Packet Reordering Response for MPTCP under Wireless Heterogeneous Environment

Amani Alheid, Angela Doufexi
Smart Internet Lab
University of Bristol,
Bristol, United Kingdom
{Amani.alheid, a.doufexi} @bristol.ac.uk

Dritan Kaleshi
5G Fellow at
Digital Catapult
London, United Kingdom
Dritan.kaleshi@cde.catapult.org.uk

Abstract—**Multipath Transmission Control Protocol (MPTCP) promises higher bandwidth and higher resilience against network path failures as an evolving technology for the future Internet. MPTCP allows multiple paths between two devices to be pooled and appear to the application as a single end-to-end transport connection. In this paper, we propose a solution for the packet reordering problem, to improve path utilization and aggregate throughput. The proposed method maintains the congestion window size when three duplicate acknowledgements occur to avoid unnecessary reductions in transmission rate. We evaluate the performance of the proposed solution using an NS-3 in different wireless scenarios and compare it against the performance of other existing reordering methods. The results show that the proposed method improves the aggregate throughput and path utilization when the packet drop rate is high.**

Keywords— MPTCP; Packet Reorder (PR); PR-R; D-SACK; TCP-DOOR; Wi-Fi; Duplicate Acknowledgement

I. INTRODUCTION

Multipath transmission protocols and data offloading techniques can overcome the limitations of the current Internet by better bandwidth aggregation and resource utilization [1-5]. The Multipath transmission control protocol (MPTCP) is a new standard supported by IETF for multipath transmission at the transport layer for the future Internet [1]. The key motivation behind the MPTCP is to provide reliable and resilient connectivity in the current and future Internet. The research in this context is focusing on faster downloads, lower data transfer costs, and seamless switching between different interfaces, particularly wireless ones, such as Wi-Fi and cellular networks. Besides the improvements that MPTCP can provide compared with a single-path transmission control protocol (TCP), particularly in aggregate throughput (gThroughput), the packet reordering problem limits its performance. In conventional single-path TCP, the reordering problem arises from packet-level multipath routing or link layer retransmission [6]. However, in a multipath transport context, the reordering comes from the heterogeneity of multiple paths or sub-flows (SF).

Several researches proposed solutions for the reordering problem in multipath transmission. These solutions vary in their techniques from simple calculations or modifications to the conventional MPTCP [7-9] to sophisticated scheduling methods [10, 11]. Some studies have proposed solutions to mitigate receiver's buffer blocking particularly for the case of constrained buffer size [12]. However, the reordered packets that frequently trigger the fast retransmission and unnecessarily reduce the size of the congestion window (CWND) result in lower throughput. Therefore, the need for a reordering solution is essential even with large receiver buffer size. In MPTCP Linux implementation [13], the duplicate selective acknowledgements (D-SACK) is implemented as a solution for the reordering problem regardless of the scheduling method.

In this paper, we examine the influence of packet reordering on the behaviour of CWNDs of MPTCP, particularly when at least one of the paths ends in a wireless link. The paper compares the performance of the end-to-end connection for different types of reordering solutions integrated with MPTCP at the connection level (and not at SF level). An interesting observation is that MPTCP, without any reordering solution, is unable to aggregate the available bandwidth and fully utilize the capacity of the links even with symmetrical links and large receiver buffer size. Consequently, we propose a solution for the packet reordering problems in which the amount of sending data is not immediately reduced by receiving a third duplicate acknowledgement (DUPACK) for a packet as the single-path TCP does. This paper has two main contributions.

- To analyse the impact of reordering recovery methods on the throughput gain and path utilization of the conventional MPTCP.
- To propose the packet reordering response (PR-R) as a solution to the reordering problem for MPTCP when the DUPACKs are triggered by out-of-order (OOO) packets whilst the retransmission timeout (RTO) technique of the conventional TCP is unchanged.

The study assumes MPTCP is operating over heterogeneous communication links that include two Wi-Fi links with different links characteristics. The performance of the proposed method (PR-R) is compared with the performance of different packet reorder (PR) recovery methods (D-SACK and TCP out of order detection and response (TCP-DOOR)) where their improvement to the aggregate throughput (gThroughput) of MPTCP is found to be substantial, according to our previous studies [14, 15]. The results show that the proposed method mitigates the influence of OOO packets and improves the gThroughput of MPTCP.

The Public Authority for Applied Educational and Training (PAAET), Kuwait.

This paper is organized as follows. Section II presents the fundamental details of MPTCP, reordering problems and the recovery mechanisms. Section III explains in detail the proposed method (PR-R). Section IV describes the system model setup and parameters, simulated scenarios, and performance evaluation metrics. Section V presents the analysis of the results and compares the performance of the PR-R with other PR recovery algorithms for MPTCP under different simulated scenarios. Section VI concludes the paper.

II. Multipath TCP

The MPTCP is an extension to the regular single-path TCP that allows single end-to-end connection data traffic to be split across multiple TCP paths. 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 SF. More SFs can be added later by any MPTCP end-point using the same application socket. Outgoing data packets are then scheduled between opened SFs and incoming data packets from all SFs are reordered to maintain the in-order byte-stream abstraction of TCP, as seen by the application. It has been shown that MPTCP delivers improved network resilience, increased throughput, and load balancing efficiency at the data centre [16].

A. Sequence Space in MPTCP

The MPTCP uses two levels of sequence spacing: a connection-level sequence number that is used by the TCP socket and is seen by the application and another sequence number called SF-level sequence number, which is used independently for each SF or each physical path; MPTCP uses data sequencing mapping (DSM) to convert between the two sequence spacing [1]. Since the sender is able to send data through more than one interface, it is very likely that the received packets reach the destination in a different order than the sending order, particularly when the links have different characteristics (i.e., path delay). In this case, the receiver has to store OOO packets into an OOO buffer before sending them to a received buffer, which stores all in-order packets that are ready to be sent to the application. The arrival packet is said to be in sequence if and only if both sequence numbers (connection-level and SF-level) are in sequence. The flowchart in Fig. 1 explains how the MPTCP receiver node examines the newly arrived packet to decide whether to save it in the received buffer (in-order packet) or in the OOO buffer (OOO packet).

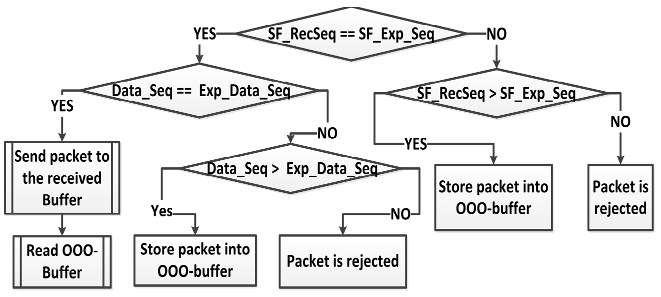

Fig. 1. Packet classification at MPTCP receiver node.

The receiver first checks the SF sequence and then the connection sequence. If the SF sequence number of the received packet (SF_RecSeq) is equal to the expected SF sequence number, (SF_Exp_Seq), and the connection (or data) sequence number (Data_Seq) is equal to the expected data sequence number (Exp_Data_Seq), then the packet is considered in sequence. If either of the sequence numbers is greater than expected, then the received packet is considered to be OOO.

B. Packet Reordering in MPTCP

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 order. The receiver responds to an OOO packet with a DUPACK, inducing the sender to infer a packet loss erroneously and unnecessarily enter the congestion control (CC) stage, resulting in lower overall end-to-end performance.

In a multipath context, packets may arrive OOO because different SFs routinely have different characteristics, particularly the end-to-end delay. The OOO arrival of the data packets will create a fundamental problem for MPTCP, while reassembling them at the connection level. When the receiver node receives OOO packets, it will store them in the OOO buffer and wait for the sequentially preceding ones in order to deliver the in-order byte-stream to the application. As a response, the receiver node sends DUPACK back to the sender. The third DUPACK received by the sender triggers one of the proposed CC method selected for the corresponding SF. In this context, the MPTCP encounters a bottleneck in the data-reordering process at a receiver side, and the receiver needs a significant receiving buffer to save OOO packets coming from different SFs [17, 18], especially when the receiver allows to store all OOO packets without reducing the transmission rate of the corresponding SF.

The reordering of the arrived packets is a significant problem, even for single-path TCP connections, and several mechanisms have been proposed for single-path TCP as a solution for PR problem. In this study, D-SACK and TCP-DOOR, where their improvement to the gThroughput of MPTCP are found to be substantial [14, 15], are used and compared with our proposed solution. Note that NoPR or MPTCP-NoPR refer to the traditional MPTCP without any PR solution.

III. Proposed Method

The proposed method is a sender-based response occurring when the third DUPACK of the lost packet is received. In legacy single-path TCP, TCP triggers the fast retransmission and recover algorithm resulting in a small size for the CWND. In addition, since the MPTCP is an extension to the single path TCP, the standard did not change this part of the congestion response. However, based on the previous studies [14, 15], the reduction in CWND due to OOO packets leads to sub-optimality of the performance of the MPTCP. For example, a client is downloading data from a server through two symmetrical links (i.e., equal data rate and round trip time (RTT)) using the MPTCP. The sender is sending a number of packets when the CWND for both SFs are equal to 32. During this RTT, the receiver starts reading packets from both SFs. Based on the time of arrival of the packets, the reading sequence, which represents the data-level sequence, for the first eight packets is [69, 73, 70, 74, 71, 75, 72, 76] as illustrated in Fig. 2. Assuming that, the SF-level sequence for both SFs are in sequence. Packet 73, which is received from SF1, is considered to be OOO. After reading two more packets from SF1, the receiver sends the third DUPACK for the lost packet (Packet 72).

It is clear from Fig. 2 that the receiver receives Packet 72 immediately after issuing the third DUPACK. However, the sender will reduce the CWND of SF1 by receiving the third DUPACK before receiving the new accumulative acknowledgement (ACK) to packet 72. This will eventually reduce the sending rate of SF1, unnecessarily resulting in a sub-optimal performance of the protocol.

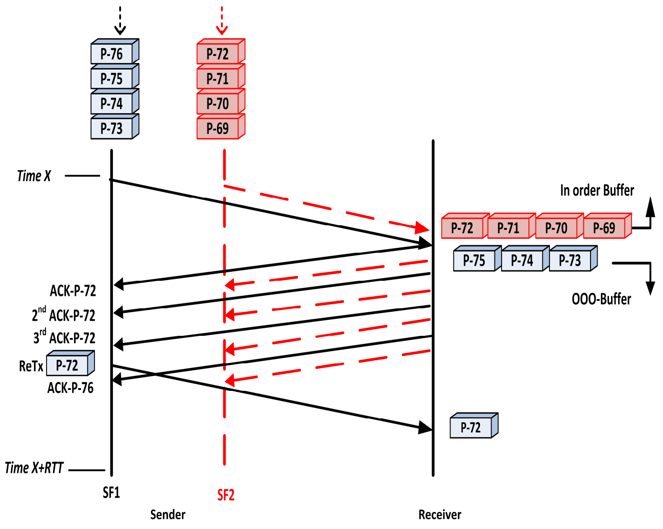

Fig. 2. DUPACK for MPTCP-NoPR

In our proposed method, when the sender receives the third DUPACK of the lost packet, it will trigger the fast retransmission and recovery algorithm without reducing the size of CWND but rather maintaining it. Then, when the sender receives the ACK of the lost packet, the CWND will increase based on the CC calculation. Otherwise, (i.e., for real loss or congestion), the RTO response will take action as legacy TCP. The coupled CC algorithm [19] is implemented with the proposed method to ensure the SF fairness.

Algorithm 1 shows the pseudo code of the proposed method called PR-R. The DUPACK function is triggered when the MPTCP discovers that the received ACK from SF_k is a duplicated ACK for a packet (Packet X) and not an accumulative ACK. The algorithm by default (i.e., NoPR) calls the CWND reduction function when the DUPACK counter for packet X (packet-X.DUPACK.Count) is equal to three as in Line 6. However, if PR-R is selected, then the algorithm does not trigger this call. In both cases, the RTT is measured and the retransmission is performed.

Algorithm 1 Proposed method for DUPACK Function

```
 1: if (packet-X.DUPACK.Count == 3)
 2: switch (Reordering Algorithm)
 3: case PR-R:
 4:         break;
 5: default:
 6:         Call CWND reduction function for SF_k;
 7:         break;
 8: end switch
 9: Notify the RTT of SF_k;
10: Set RTO for packet-X through SF_k;
11: Retransmit packet-X;
12: end if
```

IV. System Setup and Performance Metrics

A. System Setup

The performance of MPTCP is evaluated for the network topology shown in Fig. 3, where each path is ended with a wireless link using the network simulator NS-3 [20, 21] with the coupled CC [19]. The wireless links are set up to use IEEE802.11g (11g) standard with 54 Mbps as the physical data rate. The size of the OOO received buffer is set to be large (10 MB) to avoid its effect on the performance of the system. The backbone network is set to 100 Mbps data rate and 10 ms delay. The simulation consists of transferring a single large file from an FTP server, where all packets are of equal length (1400 B). Each simulation is conducted for 50 s and it is repeated 30 times, and each time it adapts different seeds.

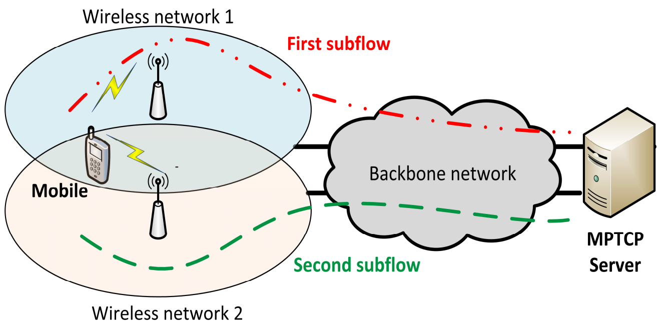

Fig. 3. Network topology for the simulated scenarios

In order to validate the behavior of the MPTCP in the deployed NS-3 package with related Linux Implementation [18], we performed several simulations for the MPTCP using NS-3 and compared the results against Linux related studies in terms of throughput gain, with respect to a single-path TCP. In [18], the results show that the gThroughput of MPTCP using two disjoint SFs with equal bandwidths is 94% higher than the throughput of the single TCP would be on the best SF with no delay difference between SFs, and 23% higher when the delay difference is 500 ms. The NS-3 results are close to the Linux study as shown in Table I, which validate our conducted simulation based study. Note that reordering solution used in both studies is D-SACK.

TABLE I. THE THROUGHPUT GAIN WITH RESPECT TO SINGLE PATH TCP USING LINUX AND NS3 IMPLEMENTATION OF MPTCP WITH DIFFERENT DELAY DIFFERENCE BETWEEN TWO SFS.

Delay Difference (ms)	Linux Throughput Gain (%)	NS-3 Throughput Gain (%)
0	94	90
100	89	80
500	23	20

The study in this paper considers two main scenarios, as shown in Table II. The first scenario studies the impact of the distance between the mobile node and the access point (AP) on the performance of the protocol using different reordering solutions. The second scenario is looking at the performance of the system as a function of the packet drop rate (PDR) variation. The proposed solution is implemented and its performance is compared with other existing reorder solutions, D-SACK and TCP-DOOR

TABLE II. SIMULATED SCENARIOS.

	First SF		Second SF	
	Distance (m)	PDR (%)	Distance (m)	PDR (%)
Scenario-1	10 - 100	0.0	10 -100	0.0
Scenario-2	10	0.0	10	0.0 - 0.9

B. *Performance Metrics*

The following performance metrics are used here for the result comparisons and analysis:

- Out-of-Order Ratio: The OOO Ratio (OOO-R) is calculated at the receiver side, and it is the ratio of the total number of received packets being stored in the OOO buffer to the total number of non-duplicate received packets.
- Out-of-Order Buffer Occupancy*:* If an OOO packet arrives at the destination, then the packet will be stored in a buffer awaiting the late packets to arrive. The OOO buffer occupancy (OOO-BO) is used to measure the maximum amount of memory required by different PR recovery methods, and it is defined as the maximum number of packets stored in the OOO buffer during simulation time.
- Link Utilization: The link utilization (LU) is obtained by observing the SF CWND. If the 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 SF over its physical data rate.
- Aggregate Throughput: The gThroughput is defined as the sum of the throughputs of all SFs used for the MPTCP connections.

V. PERFORMANCE EVALUATION

This section presents the performance analysis of the two main scenarios. The first studies the impact of wireless link reliability as a function of the distance between the mobile and AP on the performance of reordering solutions of the MPTCP. The second scenario analyses the performance of the same reordering solutions when the PDR variation increases between SFs. For completeness of discussion, it is worth mentioning that the performance of single TCP over Wi-Fi was also evaluated and found to be maximum 5 Mbps; the results have not been presented but will be referred to for comparison purposes. Note that our proposed method is denoted by PR-R.

A. *Impact of Distance from AP*

In Scenario 1, we are looking at the performance of different reordering solutions of the MPTCP as the distance between the client and both APs increases. As the distance increases, the performance will decrease due to the deterioration of link (lower signal-to-interference-plus-noise ratio (SINR)) in addition to the reordering problem. The study investigates which reordering solution will perform better with a low SINR. The results show that both the TCP-DOOR and PR-R methods outperform others up to 50 m by providing a 64% improvement in gThroughput. Beyond 50 m, PR–R is the best in terms of gThroughput by providing a 50% improvement on average, compared to the NoPR, while D-SACK comes second with a 22.7% improvement on average. Figure 4 shows the performance of the MPTCP with different PR solutions as a function of distance.

The PR-R utilizes both links more than the others along all distances, as illustrated in Table III, which gives results for 50 m and 70 m distances between mobile and both APs as a typical set of results. Table III also shows the OOO-R and the OOO-BO for all simulated PR solutions. It is clear that PR-R requires more buffer size than TCP-DOOR, which provides a comparable throughput improvement for high SINR cases.

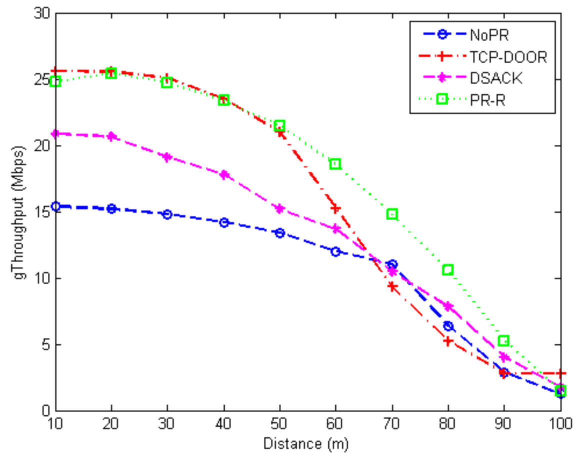

Fig. 4. Scenario 1 gThroughput against distance using different PR solutions.

TABLE III. SCENARIO 1 OOO BUFFER AND LU

Distance (m)	PR Solution	OOO-BO (Kbyte)	OOO-R (%)	LU (%)
50	NoPR	196	5.8	12.26
	D-SACK	84	12.5	14.06
	TCP-DOOR	145	39.8	19.43
	PR-R	442	42.8	19.88
70	NoPR	77	16.8	9.76
	D-SACK	106	17.5	9.67
	TCP-DOOR	126	24.8	8.62
	PR-R	252	61.4	13.69

B. *Impact of Packet Drop Rate*

This section presents the analysis of the impact of the PDR variation between different paths on the performance of MPTCP through Scenario 2. The aim of this performance analysis is to find the most sustainable recovery algorithm against losses. It is essential to test the proposed method (PR R) against losses since it maintains the value of CWND rather than reduces it when the third DUPACK is received. In Scenario 2, TCP-DOOR provides a 62% improvement in gThroughput with lossless case (PDR = 0). However, when the PDR increases, the gThroughput decreases dramatically, as shown in Fig. 5. The PR-R improves the gThroughput of MPTCP by 60% on average and provides a stable performance against the increasing rate of packet drops. The gThroughput measurements of the legacy MPTCP or NoPR do not exceed 15 Mbps, which is the maximum throughput of a single TCP throughput over one link, while D-SACK is the second best after PR-R with 21.5 Mbps at maximum.

Therefore, by suspending the immediate reduction in CWND (unlike NoPR) and retransmitting loss packets directly after receiving the third DUPACK (unlike TCP DOOR), PR-

R fully utilizes both links, as illustrated in Table IV, and outperforms other PR solutions in terms of gThroughput. Table IV represents the low and high PDR cases only as a typical set of results.

In terms of OOO packets and memory requirements, PR R provides the highest OOO-R compared to others. Higher OOO-R with higher gThroughput is desirable in MPTCP because OOO is more likely to happen when receiving packets from different paths that need to be reassembled. When the protocol depends on one path only during the transmission (i.e., NoPR in our scenario), then the OOO-R will be lower. Hence, a preferred reordering solution is one that can deal with OOO packets efficiently without degrading the overall performance of the protocol. It is clearly shown in Table IV, which represents a typical set of results that are also applicable to other cases, that PR-R has the highest OOO-R and requires more buffer space compared to D-SACK, which is the second best in terms of gThroughput.

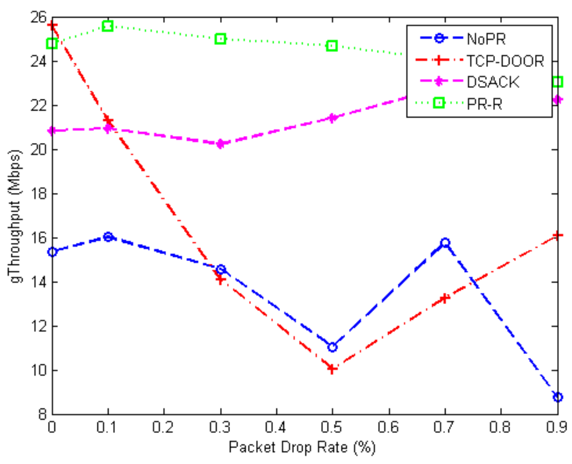

Fig. 5. Scenario 2 gThroughput against PDR of SF 2 using different PR solutions.

TABLE IV. SCENARIO 2 OOO BUFFER AND LU.

PDR (%)	PR Solution	OOO-BO (Kbyte)	OOO-R (%)	LU (%)
0.1	NoPR	128	8.9	14.82
	D-SACK	106	12.6	19.38
	TCP-DOOR	131	22.1	19.71
	PR-R	513	45.2	23.67
0.9	NoPR	68	11.7	8.09
	D-SACK	145	9.0	20.58
	TCP-DOOR	152	10.7	14.89
	PR-R	259	57.9	21.31

By looking at the CWND evolution of both SFs for four PR solutions when PDR is equal to 0.5%, as a typical set of results, it is clearly shown in Fig. 6 that the size of both CWNDs is small when using the MPTCP-NoPR. This is due to the frequent reduction in CWND when three DUPACKs are received with the same data-level sequence number during the same RTT period. Consequently, MPTCP-NoPR is unable to utilize the available bandwidth, even with lossless networks. Similarly, TCP-DOOR suffers from the small size of both CWNDs under high PDR networks because it does not take any action when receiving the third DUPACK and depends on the RTO response only. The results show that PR-R is more robust to the PDR because it triggers a retransmission when receiving the third DUPACK and maintains the value of CWND to avoid an unnecessary reduction of the CWND, which consequently results in a lower transmission rate.

On the other hand, D-SACK uses SACK rather than the accumulative ACK, which is used by NoPR, to acknowledge a discontinuous block of data. In the MPTCP, D-SACK increases the CWND twice when an OOO packet is received. The receiver sends ACK with the DSACK option back to the sender when an OOO packet is received and stored in the OOO buffer. It then sends another ACK later when the OOO packet stored in the OOO buffer becomes in order. Both cases allow the sender to increase the size of CWND. The double increase in the CWND makes the related SF to increase its transmission rate faster than the other SF, which makes the utilization of one SF better than the other.

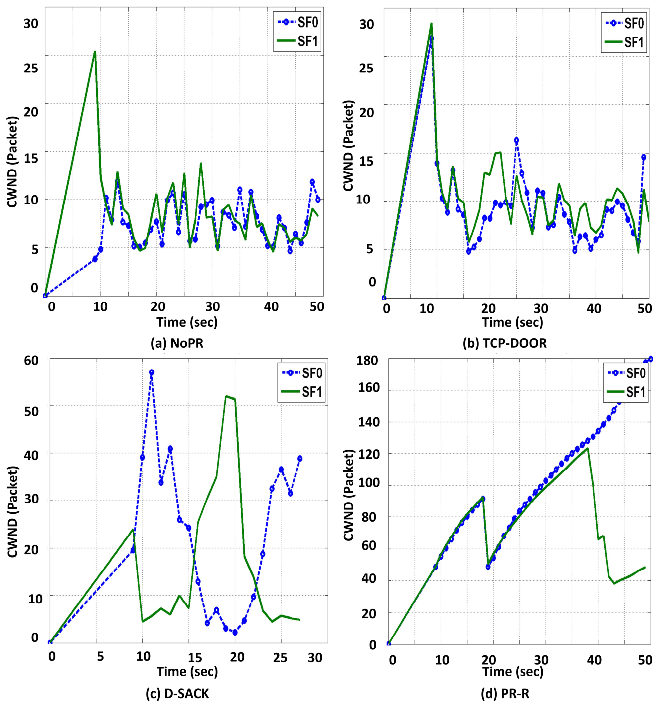

Fig. 6. Scenario 2 CWND evolution for different PR solutions when PDR equal to 0.5%.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a PR-R algorithm to improve the performance of the MPTCP over wireless network interfaces. The proposed method requires simple modifications at the sender side only, which makes its implementation easier than other schemes. The performance of PR-R has been compared to the performance of different PR recovery methods. This study shows that the proposed method adds a substantial improvement to the MPTCP and outperforms others in terms of gThroughput as well as path utilization. The proposed algorithm results in more robustness against packet losses. However, the PR-R requires more memory space, which can be addressed in future work.

REFERENCES

[1] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," *IEFT RFC 6824*, 2013.

[2] H. Hsieh and R. Sivakumar, "pTCP: An end-to-end transport layer protocol for striped connections," *in Proc. IEEE ICNP,* 2002, pp. 24-33.

[3] A. Abd, T. Saadawi, and M. Lee, "LS-SCTP: A bandwidth aggregation technique for stream control ttransmission protocol," *in Computer Communications*, vol. 27, pp. 1012-1024, 2004.

[4] D. Sarkar, "A concurrent multipath TCP and its markov model," *in. IEEE ICC*, 2006, pp. 615-620.

[5] Y. Dong, D. Wang, N. Pissinou, and J. Wang, "Multi-path load balancing in transport layer," *in EuroNGI*, 2007, pp. 135-142.

[6] L. Ka-Cheong, V. O. K. Li, and Y. Daiqin, "An overview of packet reordering in transmission control protocol (TCP): problems, solutions, and challenges," in IEEE Trans. Parallel and Distributed Systems, vol. 18, pp. 522-535, 2007.

[7] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath TCP schedulers," *in Proc. ACM SIGCOMM Workshop on Capacity Sharing Workshop*, 2014, pp. 27-32.

[8] T.-A. Le and L. X. Bui, "Forward delay-based packet scheduling algorithm for multipath TCP," *in arXiv preprint arXiv*:1501.03196, 2015.

[9] T. Kurosaka and M. Bandai, "Multipath TCP with multiple ACKs for heterogeneous communication links.,*" in IEEE CCNC*, Las Vegas, NV, USA, 2015, pp. 613-614.

[10] C. Yu and X. Mingwei, "Dual-NAT: Dynamic multipath flow scheduling for data center networks," *in IEEE ICNP*. 2013, pp. 1-2.

[11] N. Kuhn, E. Lochin, A. Mifdaoui, G. Sarwar, O. Mehani, and R. Boreli, "DAPS: Intelligent delay-aware packet scheduling for multipath transport," *in IEEE ICC*, Sydney, Australia, 2014, pp. 1228-1233.

[12] G. Sarwar, R. Boreli, E. Lochin, A. Mifdaoui, and G. Smith, "Mitigating receiver's buffer blocking by delay aware packet scheduling in multipath data transfer," *in WAINA*, 2013, pp. 1119-1124.

[13] Multipath TCP - Linux kerel implementation. Available: http://mptcp.info.ucl.ac.be/

[14] A. Alheid, D. Kaleshi, and A. Doufexi, "An analysis of the impact of out-of-order recovery algorithms on MPTCP throughput," *in AINA*, 2014, pp. 156-163.

[15] A. Alheid, D. Kaleshi, and A. Doufexi, "Performance evaluation of MPTCP in indoor heterogeneous networks," *in Proc. SIMS*, 2014.

[16] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," *in NSDI*, vol. 11, pp. 8-8, 2011.

[17] R. Costin, P. Christoph, B. Sebastien, F. Alan, H. Michio, D. Fabien, et al., "How hard can it be? designing and implementing a deployable multipath TCP," *in NSDI'12*, 2012, pp. 29-29.

[18] S. Barré, "Implementation and assessment of modern host-based multipath solutions," PhD Thesis, Université catholique de Louvain, 2011.

[19] [19] C. Raiciu, M. Handley, and D. Wischik, "Coupled congestion control for multipath transport protocols," *IETF RFC 6356,* 2011.

[20] B. Chihani and D. Collange, "A multipath TCP model for NS-3 simulator," *in Workshop on NS-3 held in conjunction with SIMUTools* 2011, 2011.

[21] (2012) NS-3 website. [Online]. Available: http://www.nsnam.org/