TCP-friendly condition. According to the design goal of TCP-friendliness in [9], MPTCP should not take up more capacity than if it was regular TCP using the best path. We need to derive the TCP-friendly condition by using the parameter $\psi_r(\mathbf{x}_s)$ in our proposed congestion control model. We have the following condition

Condition 1. *For the loss-based multipath congestion control algorithms derived from Equation (3), the parameters $\psi_r(\mathbf{x}_s)$, $\beta_r(\mathbf{x}_s)$ and $\phi_r(\mathbf{x}_s)$ for all route $r \in s$ satisfy $\psi_h(\mathbf{x}_s^*) \le 1$, $\beta_h(\mathbf{x}_s^*) = \frac{1}{2}$ and $\phi_h(\mathbf{x}_s^*)$=0, where $h = \operatorname{argmax}_{k\in s} x_k^*$ and $\mathbf{x}_s^*$ is the throughput vector at equilibrium.*

If an MPTCP algorithm satisfies Condition 1, it will satisfy the the design goal of TCP-friendliness in [9]. This can be illustrated with Equation (3). At equilibrium, we have $\frac{dx_r}{dt} = 0$, and on the best path h, we have $\frac{\psi_h(\mathbf{x}_s^*)x_h^2}{RTT_h^2(\sum_{k\in s} x_k^*)^2} - \beta_h(\mathbf{x}_s^*)\lambda_h x_h^2$. The MPTCP algorithm has the aggregate throughput $\sqrt{\frac{2\psi_h(\mathbf{x}_s^*)}{\lambda_h}}/RTT_h$ which is less than the throughput $\sqrt{\frac{2}{\lambda_h}}/RTT_h$ obtained on the best path if it is regular TCP.

The parameter $\psi_r(\mathbf{x}_s)$ also decides the responsiveness of an MPTCP algorithm. High responsiveness brings aggressive window increase, fast convergence and high throughput.

There is, however, a tradeoff between TCP-friendliness and responsiveness [9], [13]. So it is still a problem to increase throughput under the constraint of TCP-friendliness. Pareto-optimality [11] can be used to deal with the problem and improve the throughput of TCP-friendly MPTCP algorithms.

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.

Suppose there are $|S|$ multipath TCP users in the network. Each user $s \in S$ has a utility $U_s(\mathbf{x}_s)$ that 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 $l \in L$. Let $\mathbf{x}_s^*$ be the maximum of the utility as Equation (4). We use the following condition to design Pareto-optimal throughput increment by using the parameter $\psi_r(\mathbf{x}_s)$.

Condition 2. *For all routes $r \in s$, there exists a concave utility function $U_s(\mathbf{x}_s)$ that satisfies $\theta_r(\mathbf{x}_s^*)\left.\frac{\partial U_s(\mathbf{x}_s)}{\partial x_r}\right|_{\mathbf{x}_s=\mathbf{x}_s^*} = \frac{\psi_r(\mathbf{x}_s^*)x_r^{*2}}{RTT_r^2(\sum_{k\in s} x_k^*)^2}$, where $\theta_r(\mathbf{x}_s)$ is a positive function related to step size and $\mathbf{x}_s^*$ is the maximizer of Equation (4).*

As we have discussed in section IV, wVegas has step size $\delta = 1$ and $\theta_r(\mathbf{x}_s) = x_r/\lambda_r$, and the others all have step size $\delta = 0$ and $\theta_r(\mathbf{x}_s) = x_r^2$.

If an MPTCP algorithm satisfies Condition 2, it is Pareto-optimal. This can be illustrated with Equation (4). Utility $U_s(\mathbf{x}_s)$ is an increasing function of $\mathbf{x}_s$, and path price 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 $\mathbf{x}_s^*$, it is impossible to increase $\mathbf{x}_s^*$ for user s without decreasing throughput of other users or increasing congestion. Therefore, the maximizer is at Pareto-optimality and 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 the best aggressiveness a MPTCP algorithm can get based on its utility as Equation (4) under the constraint of TCP-friendliness.

B. $\psi_r(\mathbf{x}_s)$ Design for Delay-based Traffic Shifting

MPTCP algorithms are supposed to detect the bad quality paths and offload data traffic from these paths to better ones. The ability to shift traffic has important influence on MPTCP energy efficiency. We have shown in our experiments that MPTCP flow using low-delay paths consumes less CPU power than the flow using using high delay paths. In addition, using good quality paths could avoid frequent window decrease due to severe packet loss and thus reduce energy consumed by retransmission or recovery operations.

It is also important to choose proper variables to estimate path quality. In Equation (3), the parameter $\psi_r(\mathbf{x}_s)$ decides the ability of shifting traffic. Now we design the parameter $\psi_r(\mathbf{x}_s)$ and study how it changes with path delay.

Based on the results in section III, We see that the power consumption of MPTCP increases with the path delays. In section IV, we see that the parameter $\psi_r(\mathbf{x}_s)$ of Equation (3) can impact MPTCP's traffic shifting to good-quality paths, and we also show that path delay has never been used in traffic-shifting design of loss-based congestion control. Accordingly, our goal is to design the traffic-shifting function that can be improved by introducing an efficient delay-based factor. We propose the Delay-based Traffic Shifting (DTS) with the following factor:

$$\varepsilon_r = \frac{2}{1+e^{-10(\frac{baseRTT_r}{RTT_r}-\frac{1}{2})}} \tag{5}$$

where $baseRTT_r$ is the minimum RTT experienced on path r. This factor is an increasing function of $\frac{baseRTT_r}{RTT_r}$. The factor is designed to avoid moving traffic to high-delay path, to increase window properly when the path continue recovering ($\frac{baseRTT_r}{RTT_r}$ becomes smaller and smaller), and to maintain a stable throughput on low-delay paths.

According to the above analysis, the traffic-shifting parameter $\psi_r(\mathbf{x}_s)$ can be de designed to optimally increase throughout and use low-delay paths. Both throughput increment and low-delay path utilization lead to the energy efficient design of our congestion control algorithm. Our algorithm achieves throughput increment by designing the the parameter $\psi_r(\mathbf{x}_s)$ to satisfy the condition of Pareto-optimality, and it also employs the DTS to offload traffic to low-delay paths. To ensure Pareto-optimality, let $\psi_r(\mathbf{x}_s) = c\cdot\varepsilon_r$, where c is a constant. The ratio $\frac{baseRTT_r}{RTT_r}$ can be regarded as a random variable and its expectation is $\frac{1}{2}$. If $c=1$, the parameter $\psi_r(\mathbf{x}_s)$ also satisfies the fairness condition. Algorithm 1 summarizes our proposed traffic-shifting methods.

Algorithm 1: Pseudo-code of DTS

Input: TCP socket of each subflow r
Output: Congestion window $w[r]$
Initialization:
for *each subflow* r **do**
  $current\_rtt[r] \leftarrow 0$; $base\_rtt[r] \leftarrow 0$;
  $\varepsilon\_denominator[r] \leftarrow 1$; $\varepsilon\_numerator[r] \leftarrow 1$;
  $\varepsilon[r] \leftarrow 1$; $\psi[r] \leftarrow 1$;
Update of baseRTT:
for *each subflow* r **do**
  /* calculate RTT_r */
  $current\_rtt[r] =$
  $calculate\_current\_rtt(subflow\_tcp\_socket)$;
  if $current\_rtt[r] < base\_rtt[r]$ **then**
    $base\_rtt[r] \leftarrow current\_rtt[r]$;
Update of ε:
for *each subflow* r **do**
  /* calculate $baseRTT_r/RTT_r$ */
  $rtt\_div = divide(base\_rtt[r], current\_rtt[r])$;
  $rtt\_div = 10 * rtt\_div - 5$;
  /* $\exp(10 * baseRTT_r/RTT_r - 5)$ Taylor expansion*/
  $\varepsilon\_numerator[r] =$
  $100*(1+rtt\_div+50*(rtt\_div)^2+17*(rtt\_div)^3)$;
  $\varepsilon\_denominator[r] = 100 + \varepsilon\_numerator[r]$; $\varepsilon[r] =$
  $divide(2*\varepsilon\_numerator[r], \varepsilon\_denominator[r])$;
Evolution of congestion window w:
for *each ACK on path* r **do**
  $w[r] \leftarrow w[r] + \frac{\psi[r]\varepsilon[r]w[r]/current\_rtt[r]^2}{(\sum_{k\in s} w[k]/rtt[k])^2}$; //derived by Eq. (2)
for *each loss on path* r **do**
  $w[r] \leftarrow \frac{1}{2}w[r]$;
return $w[r]$

C. $\phi_r(\mathbf{x}_s)$ Design for Hierarchical Topology

The Internet is basically a hierarchy that has many end devices connected to the local aggregated switching node and then connects the local networks to the backbone. In such hierarchical network topologies, employing MPTCP may aggravate the traffic concentration on both aggregated and core nodes and result in severe network congestion. This will also delay data transmission and increase energy costs in the networks. When using MPTCP we need to deal with the balance between the number of subflows and the energy overhead. Therefore, we introduce an enegy-related price to the congestion control.

Energy proportional management has been studied in literature [22], [23]. With adaptive power management, the networks make energy cost in proportion to the workload, and thus they can choose optimal data rate to achieve energy efficiency. Let $L^{'}$ be set of links between switches. For any

$l^{'} \in L^{'}$, let $Q_{l'}$ be the queue size on link $l^{'}$, Q be the expected queue size, and ρ be the bottleneck energy cost for unit traffic. We refer to the general model in [23] and use the following utility

$$U_{ep} = \sum_{l' \in L'} (Q_{l'} - Q)^+ + \rho \sum_{l' \in L'} y_{l'} \tag{6}$$

where $y_{l'}$ is the traffic on link $l^{'}$. The first term is service quality provided by the data center, and the second term is the energy cost. Let $[R^l_{sr}]$ be the routing matrix where $R^l_{sr} = 1$ if link l is on the path 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. We add the data center cost utility to Equation (4) and obtain the following problem

$$\max \sum_{s \in S} U_s(\mathbf{x}_s) - \kappa_s U_{ep} \tag{7}$$

$$\text{s.t.} \quad y_l \leq c_l \quad \forall l \in L \tag{8}$$

where κ_s is the weight of the price. From Equation (6) we obtain the parameter $\phi_r(\mathbf{x}_s)$ of Equation (3) as $\phi_r(\mathbf{x}_s) = \kappa_s x_r^2 \frac{\partial U_{ep}}{\partial x_r}$.

We can further extend our algorithm by using the parameter $\phi_r(\mathbf{x}_s)$, and we get the following fluid model

$$\frac{dx_r}{dt} = \frac{c\varepsilon_r x_r^2}{RTT_r^2 (\sum_{k \in s} x_k)^2} - \frac{1}{2} p_r x_r^2 - \kappa_s x_r^2 \frac{\partial U_{ep}}{\partial x_r}. \tag{9}$$

VI. Performance Evaluation

In this section, we evaluate the energy efficiency of existing algorithms and our proposed algorithm via experiments in different scenarios.

A. Impact of Throughput

We measure the energy consumption of the four algorithms (LIA [9], OLIA [11], Balia [13], ecMTCP [7]), among which only OLIA is Pareto-optimal. Consider the network scenario as Fig. 5(a) with N MPTCP users and $2N$ TCP users sharing the two bottlenecks. To examine the performance of large number of MPTCP users, we use parallel MPTCP connections between a server machine and a client machine, both configured with MPTCP Linux kernel. On the client machine we generate N parallel MPTCP senders, each of which sends 16MB data to the server machine. In the experiments, we install the MPTCP Linux kernel of version 0.90 [17] on Ubuntu 14.04 and read energy consumption values from Intel's RAPL driver in each host's CPU. The algorithms in section IV are implemented in the kernel.

Fig. 6 shows energy consumption results under different values of N by using box-whisker plot (indicating minimum, 25th percentile Q1, median, 75th percentile Q3, and maximum, as well as the outliers out of the range between Q1-1.5*(Q3-Q1) and Q3+1.5*(Q3-Q1)). OLIA consumes less average energy than the others, especially when the value of N becomes large. This is because OLIA's Pareto-optimality can efficiently increase throughput in resource pooling without losing TCP-friendliness and decrease the time of data transmission. This

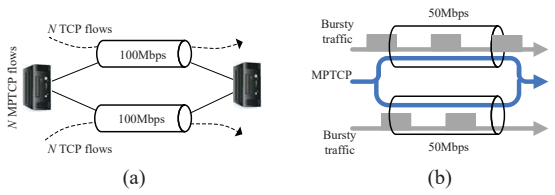


Fig. 5. Experiment scenarios: (a) increasing throughput; (b) shifting traffic.

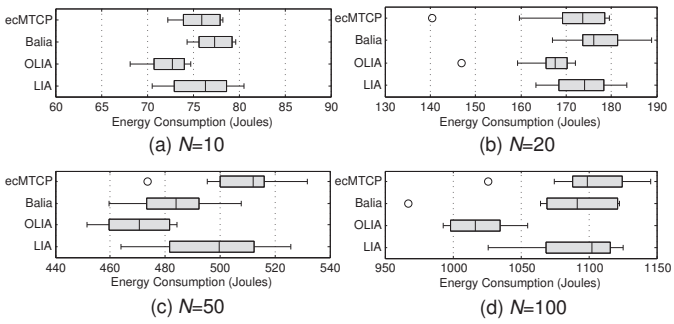


Fig. 6. Comparison of the four TCP-friendly algorithms (LIA, OLIA, Balia, ecMTCP) separately used for the scenario in Fig. 5(a) with (a) $N = 10$, (b) $N = 20$, (c) $N = 50$ and (d) $N = 100$ multipath TCP users (each transmitting 16MB data).

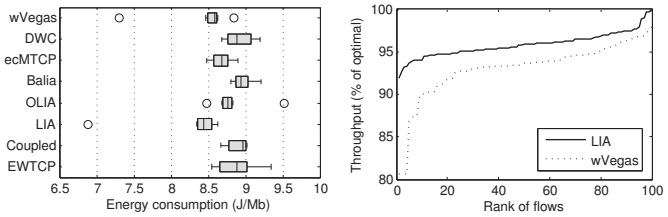


Fig. 7. Comparison of performance in the scenario of Fig. 5(b).

experiment validates that the traffic-shifting parameter that satisfies the conditions in section V can achieve energy efficiency as well as maintaining high throughput and fairness.

B. Energy-Efficiency of DTS

We need first evaluate how these shifting methods work in practice when MPTCP is confronted with dynamic quality change of multiple paths. We use the scenario in Fig. 5(b) to compare the algorithms. This scenario has four path quality states (Bad-Bad, Bad-Good, Good-Good and Good-Bad) that occur at random. This is because the scenario generates on each path a bursty traffic that follows Pareto pattern at rate 45Mbps and occurs at random intervals (average 10 seconds) and with average bursty duration of 5 seconds. Although the scenario is an extremely harsh test for the algorithms, it could happen in real world such as data centers with much bursty traffic [24]. Fig. 7 shows that LIA outperforms the other existing algorithms in traffic shifting.

We implement DTS in the MPTCP Linux kernel of version 0.90 [17]. This kernel has implemented four existing multipath congestion control algorithms. We can design and implement our congestion control algorithm by compiling the source

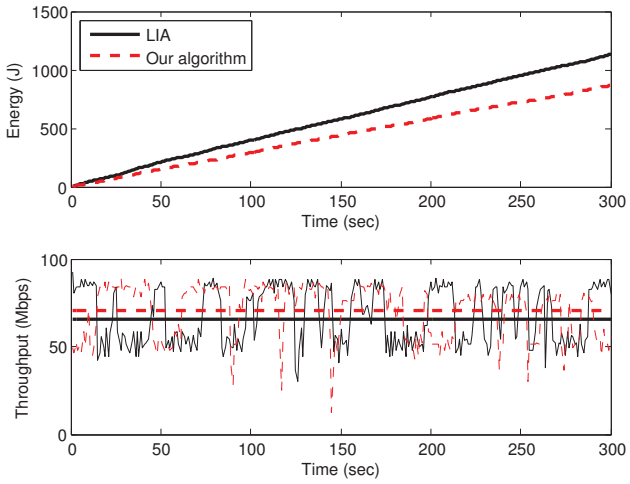


Fig. 8. Comparison of LIA and Modified LIA.

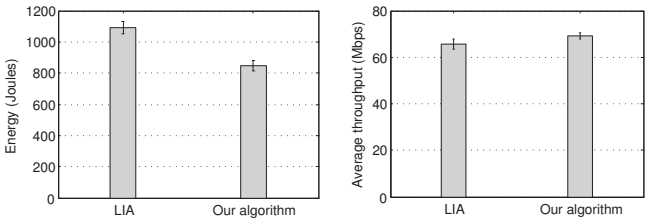


Fig. 9. Performance of DTS in testbed experiments.

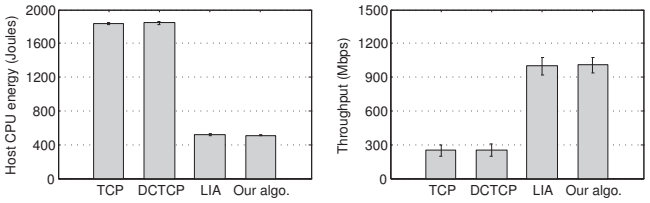
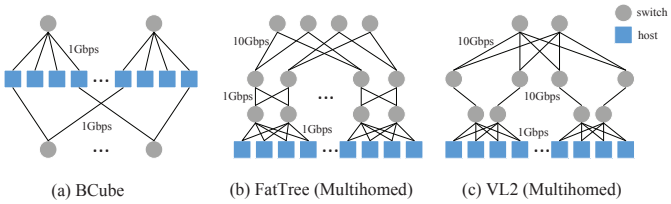


Fig. 10. Performance of the four algorithms on EC2.



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

Fig. 11. The three datacenter topologies with multihomed hosts.

code file as module, and we can enable it by modifying the value in '/proc/sys/net/ipv4/tcp_congestion_control'. Each MPTCP subflow is associated with a subsocket and a congestion window. For each subflow, TCP socket information can be obtained from its subsocket. To obtain $baseRTT$ for each subflow, we define the `struct baseRTT` and use a function to update the value in congestion control. Then the congestion control function can calculate the parameter $\psi_r(\mathbf{x}_s)$ and behave the window evolution.

We use the same scenario as Fig. 5(b) over our testbed to evaluate the modified algorithm. Fig. 8 shows the trace of LIA and our algorithm. The results show that our algorithm can save energy without degrading its throughput. Fig. 9 shows that our algorithm can reduce energy consumption by up to 20% compared to LIA. We see that the parameter ε_r can improve energy consumption without sacrificing responsiveness in adaptation to dynamic change of path quality.

C. Performance in Advanced Networks and Discussion

1) Datacenter Network Topologies: We also evaluate the performance of our algorithm in the experiments on large-scale datacenters. We rent virtual machine instances on EC2. We build up a virtual private cloud and four private subnets. We create 40 instances as hosts. Each host has four Elastic Network Interfaces (ENIs), and each ENI is set the capacity of 256Mbps. Each ENI has a private IP address and uses it to connect to a subnet. Hence, there are four routes between each pair of hosts. We implement our algorithm in the MPTCP Linux kernel of version 0.90 and copy the kernel image on the instances. The related configuration includes: c4.xlarge instance, Quad-cores Intel Xeon E5-2666 v3 CPU, 40GB storage, 7.5GB Memory, availability zone of us-west-2c, Ubuntu Server 14.04 LTS, Linux 3.18.34 and MPTCP v0.90 kernel. We use iperf to generate traffic between each pair of hosts and record their throughputs. We read host CPU's instant power consumption from Intel's RAPL driver. The experiments take 5 hours in total. We evaluate the performance of the four algorithms: TCP, DCTCP [25], LIA, and our algorithm. Each host has one connection. Each connection transmits 10GB data. As shown in Fig. 10, our algorithm saves up to 70% of aggregated energy of the single path congestion control algorithms TCP and DCTCP. Fig. 10 also shows that our algorithm has similar performance as LIA in general datacenter networks.

From Equation (2), we see that increasing link utilization is a reasonable way to reduce energy costs in wireline networks. It has been shown in [2], [11] that increasing the number of MPTCP subflows can improve throughput and link utilization of the data center networks. Thus in our evaluation we compare the eight algorithms and analyze their energy overhead change across the increasing number of MPTCP subflows.

For performance evaluation in data centers, we use *htsim* [26], a C++ based simulator that is fit for large scale network traffic simulation and provided by Raiciu *et al.* to evaluate performance of MPTCP in data centers of various topologies [2]. Three topologies, FatTree [27], VL2 [28] and BCube [29], have been proposed to address traffic concentration on a small number of bottleneck links in data center networks. As shown in Fig. 11, FatTree and VL2 follow a hierarchical topology to organize switches in access, aggregate, and core layers, and VL2 uses faster links between switches than FatTree. BCube employs the generalized hypercube topology rather than hierarchical organization of switches, and it makes use

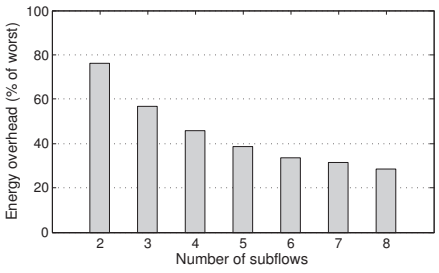


Fig. 12. Energy overhead of LIA in BCube

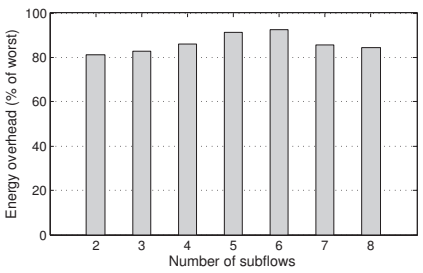


Fig. 13. Energy overhead of LIA in FatTree

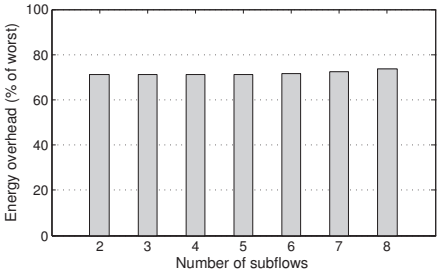


Fig. 14. Energy overhead of LIA in VL2

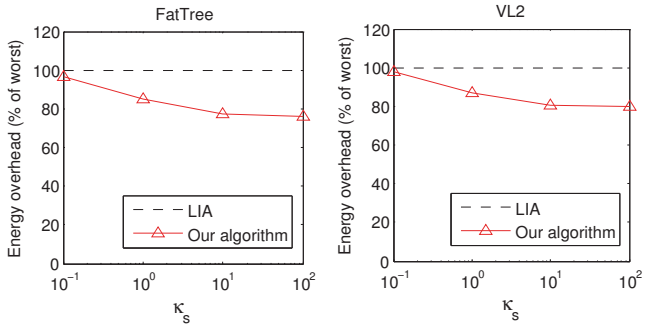


Fig. 15. Performance improvement by using $\phi_r(\mathbf{x}_s)$ of Equation (3).

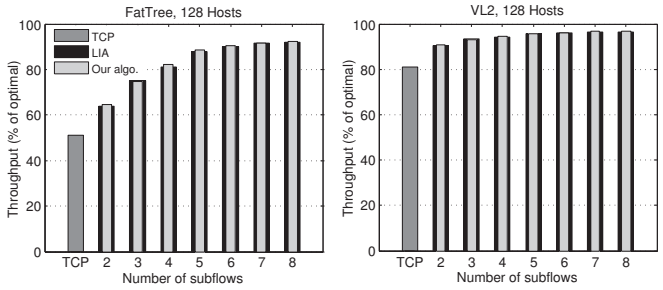


Fig. 16. Aggregated throughput in FatTree and VL2 topologies. Our algorithm gets as good utilization as LIA

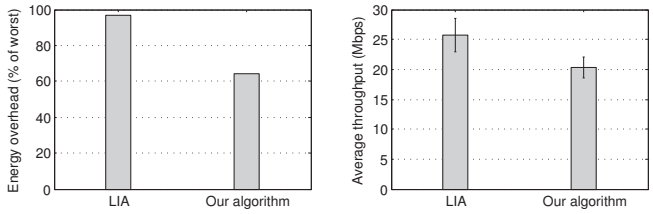


Fig. 17. Performance of DTS in the heterogeneous wireless network scenario.

of some hosts to relay traffic. We build 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 (1000 sec) MPTCP flow to another host, which is chosen at random. For each algorithm in each case (the number of subflows) under each type of topology, we simulate ten times and record its average energy overhead. Fig. 12 shows that Increasing the number of subflows can greatly reduce energy overhead in BCube, while Fig. 13 and Fig. 14 show that increasing the number of subflows fails to save energy for FatTree and VL2.

We then evaluate performance of the extended DTS in simulations for both FatTree and VL2 topologies with 8 subflows of each MPTCP connection. Fig. 15 shows that our algorithm can save up to 20% energy costs. Fig. 16 shows that our algorithm has similar performance to LIA in the FatTree and VL2 topologies.

2) Heterogeneous Wireless Networks: Multipath transmission has great potential to enhance communications between mobile devices or vehicles in heterogeneous wireless networks. We also investigate how DTS performs in heterogeneous wireless networks. We do the simulations by using the network simulator (ns-2.35). We use the MPTCP patch for ns-2.35 [30]. This patch includes the MPTCP Agent that uses the algorithm LIA in the phase of multipah congestion avoidance, and it retains the other three phases (Slow Start, Fast Retransmit and Fast Recovery) of regular TCP. We can modify the MPTCP Agent by replacing the congestion avoidance phase with different congestion control algorithms. We use the simple heterogeneous wireless network topology where a sender uses both WiFi and 4G interfaces to transmit data to a receiver. The wireless link type is set DropTail and its queue limit is 50 packets. The receiver buffer size is the default 64KB. For the path representing WiFi link, we set 10Mbps bandwidth and 40ms transmission delay. For the path representing 4G link, we set 20Mbps bandwidth and 100ms transmission delay. We generate cross traffic on both links to simulate a dynamic wireless network environment. The MP-TCP Agent is attached with infinite FTP flow. The simulation time is 200 seconds. Fig. 17 shows the energy and throughput performance. Our results show that DTS outperforms LIA in energy consumption and it can save up to 30% energy compared to LIA. Our results validate the efficiency of using the compensative parameter $\phi_r(\mathbf{x}_s)$. Fig. 17 also shows the throughput performance of DTS and LIA, and it indicates that for DTS there exists a tradeoff between energy consumption and throughput.

VII. Conclusion and Future Work

This paper has analyzed the existing multipath congestion control algorithms from energy efficiency perspective. A

general analytical model has been proposed to explore the decisive parameters for MPTCP energy efficiency. Based on the analysis results, new algorithms has been proposed and their efficiency has been validated in the experiments. 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 energy performance evaluation of MPTCP in virtualized cloud environments and energy-efficient designs for multimedia applications over MPTCP.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for valuable and insightful comments. This work was supported in part by the Canada Technology Demonstration Program, in part by a Canada NSERC Discovery Grant, in part by the NSERC E.W.R. Steacie Memorial Fellowship, and in part by a EVCAA R&S grant from the University of Minnesota, Duluth.

REFERENCES

[1] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses," IETF RFC 6824, 2013.

[2] 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.

[3] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing Network Energy Consumption via Sleeping and Rate-Adaptation," *Usenix NSDI*, vol. 8, pp. 323-336, 2008.

[4] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J. A. Maestro, "IEEE 802.3az: The Road to Energy Efficient Ethernet," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 50-56, 2010.

[5] Y. S. Lim, Y. C. Chen, E. M. Nahum, D. Towsley, R. J. Gibbens, and E. Cecchet, "Design, Implementation, and Evaluation of Energy-Aware Multi-Path TCP," in *Proc. ACM CoNEXT*, 2015.

[6] C. Pluntke, L. Eggert, and N. Kiukkonen, "Saving Mobile Device Energy with Multipath TCP," in *Proc. ACM MobiArch*, 2011.

[7] T. A. Le, C. S. Hong, M. A. Razzaque, S. Lee, and H. Jung, "ecMTCP: An Energy-Aware Congestion Control Algorithm for Multipath TCP," *IEEE Communications Letters*, vol. 16, no. 2, pp. 275-277, 2012.

[8] Q. Peng, M. Chen, A. Walid, and S. H. Low, "Energy Efficient Multipath TCP for Mobile Devices," in *Proc. ACM MobiHoc*, 2014.

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

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

[11] 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.

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

[13] Q. Peng, A. Walid, and S. H. Low, "Multipath TCP Algorithms: Theory and Design," in *Proc. ACM SIGMETRICS*, 2013.

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

[15] 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.

[16] 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. Networking*, vol. 14, no. 6, 2006.

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

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

[19] 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.

[20] https://multipath-tcp.org/pmwiki.php/Users/Android

[21] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A Close Examination of Performance and Power Characteristics of 4G LTE Networks," in *Proc. ACM MobiSys*, 2012.

[22] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu, "Energy Proportional Datacenter Networks," in *Proc. ACM ISCA*, 2010.

[23] M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, "Dynamic Right-Sizing for Power-Proportional Data Centers," *IEEE/ACM Trans. Networking*, vol. 21, no. 5, pp. 1378-1391, 2013.

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

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

[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] 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.

[29] 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.

[30] https://code.google.com/p/multipath-tcp/downloads/list