International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
International Conference on Industrial Automation and Computing (ICIAC- 12-13th April 2014)

RESEARCH ARTICLE OPEN ACCESS

Analysis on MPTCP combining Congestion Window Adaptation and Packet Scheduling for Multi-Homed Device

Mr. Prathmesh A. Bhat, Prof. Girish Talmale
Dept. of Computer Science and Technology , GHRCE Nagpur University, Nagpur
Email: prathmabhat28@gmail.com, girish.talmale@raisoni.

ABSTRACT
Evolution of use of wireless technologies in laptops and mobile terminals, which are equipped with several network interfaces, has provided users to take advantage from multi-homing to access network services anywhere, at any time and from any network. Advantage with multihomed host is that some of the traffic from more congested paths can be shifted to less congested path, thus controls congestion. In this paper we consider about Multipath TCP (MPTCP), which suffers from the degradation of goodput in the presence of changing network conditions on the available subflows due to out-of-order received packets. Cause of degradation is the large variation of end-to-end delay for multiple paths over wireless channels. To diminish the variation of end-to-end path delay, the proposed scheme uses congestion window adaption (CWA) algorithm to employ MPTCP source. Also to reduce the time of packet reordering at the receiver, a scheduling algorithm is employed for the MPTCP sender. Experiments are conducted to evaluate the goodput performance of the two enhancements to MPTCP. Significant performance gain is achieved in terms of good put, while the reordering time is minimized.
Keywords **-** Goodput, Congestion, Reordering, MPTCP

I. INTRODUCTION

Modern laptops have often found more than one network interface for accessing the Internet. Also in the case with mobile, there are more than one network interface. A mobile user access the Internet via a wireless wide area network such as general packet radio service [GPRS].Such laptops and mobiles are referred as "Multi-Homed devices". Today's processor are has fast enough to handle data transfer on multiple network interface simultaneously. This provides a good prospect to explore several interfaces for multipath transmission, so as to aggregate the bandwidth among multiple wireless links and further improve the quality of service (QoS) for bandwidth-intensive applications, such as video streaming and video conference. The standard for the transport layer is the Transport Control Protocol [TCP]. TCP however, fails to transmit packet over multiple paths for Multi-Homed Device due to the high level of out-of-order packets. In conventional TCP, such as TCP Reno and selective acknowledgment (SACK), the source node decreases its congestion window once three duplicate acknowledgments (ACK) are received from the sink node. That is, three duplicate ACKs are viewed as an indicator of packet loss in transmission. In a multipath transmission scenario, because the round-trip time (RTT) of each path varies, there is a high probability that packets with lower sequence numbers sent over a slower path arrive at the sink later than packets with higher sequence numbers sent over a faster path. As a result, the sink node receives out-of-order packets and then returns duplicate ACKs, which is misinterpreted by the source as packet loss. Then, the source reduces its congestion window and enters fast retransmit and recovery stage. This behavior puts the efficiency of TCP transmission in danger because the sending window can be mistakenly set to a small value [1]

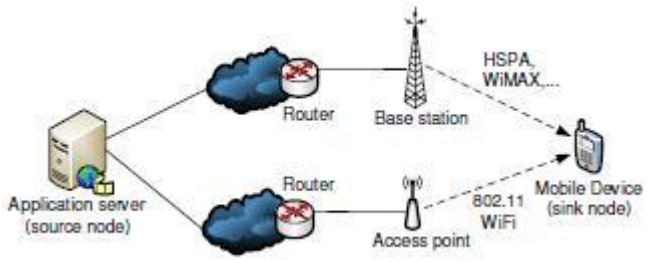


Fig. 1 Multi-home scenario in wireless network

Fig. 1 shows a multi-home scenario where a Mobile device is connected to both Base Station and Access point via its multiple interfaces. MPTCP works well for multi-homed mobile devices to simultaneously deliver TCP packets over multiple paths and pool the available bandwidth together. Although MPTCP has a better available throughput for the upper layer, there

is still another unresolved issue caused by out-of-orders packets. Throughput represents the overall receiving capacity of successful packet delivery over multiple paths. Nonetheless, it is goodput that reflects the real application-level throughput, which is the amount of useful data available to the receiver application per time unit. Specifically, in-order packets received at the transport layer can be forwarded to the application layer and counted for goodput. Most recent study [7] introduced CWA with a proactive scheduler for wired communication. This study show that MPTCP goodput is near optimal when the end-to end delays of two transmission paths are very close. However these study show that it takes a lot of time to reorder packets at receiving end. Some more recent work in 2012 tries to improve goodput for MPTCP, by using network coding [2] and packet retransmission over fast path [3]. However, these studies only show the average goodput improvement over a long term. In fact, stable goodput with minimal variation is preferable for QoS assurance to real-time applications. Author in [6] has studied different congestion control variants for Multipath TCP have been compared. Also author has
to balance the traffic load on each path and improve throughput without exposing regular TCP users.

MPTCP sublayer is responsible for coordinating data packets on multiple paths, such as reordering packets received from each path at the sink, scheduling packets toward each path at the source, and balancing the congestion window of each subflow TCP. MPTCP also look after packet reordering for multiple paths. Since each TCP subflow maintains an independent sequence number space, the sink may receive two packets of the same sequence number. Further, packets received at the sink can be out-of-order because of mismatched round-trip time (RTT) of multiple paths. Therefore, the source needs to address the sink about the reassembly of the data forwarded to the application. MPTCP solves this problem by using two levels of sequence numbers. First, the sequence number for TCP subflow is referred to as *subflow sequence number* (SSN), which is similar to the one in regular TCP. The subflow sequence number independently works within each subflow and ensures that data packets of each subflow are successfully transmitted to the sink in order.
investigated a couple of relevant hybrid scheduler algorithms that are based on the two implementation strategies, Push and Pull.

In this paper, we use CWA-MPTCP, in which the MPTCP source dynamically adjusts the congestion window of each TCP subflow so as to maintain similar end-to-end delays over multiple paths, and packet scheduling algorithm, which reduces time required at to rearrange packets at receiving end.

II. OVERVIEW OF MPTCP

MPTCP is an extension to TCP that allows the concurrent data transmission. From the performance perspectives, MPTCP has two main objectives:

a) Improve the throughput by combining bandwidth over multiple available paths.
b) Improve the reliability by providing multiple paths and switching traffic upon path failure.

As shown in Fig. 2, MPTCP roughly divides the transport layer into two sublayers, specifically, MPTCP and subflow TCP. Based on this architecture, MPTCP can be easily employed within current network stack. Each path has its subflow to reuse most function of regular TCP. The key transformation between subflow TCP and regular TCP is that congestion control on each path is assigned to MPTCP sublayer [5]. Although each subflow TCP maintains a congestion window at the source, the congestion window is updated by a coupled congestion control algorithm which aims

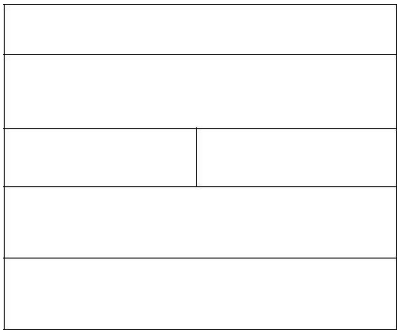


Fig. 2 Network protocol stack with MPTCP

The sequence number at the MPTCP level is called *data sequence number* (DSN). Each packet received at the sink has a unique DSN no matter which path it is sent over. Hence, the sink can easily sequence and reassemble packets from different paths by DSN.

III. GOODPUT IMPROVEMENT FOR MPTCP

A. Problem Analysis

In this work, we give special attention on important performance metric, i.e., *goodput*. The goodput of MPTCP is defined as the data throughput of inorder packets forwarded by MPTCP to the application layer. Intuitively, we have,

Goodput = Size of N in-order packets (1) Total receiving time of N packets

Next, to find out reasons for poor goodput performance, consider two special scenarios of MPTCP.

Suppose that there are two available paths. Let Γ_i denote the packet sending interval at the MPTCP source for path i, i = 1, 2. Consider that the throughput of path 2 is smaller than that of path 1. Denoting the end-to-end delay of path i by d_i, we have $d_1 < d_2$. Consider a block of N packets with continuous DSN numbers, among which N − 1 packets are received on path 1 and only 1 packet is from path 2. Such a block of data packets is referred to as an *in-order unit*. Let S and T denote the total size in the unit of maximum segment size (MSS) and the total receiving time of an in-order unit, respectively. Then, we can evaluate the goodput by G = S/T.

Consider two special cases illustrated in Fig. 3. The in order unit comprises 4 packets of DSN numbers 1, 2, 3, and 4.Suppose that packet 1 and packet 2 are sent at the same time to path 1 and path 2, respectively. Fig. 3(a) shows the case with D, $|d_2 - d_1| > \Gamma_1$. We can easily obtain T = D and the goodput, given by

$$G = \frac{S}{T} = \frac{\Gamma_2/\Gamma_1 + 1}{\Delta D} \quad (2)$$

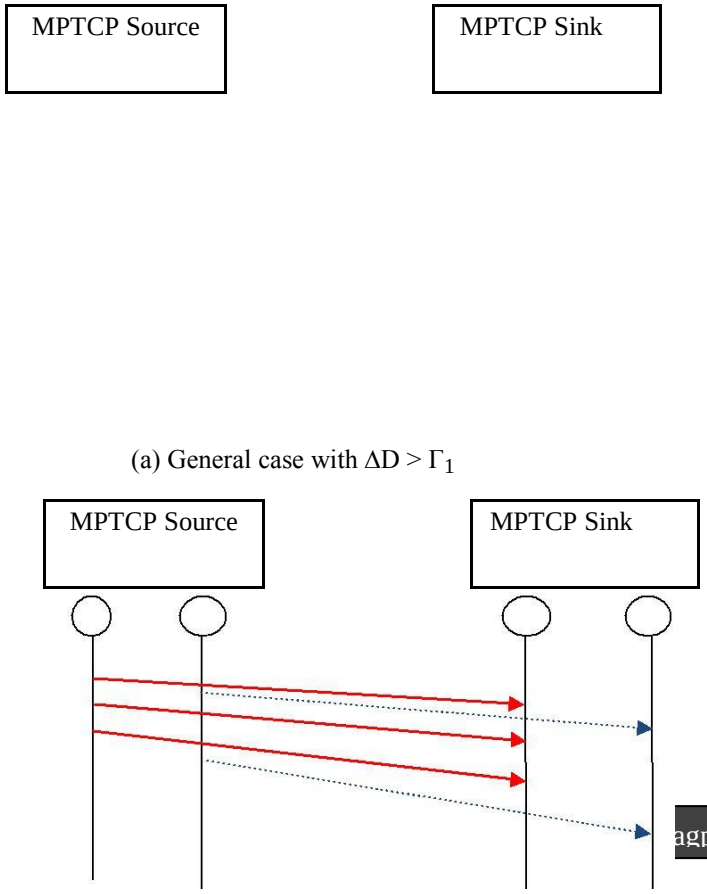


Fig. 3 Special cases with two transmission path for goodput analysis

Fig. 3(b) shows another special scenario with D ≤ Γ_1. In this case, the MPTCP sink needs less time to receive all packets within the in-order unit. Here, the total time to receive all N packets of the in-order unit is just the time for path 1 to receive all N-1 packets re-sent over it. Obviously,

$$G = \frac{S}{T} = \frac{\Gamma_2/\Gamma_1 + 1}{\Gamma_2} \quad (3)$$

Actually, Eq. (3) is also the aggregate throughput (denoted by γ) over two paths. That is,

$$\Gamma = \frac{1}{\Gamma_2} + \frac{1}{\Gamma_1} \quad (4)$$

This observation implies that goodput is inversely proportional to the end-to-end path delay difference ΔD.

B. Congestion Window Adaptation

In conventional TCP, the TCP sender maintains a congestion window to control the maximum amount of packets to send at a time. The indication for packet loss is either Timeout or triple duplicate ACKs received from receiver. The source node reacts on packet loss and reduces its congestion window to bring the traffic load to stability. In MPTCP, each TCP subflow maintains its own congestion window and triggers a decrease of the congestion window by receiving duplicate ACKs. In contrast, the increase of the congestion windows of all subflows is controlled by a coupled algorithm [4] at the MPTCP flow level. This congestion window control algorithm can combine the available bandwidth of each path and prevent a MPTCP source from taking up too much resource to assure TCP friendliness. In this congestion control algorithm, the only reason to decrease the congestion window is packet loss indicate d by duplicate ACKs. Consequently, the congestion window of each path may greatly differ from each other and lead to a large path delay difference, which is harmful to the goodput performance.

Algorithm: Congestion Window Adaptation.

1. **if** $\theta_{min} \leq \theta \leq \theta_{max}$ **then** //High delay ratio detected
2. i= arg max_p (end-to-end delay of path p)
3. m=max adaptation limit
4. **if** $count_i < m$ **then**
5. $cwnd_i$ $cwnd_i/\theta$
6. **if** $ssthresh_i > cwnd_i$ **then**
7. $ssthresh_i = cwnd_i$
8. **end if**
9. $count_i$ $count_i + 1$
10. **else**
11. $count_i = 0$
12. **end if**
13. **end if**

The algorithm monitors the end-to-end delays of multiple paths. Whenever large delay ratio is detected, congestion window adaptation takes place at source compared with regular TCP, where adaptation takes place only when source receives three duplicate ACK. Here, delay ratio refers to ratio of maximum path delay over minimum path delay. The objective is to decrease the delay ratio in order to increase goodput.

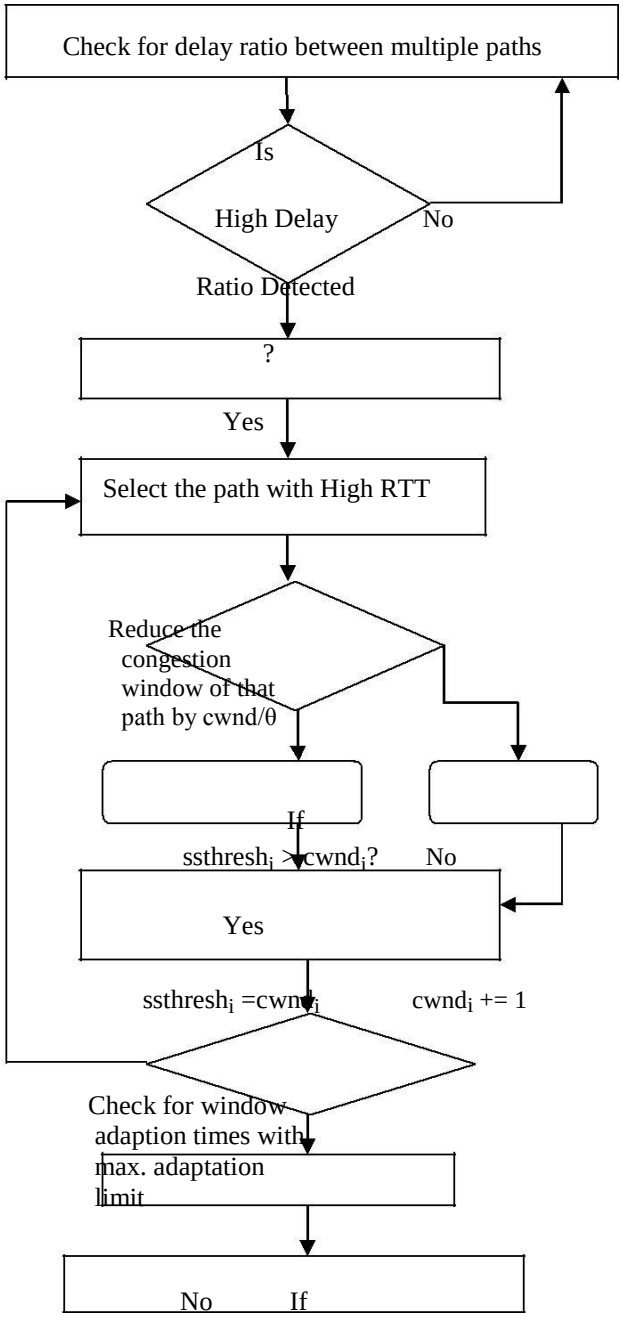


Fig. 4 Flowchart for Congestion Window Adaptation

Delay ratio range is from θ_{min} to θ_{max}. Whenever delay ratio θ falls in the range, congestion window adaptation takes place. Assume path i has maximum delay, so congestion window ($cwnd_i$) is decreased proportionally to the delay ratio. This is done because a larger delay ratio indicates that the high delay path is congested. Its congestion window needs to be decreased to release traffic and reduce path delay. Here, θ_{max} is presented to avoid over-blocking slow path and severely risking aggregate throughput. Meanwhile, if $ssthresh_i > cwnd_i$ then the TCP slow start threshold ($ssthresh_i$) is updated with the new $cwnd_i$. Otherwise, $cwnd_i$ will be recovered quickly with the slow start procedure (i.e. $cwnd_i$ is linearly increased by 1 for each successful ACK received at the source). As a consequence, it would be hard to guarantee that the congestion window of the slow path is decreased for sufficient time to reduce the end-to-end delay.

The above procedure alone cannot reduce the end-to-end delay variation of multipath variation. This is because there are other sources affecting end-to-end delay. The sources causing problem are transmission, processing, and queuing delays at routers, base stations, and intermediate nodes between communication peers. The path delay variation can be reduced by decreasing the congestion window of the slow path and relieving its carried traffic load. Since the transport-layer control itself cannot completely eliminate the path delay variation, the parameter $count_i$ is used to restrict the number of continuous reductions of congestion window for a single path i by m, which is the maximum adaptation limit.

After the cwnd of a high-delay path is reduced according to Algorithm, the corresponding TCP subflow is blocked from sending more packets, because of the gap between the original cwnd and the adapted new one, i.e., $(cwnd_i - cwnd_i/\theta)$. The TCP subflow is blocked since the highest acknowledged DSN plus the adapted smaller cwnd becomes less than the highest DSN of packets that are sent to the sink node. This subflow is then blocked for a period T, given by

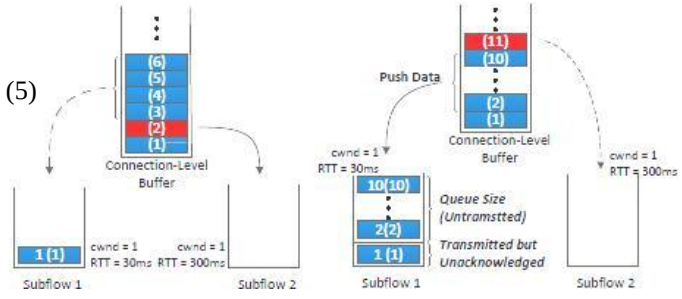
$$\Delta T = (cwnd_i - cwnd_i/\theta) * \Gamma_i. \quad (5)$$

For instance, when $1 \leq \theta \leq 3$, $\Gamma_i = 5$ ms, and $cwnd_i = 100$ packets, ΔT ranges from 170 ms to 340 ms. During this short period, although one slow path is blocked and the overall throughput slightly decreases, more significant performance gain is achieved for goodput.

C. Scheduling Algorithm

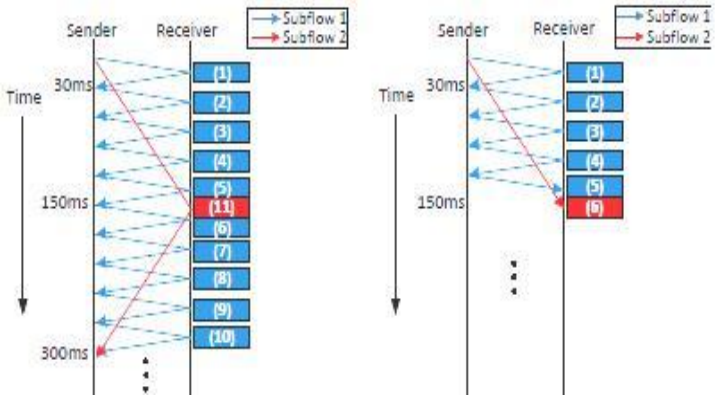
The key design objective for a multipath solution is that it should be able to give a good performance under various network constraints of dissimilar subflows. Therefore, the scheduler, which performs the distribution of the individual packets of an application flow over several available subflows, is a critical design issue for efficient operation of multipath TCP. As Multipath TCP makes use of several paths between two endpoints to transmit data simultaneously, an efficient multipath scheduler is required at the sender. The scheduler should specify the order in which the new data is scheduled on the different flows of an MPTCP connection. The scheduling decision is done based on several variables such as the capacity of the subflow, the delay on the subflow, queue size at the sender or buffer size of a subflow.

The best approach for an MPTCP scheduler is based on hybrid strategy using both push and pull strategy [11]. This strategy operates efficiently by allocating data segments to active flows with dynamic size. The authors had recognized in [11] that the Push strategy based on the Delivery Delay of the data segment earns the best performance. In this work, the Hybrid Delivery Delay scheduler is presented and compared with the Hybrid Acknowledgement (ACK) Delay scheduler as well as the basic Pull strategy based scheduler. The operation of the different schedulers is showed with the help of Figures 5 and 6 where it is assumed that the one path has 10 times the round trip time (RTT) when compared to the other. The Pull scheduler simply allocates segments as soon as an acknowledgement arrives and hence the cwnd is open to transmit new data segments, refer Figure 5(a). On the other hand, the Hybrid Acknowledgement Delay scheduler aims at allocating data segments in an ordered way based on the expected acknowledgment over the two paths.



(a) Pull Strategy Scheduler (b) Hybrid Ack

Delay Scheduler Fig. 5 Pull and Hybrid Ack Delay Scheduler



(a)Hybrid Ack Delay Scheduler (b) Hybrid Delivery Delay

Scheduler

As showed in Figure 5(b), due to an RTT ratio of 10 between the two paths, the data segments that would have been sent in the 11th RTT slot (shown in red color) are scheduled on the path with higher RTT so that its acknowledgment arrives near to the acknowledgment of the data segments that are scheduled on the lower RTT path in the 10th RTT slot, as shown in Figure 6(a). It is also clear from Figure 6(a) that this strategy will lead to a reordering delay for the data segments transmitted on the path with higher RTT as it arrives earlier at the receiver than the other data segments that are still queued at the lower RTT path. The scheduler variant that aims at removing the reordering delay at the receiver will have to follow the trend presented in Figure 3b i.e., the Hybrid Delivery Delay scheduler. Thus this scheduler will reduce the packet reordering time at the receiver.

IV. Results and Discussion

To assess the performance of the proposed scheme, we extend MPTCP in NS-2 with congestion window adaptation and packet scheduler. Performance is access on the basic of goodput and reordering time at receiver for wireless scenario. In Wireless scenario, the multi-radio receiver is equipped with multiple interfaces and connected to a base station over wireless links. The detailed system parameters are given in table1.

Table 1

Parameter	**Sample Value**
Number of Transmission Path	2
Avg. Bandwidth on path 1	8Mbit/s
Avg. Bandwidth on path 2	2Mbit/s
Application Used	FTP
Min Delay Ratio θ_{min}	1
Max Delay Ratio θ_{max}	3

Fig. 6 Message Sequence Diagram

First we compare the goodput when regular MPTCP and CWA-MPTCP are used. The goodput of MPTCP combining CWA and packet scheduler is better when compared with regular MPTCP. The goodput of CWA-MPTCP is consistent over a period of time whereas in original MPTCP, goodput rises in some part and falls in other. This is due to the large delay of the slow path in some periods introduce out-of-order packets. In CWA-MPTCP, the end-to-end delay difference between the two paths is reduced to some extend and thus improved and consistent goodput.

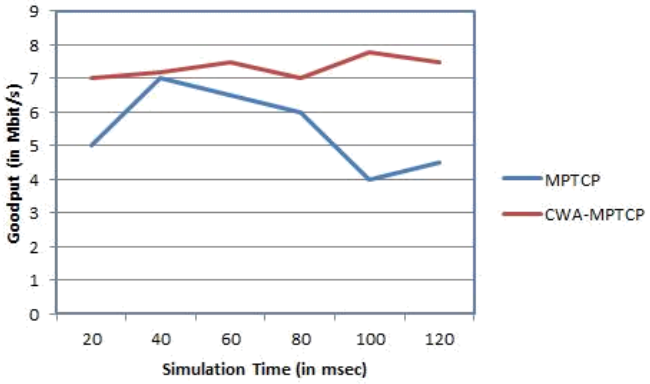


Fig. 7 compares the goodput when original MPTCP and CWA-MPTCP are used.

Fig. 7 Compared Goodput of MPTCP and CWA-MPTCP After goodput, reordering time is considered for analysis. Packet Scheduling used in the paper is Push strategy based on the Delivery Delay of the data segment. This scheduler based on the acknowledgement delay has a very low reordering delay. For analysis, the Delivery Delay scheduler is compared with the scheduler based on the Pull Strategy. The

pull strategy scheduler have high reordering time since the segments on the lower delay path have to wait, but this time decreases with each segment as the latter segments are transmitted later in time. Fig. 8 compares both these schedulers.

Fig. 8 Comparison of Pull and Delivery Delay Scheduler

The results show that Delivery Delay scheduler has less reordering time when compared to pull scheduler.

Simulation results show that MPTCP combining both CWA and packet scheduler may give better goodput than the previous work. Goodput may approach the upper bound of aggregate throughput.

V. Conclusion

In this paper, we combined a congestion window adaptation algorithm (CWA-MPTCP) and packet scheduling technique to enhance the goodput of MPTCP and decrease the receive buffer requirement for the sink node. The adaptation takes place only when high delay ratio is detected. By reducing delay ratio, high goodput can be achieved for multipath transmission over wireless links. The scheduling at the sender side helps to reduce reordering time at receiver end.

Simulation results demonstrate that our solutions achieve stable goodput with significant improvement and reduced reordering time requirement for the sink node.

[12] IEEE Globecom 2012

REFERENCES

[1] M. Zhang, B. Karp, S. Floyd, and L. Perterson, "RR-TCP: A re ordering robust TCP with DSACK," in *Proc. IEEE ICNP*, Nov. 2003.

[2] M. Li, A. Lukyanenko, and Y. Cui, "Network coding based multipath TCP," in *Proc. IEEE INFOCOM Computer Communication Workshop*, Mar. 2012.

[3] C. Raiciu, C. Paasch, S. Barre, and A. Ford, "Designing and implementing a deployable multipath TCP," in *Proc. USENIX NSDI*, Apr. 2012.

[4] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," IETF RFC 6356, Oct. 2011.

[5] Sébastien Barré, Christoph Paasch, and Olivier Bonaventure "MultiPath TCP: From Theory to Practice"

[6] Amanpreet Singh, Mei Xiang, Andreas Konsgen and Carmelita Goerg, Yasir Zaki "Enhancing Fairness and Congestion Control in Multipath TCP" in WMNC,2013

[7] Dizhi Zhou, Wei Song, Minghui Shi "Goodput Improvement for Multipath TCP by Congestion Window Adaptation in Multi-Radio Devices" in IEEE CCNC-Wireless Networking Track 2013.

[8] D. Zhou, P. Ju, and W. Song, "Performance enhancement of multipath TCP with cooperative relays in a collaborative community," in *Proc. IEEE PIMRC*, Sep. 2012.

[9] Y. Cui, X. Wang, H. Wang, G. Pan, and Y. Wang, "FMTCP: A fountain code-based multipath transmission control protocol," in *Proc. IEEE ICDCS*, Jun. 2012.

[10] Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath TCP development," IETF RFC 6182, Mar. 2011.

[11] Amanpreet Singh, Carmelita Goerg Andreas Timm-Giel Michael Scharf,Thomas-Rolf Banniza "Performance Comparison of Scheduling Algorithms for Multipath Transfer," in