A Hybrid Multipath Congestion Control Algorithm for High Speed and/or Long Delay Networks

Bich-Phuong Ha
Faculty of Information Technology,
HCMC University of Transport
Hochiminh City, Vietnam
Email: bphuong_dv@hcmutrans.edu.vn

Bao-Yen Tran, Tuan-Anh Le, Cong-Hung Tran
Faculty of Information Technology,
Posts and Telecommunications Institute of Technology
Hochiminh City, Vietnam
Email: ttbyen.hcm@vnpt.vn, {letuanh,conghung}@ptithcm.edu.vn

Phuong L. Vo
School of Computer Science and Engineering,
Vietnam National University HCMC, International University
Hochiminh City, Vietnam
Email: vtlphuong@hcmiu.edu.vn

Abstract— **As the prevalence of producing devices equipped with multiple interfaces, the idea to enhance network resource usage by developing a multipath transmission control protocol has recently captured great attention from many researchers. Moreover, there is a considerable increase in user demand of transmitting data over high speed and/or long delay networks. Various algorithms have been proposed lately to improve the performance of end-to-end connections in such environments. However, most existing schemes belong to either loss-based approach or delay-based approach, which have well-known problems about fine-grained of load balancing achievement, or fairness, respectively. In this paper, we propose a hybrid multipath congestion control algorithm for high speed and/or long delay networks, named MCompound. As a design extended from Compound TCP, it can achieve high link utilization while remains good fairness. The performance of the algorithm shown in the simulation results is meliorated as compared to other algorithms.**

I. Introduction

For decades, Transmission Control Protocol (TCP) [1] has been broadly used on the Internet for its reliability and fairness on competing flows. Howbeit, as pointed out in [2], it under-utilizes the network bandwidth over high speed and/or long latency networks because of its congestion control algorithm. Thus, many algorithms have been proposed to address this issue. Based on the input of algorithm using to control window size, they are classified into three main categories, namely loss-based approach, delay-based approach, and hybrid approach.

Loss-based approach uses packet loss as an indication of network congestion to reduce its sending rate. According to [3], it needs to be highly aggressive to achieve efficiency requirement, resulting in its poor fairness. On the other hand, delay-based approach uses queuing delay as a congestion decision. This makes it effectively utilize network resources while still achieves achieving good fairness. Nevertheless, it is said to be too good in its fair share to compete with loss based flows [4]. Therefore, to combine the advantages of aforementioned kinds, a hybrid approach, called CTCP [3], has been presented by Microsoft Research. CTCP proves that it outperforms them in terms of efficiency, RTT fairness, and TCP fairness. Moreover, it is already implemented in the latest versions of Windows operating systems, e.g., Windows Server 2008, Windows 7, Windows 8.

Additionally, transmitting data over high speed and/or long latency links via multiple interfaces will become a new trend in the next few years. This leads to some recent proposals for multipath congestion control protocol.

Inspired by the idea to help users with multi-interface devices can take advantage of extra network resources, Ford et al. have proposed MPTCP [5]. It efficiently increases the throughput and robustness of applications by creating a TCP sub-flow for each interface and allowing them to transmit data simultaneously. Even so, MPTCP cannot work well in high speed and/or long latency networks. Hence, MPCubic [6], an extension of TCP Cubic [7], has been proposed lately. Even though Cubic can attain high throughput and fairness independent of RTT, it is said to aggressively enlarge its window size, inflating queue and bloating RTT [8]. Consequently, MPCubic is inherited these strong points and weak points from Cubic. Another algorithm, by contrast, is MPFAST [9], originated from FAST TCP [10]. MPFAST is known to work well as a fine-grained load balancing in high speed and/or long latency networks, yet it may drastically suffer throughput degradation when competing with loss-based flows due to fair sharing property of delay-based scheme [11].

In this paper, we propose a hybrid multipath congestion control protocol for high speed and/or long latency networks, named MCompound. Based on CTCP, we follow three design goals of multipath TCP congestion control, which are fairness, load balancing, and efficiency, to develop MCompound. The performance of the algorithm shown in simulation results is meliorated as compared to algorithms mentioned before.

The rest of paper is structured as follows. Section II is a brief express of some related works. Then, we describe details of MCompound design in Section III. Simulation results are discussed in Section IV and a brief conclusion is in Section V.

II. Related Work

In an attempt to exploit path diversity, many multipath TCP variants have been proposed and demonstrated their pretty good performance on transmitting data over the networks. However, existing protocols are considered inefficient in high speed and/or long delay environments or inaccurate in estimating congestion information. These protocols use two kinds of approaches for TCP congestion control which are loss-based approach and delay-based approach. With loss-based approach, TCP source increases its sending rate until a packet loss happens due to a fully bottleneck link buffer. This packet loss provides an implicit evidence for congestion signal. To be evaluated more quickly than loss-based approach in detecting congestion, delay-based leans on the sudden increase of end-to-end delay as a sign of congestion signal. Therefore, congestion can be detected sooner, before packet loss occurs. However, when being deployed in the same link, the presence of aggressive loss-based flows can completely fill queue buffers at network link, and significantly deteriorate throughput of delay-based flows [11]. We abridge some protocols below.

The design of MPTCP with opportunistic linked increase algorithm (OLIA) [12] can solve the problem of the linked increase algorithm (LIA) of MPTCP. That is it is impossible to increase throughput of upgraded MPTCP users without decreasing throughput of others, or without increasing the congestion cost. MPTCP, however, is originated from TCP Reno which will eventually become a performance bottleneck itself in the environment that as bandwidth delay product continues to grow. That is the reason contributes to a poor performance of MPTCP in high speed and/or long delay networks.

A delay-based solution, named MPFAST [9], which can estimate more accurately than loss probability, has recently been proposed for multipath congestion control schemes in high speed and/or long delay networks. The reasons are two-fold, firstly, packet loss rarely happens in such networks under TCP Reno and its variants, and secondly, loss samples provide coarser information than queuing delay samples. Notwithstanding, MPFAST is not competitive to loss-based approaches in the same link [13].

Before MPFAST, MPCubic [6] has been developed for transmitting data over large bandwidth delay product. Therefore, it effectively utilizes the network resources. Howbeit, MPCubic is a loss-based approach, which leads to the result of grabbing throughput of other protocols existing in that path.

III. Design of Multipath Compound TCP

A. Design of single-path CTCP

In this section, we briefly summarize the CTCP algorithm for single-path transmission. Consider a source i of single-path CTCP [3] adjusts its window size w_i periodically as followings. It is a sum of a regular TCP loss-based window w_{ic} and a scalable delay window w_{id}. w_{ic} is increased by one every RTT, w_{id} is increased in such that the total window w_i at that RTT follows $w_i = w_{i-1} + \alpha w_i$ in absence of packet losses; when there is a loss, w_i reduces as $w_i = w_{i-1}(1-\beta)$. Fig. 1 illustrates three-phase evolution of a single-path CTCP window.

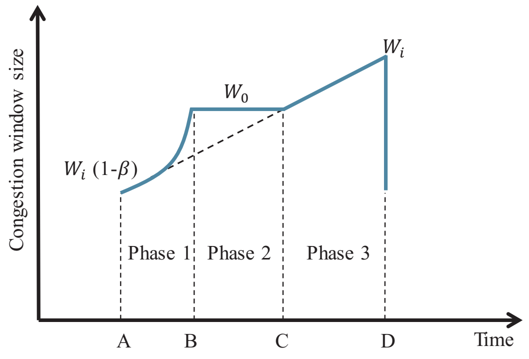

Fig. 1. The evolution of CTCP window.

After experiencing a loss event and decreasing window size, during time A to B, the queue has been drained so the window increases as

$$\frac{dw_i}{dt} = \frac{\alpha {w_i}^k}{R} \implies \frac{{w_i}^{1-k}}{1-k} = \frac{\alpha t}{R_i}. \quad (1)$$

Thus, we have the interval of phase 1 calculated as

$$T_{iAB} = \frac{R_i}{\alpha(1-k)}\left[{w_{i0}}^{1-k} - (1-\beta)^{1-k}{w_i}^{1-k}\right]. \quad (2)$$

And the number of transmitted packets during phase 1 is:

$$\begin{aligned} N_{iAB} &= \int_{t_{iA}}^{t_{iB}} \frac{w_i}{R_i} dt \\ &= \frac{1}{\alpha(2-k)}\left[{w_{i0}}^{2-k} - (1-\beta)^{2-k}{w_i}^{2-k}\right]. \end{aligned} \quad (3)$$

When the amount of backlogged data is larger than a threshold γ (usually set to 30, [3]), the sender now switches to phase 2. In this phase, w_{ic} continues to grow while w_{id} reduces gracefully so the window remains stable around w_{i0}. Then, we have the number of transmitted packets during phase 2 is $N_{iBC} = w_0 T_{iBC}$.

At time C, w_{id} drops to zero, $w_i = w_{ic}$. And just like regular TCP, window increases linearly in phase 3 as following:

$$\frac{dw_i}{dt} = \frac{1}{w_i}(1-p_i). \quad (4)$$

Hence, we get the average throughput $\bar{\wedge}_i = \sqrt{2/p_i}/R_i$. And the number of transmitted packets in this phase is $N_{iCD} = \frac{1}{2}\left(w_i^2 - w_{i0}^2\right)$.

Thus we have the total packets sent from time D to F is $Y_i = N_{iAB} + N_{iBC} + N_{iCD} \approx 1/p_i$. And the throughput is

$$\wedge_i = \frac{1/p_i}{R_i\left(T_{iAB} + T_{iBC} + T_{iCD}\right)}. \quad (5)$$

.

With consideration that loss rate is quite high compared to link speed so that the throughput is limited by packet loss ratio [3], we have $\frac{1}{\alpha(2-k)}\left[{w_i}^{2-k} - (1-\beta)^{2-k}{w_i}^{2-k}\right] = 1/p_i$.

Hence, we have the average throughput in this case as

$$\bar{\wedge}_i = \frac{1}{\alpha^{\frac{1}{2-k}}\left[1-(1-\beta)^{1-k}\right]}\left[\frac{1-(1-\beta)^{2-k}}{2-k}\right]^{\frac{1-k}{2-k}} \frac{1}{\bar{R}_i \bar{p}_i^{\frac{1}{2-k}}}. \tag{6}$$

CTCP parameters α, k, and β determine aggressiveness of CTCP (α = 0.125, k = 0.75, β = 0.5, as suggested in [3]), R_i is the round-trip time measured at source i. And $\bar{R}_i$ and $\bar{p}_i$ denote the average values of $R_i(t)$, and $p_i(t)$, respectively.

B. Design of Multipath CTCP

In this section, we propose a multipath version of single-path CTCP. Design of MCompound is started from the single-path model also with three phases as follows. Please note that, decrease phase remains untouchable, and the CTCP window keeps stable in phase 2 in the increase phase, thus we only focus on phase 1 and phase 3. Let us denote r is a sub-flow of MCompound which sends its data packets on path r.

Phase 1: MCompound's congestion control algorithm of source s adapts periodically the window $w_{s,r}$ on path r in this phase according to:

$$\frac{dw_{s,r}}{dt} = \frac{\alpha {w_{s,r}}^k}{R_{s,r}}\theta_{1s,r}. \tag{7}$$

where $\theta_{1s,r} \in [0,1]$ (note that $\sum_r \theta_{1s,r} = 1$) denotes the weight of source s on path r. Let us define N to be the number of sub-flows (i.e., paths) of a MCompound connection. The average throughput of sub-flow r of source s is calculated as similarly as the average throughput of single-path CTCP

$$\bar{\wedge}_{s,r} = \frac{1}{\alpha^{\frac{1}{2-k}}\left[1-(1-\beta)^{1-k}\right]}\left[\frac{1-(1-\beta)^{2-k}}{2-k}\right]^{\frac{1-k}{2-k}} \cdot \frac{\bar{\theta}_{1s,r}^{\frac{1}{2-k}}}{\bar{R}_{s,r}\bar{p}_{s,r}^{\frac{1}{2-k}}}. \tag{8}$$

We now get

$$\bar{p}_{s,r} = \left[\frac{\frac{1}{\alpha^{\frac{1}{2-k}}\left[1-(1-\beta)^{1-k}\right]}\left[\frac{1-(1-\beta)^{2-k}}{2-k}\right]^{\frac{1-k}{2-k}}}{\bar{\wedge}_{s,r}\bar{R}_{s,r}}\right]^{2-k} \bar{\theta}_{1s,r}. \tag{9}$$

With $\bar{\theta}_{1s,r}$ indicates the aggressiveness level of MCompound window growth so that it can share bandwidth fairly with a single-path CTCP since MCompound sub-flows compete with a CTCP flow at a common bottleneck. To satisfy fairness goal, the average total throughput of source s should equal to that of a regular CTCP flow. Thus, we have:

$$\sum_{r=1}^{N} \bar{\wedge}_{s,r} = \bar{\wedge}_i = \frac{1}{\alpha^{\frac{1}{2-k}}\left[1-(1-\beta)^{1-k}\right]}\left[\frac{1-(1-\beta)^{2-k}}{2-k}\right]^{\frac{1-k}{2-k}} \cdot \frac{1}{\bar{R}_i \bar{p}_i^{\frac{1}{2-k}}}. \tag{10}$$

We assume that they are competing in a single bottleneck, so that they suffer the same packet loss probability. By substituting (9) into (10), then (10) becomes

$$\sum_{r=1}^{N} \bar{\wedge}_{s,r} = \frac{1}{\alpha^{\frac{1}{2-k}}\left[1-(1-\beta)^{1-k}\right]}\left[\frac{1-(1-\beta)^{2-k}}{2-k}\right]^{\frac{1-k}{2-k}} \frac{1}{\bar{R}_i} \cdot \frac{\bar{\wedge}_{s,r}\bar{R}_{s,r}}{\frac{1}{\alpha^{\frac{1}{2-k}}\left[1-(1-\beta)^{1-k}\right]}\left[\frac{1-(1-\beta)^{2-k}}{2-k}\right]^{\frac{1-k}{2-k}} \bar{\theta}_{1s,r}^{\frac{1}{2-k}}}. \tag{11}$$

To determine $\bar{\theta}_{1s,r}$, we consider all paths experienced the same RTT. Hence, we have

$$\sum_{r=1}^{N} \bar{\wedge}_{s,r} = \frac{\bar{\wedge}_r}{\bar{\theta}_{1s,r}^{\frac{1}{2-k}}} \tag{12}$$

$$\Longrightarrow \bar{\theta}_{1s,r} = \left(\frac{\bar{\wedge}_{s,r}}{\sum_{r=1}^{N}\bar{\wedge}_{s,r}}\right)^{2-k}, \tag{13}$$

where $\bar{\theta}_{1s,r}$, $\bar{R}_{s,r}$ denote the average values of $\theta_{1s,r}(t)$, and $R_{s,r}(t)$, respectively.

Phase 3: Similar to the single-path CTCP algorithm expressed above, we have the average throughput in this phase calculated as $\bar{\wedge}_{s,r} = \frac{\sqrt{2/p_{s,r}\bar{\theta}_{2s,r}}}{R_{s,r}}$. And the total throughput of source s is

$$\sum_{r=1}^{N} \bar{\wedge}_{s,r} = \frac{\sqrt{2/p_i}}{R_i} \tag{14}$$

with the same assumption as phase 1.

Hence, we now get

$$\bar{\theta}_{2s,r} = \left(\frac{\bar{\wedge}_{s,r}}{\sum_{r=1}^{N}\bar{\wedge}_{s,r}}\right)^{2}. \tag{15}$$

The sub-flow on path r changes its window size described by the pseudo-code of MCompound shown in Algorithm 1.

IV. Performance Evaluations

So as to evaluate the performance of the introduced MCompound, several scenarios shown in Fig. 2 are run on NS2 [14]

Algorithm 1 Multipath CTCP Algorithm for the sender on path r.

```
Initialization: $total\_rate \leftarrow 0$, $limi[r] \leftarrow 0$, $\bar{\theta}_1[r] \leftarrow 1$, $\bar{\theta}_2[r] \leftarrow 1$, $\bar{\theta}_2[r] \leftarrow 1$, $inc[r] \leftarrow 0$
**On receipt of each update every RTT of a sub-flow $r$:**
/* cwnd, dwnd is the loss-based window, and delay-based window, respectively */
    $limi[r] \leftarrow limi[r] + \bar{\theta}_2[r]$;
**if** ($limi[r] \geq 1$) **then**
    $snd\_cwnd\_cnt[r] + +$;
    $limi[r] \leftarrow 0$;
**end if**
**if** ($snd\_cwnd\_cnt[r] \geq snd\_cwnd[r]$) **then**
    $inc[r] \leftarrow 1$;
    $snd\_cwnd\_cnt[r] \leftarrow 0$;
**end if**
**if** ($inc[r] \& snd\_cwnd[r] < snd\_cwnd\_clamp[r]$) **then**
    $cwnd[r] + +$;
**end if**
**if** ($diff[r] < \bar{\theta}_3[r] * \gamma$) **then**
        $dwnd[r] \leftarrow \alpha * W[r]^k * \bar{\theta}_1[r]$;
        $dwnd[r] + = dwnd[r]$;
**end if**
    $dwnd[r] = dwnd[r]$;
$W[r] = cwnd[r] + dwnd[r]$;
**Adjust_theta():**
    Calc_theta();
/* Calculate smoothed rate for each sub-flow */
$rate[r] = 0.875 * rate[r] + 0.125 * W[r]/rtt[r]$;
$total\_rate + = rate[r]$;
/* Calculate $\bar{\theta}_1[r]$, $\bar{\theta}_2[r]$, $\bar{\theta}_3[r]$ for each sub-flow */
    **Calc_theta():**
$\bar{\theta}_1[r] = pow(rate/total\_rate, 2 - k)$;
$\bar{\theta}_2[r] = pow(rate/total\_rate, 2)$;
$\bar{\theta}_3[r] = rate/total\_rate$;
```

with the selective acknowledgment 1 (SACK1) option, a 1448-byte data packet and router buffer size of BDP. Our simulations are divided into three parts, including: load balancing, efficiency, and fairness. In all simulations, the samples of data rate are taken every 0.5 seconds.

A. Load Balancing

In order to know how well the MCompound balances congestion in networks, we consider the topology in Fig. 2(a) with propagation delay is 64ms. Specifically, two MCompound sub-flows and a regular CTCP on link 1 concurrently start at 0s. Next, two regular CTCP flows come up on link 2, and link 3, respectively, at 70s. All flows stop at 300s. Fig. 3 shows that MCompound can quickly shift traffic away from link 2 and link 3 which are more congested than link 1. Firstly, sub-flow 1 and sub-flow 2 drastically increased and reach their average rate at about 161.4Mbps, and 124.61Mbps respectively. As a result, MCompound obtains its average optimal rate of 286Mbps. And when 2 regular CTCP flows

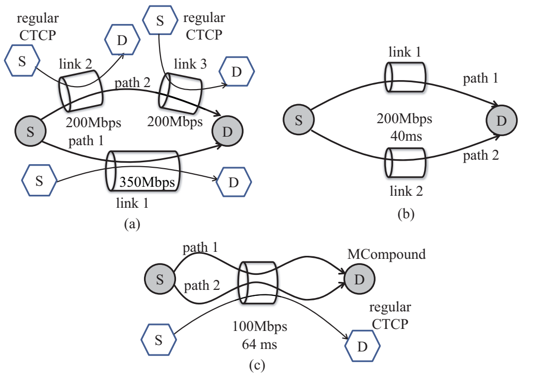

Fig. 2. Simulation topologies.

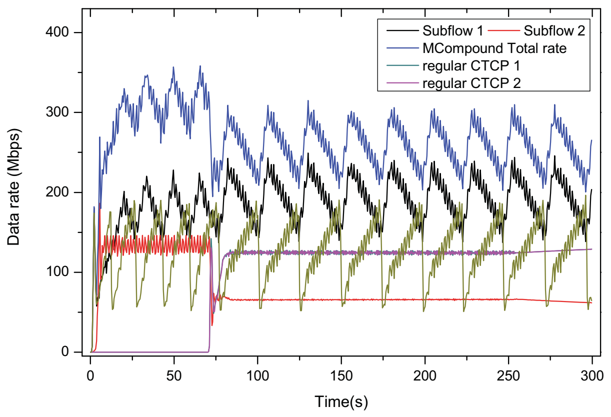

Fig. 3. MCompound adjusts sub-flow's rate to cope with network congestions.

begin to run, link 2 and link 3 become more congested, MCompound instantaneously moves its traffic to link 1. This leads to approximate 30Mbps increase in average rate of sub-flow 1, and 58.74Mbps decrease in that of sub-flow 2.

MCompound proves its effectiveness through balancing congestion in networks by adjusting reasonably sub-flows rate according to the environment that exists congestion or not.

B. Efficiency

This section substantiates MCompound efficiency amelioration in comparison with MPTCP. Scenario shown in Fig. 2(b) is a symmetric scenario where each sub-flow is at a distinct bottleneck with 200Mbps link capacity and 40ms propagation delay. Fig. 4 compares the throughput of MCompound and MPTCP during time simulation. It can be clearly seen that MCompound utilizes the network bandwidth more efficiently than MPTCP. As expected, after a swift increase, MCompound throughput peaks well over 380Mbps, and remains steady for nearly 400s. It then slightly drops to about 320Mbps until at which point the figures rapidly increases and reaches its peak throughput. In contrast, MPTCP throughput sluggishly climbs to the value that network paths are used effectively. In spite of its effort to stay constant for about 80s, less than 5 times

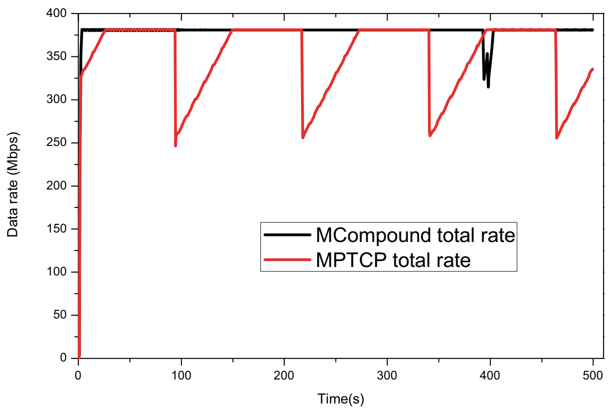

Fig. 4. Efficiency melioration of MCompound compared to MPTCP.

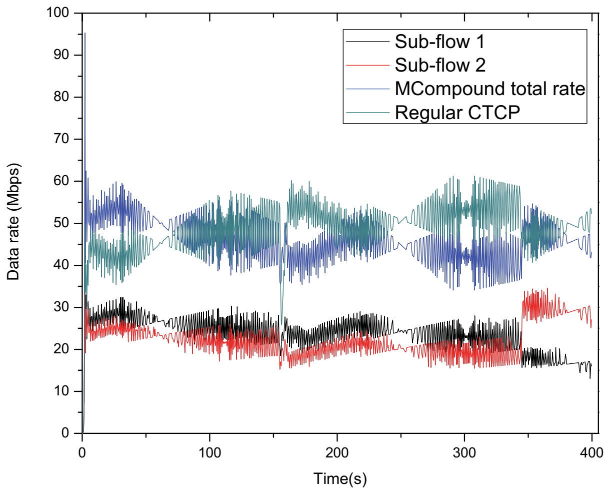

Fig. 5. MCompound fairly shares with regular CTCP at a common bottleneck.

compared to MCompound, it reduces dramatically to around 250Mbps, then slowly goes up to attain the target throughput.

Overall, this result implies that MCompound can be made to utilize the available transmission capacity much better than MPTCP.

C. Fairness to single-path CTCP

Finally, this topology aims to test the intra-fairness of MCompound. It can be seen in Fig. 2(b) that two MCompound sub-flows compete with a regular Compound at a common bottleneck with link capacity of 100Mbps and propagation delay of 64ms. Fig. 5 plots the competition result between them. Over the entire period shown, the total rate of MP-Compound is almost double of the sub-flow, and there is a slight oscillation in window size of two sub-flows due to the RTT oscillation. As a result, MCompound total rate is similar to that of single path CTCP. In summation, MPCompound is a good fairness multipath transport protocol for high speed and/or long latency because it satisfies the requirement that fairly shares the network resource with a CTCP flow.

V. Conclusions

Coming from the idea to help users transmit data over high speed and/or long delay networks through multiple paths, we propose a hybrid multipath transport protocol, named MCompound. As the name indicates, it is based on Compound TCP which inherits the advantages from loss-based schemes and delay-based schemes. Through simulation results, MCompound proves that it can meliorate the utilization of available network bandwidth, alleviate network congestion while preserving a good fairness to other flows.

Acknowledgments

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.02-2013.48.

References

[1] J. Postel, “Transmission control protocol,” IETF RFC 793, Sept. 1981.
[2] S. Floyd, “Highspeed tcp for large congestion windows,” *SIGCOMM Comput. Commun. Rev.*, Dec 2003.
[3] K. Tan, J. Song, Q. Zhang, and M. Sridharan, “A Compound TCP approach for high-speed and long distance networks,” in *INFOCOM*, 2006.
[4] J. Mo, R. La, V. Anantharam, and J. Walrand, “Analysis and Comparison of TCP Reno and Vegas,” in *INFOCOM*, 1999.
[5] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, “Architectural guidelines for multipath TCP development,” IETF RFC 6824, 2011.
[6] T. Le, R. Haw, C. Hong, and S. Lee, “A multipath Cubic TCP congestion control with multipath fast recovery over high bandwidth-delay networks,” *IEICE Trans. Comm*, pp. 2232–2244, 2012.
[7] S. Ha, I. Rhee, and L. Xu, “CUBIC: A new TCP-friendly high-speed TCP variant,” in *ACM SIGOPS Operating System Review*, July 2008.
[8] K. Winstein and H. Balakrishnan, “TCP ex machina: Computer-generated congestion control,” 2013, pp. 123–134.
[9] Bich-Phuong Ha, Bao-Yen Tran, Tuan-Anh Le, and Cong-Hung-Tran, “Multipath FAST TCP for Large Bandwidth-Delay Product Networks,” in *Proc. of the International Conference on Green and Human Information Technology ICGHIT Conf.*, 2014.
[10] C. Jin, D. Wei, and S. Low, “FAST TCP: Motivation, architeccture, algorithm, performance,” in *IEEE Infocom*, 2004.
[11] M. Podlesny and C. Williamson, “Providing fairness between TCP NewReno and TCP Vegas with RD network services,” in *Quality of Service (IWQoS), 2010 18th International Workshop on*, June 2010, pp. 1–9.
[12] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, “MPTCP is not pareto-optimal: performance issues and a possible solution,” in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, ser. CoNEXT ’12. New York, NY, USA: ACM, 2012, pp. 1–12.
[13] J. Wang, F. Gao, J. Wen, C. Li, Z. Xiong, and Y. Han, “Achieving TCP Reno friendliness in FAST TCP over wide area networks,” in *Computing, Networking and Communications (ICNC), 2014 International Conference on*, Feb 2014, pp. 445–449.
[14] NS-2 network simulator. [Online]. Available: http://www.isi.edu/nsnam/ns/