Coordinated TCP Westwood Congestion Control for Multiple Paths over Wireless Networks

Tuan Anh Le, Choong Seon Hong, and Eui-Nam Huh

Department of Computer Engineering, Kyung Hee University
1 Seocheon, Giheung, Yongin, Gyeonggi, 446-701, Korea
{letuanh, cshong, johnhu}@khu.ac.kr,

***Abstract*— Multipath TCP (MPTCP) has just been approved by the IETF. It was designed to be fairly shared with regular TCP, so its performance is equivalent that of a regular TCP flow that on the best path for it. However, regular TCP has been proven its performance very poor in wireless networks, where packet loss often is caused by random error rather than by network congestion as in wired networks. TCP Westwood (TCPW) uses the available bandwidth estimation technique to improve TCP performance in such environment. In this work, we propose an extended version of TCP Westwood for multiple paths, called MPTCPW. To start with the analysis model of TCPW, MPTCPW is designed as a coordinated congestion controller between paths which allows load-balancing, fair sharing to regular TCPW at bottleneck. Our simulation results show that MPTCPW can achieve higher throughput than MPTCP in wireless environments, fairness to regular TCPW, and greater load-balancing than uncoordinated MPTCPW.**

I. Introduction

Nowadays portable devices with multiple wireless interfaces are becoming more popular. If multiple links are used simultaneously for a transport protocol session, multipath transport protocol would be able to avoid sending traffic on hotspot and failure links, and to support mobility [1]. So far, the performance and resilience are improved.

Regular TCP [2] performance is degraded significantly as packet losses are occurred by random error (caused by signal fading, or multiple-access interference) in wireless links rather than by network congestion as in wired networks. This is because regular TCP reduces blindly its congestion window by haft when a packet drop is detected. To overcome this problem, TCP Westwood (TCPW) [3] [4] has been proposed. In TCPW scheme[1], the sender monitors ACK inter-arrival time in order to estimate the bottleneck bandwidth available. Instead of reducing congestion window by haft, TCPW updates both congestion window and slow-start threshold to the estimated bandwidth (in terms of the number of packets) whenever detecting packet drop. TCPW performance has been proven significant improvement in wireless networks.

A multipath TCP protocol session consists of simultaneous multiple sub-flows amongst paths between two end-hosts [5], where each sub-flow performs flow control function on a path. So there are some requirements for congestion control design for multiple paths [6] as follows: (i) A multipath TCP protocol should fairly share bandwidth with single-path TCP protocol without any common bottleck detection mechanism; (ii) To improve multipath TCP performance in multipath environments. (iii) To exploit load-balancing between paths.

Multipath TCP (MPTCP) [6] has just been approved by the IETF. Its design aims to satisfy the above requirements. MPTCP sends more traffic on lightly loaded paths by grouping a set of resources on all paths as a single resource, as suggested by the resource pooling principle [7]. Since MPTCP performance is equivalent that of a regular TCP flow that on the best path for it, MPTCP faces with performance degradation over wireless networks.

In this paper, we are interested on an extended version of TCP Westwood for multiple paths, called MPTCPW. To start with the analysis model of TCPW, MPTCPW is designed as a coordinated congestion controller between paths which allows balancing the loads between the paths, while ensuring fair bandwidth allocation to regular TCPW at the shared bottleneck.

The rest of this paper is organized as follows: Section II summarized the previous works relevant to multipath congestion control. We describe the details of the TCPW analysis model, and then its extension for multiple paths in Section IV. We evaluate our proposal in terms of fluctuation, fairness, throughput over wireless links and load-balancing capability. Finally, we conclude our work in Section VI.

II. Related Work

In this section, we summarize several solutions has been proposed for multipath congestion control.

Multiple paths TCP (mTCP) [8] aggregates the available bandwidth of all paths in parallel, and detects and removes paths at shared bottleneck to alleviate unfair allocation.

The congestion controller of each sub-flow in Parallel TCP (pTCP) [9], Concurrent Transfer Multipath (CMT) over SCTP [10] operates independently each others as used in regular TCP. Therefore, they can not compete fairly with regular TCP flows at shared bottleneck because of no extra mechanism to handle or detect the bottleneck.

Reliable Multiplexing Transport Protocol (R-MTP) [11] was designed for wireless networks. Based on inter-arrival time, R-MTP's sub-flow adapts to rate control according to changes in bandwidth on each channel and congestion.

[1]The original version of TCP Westwood [3] overestimates the available bandwidth in the presence of ACK compression. TCP Westwood+ [4] solves this problem.

To improve fairness to regular TCP at the shared bottleneck, the weighted TCP [12] uses a fixed weight in allocating bandwidth to each sub-flows. The weight is chosen by $1/N^2$, where N is the number of paths in a multipath TCP session. Obviously the weighted TCP could fairly share if RTTs of flows are equal. This mechanism takes advantage of the simplicity of deployment and fair allocation without any shared bottleneck detection mechanism.

III. TCP Westwood Background

In this section, we briefly describe TCPW algorithm [4] and then its performance analysis model.

TCPW estimates bandwidth available based on inter-arrival time of ACK packets at the sender. Whenever the sender receives an ACK, it computes a bandwidth sample as

$$b_k = d_k/(t_k - t_{k-1}),$$

where d_k denotes the amount of data acknowledged by the k^{th} ACK. t_k and t_{k-1} denote the inter-arrival time of the k^{th} ACK and the $(k-1)^{th}$ ACK, respectively. To smooth out short-time variants in the bandwidth samples b_k, a discrete time-varying low-pass filter is used as

$$\hat{b}_k = \frac{2\tau - \Delta_k}{2\tau + \Delta_k}\hat{b}_{k-1} + \frac{b_k + b_{k-1}}{2\tau + \Delta_k}\Delta_k,$$

where $\hat{b}_k$ is the smoothed bandwidth at time t_k, $1/\tau$ is the cutoff frequency of the filter. Δ_k denotes the inter-arrival time between $(k-1)^{th}$ ACK and the k^{th} ACK. The sender uses the estimated bandwidth to update the congestion window size (w^{TCPW}) and the slow-start threshold ($ssthresh$) after receiving duplicated ACKs or detecting a timeout event, as

$$ssthresh = \hat{b} \times RTT_{min}/MSS,$$

where RTT_{min} denotes the minimum RTT during TCPW connection.

Now, we determine the fluid model of TCPW. In the congestion avoidance phase, TCPW remains the window growth based on the additive increase function as used in regular TCP (TCP Reno) as follows: Whenever the sender receives a positive ACK, it increase w^{TCPW} by $1/w^{TCPW}$. Whereas, after detecting a lost packet, it sets w^{TCPW} and $ssthresh$ to $\hat{B} \times RTT_{min}$, where $\hat{B}$ ($\hat{B} = \hat{b}/MSS$) is the estimated available bandwidth in terms of the number of packets. The TCPW algorithm is expressed as

$$\text{Increase: } w^{TCPW}(t) \leftarrow w^{TCPW}(t-1) + 1/w^{TCPW}(t-1),$$
$$\text{Decrease: } w^{TCPW}(t) \leftarrow \hat{B}(t) \times RTT_{min},$$

where $\hat{B}(t) = \hat{w}^{TCPW}(t)/RTT$ [13]. The fluid model of TCPW corresponding to such increase-decrease function is expressed in respect with continuous time t as

$$\frac{d}{dt}w^{TCPW}(t) = \frac{w^{TCPW}(t)}{RTT}\left(\frac{1-p}{w^{TCPW}(t)} - \frac{q}{RTT}w^{TCPW}(t)p\right), \quad (1)$$

where $q = RTT - RTT_{min}$ denotes the total queueing delay measured at two-end hosts. p denotes the packet drop probability. Suppose that the packet loss rate p is small (i.e., $1 - p \approx 1$), the fixed point equation of (1) gives

$$\frac{1}{\hat{w}^{TCPW}} = \frac{q}{RTT}\hat{w}^{TCPW}p,$$

or

$$p = \frac{RTT}{q(\hat{w}^{TCPW})^2}, \quad (2)$$

where $\hat{w}^{TCPW}$ is the equilibrium value of $w^{TCPW}(t)$.

IV. Extending TCPW towards MPTCPW

In this section, based on the fluid model of TCPW above, we carry out extending TCPW for multiple paths with constrains about performance improvement, load-balancing between paths and fair sharing [6] [14] to regular TCPW, which are the goals to design MPTCP [6].

Now, we propose MPTCPW's congestion window growth in additive increase function over path s as follows: Whenever the sub-flow receives a positive ACK on that path, it increase its congestion window (w_s) by $\min(\delta/w_s(t), 1/w_s(t))$; when receiving duplicate ACKs, it updates w_s to $\hat{B}_s \times RTT_{min,s}$. Therefore, the sub-flow's congestion control algorithm on path s is

$$\text{Increase: } w_s(t) \leftarrow w_s(t-1) + \min(\delta/w_s(t-1), 1/w_s(t-1)),$$
$$\text{Decrease: } w_s(t) \leftarrow \hat{B}_s(t) \times RTT_{min,s}.$$

So the fluid model corresponding to such increase-decrease function is

$$\frac{d}{dt}w_s(t) = \frac{w_s(t)}{RTT_s}\left[\min\left(\frac{\delta}{w_s(t)}, \frac{1}{w_s(t)}\right) - \frac{q_s}{RTT_s}w_s(t)p_s\right], \quad (3)$$

where $q_s = RTT_s - RTT_{min,s}$, δ denotes the linking between paths within an MPTCPW flow. $\min(.)$ function ensures that sub-flow's window increase (the left argument) does not exceed that of single-path TCPW (the right argument) over that path. Similarly, we have the fixed point equation of (3) as

$$\min\left(\frac{\delta}{\hat{w}_s}, \frac{1}{\hat{w}_s}\right) = \frac{q_s}{RTT_s}\hat{w}_s p_s. \quad (4)$$

If TCPW was on path s, from (2) its packet loss rate would be $p_s = RTT_s/q_s(\hat{w}_s^{TCPW})^2$. We substitute p_s into (4) to obtain

$$\hat{w}_s^{TCPW} = \max\left(\frac{\hat{w}_s}{\sqrt{\delta}}, \hat{w}_s\right). \quad (5)$$

To improve throughput and to preserve fairness, the total throughput of a MPTCPW flow should be equivalent that of a TCPW flow on the best path for it [15]. This implies that

$$\sum\nolimits_s \frac{\hat{w}_s}{RTT_s} = \max\nolimits_s\left\{\frac{\hat{w}_s^{TCPW}}{RTT_s}\right\}. \quad (6)$$

By substituting (5) into (6), we obtain

$$\sum_s \frac{\hat{w}_s}{RTT_s} = \max_s \left\{ \max \left(\frac{\hat{w}_s}{\sqrt{\delta} RTT_s}, \frac{\hat{w}_s}{RTT_s} \right) \right\}.$$

Observably, in the above equality, the total throughput cannot equal that on a path since $\hat{w}_s$ never equals zero, therefore

$$\sum_s \frac{\hat{w}_s}{RTT_s} = \max_s \left\{ \frac{\hat{w}_s}{\sqrt{\delta} RTT_s} \right\}.$$

By solving for δ, we obtain

$$\delta = \max_s \left\{ \left(\frac{\hat{w}_s}{RTT_s} \right)^2 \right\} \Big/ \left(\sum_s \frac{\hat{w}_s}{RTT_s} \right)^2. \qquad (7)$$

δ in the equation above is shared between paths and depends on only the maximum throughput among paths. However, the congestion window on the congested path is always smaller than that on the lightly loaded path. To perform load-balancing between the paths, the congestion window on the congested path must be grown more gradually than that on the lightly loaded path [14]. This implies that δ for the worse path would be the smaller value. Therefore, we slightly modify δ, and then replace δ with δ_s as

$$\delta_s = \hat{w}_s^{\gamma} \max_s \left\{ \frac{\hat{w}_s^{2-\gamma}}{RTT_s^2} \right\} \Big/ \left(\sum_s \frac{\hat{w}_s}{RTT_s} \right)^2, \qquad (8)$$

where γ is a trade-off parameter between load-balancing and protocol fluctuation, $0 \leq \gamma \leq 2$. In our experiments (not reported in this paper due to space limit), $\gamma = 1$ gives a reasonable trade-off between fluctuation and load-balancing.

V. Performance Evaluations

In this section, we evaluate fluctuation, fairness and load-balancing of MPTCPW, and compare its performance with MPTCP [6] in wireless networks. Load-balancing capability of MPTCPW is compared with uncoordinated MPTCPW (named unMPTCPW), in which each sub-flow performs the flow control function independently. To ensure fairness to regular TCPW at the shared bottleneck, we choose the traffic splitting factor value of $(1/2)^2$ for two-path unMPTCPW such that each sub-flow gets half throughput of what a regular TCPW gets.

We use NS-2 [16] in our experiments, and the selective acknowledgment (SACK) option [17]. The scenarios shown in Fig. 1 with a 1000-byte data packet, and random early drop (RED) queue management [18] to prevent routers from globally synchronized packet drops.

A. Fluctuation Investigation

Assume that the traffic load on two paths is the same. Is a multipath protocol *stable* when some packets are sent on one path for moment, and then on the other, and so on?. This implies that the paths are not used simultaneously at almost time [15] [14]. So any multipath transport protocol performance and resource pooling would be poor in such state.

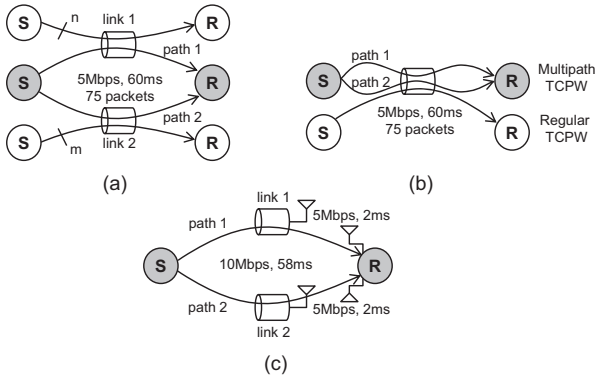

Fig. 1. Simulation scenarios.

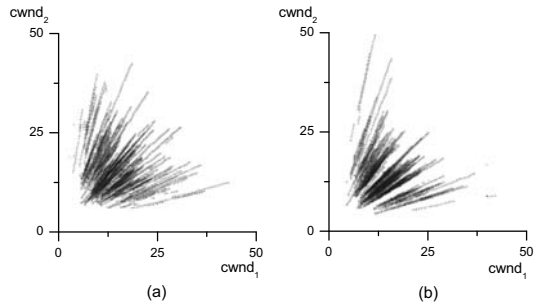

Fig. 2. (a) The congestion window evolution of two-path MPTCPW flow; (b) that of two-path MPTCP flow.

Now, we investigate protocol fluctuation in a typical network configuration, where two-separate path multipath protocol competing with background load generated by ten regular TCPW/TCP flows[2] on each path ($n = m = 10$) as shown in Fig. 1(a). Because of the same load on both paths, each sub-flow experiences the same congestion condition. Fig. 2(a) shows that the congestion windows on path 1 and 2 ($cwnd_1$, $cwnd_2$) in MPTCPW are increased regularly rather than biasing on any side (i.e., the protocol is fluctuation in its data rate on each path). This result is similar to that of MPTCP as shown in Fig. 2(b). Another simulation result in the same bottleneck is shown in Fig. 3(a), the evolution of congestion window on two paths is almost same.

B. Fair Sharing

In this section, we show how a multipath transport protocol fairly shares with a single-path protocol at a common bottleneck without any common link detection mechanism. We use a dumbbell scenario as shown in Fig. 1(b), where a two-path MPTCPW flow competes against a regular TCPW flow at the shared link. Fig. 3(a) shows that the single-path TCPW gets its congestion window twice. It is reasonable when total of data sent by the two-path MPTCPW flow is close the amount of data sent by the single-path TCPW flow as shown in Fig. 3(b).

[2]If multipath transport protocol is MPTCPW, we will use ten regular TCPW flows as background load. Otherwise, use regular TCP flows.

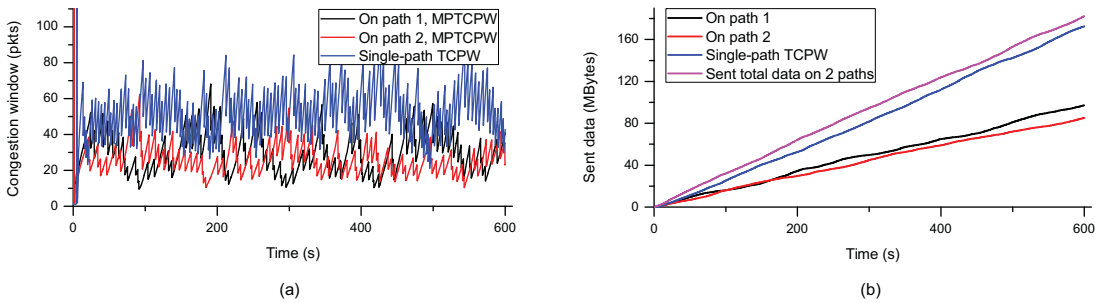

Fig. 3. (a) The congestion window evolution of a two-path MPTCPW flow competing against a regular TCPW flow in the shared bottleneck; (b) their amount of data transferred.

Another experiment result is shown in Fig. 5(a) with the network configuration detailed in Section V-B. In Fig. 5(a), MPTCPW moves more traffic from the congested path (path 1) to the lightly loaded path (path 2), so the sub-flow's rate on path 2 surges, but not exceed the average throughput of single-path TCPW flows on that path. Therefore, MPTCPW can preserve fairness to single-path TCPW in the general configuration.

C. Throughput over Wireless Links

In this section, we evaluate throughput performance of MPTCPW compared with that of regular MPTCP. The simulation scenario is shown in Fig. 1(c). The experiments were run for 600 seconds under wireless link's random error varying from 0.001% to 5% in packet loss rate.

Fig. 4(a) shows the average throughput of MPTCPW and MPTCP when the packet loss rates on two paths are equal. In Fig. 4(a), MPTCPW throughput is higher than that of MPTCP in any packet loss rate. Outperforming of MPTCPW is come from adaptive updating $ssthresh$ and $cwnd$ to the estimated available bandwidth on paths as detecting a packet loss or timeout event.

In the next experiment, we fix the packet loss rate at 0.01% on path 1 and vary the loss rate from 0.001% to 5% on path 2. Fig. 4(b) shows that MPTCP throughput is degraded significantly when packet loss rate on path 2 increases to 1%. Since MPTCPW can recover from random loss, it outperforms at any condition. In Fig. 4(b), when random loss on path 2 is very high (greater than 1%), the total throughput of two-path transport protocols is dominated by only the best path, path 1.

D. Load-Balancing

In this section, we investigate the capability of shifting traffic from the congested path in an adverse configuration, where RTT on the lightly loaded path is longer than that on the congested one[3]. We use such adverse configuration in order to demonstrate the prominent feature of MPTCPW compared with uncoordinated MPTCPW.

[3]For packet loss-based transport protocols such as regular TCP, TCPW, MPTCP, and MPTCPW, their congestion window sizes on the longer RTT paths are increased more slowly than that on the shorter RTT paths.

The coordinated congestion control in multipath is able not only to gather bandwidth on all paths into a bigger bandwidth but also to move traffic among paths according traffic load on each path. For such configuration, if we did not use δ_s as (8), the congestion window of sub-flow on the lightly loaded and longer RTT path was increased more slowly than expected. Based on computing δ_s as (8), the sub-flow can compensate congestion window increase for the longer RTT path.

Experiments were run in the simulation scenario as shown in Fig. 1(a), where link capacity and propagation delay of link 1 and 2 are set (10Mbps, 40ms) and (6Mbps, 160ms), respectively. To generate heavy load in link 1 and light load in link 2, we use TCPW flows as the background traffic with eight flows on link 1, and three flows on link 2. The observed packet loss rates at link 1 and 2 after experiment are 2.00% and 0.56%, respectively. Fig. 5(a) shows that the data rate of sub-flow on the congested and short RTT path (path 1) is very low. While the longer RTT path (path 2) has light load, the sub-flow on that path surges its rate up. Therefore, the total throughput of two sub-flows reaches the expected value, which is equivalent to the average rate of three TCPW flows that over the best path, path 2.

In contrast to load-balancing capability of MPTCPW, unMPTCP just splits the traffic across paths with factor of $(1/2)^2$. We alternated MPTCPW protocol in the above experiment with unMPTCP. Fig. 5(b) shows that each sub-flow gets half TCPW flows' average rate that they are sharing. The packet loss rates at link 1 and 2 are measured to be 2.08% and 0.35%, respectively. Although path 1 is congested, the sub-flow on that path still sends blindly packets at haft rate. Therefore, two-path unMPTCP can get the total throughput lower than TCPW flows in such adverse configuration.

VI. Conclusions

In this paper, we propose an extended version of regular TCP Westwood for multiple paths over wireless networks, called MPTCPW. To start with the analysis model of TCPW, MPTCPW congestion control is designed as a coordinated control between paths which allows load-balancing feature beween paths, fair sharing to regular TCPW at bottleneck. Our simulations show that MPTCPW can achieve stability, higher throughput compared with MPTCP, fairness to regular TCPW,

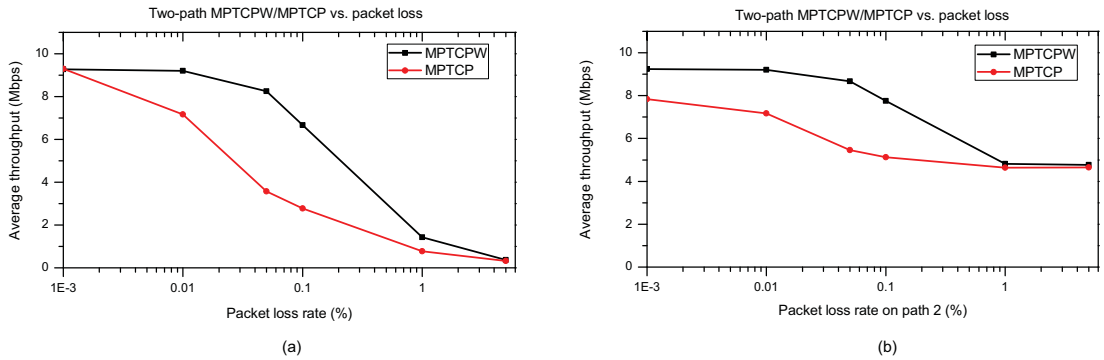

Fig. 4. (a) The average throughput of two-path MPTCPW and MPTCP vs. varying packet loss on path 1 and 2; (b) that of two-path MPTCPW and MPTCP vs. varying packet loss on path 2.

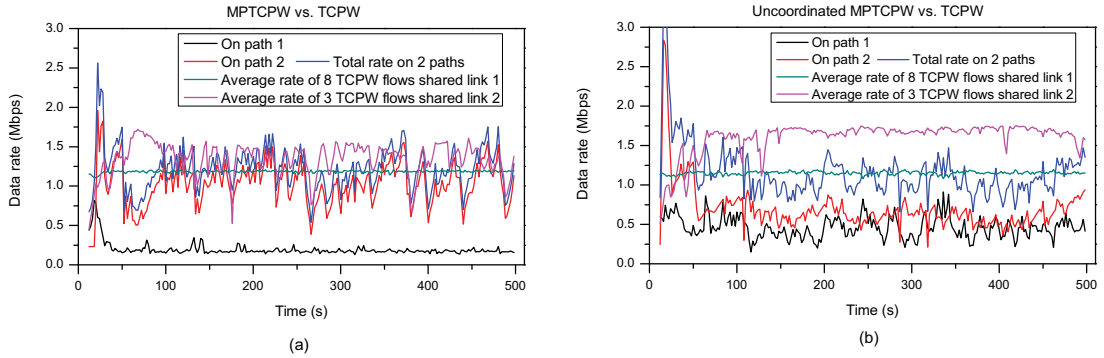

Fig. 5. (a) Load-balancing capability of MPTCPW; (b) lack load-balancing capability of uncoordinated MPTCPW when two-path MPTCPW competes against eight TCPW flows at link 1 and three TCPW flows at link 2.

and greater load-balancing than uncoordinated MPTCPW under various network conditions.

We believe that our extension technique can be generalized for extending current transport protocols, such as TCP Vegas, FAST TCP, and Compound TCP for multiple paths.

ACKNOWLEDGMENTS

This research was supported by the IT R&D program of KCA (10913-05004: Study on Architecture of Future Internet to Support Mobile Environments and Network Diversity). Dr. CS Hong is the corresponding author.

REFERENCES

[1] C. Raiciu, D. Niculescu, M. Bagnulo, and M. Handley, "Opportunistic mobility with multipath TCP," in *Proc. of the MobiArch Workshop*, 2011.
[2] V. Jacobson and M. J. Karels, "Congestion avoidance and control," in *ACM SIGCOMM*, 1988.
[3] M. Gerla, B. K. F. Ng, M. Y. Sanadidi, M. Valla, and R. Wang, "TCP Westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs," in *Computer Communications*, Jan. 2004.
[4] S. Mascolo, L. A. Grieco, R. Ferorelli, P. Camarda, and G. Piscitelli, "Performance evaluation of Westwood+ TCP congestion control," in *Performance Evaluation*, Jan. 2004.
[5] A. Ford, C. Raiciu, and M. Handley, "TCP extensions for multipath operation with multiple addresses," IETF Internet Draft, July 2011.
[6] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," IETF RFC 6356, Oct. 2011.
[7] D. Wischik, M. Handley, and M. B. Braun, "The resource pooling principle," in *ACM SIGCOMM CCR*, Oct. 2008.
[8] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in *Proc. of the USENIX'04 Annual Technical Conf.*, June 2004.
[9] H. Hseih and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts," in *Proc. of the MobiCom' 02 Conf.*, 2002.
[10] J. R. Iyengar, K. C. Shah, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Transactions on Networking*, pp. 951–964, 2006.
[11] L. Magalhaes and R. Kravets, "Transport level mechanisms for bandwidth aggregation on mobile hosts," in *Proceedings of the Ninth International Conference on Network Protocols*. Washington, DC, USA: IEEE Computer Society, 2001, pp. 165– 171.
[12] M. Honda, Y. Nishida, L. Eggert, P. Sarolahti, and H. Tokuda, "Multipath congestion control for shared bottleneck," in *Proc. PFLDNeT*, May 2009.
[13] L. A. Grieco and S. Mascolo, "Mathematical analysis of Westwood+ TCP congestion control," in *IEE PROCEEDINGS-CONTROL THEORY AND APPLICATIONS*, Jan. 2005.
[14] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proc. of the USENIX NSDI'11 Conf.*, Mar. 2011.
[15] D. Wischik, M. Handley, and C. Raiciu, "Control of multipath TCP and optimization of multipath routing in the Internet," in *Network Control and Optimization*, 2009.
[16] NS-2 network simulator. [Online]. Available: http://www.isi.edu/nsnam/ns/
[17] M. Mathis, J. Mahdavi, S. Floyd, and A.Romanow, "TCP selective acknowledgment options (SACK)," IETF RFC 2018, Oct. 1996.
[18] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: An algorithm for increasing the robustness of RED's active queue management," Tech. report, Tech. Rep., 2001.