International Journal of Control and Automation
Vol.7, No.5 (2014), pp.201-210
http://dx.doi.org/10.14257/ijca.2014.7.5.22

A Dynamic Weight Congestion Control Algorithm to Achieve Fairness in the MPTCP

Zun-liang Wang, Yue Ma and Jin-li Zhang

School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, P.R. China

wangzl@bupt.edu.cn

Abstract

Nowadays, a growing number of mobile equipments have more than one single network interface. For instance, smart phones have 3G/4G and Wi-Fi interfaces at the same time. Regular TCP restricts communications to a single path per transport connection. MPTCP (Multipath TCP) is a set of extensions to traditional TCP to improve the robustness and performance of end-to-end transport layer connections using more than one interface simultaneously. A key issue in the design of MPTCP is the fairness to traditional TCP flows. In this paper, we do a research on the congestion control mechanism of MPTCP and propose a dynamic weighted congestion control algorithm, which can improve the fault of the congestion control algorithm in the RFC 6356 when the subflows have share bottleneck. When the subflow increases the window size, dynamic weight algorithm will increase weight of the congestion window by the size of the congestion window. A MPTCP flow with the dynamic weighted congestion algorithm can allocate their traffic by the throughput of the subflow and get more fairness to traditional TCP flows than the MPTCP in RFC 6356.

Keywords: *MPTCP; Congestion control algorithm; Computer Network*

1. Introduction

With the development of network technology and multi-homed technology, more and more endpoints have multiple network adapters, such as notebook computers with wired (Ethernet) and wireless (Wi-Fi) network adapters, and smart phones with 3G/4G and Wi-Fi interface. All these endpoints can create multiple network connections at the same time. As shown in Figure 1, there are two flow paths between the smart phone and the server using 3G/4G and Wi-Fi interface simultaneously. if the multi-homed endpoints use multiple network interfaces [1], it can indeed balance the load between the different paths, switching dynamically and automatically the traffic from congested, disrupted or broken links to the best paths, which will greatly enhance the user experience effect. But the traditional TCP network architecture cannot use these multiple network interfaces resources. In order to achieve the goal of improving the robustness and performance of end-to-end connections in transport layer, the MPTCP protocol is proposed by IETF, which can use multiple connections transmit data at the same time in transport layer. MPTCP can balance the traffic of the subflow by connecting situation with the throughput of the subflow. The current research about MPTCP mainly comprises connecting management, flow control and congestion control. Congestion control algorithms mainly focus on improving the network throughput and guaranteeing the fairness between MPTCP and traditional TCP. In this paper we analyzed the congestion control algorithm in RFC6356 and proved the unfairness between subflows when subflows have a

Copyright ⓒ 2014 SERSC

share bottleneck. The congestion control algorithm in RFC6356 puts more traffic on the link which have lower loss rate. We present the dynamic weighted congestion algorithm to improve the fault of the algorithm in RFC6356. Dynamic weighted congestion control algorithm can guarantee the fairness between MPTCP and tradition TCP, which is based on the EWTCP. The dynamic weight algorithm increases the congestion widow by the weight, so it can allocate the traffic to subflows as the throughput of the sublow.

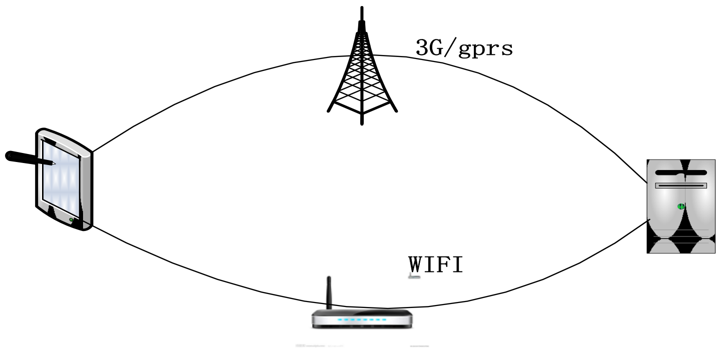


Figure 1. MPTCP network structure

In this paper we first introduce the architecture of MPTCP in Section 2, as it is specified in the current versions of the IETF drafts. In Section 3 we analyze the problems of the algorithm in RFC6356.Then in Section 4 we discusse the process of modeling the dynamic weight of congestion control algorithm. In Section 5 we describe the simulation scenario, and comment the behavior of the dynamic weight of congestion control algorithm. Finally, in Section 6 we conclude the paper.

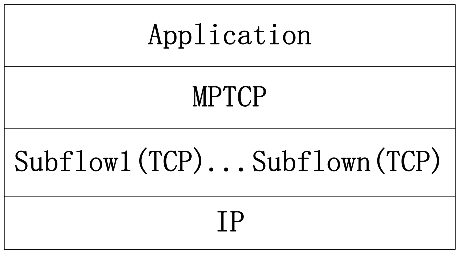


Figure 2. MPTCP architecture

2. MPTCP Architecture

In order to achieve the purpose of MPTCP, an IETF working group has recently been created to specify a multipath protocol for the transport layer. The architecture of the MPTCP is putted forward as shown in Figure 2. IETF adds a MPTCP layer based on the traditional TCP/IP network model. MPTCP layer is transparent to both higher application and lower IP network layer. It is a set of additional functions to regular TCP. MPTCP allows one TCP connection to be spread across more than one paths. MPTCP makes use of standard TCP sessions as subflows to provide the underlying transport per path. MPTCP distributes load through the creation of separate subflows across disjoint paths. MPTCP layer is responsible for the subflow scheduling and submits data to the application layer. The application layer needs not consider the detail of the transport layer. Traditional TCP layer in the protocol

Copyright ⓒ 2014 SERSC

ensures the reliable transmission of subflow and congestion control, and submits data to the MPTCP layer.

There are some functions MPTCP must implement, path management, packet scheduling, subflow interface and congestion control. The path management looks after the discovery of multiple paths between two hosts. The packet scheduler receives data from the application and make the necessary operations before sending data to a subflow. The subflow adds sequence number and passes them to network. The receiving subflow re-orders data and passes it to the packet scheduling component, which performs re-ordering and sends to the application.

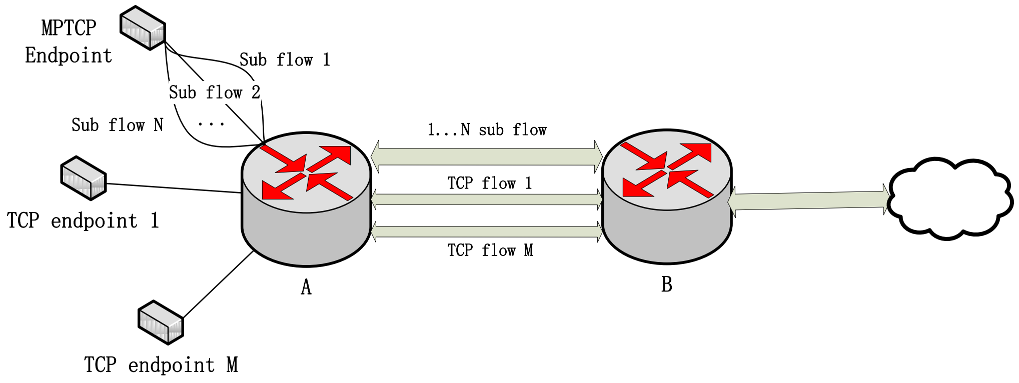


Figure 3. MPTCP fairness schematic

MPTCP should perform congestion control as traditional TCP does. As a TCP flow each subflow of MPTCP must have its own congestion control state so that capacity on that path is matched by offered load. The simplest way to achieve this goal is to simply run TCP Reno congestion control algorithm on each subflow. But this solution is unsatisfactory as it gives the multipath flow an unfair share when the paths taken by its different subflows share a common bottleneck. As shown in Figure 3, when a MPTCP flow containing N subflows has a share bottleneck with a traditional TCP flow, if each subflow run congestion control algorithm as traditional TCP does, the MPTCP is approximately N times as aggressive as each TCP flow. In Figure 3, a MPTCP connection that contains N subflows competes with M traditional TCP flows at the shared bottleneck between router A and router B. While each traditional TCP flow receives a 1/(N +M) share of the bottleneck, the MPTCP flow receives a N/(N +M) share.

In order to keep the fairness between traditional TCP and MPTCP, reference [2] proposed several principles of MPTCP congestion control algorithm:

1. A multipath flow should perform at least as well as a single path flow would be on the best path available to it.

2. A multipath flow should not take up more capacity from any of the resources shared by its different paths if it were a single flow using only one of these paths. This guarantees that it will not unduly harm other flows.

3. A multipath flow should move off its most congested paths as much traffic as possible.

The first principle means to improve throughput, the second principle means to do no harm, and these two principles together ensure fairness at the bottleneck. The third principle means to balance congestion, it captures the concept of resource pooling: if each multipath flow sends more data through its least congested path, the traffic in the network will move away from congested areas. This improves robustness and overall throughput.

3. Problem Definition and Analysis

As described in Section 2, the congestion control of MPTCP aims to set the multipath flow's aggregate bandwidth to be the same as that of a regular TCP flow would get on the best path available to the multipath flow. An algorithm named coupled congestion control algorithm has proposed in RFC 6356. The algorithm extends standard TCP congestion control for multipath operation. The algorithm in RFC 6356 only changes the increase phase of the congestion avoidance state when the endpoint receives an ACK. The remaining phases such as the slow start, fast retransmit and fast recovery are the same as in traditional TCP [5]. Formula (1) is the increase value when the flow gets an ACK, which is proposed by IETF in RFC 6356. Let $cwnd\_i$ be the congestion window on the subflow i. Let $cwnd\_total$ be the sum of the congestion windows of all subflows in the connection. Let MSS_i and rtt_i be the maximum segment size on subflow i. For each ACK received on subflow i, increase $cwnd\_i$ as formula (1). The first argument is the computed increase for the subflow, and the second argument is the increase TCP would get in the same scene. The formula takes the minimum between the computed increase for the subflow and the increase TCP would get in the same scene. In this way, Any subflow of MPTCP cannot be more aggressive than a TCP flow in the same scenario, and principle 2 described in Section 2 has been achieved. ∂ is a parameter of the algorithm that describes the aggressiveness of the multipath flow.. The total throughput of all flows in MPTCP depends on the value of ∂ and the loss rates, maximum segment sizes, and RTT of its paths. Since the total throughput is no worse than the throughput a single TCP would get on the best path is required, It's impossible to choose a priori a single value of ∂ that achieves the desired throughput in every occasion. So ∂ must be computed based on the observed properties of the paths. ∂ can be calculate by formula (2) in RFC 6356, The value of ∂ is chosen such that the aggregate throughput of the MPTCP is equal to the throughput a TCP flow would get if it ran on the best path, and to meet principle 1 described in Section 2.

$$\min\left(\frac{\partial * bytes\_acked * MSS\_i}{cwnd\_total}, \frac{bytes\_acked * MSS\_\mathrm{i}}{cwnd\_i}\right) \tag{1}$$

$$\partial = cwnd\_total * \frac{MAX\left(\dfrac{cwnd\_i}{rtt\_i^2}\right)}{\left(SUM\left(\dfrac{cwnd\_i}{rtt\_i}\right)\right)^2} \tag{2}$$

$MAX\,(x\_i)$ be the maximum among all the subflow.

$SUM\,(x\_i)$ be the summation of all the subflow.

The formula (2) is derived by equalizing the rate of the flows in MPTCP with the rate of a TCP running on the best path. In general cases this method can satisfy the fairness in share bottlenecks in the network, and get the throughput as well as traditional TCP in the best path. But it does not satisfy to all scenes. Take the scene shown in Figure 4 for example, this method will move more than half of the traffic on the 3G path, and cannot get throughput as well as traditional TCP in the best path. The analysis is as follow:

Assume all subflows have the same round-trip time and are in the congestion stage. We can get formula (3) and $\partial < 1$ from formula (2). Thus, the congestion window with increasing value is $\frac{\partial * bytes\_acked * MSS\_i}{cwnd\_total}$ by each ACK.

$$\partial = \frac{MAX\left(cwnd\_i\right)}{cwnd\_total} \tag{3}$$

To get a feeling for the behavior of this algorithm, we assume a MPTCP flow has only two links, Let p_1 and p_2 be the loss rate. Let R_1 and R_2 be the round-trip time. Let W_1 and W_2 be the congestion windows. We also assume $W_1 > W_2$ and all subflows have the same maximum segment size. From formula (3) we can get the congress window increasing with value is $\frac{W_1 * bytes\_acked * MSS}{\left(W_1 + W_2\right)^2}$.

As [3] proposed that each window is made to increase $1/w$ by ACKs, and decrease $w/2$ by drops. To keep in equilibrium the increases and decreases must balance out and seen in formula (4). And we can get our balance formula (5) and (6).

$$\left(\frac{w}{RTT}(1-\mathrm{p})\right)\frac{1}{w} = \left(\frac{w}{RTT}p\right)\frac{w}{2} \tag{4}$$

$$\left(\frac{W_1}{R_1}\left(1-p_1\right)\right) * \frac{W_1 * bytes\_acked * MSS}{\left(W_1 + W_2\right)^2} = \left(\frac{W_1}{R_1} * p_1\right)\frac{W_1}{2} \tag{5}$$

$$\left(\frac{W_2}{R_2}\left(1-p_2\right)\right) * \frac{W_1 * bytes\_acked * MSS}{\left(W_1 + W_2\right)^2} = \left(\frac{W_2}{R_2} * p_2\right)\frac{W_2}{2} \tag{6}$$

From formula (5) and (6), we can deduce equation (7). If $W_1 > W_2$, we can get $p_1 < p_2$.

$$p_1 * W_1 = p_2 * W_2 \tag{7}$$

According to the above formula derivation, we can get the conclusion that the congestion control algorithm in RFC 6356 tends to select subflow link with low loss rate instead of subflow link with high throughput. According to the above formula (4), we can get the formula to calculate the throughput of link with loss rate p as follows: $w = \sqrt{2(1-\mathrm{p})/p} \approx \sqrt{2/p}$, The parameter w means the throughput of link, and the parameter p means loss rate of the link. We can calculate the throughput of link in the Figure 4 according to this formula. The throughput of single traditional TCP in WIFI link with loss rate 4% is 126.5kbps and the throughput of single traditional TCP in 3G link with loss rate 1% is 200kbps.

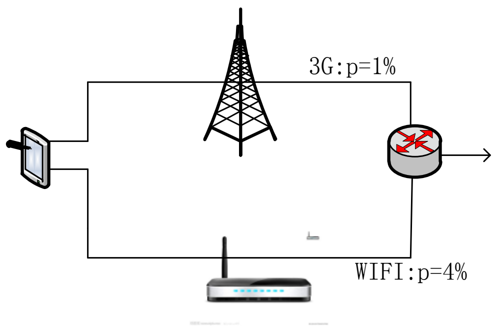


Figure 4. Low throughput of the network environment

4. Dynamic Weight Algorithm

Dynamic weight algorithm is the improvement of the EWTCP algorithm. EWTCP algorithm is made to increase a/w_r by each ACK and decrease $w_r/2$ by drops. Reference [3] points out that this algorithm guarantees the fairness between MPTCP and traditional TCP and can obtain the throughput of the traditional TCP in the best path. Meantime this algorithm can allocate traffic by the throughput of the subflow, but in some special network environment, EWTCP cannot obtain the throughput of the tradition TCP in best path.

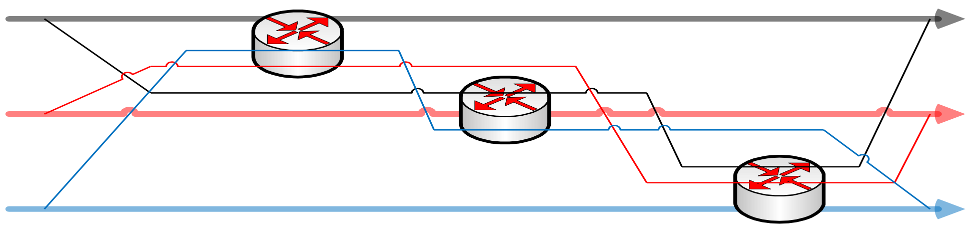


Figure 5. Ineffective usage of the network resource

In Figure 5, it supposed that the three links have capacity of 12Mb/s. When each flow splits its traffic evenly, then each subflow will get 4Mb/s, hence the throughput of the subflow will be 8Mb/s. But if each flow uses only the one-hop shortest path, it could get 12Mb/s [3]. EWTCP algorithm cannot work well in this situation.

In order to improve EWTCP in this situation, we propose dynamic weight algorithm. When the subflow receives an ACK, dynamic weight algorithm will change the weights according to $cwnd\_i/cwnd\_total$ .Reference [4] and [5] models throughput of TCP T as a function of RTT.

$$T = \frac{s}{R\sqrt{\frac{2p}{3a}} + t\left(3\sqrt{\frac{3p}{8a}}\right)p\left(1+32p^2\right)} \qquad (8)$$

Where T is a function of RTT; t is the retransmission timeout value; s is packet size; p is packet loss rate. If a=1, we can get the throughput of traditional TCP.

$$T_{tcp} = \frac{s}{R\sqrt{\frac{2p}{3}} + t\left(3\sqrt{\frac{3p}{8}}\right)p\left(1+32p^2\right)} \tag{9}$$

According to the fairness rules of the MPTCP, we can get the increasing weights by the congestion window. We assume that the throughput of the subflow i should be D_i times of the traditional TCP and $\sum D_i = 1$. According to formula (9), we can get the throughput T_i of subflow i.

$$T_i = D_i \frac{s}{R\sqrt{\frac{2p}{3}} + t\left(3\sqrt{\frac{3p}{8}}\right)p\left(1+32P^2\right)} \tag{10}$$

Reference [4] points out that $a = D_i^2$, so we can cumulate the increasing weight by the size of the congestion window. As a result the greater size of the congestion window provided the greater contribution to the throughput of the MPTCP. Thus we can get the equation (11) and communicate the weight when increasing the congestion window.

$$w_1 : w_2 : ... : w_i = D_1 : D_2 : ... : D_i \tag{11}$$

In Figure 5 dynamic weight algorithm can improve the throughput of MPTCP. Let w_i be the window size of i. When each flow splits its traffic evenly across its two paths, the flow through more nodes will drop more times than the flow though fewer nodes, so we can get $w_1 > w_2$. According to the dynamic weight algorithm, the flow with only one node will obtain larger weight, and then improve the increasing speed of the congestion window. In the long-term stability, MPTCP will focus their traffic on the link which passes through only one node and the MPTCP will obtain 12Mb/s. Dynamic weights can allocate the traffic more fair when subflows share a bottleneck.

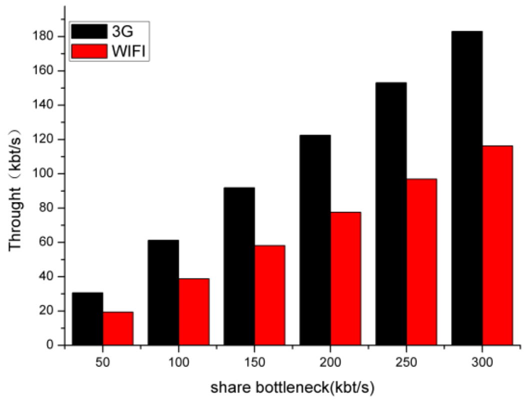


Figure 6. Subflows throughput using RFC6356 algorithm

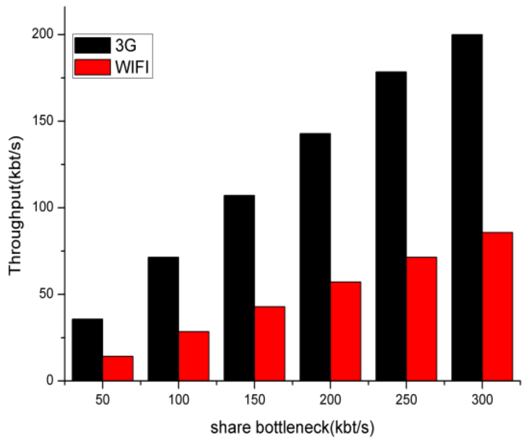


Figure 7. Subflows throughput using dynamic weight algorithm

5. Simulation

We simulate the network environment showed in Figure 4. The simulated system is composed of a FTP application, to transfer a big file, running on Client/Server architecture where the two hosts are linked by two links. The simulation verifies the dynamic weigh algorithm from the fairness of subflows and the throughput of the MPTCP. In Figure 4, there are two links which have a share bottleneck A. By changing the throughput of the share bottleneck, we can detect the performance of the dynamic weight algorithm in subflow fairness. In the simulation, the throughput of the share bottleneck will be 50kb/s，100kb/s，150kb/s，200kb/s，250kb/s，300kb/s. In Figure 4, the throughput of the 3G path and Wi-Fi path will be 200kb/s and 126.5kb/s when they do not have the share bottleneck. According to the Table 1, MPTCP will allocate the throughput of the share bottleneck as 200:126.5, but the algorithm in RFC 6356 cannot work well.

In Figure 6, the congestion control algorithm in RFC 6356 allocate more traffic on the 3G path, it will cause the unfairness of the subflow. When the throughput of the share bottleneck bigger than 300kb/s, the algorithm in RFC 6356 allocate 200kb/s and 87kb/s on the 3G and Wi-Fi path, so algorithm in RFC 6356 cannot improve the network resource utilization. In Figure 7, the result of the dynamic weight algorithm will allocate the traffic to subflows as the throughput of subflows, and it can obtain higher throughput when the limitation of the share bottleneck is 300kb/s. Figure 8 shows the behavior of the congestion window in case the algorithm in RFC 6356 is used when the limitation of the share bottleneck is 300kb/s. Figure 9 shows the behavior of the congestion window in case the dynamic weight algorithm is used when the limitation of the share bottleneck is 300kb/s. The result of the simulation proves the correctness of the dynamic weight algorithm.

Table 1. Ration of 3G and WI-FI

Algorithm / Throughput	Tradition TCP	RFC 6356	Dynamic weight
3G:Wi-Fi	100:63	100:40	100:60

Copyright ⓒ 2014 SERSC

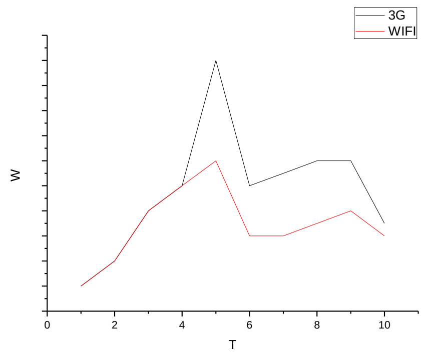
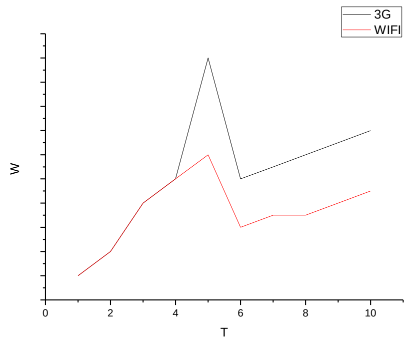


Figure 8. Congestion window evolution using RFC6356



‘

Figure 9. Congestion window evolution used dynamic weight algorithm

6. Conclusion

In this paper, a dynamic weight congestion control algorithm is proposed. And this algorithm can work better than the algorithm in RFC6356 when subflows have share bottleneck. According to the result of the simulation and the proof, the dynamic weight congestion control algorithm has advantages as flows:

1. The dynamic weight congestion control algorithm can allocate traffic to subflows by the throughput of subflows.

2. The dynamic weight congestion control can obtain higher throughput in some limitation of the share bottleneck.

Acknowledgements

This work was supported by the Fundamental Research Funds for the Central Universities 2012RC207.

References

[1] M. Xue and Z. Zhang, “MPTCP joint Markov Model for congestion control mechanisms”, Journal of Tsinghua University, Natural Science, vol. 52, no. 9, (**2012**), pp. 1281-1285.
[2] Raiciu, C. M. Handley and D. Wischik, “Coupled congestion control for multipath transport protocols”, draft-ietf-mptcp-congestion-01 (work in progress), (**2011**).

[3] D. Wischik and C. Raiciu, A. Greenhalgh and M. Handley, “Design, implementation and evaluation of congestion control for multipath TCP”, Proceedings of the 8th USENIX conference on Networked systems design and implementation, (**2011**).
[4] M. Honda, Y. Nishida, L.Eggert, P. Sarolahti and H. Tokuda, “Multipath congestion control for shared bottleneck”, In Proceedings of PFLDNeT workshop, (**2009**).
[5] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, “Modeling TCP throughput: A simple model and its empirical validation”, In ACM SIGCOMM Computer Communication Review, vol. 28, no. 4, (**1998**) October, pp. 303-314.
[6] V. Paxson and M. Allman, “Computing TCP’s retransmission timer”, IETF RFC 2988, (**2000**) November.
[7] A. Ford, C. Raiciu, M. Handley, S. Barre and J. Iyengar, “Architectural Guidelines for Multipath TCP Development”, IETF RFC 6182, (**2011**) March.
[8] P. Eardley, “Survey of MPTCP Implementations”, IETF blank version for implementers to fill in”, (**2013**) March.

Copyright ⓒ 2014 SERSC