Alheid, A. A. Y., Kaleshi, D., & Doufexi, A. (2014). An Analysis of the Impact of Out-Of-Order Recovery Algorithms on MPTCP Throughput. In 2014 IEEE 28th International Conference on Advanced Information Networking and Applications (AINA 2014): Proceedings of a meeting held 13-16 May 2014, Victoria, British Columbia, Canada. (pp. 156-163). [6838660] (Proceedings of the IEEE International Conference on Advanced Information Networking and Applications (AINA)). Institute of Electrical and Electronics Engineers (IEEE). DOI: 10.1109/AINA.2014.50

An Analysis of the Impact of Out-Of-Order Recovery Algorithms on MPTCP Throughput

Amani Alheid, Dritan Kaleshi, Angela Doufexi

Department of Electrical and Electronic Engineering,
University of Bristol,

Bristol, UK

{amani.alheid, dritan.kaleshi, a.doufexi}@bristol.ac.uk

***Abstract*—In this paper we evaluate and compare the end-to-end performance of different multipath TCP (MPTCP) congestion controllers when run in conjunction with different TCP packet reordering recovery algorithms. The paper answers the following questions: what is the impact of out-of-order events on the end-to-end throughput when using MPTCP, how do out-of-order recovery algorithms that have been proposed for single-path TCP perform with multi-path TCP, and how sensitive this performance is against the delay difference between the paths used. The paper compares three different MPTCP congestion control algorithms used in conjunction with four current TCP packet reordering solutions: D-SACK, Eifel, TCP-DOOR, and F-RTO. Simulation results show that whilst TCP-DOOR and, second, D-SACK perform generally better across all congestion control MPTCP implementations, the choice of packet reordering algorithm is not always fixed and straightforward – when MPTCP is used with some form of coupled congestion control the performance degrades towards that of single-path usage when the delay difference of the paths is over 200ms. The paper identifies combinations of congestion control and packet reordering algorithms that give better aggregate throughput performance for different path delay differences.**

I. INTRODUCTION

The success of the Internet comes from its capability to provide a wide range of robust and reliable end-to-end data transmission services for various applications, such as email, file sharing and media streaming. On the other hand, demand for network efficiency and performance, robustness and reliability is becoming essential. In particular, using more than one network path for data transmission simultaneously can increase the reliability of the connection as well as the end-to-end throughput.

Most of the portable devices today have more than one interface, allowing the users to use different wired/wireless access technologies (e.g. xDSL, Ethernet, WiFi, Cellular/Cellular LTE). Using available access technology links simultaneously with the ability to shift traffic from congested paths to uncongested paths are the basic goals for any new proposed protocol in this area. Many solutions have been proposed for multipath transmission from a transport layer perspective that based either on TCP or SCTP. Different techniques are used with these protocols in order to improve throughput compared to single-path transports. Some of them depend on bandwidth aggregation techniques [1]-[3], concurrent transmissions [4]-[6], and connection delay [7].

Multipath TCP (MPTCP) is a modification of the classical TCP that allows end-to-end data traffic to be split across multiple paths, whilst maintaining TCP connections at the end points (applications) [1]. The design objectives of MPTCP are to support unmodified applications that use TCP (MPTCP operation at transport layer is hidden to other layers), work over current networks and work whenever TCP would work.

Several studies have proposed and analysed the performance of MPTCP congestion control (CC) algorithms against its goals [8], [9]. They differ in terms of transmission robustness, fairness and path selection stability (flapping), and are discussed more extensively in Section II-B. They all report improvements in throughput measurements compared to the standard TCP. However, sending data through different paths increases the possibility that the order of the packets received at the destination is different from that of the sender (out-of-order (OOO) events). There exist many causes for packet reordering for a single connection; packet-level multipath routing, route fluttering, inherent parallelism in modern high speed routers, link layer retransmission, and router forwarding lulls [10]. Consequently, when one end-to-end data connection uses more than one path in its transmission then the diversity of path characteristics (both loss and delay), will create OOO events even when the single paths behave ideally.

In this paper we evaluate and compare the behaviour of different MPTCP congestion controllers in combination with different packet reordering (PR) recovery mechanisms. Four of these have been used in this study: D-SACK [11], Eifel [12], TCP-DOOR [13], and F-RTO [14]. We use delay as the main network variable parameter, as our objective is to have a representative topology that introduces OOO events. Considering, in addition, lossy paths would only create more OOO events due to loss-induced retransmissions without, fundamentally, changing the behaviour of the PR techniques.

This paper is organized as follows: Section II presents the MPTCP protocol and the different congestion control algorithms specifically proposed for it. Section III discusses the four packet reordering techniques proposed for single-path TCP to make it more robust to OOO events and how they have been adopted to MPTCP. In section IV, the experimental results on the performance evaluation of all combinations of CC and PR algorithms for MPTCP for the same network are presented. Section V presents conclusions and future work.

II. MPTCP: MULTIPATH TCP

MPTCP is a modification of the regular TCP that allows single data traffic to be split across multiple paths [1]. One of the main design goals behind MPTCP was to be completely transparent to both the application and the network. The application opens a regular TCP socket which initially starts one regular TCP subflow. More subflows can be added later by any MPTCP end point using the same application socket. Outgoing data is then scheduled according to some implementation management policy and incoming data from all TCP subflows is reordered to maintain the in-order byte-stream abstraction of TCP, as seen by application. For this to work, at least one end (preferably both ends) must have at least two IP addresses, and both ends must implement the multipath TCP extensions. Packets are sent down different paths by addressing them to the different destination addresses available for the remote system. The multi-addressed multipath TCP has a second sequence number space carried in TCP options, so that the regular sequence number and acknowledgement fields can remain compatible with existing middle-boxes such as NATs (network address translations). It has been shown that MPTCP delivers improved network resilience and increased throughput. It can also benefit load balancing at multi-homed servers and data centres [8].

A. Sequence Space

MPTCP protocol uses two levels of sequence spacing: a connection-level sequence number and another sequence number called subflow-level sequence number for each path or subflow (SF). The connection-level sequence is the data sequence number seen by the application. When the MPTCP sender starts transmitting data through different SFs, connection-level data sequence number has to be mapped to the subflow sequence number. Each SF has to send data as a regular TCP connection independently from other SF(s) with its own sequence numbers and cumulative acknowledgments (ACKs). The MPTCP receiver uses the connection-level sequence number to reassemble the data streams coming from different SFs inorder to pass them to the application layer in-sequence. Therefore, MPTCP uses a data sequencing mapping (DSM) to convert between the two sequence spacing [1]. The DSM can be depicted clearly in Fig.1 where packet (5-S2) for example has a data-sequence-number equal to 5 and a subflow-sequence number equal to 2.The arrival packet is said to be in-sequence if and only if both the subflow-sequence and data-sequence are as expected.

Fig.2 explains how the MPTCP receiver node examines the newly arrived packet to decide whether to save it in the receiver buffer (in-order packet) or in the OOO-buffer (OOO packet). Otherwise it will be rejected (most likely a duplicate packet). The receiver first checks the sequencing of the subflow then the sequencing of the connection. When the subflow sequence number of the received packet (SF_RecSeq) is equal to the expected subflow sequence number, (SF_ExpectSeq) and the connection (or Data) sequence number (D_RecSeq) is equal to the expected Data sequence number (D_ExpSeq) then the packet is considered in-sequence. The received packet is considered to be OOO if either of the sequence numbers is greater than the expected, otherwise it is rejected.

B. Congestion Control

The congestion control (CC) algorithm is the most important part of MPTCP protocol. In the regular (i.e. single-path) TCP protocol, only one congestion window (CWND) exists between the sender and receiver nodes. However MPTCP has more than one congestion window depending on the number of subflows between the two end points. The MPTCP sender has a CWND for each subflow to control the local traffic in each path, whilst the MPTCP receiver has a single global receiving window shared between all subflows.

Three major goals for the congestion control have to be satisfied by the MPTCP protocol [9]:

1) *Improve throughput:* A multipath flow should perform at least as well as a single path flow would on the best of the paths available to it.
2) *Do not harm:* A multipath flow should not take up more capacity from any of the resources shared by its different paths, than if it was a single flow using only one of these paths.
3) *Balance congestion:* A multipath flow should move as much traffic as possible off from its most congested paths.

As an improvement to the previous goals another goal was added later by the same authors and it is about the path's fluctuation

4) *Adapt quickly and do not oscillate:* A multipath flow should adapt quickly when congestion changes and without flapping.

Different CC algorithms have been proposed [9]; Uncoupled (Un-CC), Fully Coupled (FC-CC), and Coupled (Co-CC); and extensive simulation studies have been done for them to test MPTCP goals [9]. These studies concluded that Un-CC does not satisfy the fairness condition and the FC-CC suffers from flappiness. On the other hand the Co-CC solves these problems because it deals with different RTTs for different paths [15]. Un-CC uses Additive-Increase/Multiple-Decrease (AIMD) congestion control used with regular-TCP in each path independently. The increase equation is given by (1) and the decrease is given by (2). However, FC-CC takes total CWND of all paths in consideration in order to couple both the increase and decrease cases for each path using the set of equations (3) and (4). The Co-CC couples only the increase case for each path and keeps the decrease similar to regular-TCP. Co-CC increases CWND of each path by (5) and decreases by (2) where Wr is CWND of path r, W is the summation of all CWNDs, and α is calculated using (6) [15].

Uncoupled-CC:

$$W_r = Wr + \frac{1}{Wr} \quad (1)$$

$$Wr = \frac{Wr}{2} \quad (2)$$

Fully Coupled-CC:

$$W_r = Wr + \frac{1}{W} \quad (3)$$

$$Wr = max\,(Wr - \frac{W}{2}\ ,\ 1) \quad (4)$$

Coupled-CC:

$$W_r = Wr + \text{Min}\left(\frac{\alpha}{W}, \frac{1}{Wr}\right) \quad (5)$$

$$\alpha = W * \frac{\text{Max}\frac{Wr}{(\text{RTT})^2}}{(\Sigma_r \frac{Wr}{\text{RTT}})^2} \quad (6)$$

Although Co-CC adjusts CWND size for each path taking in consideration RTT measurement, MPTCP cannot saturate link with higher RTT, because OOO data arrival on the receiver endpoint at the connection level causes a bottleneck in data re-sequencing process. Section IV of this report shows that sending data using the best path will be a suitable solution however; it limits the aggregate throughput to be no more than the throughput of the best path.

III. Packet Reordering Solutions

A sender generates a traffic stream with an in-order sequence of data packets. For many reasons the ordering of the packets received at the destination may be different from the sender generated one. An out-of-order packet makes the receiver responds with duplicated acknowledgements (dupACK) inducing the sender to infer wrongly a packet loss and then enter congestion control stage unnecessarily, resulting in lower overall end-to-end performance.

It has been shown that packet reordering is not a rare event [16], [17]. With persistent and substantial packet reordering, TCP spuriously retransmits segments - the sender keeps its congestion window unnecessarily small, loses its ACK-clocking, and understates the estimated RTT and RTO (Retransmission Timeout) [10]. This can result in significantly lower application throughput and network performance.

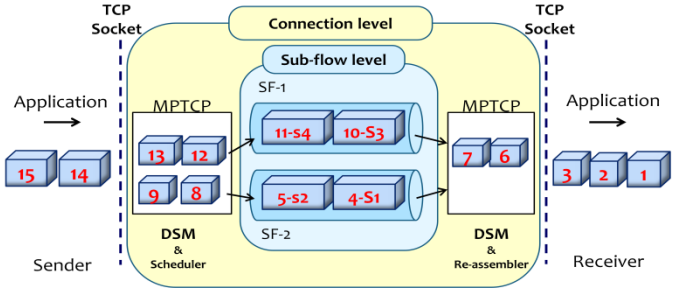


Fig. 1. Out-Of-Order example in Multipath TCP

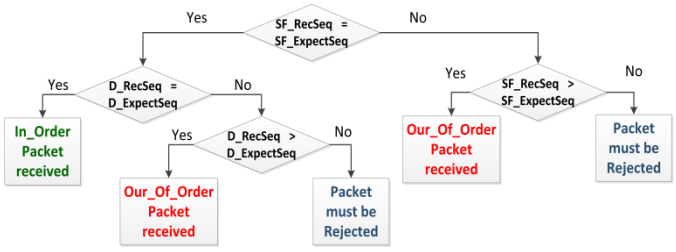


Fig. 2. Packet classification at MPTCP receiver node

In the multipath context, received packets are out of order because different SFs may have different characteristics, such as end-to-end delay. The OOO arrival of the data packets will create a substantial problem for multipath TCP while reassembling them at the connection level, and not at SF level because the SFs are independent. When the receiver node receives OOO packets it will store them into OOO buffer waiting for the packets expected to precede them. However, when the sender receives dupACKs it will trigger one of the proposed methods for solving reordering in addition to the CC selected for the corresponding SF. Referring to Fig.1. Let the two end points be connected by two SFs, SF-1 and SF-2. Under symmetric conditions of the SFs and without considering loss events, the transmitted data packets mostly arrive to the destination node in-sequence. However, when the SF-2 has a large RTT compared to SF-1 then data will most likely arrive out of sequence at connection level, although it may be in-order at SF level. This is illustrated in Fig.1 where packet 6 and 7 are considered OOO because they have been transmitted through the faster path, SF1, and arrive before packets 4 and 5. Since the sender cannot distinguish between the losses or delays of packets, it will enter the congestion control stage and reduce the CWND for SF-2. In the worst case, the sender will continue halving the CWND unnecessary and keeping SF-2 in slow start most of the time.

Many mechanisms have been proposed for TCP as a solution for the packet reordering problem and four of them named D-SACK, Eifel, TCP-DOOR and F-RTO will be discussed in this section.

A. D-SACK

D-SACK is an extension of the selective acknowledgment SACK option for TCP[11] that depends on duplicate selective acknowledgement (D-SACK) to detect segment reordering and retracts the associated spurious congestion response. When congestion is detected, CWND is saved before reduction and when a sender finds that it has made a spurious congestion response based on the arrival of a D-SACK it performs "slow start" to increase the current CWND to the stored CWND before congestion avoidance.

B. Eifel

Ludwig and Katz proposed the Eifel algorithm to eliminate the retransmission ambiguity and solve the performance problems caused by spurious retransmissions [12]. The sender uses the TCP timestamp option to inset the current timestamp into the header of each outgoing segment to a destination. The receiver then copies those timestamps in the corresponding ACKs. When a packet loss is assumed, the sender retransmits the lost segment and always uses the stored timestamp of the first retransmission in addition to the Slow-Start thresh hold (SSThreshold) and the CWND. Upon receiving the ACK of the corresponding segment, the sender compares the timestamp of the arrived ACK with the stored one. If the ACK's timestamp is smaller, then the retransmission was spurious. Subsequently, the sender simply restores the SSThreshold and the CWND to the stored values.

C. TCP-DOOR

TCP-DOOR has been proposed to improve TCP performance over mobile Ad-hoc networks [13]. It is commonly known that TCP protocol performs poorly in wireless networks since it assumes all packet losses are due to congestion. TCP-DOOR (Detection of Out-of-Order and

Response) is similar to Eifel in using packet timestamp. Once the OOO is detected the TCP-DOOR responds by temporarily disabling the congestion control and instant recovery during congestion avoidance. The sender keeps its state variables constant for a time period, such as RTO and CWND, and then recovers immediately to the state before congestion avoidance action was invoked.

D. F-RTO

The Forward RTO Recovery (F-RTO) algorithm is a TCP sender method that does not require any TCP options to operate [14]. After retransmitting the first unacknowledged segment triggered by a timeout, the F-RTO algorithm at a TCP sender monitors the incoming ACKs to determine whether the timeout was spurious or not and also to decide whether to send new segments or retransmit unacknowledged segments. However, if packet reordering or packet duplication occurs on the segment that triggered the timeout, the F-RTO algorithm may not detect the spurious timeout due to incoming dupACK.

Many comparisons have been made to classify and evaluate several PR recovery methods for single TCP [10]. They conclude that by performing slow start during state restoration, D-SACK allows TCP to reacquire ACK-clocking and avoid injecting traffic bursts into the network. On the other side, the response of D-SACK is slower than the other algorithms such as Eifel and TCP-DOOR. Also, it has been stated that Eifel does not work when the original and retransmitted segments are reordered. While TCP-DOOR can improve the TCP throughput significantly (50% on average [13]), it may lead to congestion collapse from undelivered packets by disabling the congestion control for a time period every time an OOO event is detected. Thus, TCP-DOOR does not perform well in a very congested network.

In the following section, the PR solutions mentioned previously will be simulated with MPTCPto evaluate their influence on the link utilization and the application throughput. The throughput will then be compared with MPTCP throughput when no recovery method is in use (NoPR).

IV. Performance Evaluation

In this section, we present our simulation results and discuss the path utilization using various packet reordering recovery algorithms mentioned in the previous section. MPTCP has been simulated using ns-3 [18], [19] and the performance has been evaluated with four different solutions for PR (DSACK, Eifel, TCP-DOOR, and F-RTO). The simulated scenarios evaluate the impact of PR on the aggregate throughput (gThroughput) of the protocol.

A. Simulation setup

The simulated system shown in Fig. 3 assumes an FTP application to transfer a 50MB file running a Client/Server architecture. Two nodes are implemented and connected by two Point-to-Point links that represent two possible disjoint paths for MPTCP. The data rates for both links are set to 0.5Mbps with 0 error rate (lossless paths). The delay of the first subflow (SF-0) is set to 10ms while the delay of the other subflow (SF-1) is set initially to 10ms, and varied in different experiments. The delay of both paths varies during the simulation runtime by ±5% of the initial value. The size of OOO receiver buffer is set to be large enough for all OOO packets so as not to limit our performance study by its size.

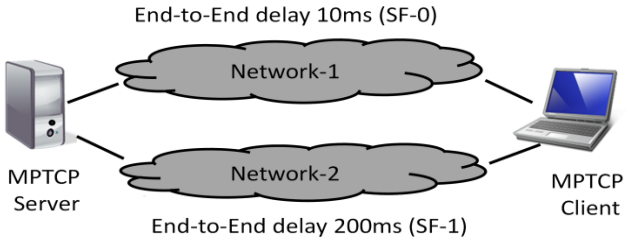


Fig. 3. The simulated scenario

B. Performance Metrics

In this paper, the following performance metrics are used for the results comparisons and analysis.

1) Reorder Buffer-occupancy-Density

As MPTCP receiver requires an OOO-buffer to store the OOO packets received from different paths and before sending them to the shared receive buffer that save them in-order. Reorder Buffer Occupancy Density (RBD) is used to measure the amount of space each PR solution needs. It reflects the ability of each PR solution in recognizing OOO packets. RBD is defined as the buffer occupancy frequencies normalized by the total number of non-duplicate packets [20] where B is the number of packets presented in the OOO-buffer.

2) Out-Of-Order Ratio

MPTCP maintains two sequence numbers for each packet. Data sequence for MPTCP connection and subflow sequence for each TCP subflow. In-order packets arrive from the same subflow may wait in the OOO-receive-buffer before their data sequence numbers become in-order, this due to the late arrivals of packets from other SFs. Therefore, a key performance metric of using PR solution with MPTCP is to measure Out-Of-Order-Ratio (OOO-R) at the receiver side. OOO-R is measured to be total number of received packets being stored in OOO-buffer over the total number of non-duplicate received packets (the size of the FTP file).

3) Link-Utilization

The link utilization (L-Utilization) can be obtained by observing the SF-CWND. If MPTCP is able to increase the value of CWND then more data can be sent through this SF. The lack of competition in the link from other flows in our scenario makes all bandwidth available to the MPTCP connection. Link Utilization is defined by the throughput of the Link (SF) over its data rate.

4) Aggregate Throughput

As our goal is to study the PR impact on the overall performance of MPTCP, we focus on measuring the aggregate throughput (gThroughput) of this protocol. The aggregate throughput is defined by the summation of the throughputs of

all available paths for MPTCP connections (SFs). Optimal throughout used in this paper presents the maximum possible throughput that can be achieved by the protocol when consuming all the available bandwidth of the links.

C. Results Analysis

The results are divided into two parts, the first part is the evaluation of original MPTCP without any PR solutions (NoPR), and the second is the behaviour of MPTCP with PR solutions. The influence of PR solutions on MPTCP protocol is studied by comparing their performance with MPTCP behaviour in the first part.

1) MPTCP with NoPR

The benchmark simulation uses MPTCP with the three CC algorithms (Uncoupled, Fully-Coupled, and Coupled) under four network scenarios without any packet reordering. The first scenario uses equal and constant delays (10ms) for both SFs. The second uses equal delays but SF-0 suffers from delay fluctuations during the simulation. The third evaluation uses different delays between the SFs (10ms and 200ms) without fluctuations, while the last one uses different delays with small fluctuations in both SFs.

We observe that the gThroughput of MPTCP protocol using all mentioned CC algorithms can reach the optimal value if and only if the delays in both links are stable and equal. However, when one or both links have a small variation in the delay during the transmission then only one link will dominate and the gThroughput will be equal to the throughput of the dominant link. In Fig.4a both links have fixed (no fluctuation) delays equal to 10ms and gThroughput is optimal and equal to the sum of the available path throughputs (1Mbps). Fig.4b shows the gThroughput obtained when the SFs experience either unequal delays or fluctuations in their delays - gThroughput in this case is equal to the throughput of only one subflow (0.5Mbps SF-1 in this case). This can be also understood from the RBD distribution for all MPTCP CCs in Fig.5, which shows that the OOO memory occupancy were very low because most of the packets arrive in-sequence due to MPTCP using only one SF instead of two. The same results we observed when the delay differences between SFs were 50ms, 200ms, and 500ms.

Although the first goal of the MPTCP protocol design is satisfied, the gThroughput is not optimal as the capacity of SF-0 is not used. This is because when the transmission starts just after establishing the connection one of the subflows suffers from late packet arrival and cannot recover.

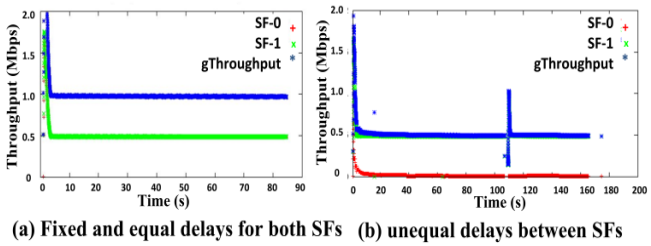


Fig. 4. The throughput of the MPTC and its SFs

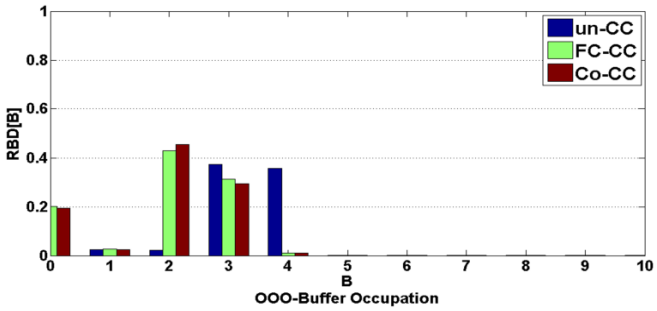


Fig. 5. RBD [0-10] for MPTCP

2) Out-of-order performance for MPTCP with PR Solutions

In this part of our evaluations, the MPTCP is simulated with four mechanisms proposed for single-TCP to recover from PR using the same topology of Fig.3. Both SFs suffer from small fluctuation in the delays (±5%). The delay of SF-0 has been set to 10ms while the delay of SF-1, the key study of our evaluations, is increased from 10ms to 500ms in 100ms steps. This section presents a complete analysis when the delay of SF-1 is equal to 200ms, as a typical set of results. Most of these PR solutions behave effectively when the delay difference between SFs is less than 200ms. The performance analysis of these PR solutions is compared with the baseline evaluations presented previously.

From our observations, PR solutions increase OOO-R up to eight-fold compared to original MPTCP shown in Table I. OOO-R reaches 47.3% in maximum with TCP-DOOR and 40.6% on average with D-SACK. The increase of the OOO-R indicates that the sender, with the help of a PR solution, is able to realise the late arrival of packets and therefore rolls back CWND to its state exactly before retransmission was triggered and continue sending more data. The OOO-buffer occupation increases as more packets are stored waiting for their data-sequence to be in-order. Table II shows link utilization for both SFs and gThroughput obtained in all studied cases; L-Utilisation is the proportion of maximum single-path capacity used by the respective MPTCP subflow. The results indicate the ability of PR solutions to increase gThroughput by also utilizing the path that suffers from large end-to-end delay (SF-1) for all PR solutions except Eiffel. The gThroughput improvement is less than the others and very close to the original MPTCP specifically with FC-CC which also requires double data transfer completion time.

TABLE I: OOO-R AND MAXIMUM OOO BUFFER SIZE OCCUPIED BY DIFFERENT PR SOLUTIONS AND MPTCP CCS

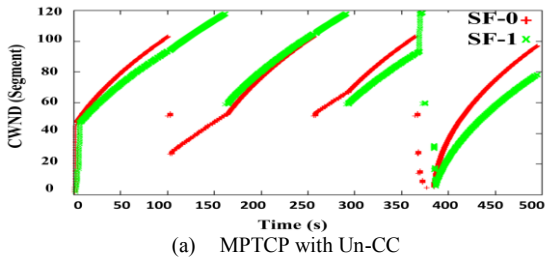
Packet Reorder Solution	Congestion Controller					
	Uncoupled		Fully Coupled		Coupled	
	OOO-R (%)	OOO Buffer (KB)	OOO-R (%)	OOO Buffer (KB)	OOO-R (%)	OOO Buffer (KB)
NoPR	6.49	1.1872	5.0	1.1648	5.1	1.1648
DSACK	44.2	2.296	34.5	2.128	43.1	2.128
Eifel	27.9	1.68	9.3	1.176	37.0	2.128
TCP-DOOR	47.3	2.464	22.0	2.128	44.4	2.128
F-RTO	27.1	0.3248	18.7	0.1456	21.4	0.1344

TABLE II: MPTCP GTHROUGHPUT AND L-UTILIZATION COMPARISONS BETWEEN DIFFERENT PR SOLUTIONS AND MPTCP CCS

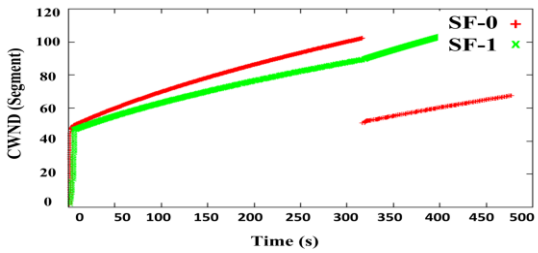
Packet Reorder Solutions	Congestion Controller								
	Uncoupled			Fully Coupled			Coupled		
	L-Utilization (%)		gThroughput (Mbps)	L-Utilization (%)		gThroughput (Mbps)	L-Utilization (%)		gThroughput (Mbps)
	SF-0	SF-1		SF-0	SF-1		SF-0	SF-1	
NoPR	97.0	4.2	0.50	97.0	3.6	0.50	96.0	3.4	0.50
DSACK	91.8	84.4	0.86	74.0	50	0.80	88	86	0.86
Eifel	40.4	94.6	0.68	94.6	50	0.50	84	84.2	0.82
TCP-DOOR	88.6	91.8	0.90	96.4	68.8	0.61	96	67.7	0.87
F-RTO	100	65.6	0.72	89	58.6	0.60	90	60	0.64

We observe that DSACK is able to improve the gThroughput of MPTCP by 60-70% against NoPR case with all CC algorithms when the delay difference between the two SFs is less than 200ms. Fig.6 illustrates the behaviour of CWND using DSACK. It shows that the increasing rate of CWND with un-CC is faster than Co-CC where the latter forces the CWND to increase smoothly while balancing the load between SFs.

Eifel performs worse than the other solutions, particularly with FC-CC where the gThroughput is found to be less than the benchmark measurements. The improvement in gThroughput with Eifel can be achieved with un-CC but not with the coupling methods. It can be clearly observed from Fig.7 and Table II that the MPTCP with Eifel could not saturate SF-1 with coupled CCs as compared to the other solutions. On the other hand, Eifel with Co-CC can behave better when the transmission is handled to the slower link and its CWND get a chance to increase rapidly. In this case, the CWND of the faster link can also send more data even with many spurious retransmission detections. However, once the latter link handles the transmission, its CWND will increase quickly preventing the other link from sending more data. Therefore, the performance of Eifel is not stable with MPTCP protocol.



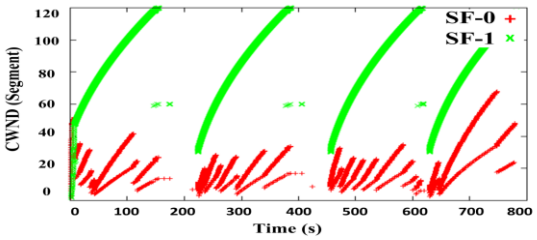
(a) MPTCP with Un-CC



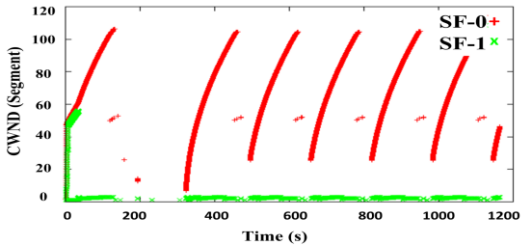
(b) MPTCP with Co-CC

Fig. 6. CWND of MPTCP SFs with DSACK

As expected, TCP-DOOR has a significant impact on the gThroughput and link utilization as it suspends the congestion response for a certain time period upon detecting a spurious retransmission. This can be clearly seen in Fig.8 that depicts the behaviour of the CWND under TCP-DOOR. The performance of TCP-DOOR approaches DSACK with 70% improvement in gThroughput using both un-CC and Co-CC. However, the DSACK outperforms others by at least 40% with FC-CC. On the other hand, TCP-DOOR will not perform well in a very congested network [10].

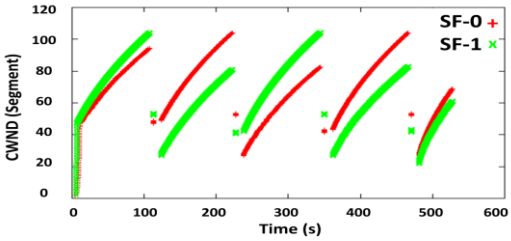


(a) MPTCP with Un-CC

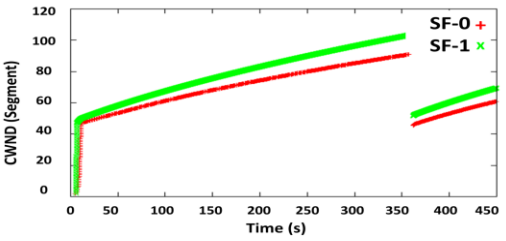


(b) MPTCP with FC-CC

Fig.7 CWND of MPTCP SFs with Eifel



(a) MPTCP with Un-CC

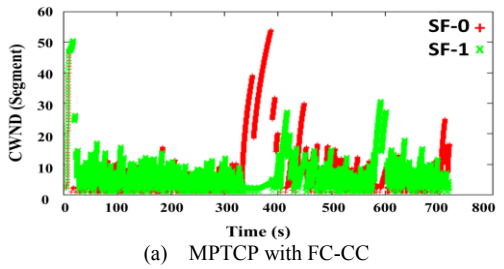


(b) MPTCP with Co-CC

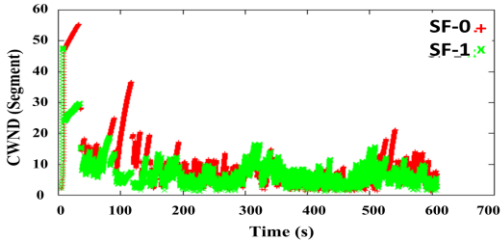
Fig. 8. CWND of MPTCP SFs with TCP-DOOR

The performance of F-RTO is worse than both DSACK and TCP-DOOR under un-CC and Co-CC and approaches the TCP-DOOR under FC-CC. MPTCP with F-RTO has a better balance in data transmission split between SFs as long as no critical congestion occurs to any of available paths. Fig.9 depicts this situation where the transmission uses both SFs, until a critical RTO occurs and then one SF dominates. The

gThroughput improvement with this solution can reach 43.1% in maximum as presented in Table II. Fig10 summarises the application throughput with all PRs being simulated with MPTCP.



(a) MPTCP with FC-CC



(b) MPTCP with Co-CC

Fig. 9. CWND of MPTCP SFs with F-RTO

By examining the OOO-Buffer, F-RTO outperforms the other combinations in terms of memory requirements (high RBD for in-sequence received packets at the same time). We notice Eifel approaches the NoPR under FC-CC where the algorithm occupies less than 5 memory locations most of the time with high RBD and low OOO-R, whilst both DSACK and TCP-DOOR use more memory locations with small densities. As F-RTO occupies less memory space (around 300KB) as compared to other PR solutions, this makes F-RTO preferable to others in terms of memory utilization, as shown in Fig.11 and Table II. All PR solutions under Un-CC occupy more memory space because Un-CC method injects more data into network without balancing loads between SFs. Due to space limitation, RBD figures of FC-CC and un-CC are not presented - only the RBD for all PR solutions under Co-CC are shown as typical results. Table II presents each PR solutions with its maximum memory space occupied by OOO packets classified by CC of MPTCP.

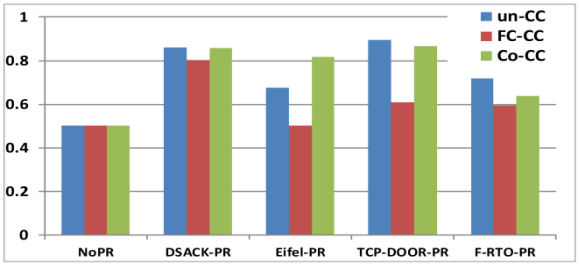


Fig. 10. MPTCP gThroughput with and without PR solutions

In order to study the behaviour of MPTCP under different networks, we fixed the end-to-end delay of one link and change the delay of the other by increasing the difference between them (0ms to 500s in steps of 50ms and 100ms).

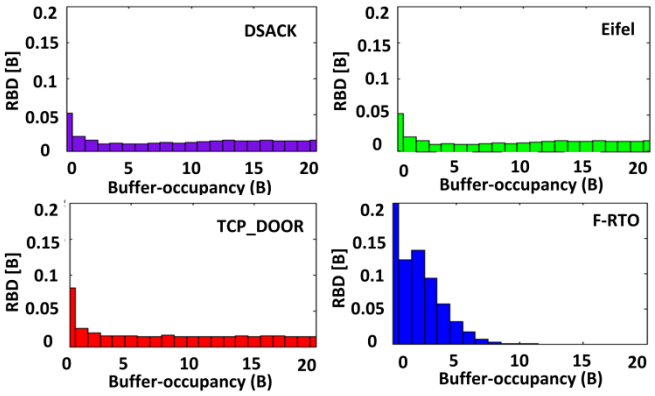


Fig. 11. OOO-Buffer-RBD under Coupled-CC

The aggregate throughputs of MPTCP with all PR solutions being simulated as a function of path delay difference are shown in Fig.12, Fig.13, and Fig.14 under un-CC, FC-CC, and Co-CC labels respectively. Two main observations can be obtained from this experiment. First, all PR methods (except Eifel) are able to substantially improve gThroughput of MPTCP up to a value of 200ms difference for path delay difference; Eifel has a small impact particularly with FC-CC. Both DSACK and TCP-DOOR outperform others by providing better application throughput as the delay variation increases. However they need at least 2MB memory space for OOO-buffer while the FRTO provides less application throughput but with less memory space. Second, when the delay difference becomes more than 200ms, all PR solutions behave less effectively with coupling methods and the gThroughput improvement is less than 20%.

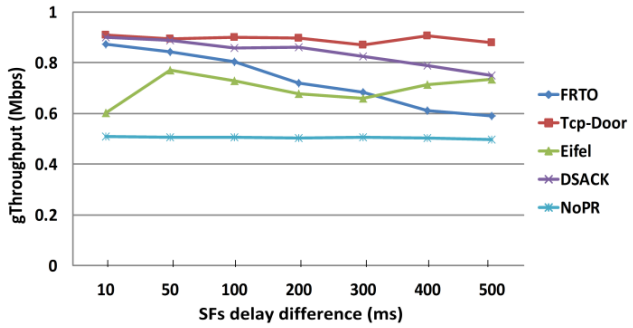


Fig. 12. The gThroughput of MPTCP with uncoupled-CC and various packet reorder solutions as delay variation between two subflows increases

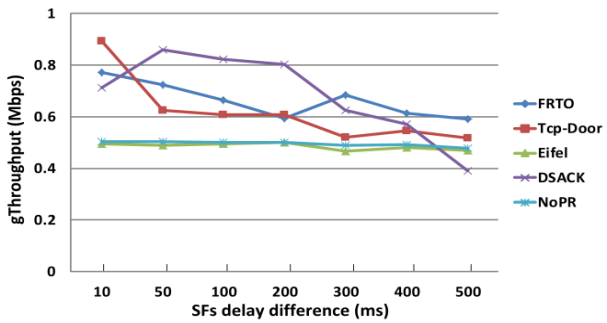


Fig. 13. The gThroughput of MPTCP with Fully coupled-CC and various packet reorder solutions as delay variation between two subflows increases

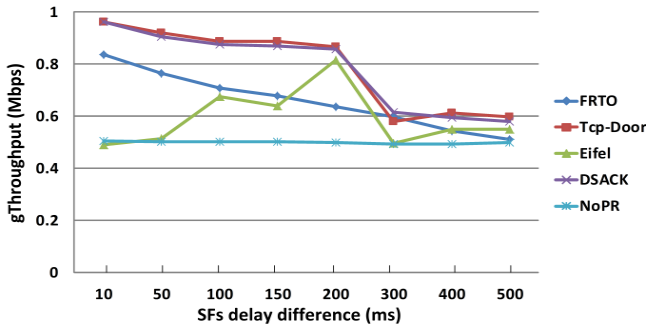


Fig. 14. The gThroughput of MPTCP with Coupled-CC and various packet reorder solutions as delay variation between two subflows increases

V. CONCLUSIONS

Many different solutions have been proposed to solve the packet reordering problem in single-path TCP. However, none of them has been intensively evaluated in the context of multipath protocols, neither have they been comprehensively compared when run in conjunction with MPTCP. This paper presents results of the performance of MPTCP with four TCP packet reordering solutions, namely D-SACK, Eifel, TCP-DOOR, and F-RTO, and benchmarks them against the performance of MPTCP without any packet reordering recovery methods. The results show that when the two subflows have symmetrical attributes then the behavior is much better than in the asymmetrical case.

Whilst the Coupled Congestion Control (Co-CC) algorithm provides a robust data transmission and solves the fairness and floppiness problems that exist with other congestion control methods for MPTCP, the results show that the Co-CC sends most of the data using the best path and is unable to effectively use the others even under a small delay variation scenario. At the same time, the results clearly show that the packet reordering solutions bring a substantial performance improvement for MPTCP by increasing the aggregate throughput as well as the path utilization particularly when delay difference between SFs is less than 200ms.

The analysis also shows that MPTCP using uncoupled congestion control is less sensitive to path delay differences up to 500ms, and that both TCP-DOOR and DSACK utilize both paths effectively. MPTCP using DSACK is less sensitive to path delay difference (up to 200ms) independently of which CC algorithm is used. TCP-DOOR approaches the DSACK in aggregate throughput under both Co-CC and un-CC algorithms. MPTCP should use F-RTO as a PR solution if memory is a constraint. MPTCP using Eifel PR solution gives very little throughput gain even when path capacity is available; whilst it still provides connectivity redundancy, it is not the best choice for throughput maximization.

Whilst this study considered only lossless delay asymmetrical links, the future work will consider how the packet reordering will behave with MPTCP used over lossy asymmetrical links.

ACKNOWLEDGMENT

This research is supported in part by the Public Authority for Applied Educational and Training (PAAET), Kuwait.

REFERENCES

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

[2] A. Adu-Al, T. Saadawi, and M. Lee, “LS-SCTP: a bandwidth aggregation technique for stream control transmission protocol,” *Comput. Commun.*, vol. 27, no. 10, pp. 1012–1024, 2004.

[3] H. Hsieh.and R. Sivakumar, "pTCP: an end-to-end transport layer protocol for striped connections," *Network Protocols, 2002. Proceedings 10th IEEE International Conference on* , pp.24,33, 12-15 Nov. 2002 ”

[4] R. Stewart, Q. Xie, et al., Stream Control Transmission Protocol, *RFC 2960,* 2000.

[5] D. Sarkar, "A Concurrent Multipath TCP and Its Markov Model," *Communications, 2006. ICC '06. IEEE International Conference on* , vol.2, pp.615,620, June 2006.

[6] Y. Dong, D. Wang, N. Pissinou, and J. Wang, "Multi-path load balancing in transport layer." In *Next Generation Internet Networks, 3rd EuroNGI Conference on*, pp. 135-142. IEEE, 2007.

[7] Y. Hasegawa, I. Yamaguchi, T. Hama, H. Shimonishi, and T. Murase.“Improved ata distribution for multipath tcp communication”.*Global Telecommunications Conference, 2005.GLOBECOM'05. IEEE,* vol.1,pp. 5-pp.,IEEE, 2005

[8] D. Wischik,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*, pp.8-8, 2011

[9] C. Raiciu, D. Wischik, and M. Handley. "Practical congestion control for multipath transport protocols.", *University College of London,* Technical Report, 2009.

[10] K. Leung, V. Li. & D. Yang “An overview of packet reordering in transmission control protocol (TCP): problems, solutions, and challenges”. *Parallel and Distributed Systems, IEEE Transactions on, IEEE, vol. 18*, pp. 522-535,2007.

[11] S. Floyd, J. Mahdavi, M. Podolsky, and M. Mathis “An extension to the selective acknowledgement (SACK) option for TCP”. *RFC 2883,* 2000.

[12] R. Ludwig, and H. Katz “The Eifel algorithm: making TCP robust against spurious retransmissions”. *ACM SIGCOMM Computer Communication Review, ACM,* vol. 30, pp.30-36, 2000.

[13] F. Wang and Y. Zhang “Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response”, *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing,* pp. 217-225, 2002.

[14] P. Sarolahti, M. Kojo, K. Yamamoto, and M. Hata “Forward RTO-recovery (F-RTO): An algorithm for detecting spurious retransmission timeouts with TCP”*, RFC5682,* 2009

[15] C. Raiciu, M. Handley, and D. Wischik “Coupled congestion control for multipath transport protocols”, *RFC 6356*, Oct. 2011.

[16] J. Bennett, C. Partridge, and N. Shectman, “Packet reordering is not pathological network behaviour.”, *Networking, IEEE/ACM Transactions on, IEEE,*vol. 7, pp. 789-79, 1999.

[17] M. Laor & L. Gendel “The effect of packet reordering in a backbone link on application throughput.”, *Network, IEEE,* vol. 16, pp. 28-36, 2002.

[18] (2012) ns-3website. [Online]. Available: http://www.nsnam.org/.

[19] B. Chihani and D. Collange, “A Multipath TCP Model for ns-3 simulator”, *Workshop on ns-3 held in conjunction with SIMUTools 2011*, Spain, 2011

[20] A. Jayasumana, N. Piratla, T. Banka, A. Bare and R. Whitner, “Improved Packet Reordering Metrics”. *RFC 5236,* 2008.