Performance Enhancement of Multipath TCP with Cooperative Relays in a Collaborative Community

Dizhi Zhou, Peijian Ju, and Wei Song, *Member, IEEE*

Faculty of Computer Science
University of New Brunswick, Fredericton, Canada
Emails: q5frc@unb.ca, i1689@unb.ca, wsong@unb.ca

***Abstract*—Pooling mobile nodes in vicinity as a collaborative community offers an opportunity for multi-homed mobile nodes to enable multipath transmission even when there is no multiple access coverage. However, packet collisions exist within the collaborative community when a contention-based channel access such as IEEE 802.11 is applied. As a result, the overall multipath performance may degrade if a regular multipath transmission protocol is used in a collaborative community consisting of multiple relays. In this paper, we extend the multipath transport control protocol (MPTCP) at the receiver side for a collaborative community. Our proposed extensions, referred to as *Co-MPTCP*, take advantage of fast ACKs and receive buffer sharing at relays. A collaborative community consisting of a root node and multiple relays presents as a *virtual* multi-homed receiver to the sender. Extensive simulations are conducted to demonstrate the effectiveness of our extensions in terms of throughput, goodput, and receive buffer relief.**

***Index Terms*—MPTCP, multipath transmission, cooperative networking, relay, collaborative community.**

I. Introduction

In recent years, the mainstream mobile devices become equipped multiple radio interfaces. Multi-radio mobile devices usually have at least one built-in wireless wide area network (WWAN) interface, such as high speed packet access (HSPA). Also, the multi-radio mobile devices often have one or more short-range wireless network interfaces such as WiFi and Bluetooth. The multi-homed capability of multi-radio devices offers a good opportunity to explore multiple interfaces for multipath transmission, so as to optimize the quality of service (QoS) of bandwidth-intensive applications such as video streaming and video conferencing [1]. As one possible solution, multipath transport control protocol (MPTCP) [2] is standardized by Internet Engineering Task Force (IETF) in 2011. MPTCP runs in multi-homed mobile devices to simultaneously deliver transport control protocol (TCP) packets over multiple interfaces.

However, the WiFi link as a potential transmission path is not ubiquitously available. When there is no WiFi coverage (e.g., within subways), only the WWAN interface is active for wireless access. Hence, multipath transmission may not be always feasible even if the mobile device is equipped with multiple interfaces. Pooling mobile devices in vicinity together as a collaborative community [3] is an alternative way to enable multipath transmission for multi-homed mobile devices even when there is no multiple access coverage. Fig. 1 shows a collaborative community, in which one mobile device, referred to as *root node*, can aggregate the available bandwidths of two nearby mobile devices, referred to as *relay nodes*, by utilizing short-distance communications (e.g., WiFi or Bluetooth). That is, the relay nodes can receive packets on behalf of the root node via their own WWAN interfaces and then forward packets toward the root node via WiFi or Bluetooth.

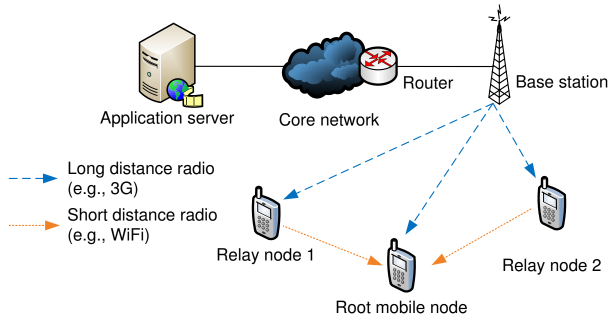

Fig. 1. Multipath with cooperative relays in a collaborative community.

In the literature, some solutions are proposed to enable a collaborative community at the network layer [4], the transport layer [5], and the application layer [6]. One common rationale behind many existing solutions is to hide the internal structure and present the collaborative community as a virtual multi-homed node to the other connection peer. If the relay nodes work as intermediate nodes at the Internet protocol (IP) layer, we can directly apply traditional multipath protocols such as MPTCP. However, the achievable performance of multipath transmission is undermined if the interference among the root node and relay nodes is not addressed well. For example, if IEEE 802.11 is used within a collaborative community, the root node and relay nodes form an ad hoc wireless network sharing a common channel. Because the multiple access control (MAC) in 802.11 is based on carrier sense multiple access with collision avoidance (CSMA/CA), the more relay nodes in a collaborative community, the higher the packet collision probability. Such interference if not addressed properly can limit the achievable performance gain of multipath solutions for a collaborative community. Although multichannel MAC protocols can be used for an ad hoc network [7], additional channel allocation is further required.

This research was supported by a research grant from Natural Sciences and Engineering Research Council (NSERC) of Canada.

In this paper, we focus on a transport-layer solution to better address the interference issue and minimize modification to network stacks. Particularly, we extend the MPTCP protocol [2] for a collaborative community to enable multipath transmission among the root node and cooperative relay nodes. A transport-layer solution is adopted so as to exert more control intelligence to packet streams at the connection endpoints than a network-layer solution. The extended MPTCP solution is not limited to any specific application but provides generic multipath transmission support. As an IETF standard, MPTCP is much easier to deploy than many other multipath solutions [5]. Also, MPTCP follows a layered protocol stack and addresses the TCP-friendly issue with non-MPTCP traffic [8]. To fully utilize the capacities of relay nodes for multipath transmission, we propose the following extensions to MPTCP for a collaborative community:

- Fast ACK via relays: Relay nodes return ACK messages to the sender on behalf of the root node. As such, the interference of packet collisions in the collaborative community is resolved locally without the sender's awareness.
- Receive buffer sharing: To mitigate the out-of-order problem with fast ACK, relay nodes share their receive buffer with the root node. The root node informs each relay of the expected range of packet sequence numbers. Then, relay nodes buffer the packets outside the range until they are requested by the root via a range update message.

To justify our extensions, we evaluate and compare the performance of the modified MPTCP protocol (referred to as *Co-MPTCP*) with that of normal MPTCP over a single multi-homed mobile device. The experimental results show that Co-MPTCP with cooperative relays can achieve the benchmark performance of normal MPTCP over multiple interfaces of a single device. It is also observed that our transport-layer solution greatly outperforms IP-layer relays. The receive buffer at the root node is also significantly offloaded by relays.

The rest of this paper is organized as follows. A background overview of MPTCP is given in Section II. We discuss the proposed extensions to MPTCP in Section III. Experimental results are presented in Section IV. We outline related work in Section V and conclusions in Section VI.

II. Overview of MPTCP

The IETF multipath solution MPTCP adds the capability of using multiple paths simultaneously to a regular TCP session [2]. As shown in Fig. 2, MPTCP loosely splits the transport layer into two sublayers, namely, MPTCP and subflow TCP. Based on this architecture, MPTCP can be easily implemented within current network stack. Subflow TCP runs on each path independently and reuses most functions of regular TCP. The main difference between subflow TCP and regular TCP is that congestion control on each path is delegated to MPTCP sublayer [8]. Although each subflow TCP maintains a congestion window at the source (sender), the congestion window is updated by a cooperative congestion control algorithm which aims to balance the traffic load on each path and improve overall throughput without jeopardizing

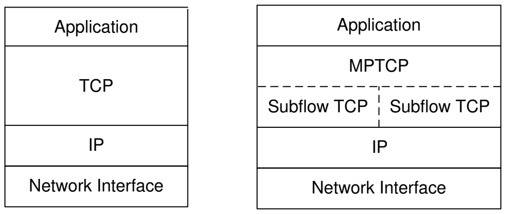

Fig. 2. Comparison of regular TCP and MPTCP protocol stacks.

regular TCP users. On the other hand, each subflow TCP at the sink (receiver) advertises the same receiving window to the source. This receiving window is a shared buffer at the sink so that any subflow from the source can send packets as long as the receiver is able to accept it.

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. In addition to the aforementioned congestion control algorithm, another main function of MPTCP is packet reordering for multiple paths. As each subflow TCP 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 [9] because of mismatched round-trip time (RTT) of multiple paths. Therefore, the source needs to tell the sink how to reassemble the data delivered to the application. MPTCP solves this problem by using two levels of sequence numbers. First, the sequence number for subflow TCP 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. 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. MPTCP Enhancement with Cooperative Relays

In this study, we extend MPTCP to enable multipath transmission with relay nodes in a collaborative community. Initially, MPTCP is designed as an end-to-end protocol without awareness of any intermediate nodes, such as relays in the collaborative community. However, we cannot maximize the performance gain by only running MPTCP at the end nodes, because packet collisions exist within the ad hoc network of the collaborative community. A basic principle of our extensions is to have each relay act as a virtual interface for the root node and share the receiving capability between the root and relay nodes. Fig. 3 illustrates a framework of our MPTCP extensions, referred to as *Co-MPTCP*. Specifically, we develop a fast ACK and receive buffer sharing mechanism.

A. Fast ACK via Relays

As introduced in Section II, MPTCP has two levels of sequence number space: subflow sequence number (SSN) and

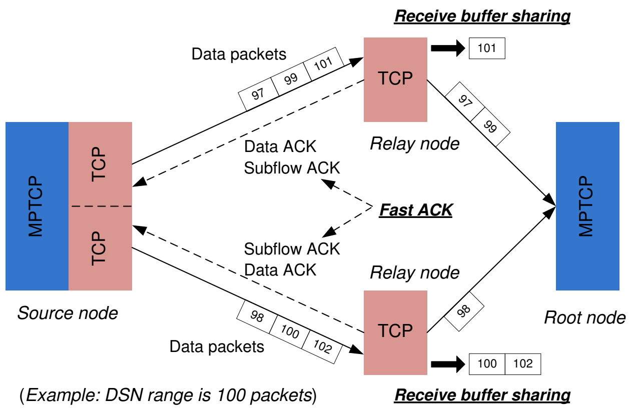

Fig. 3. General framework of Co-MPTCP extensions.

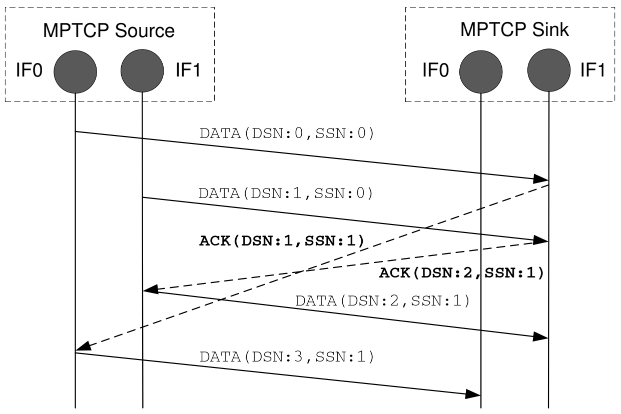

Fig. 4. MPTCP sequence numbers DSN and SSN for acknowledgement.

data sequence number (DSN). Correspondingly, MPTCP has two types of acknowledgement (ACK) messages, `Subflow ACK` and `Data ACK`. `Subflow ACK` is carried in the TCP header with ACK bit set to 1, while `Data ACK` is implemented by data sequence signal (DSS) option. In subflow TCP, the acknowledgement number in `Subflow ACK` specifies the next in-order SSN that the subflow sink expects to receive. In contrast, the acknowledgement DSN number in `Data ACK` only indicates that the sink has successfully received the packet whose DSN is the DSN in `Data ACK` minus one. The MPTCP source sends the next packet based on a scheduler rather than the DSN number in `Data ACK`. If the MPTCP source sends the next packet based on the DSN number in `Data ACK`, duplicate transmissions are possible to cause waste of bandwidth resources.

Fig. 4 shows an example of sequence numbers for MPTCP ACK messages. In this case, `Data ACK` for packet 0 arrives after `Data ACK` for packet 1. Once receiving the `Data ACK` for packet 0 at the interface 0 (IF0), MPTCP source immediately sends out packet 3 instead of packet 1, which is requested in the DSN field of the `Data ACK`. This MPTCP feature can be exploited to avoid unnecessary duplicate transmissions. As a result, it is possible to allow multiple relays to return `Data ACK` messages independently on behalf of the root node. Thus, the interference and packet collisions can be resolved locally within the collaborative community. Transmission errors between the root and relay nodes do not exacerbate the head-of-line blocking problem. In other words, the contention or retransmission overhead do not block transmission between the MPTCP source and relays.

In order to provide timely ACK feedback to the source, MPTCP can be extended to migrate one connection endpoint from the root to relays. After setting up the first TCP subflow with the source, the root sends an address advertisement to the server with the `Add Address` option [10] which contains the address and port number of one or multiple relays. Then the source initiates a "SYN-SYN/ACK-ACK" three-way handshaking and establish one or multiple TCP subflow connections with relays. Once receiving packets from relays, the root sends the source a message with the `Remove Address` option to delete the original subflow between the root and the source. As such, the relays are visible to the MPTCP source as the endpoints and take the responsibility of returning fast ACKs on behalf of the root. This procedure reuses the signalling of MPTCP to facilitate its deployment. Two technical issues need to be handled properly. First, the root should get the address and port number of relays before sending a message with an `Add Address` option to the source. Second, relays need to know the tokens of the root and the source, which are generated during the first subflow establishment. Otherwise, the relays may fail to set up the TCP subflow with the source due to security concerns. We deal with these two issues by requiring the root and relays to exchange tokens and relay information in advance before the address advertisement is sent to the source. The authentication between the root and relays has been addressed in the literature such as [6].

B. Receive Buffer Sharing

In MPTCP, *receive window* specifies the maximum amount of data that can be received and buffered at the sink. The receive window defines the overall receiving capacity of all interfaces of the sink node. Here, we use another term *receive buffer* to represent the storage space required for the sink to accommodate N packets of continuous DSN sequence numbers (called a *DSN block*). Because packets arrived at the sink may be out of order due to multipath transmission, the actual space required to buffer a DSN block can be much larger than the nominal size of N packets. Hence, the receive buffer implies the out-of-order extent of received packets over multiple paths. The larger the DSN gap among received packets, the larger the *receive buffer* required at the sink, and the more likely the upper bound of *receive window* is reached.

In recent years, more and more people own multiple mobile devices, such as smartphones, laptops, and tablets. To relieve the constraint of devices with limited buffer, it offers a benefit to take advantage of the resources of cooperative devices. Hence, we propose Algorithm 1 for receive buffer sharing at the root node, so as to enhance receiving capacity of the root node and mitigate the impact of packet reordering. Once an MPTCP connection is established following three-way handshaking for all interfaces, the root node sends the expected

DSN range, (MIN_DSN, MAX_DSN), for the coming data packets to all relays in the collaborative community. The length of the DSN range is N packets. If any packet within this DSN range is received, the root node accepts and buffers the packet. Meanwhile, the root node counts the number of received packets of continuous DSN falling in the expected range, denoted by M. Given a threshold ρ $(0 < \rho \leq 1)$, when $M/N \geq \rho$, the root node sends update messages for the next DSN range to all relay nodes.

Algorithm 1 Receive buffer sharing at root node.

```
1:  Send DSN range, (MIN_DSN, MAX_DSN), to relays
2:  New packet is received
3:  if $MIN\_DSN \leq DSN \leq MAX\_DSN$ then
4:      Update number of continuous packets in range: $M$
5:  end if
6:  Accept and append new packet to receive buffer
7:  if $M/N \geq \rho$ then
8:      $MIN\_DSN \leftarrow MIN\_DSN + N$
9:      $MAX\_DSN \leftarrow MAX\_DSN + N$
10:     Send update messages for DSN range to relay nodes
11: end if
```

Algorithm 2 defines the buffer sharing algorithm working at each relay. Once a packet within the expected DSN range is received at a relay, it forwards the packet toward the root node. If the packet DSN is smaller than the lower bound of DSN range MIN_DSN, it indicates an urgent out-of-order packet belonging to previous DSN ranges. The relay also forwards such a packet immediately. In contrast, if the DSN of a received packet is larger than the upper bound MAX_DSN, the relay buffers and retains the packet on behalf of the root node. This is because the out-of-order packet is not useful at this moment to compose a consecutive data block that can be passed to the application. The out-of-order packet could take extra space of receive buffer at the root node if immediately forwarded by the relay. Once an update message for DSN range is received, the relay forwards all the buffered packets within the new DSN range to the root node. The DSN range update is based on a threshold ρ maintained at the root node, which indicates the occupancy level of receive buffer and the completeness of a consecutive data block. The threshold can be less than 1 so that certain margin time is reserved for the relays to forward buffered packets to the root. We will see in Section V the effects of the threshold to address data blocking.

IV. Experimental Results and Discussion

To evaluate the performance of the proposed solution, we implement the Co-MPTCP extension with network simulator ns-2 for the scenario in Fig. 1. We create a multichannel base station which is connected to two relays over different channels. Two relays and one root node share a common channel to form a single-channel ad hoc network. The root node receives all packets from two relays via a WiFi interface. The detailed simulation parameters are given in Table I.

Algorithm 2 Receive buffer sharing at relay node.

```
1:  Receive DSN range, (MIN_DSN, MAX_DSN), from root
2:  New packet is received
3:  if $MIN\_DSN \leq DSN \leq MAX\_DSN$ or
4:      $DSN < MIN\_DSN$ then
5:      Forward new packet toward root node
6:  else
7:      if $DSN > MAX\_DSN$ then
8:          if Queue length $K \geq MAX\_BF\_SIZE$ then
9:              Forward head packet of receive buffer to root
10:         end if
11:         Append new packet to the end of receive buffer
12:     end if
13: end if
```

TABLE I
Network parameters for simulation experiments.

Simulation parameter	Sample value
Number of relay nodes	2
Radius of base station	500m
3G downlink speed	8 Mbit/s
WiFi link speed	54 Mbit/s
File size for download	43 Mb (10000 MSS)
DSN update threshold ρ	0.8, 0.9, 1.0
Size of expected DSN range N	100 packets

A. Performance Metrics

In the experiments, we focus on three performance metrics, namely, subflow TCP throughput, MPTCP receive buffer size, and MPTCP goodput. Here, we define the receive buffer size as the required space at the root node to completely reconstruct a N-packet block of continuous DSN. In addition, goodput achievable at the application layer depends on the time for the transport layer to deliver a block of consecutive data segments to the application. We calculate the goodput as follows

$$Goodput_k = \frac{Block\ size\ of\ continuous\ DSN}{Recv\_time_k - Recv\_time_{k-1}}$$

where the numerator is the size of a N-packet block of continuous DSN, and the denominator is the time from when the root node receives the last packet of the $(k-1)^{th}$ block until the root node completely receives the k^{th} block.

B. Experimental Results

First, we investigate the subflow TCP throughput of each path in Fig. 5. As seen, Co-MPTCP and normal MPTCP have similar throughput in both paths. This is because Co-MPTCP limits packet collisions within the collaborative community by having relays to return ACK message on behalf of the root node. Therefore, the MPTCP source is not aware of the interference and not affected by the collision resolution and retransmissions. The source congestion window always matches the receiving capacities of a virtual multi-homed node. In contrast, MPTCP running over IP relays suffers from severe

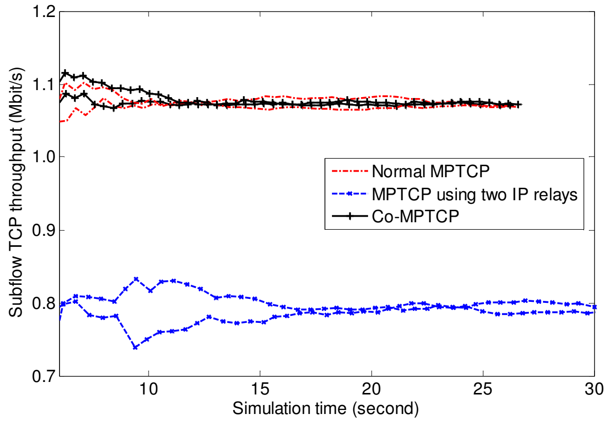

Fig. 5. Subflow TCP throughput.

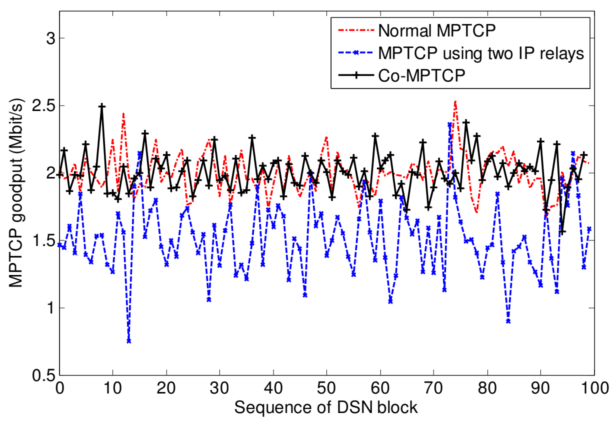

Fig. 7. MPTCP goodput at root node.

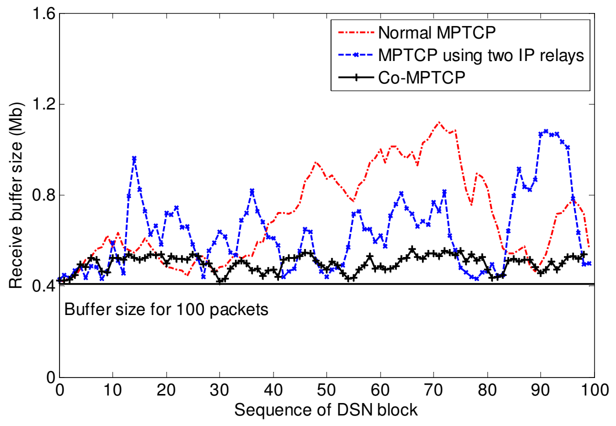

Fig. 6. Receiver buffer size at root node.

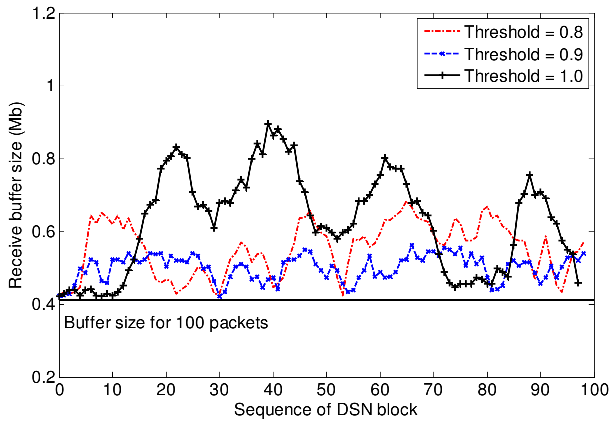

Fig. 8. Impact of DSN range update threshold on receive buffer requirement.

packet collisions and interference between two relay paths. As a result, the throughput on each path is much smaller than that of Co-MPTCP and normal MPTCP.

Fig. 6 compares the required buffer size of three solutions to completely receive a 100-packet block of continuous DSN. The x-axis shows the sequential number of DSN blocks. The horizontal black line shows the size of one DSN block, which is also the minimum receive buffer required to accommodate one block. As expected, the root node always needs a receive buffer larger than the minimum to accommodate out-of-order packets. It is also observed that Co-MPTCP is most stable and requires a receive buffer size closest to the minimum. This is due to the receive buffer sharing algorithm implemented at the root and relay nodes. Packets outside the expected DSN range are filtered and buffered at the relays. As such, Co-MPTCP efficiently increases the receiving capacity of the root node.

Fig. 7 shows the goodput of the three multipath solutions. As a primary objective of Co-MPTCP, a collaborative community is emulated as a virtual multi-homed mobile node by exploiting cooperative relays. We expect that Co-MPTCP approaches the performance of normal MPTCP in both subflow TCP throughput and MPTCP goodput. As seen in Fig. 7, Co-MPTCP achieves similar goodput as normal MPTCP, which significantly outperforms MPTCP over IP relays. By using fast ACK via relays, the sending rate of MPTCP source always matches the real receiving capacity of the collaborative community as a virtual node. Hence, the receiving rate of Co-MPTCP at the root node is very close to that of normal MPTCP. Given the same level of out-of-order packets, a larger goodput is achieved with a higher receiving rate. For MPTCP over IP relays, packet collisions slow down the sending rate of the MPTCP source and the goodput of the root node.

Next, we examine the impact of the threshold for DSN range update. Regarding receive buffer requirement, Fig. 8 shows that a threshold 0.9 is more appropriate for our simulations with Co-MPTCP. If the threshold is too small, e.g., 0.8, the relays forward more packets in the next DSN range, which increases out-of-order packets at the root node. If the threshold is too large, say, 1.0, there is a gap between two DSN blocks so that the root node is blocked for new packets. To achieve a good trade-off, a proper threshold should reserve sufficient time for the root node to send DSN range updates to relays, while each relay is able to forward buffered packets to the root node.

On the other hand, the threshold has a smaller impact on goodput. As shown in Fig. 9, the three multipath solutions achieve similar goodput with different thresholds. The only exception is the spike that occurs in the second block when the threshold is 1.0. In this case, the root sends DSN range update messages to the relays once the last packet of the first block is received. Thus, after forwarding this last packet to the root, the

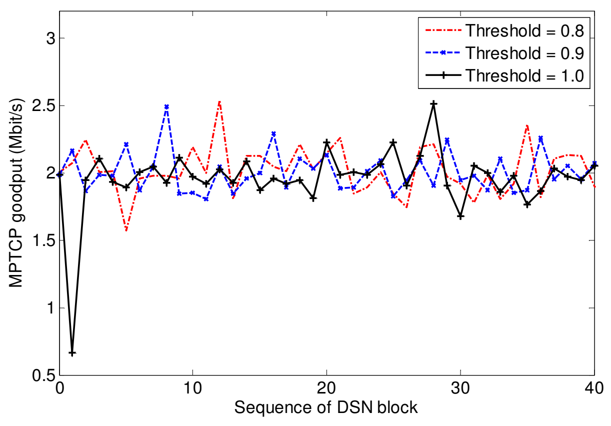

Fig. 9. Impact of DSN range update threshold on MPTCP goodput.

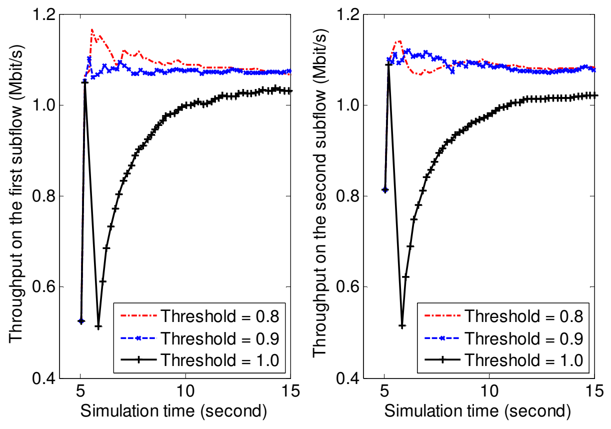

Fig. 10. Impact of DSN range update threshold on subflow TCP throughput.

relay buffers all the received packets within the second block until a DSN range update is received. However, those packets should be immediately forwarded to the root node because the first block is already received at the root node. This behavior only affects the goodput of the second DSN block because an empty receive buffer therein cannot trigger packet forwarding by the relay before a DSN range update is received. Different from goodput, subflow TCP throughput varies with previous states. Given a DSN range update threshold 1.0, the RTT of the first several packets in the second DSN block is larger than the subsequent average value. This triggers the source to reduce the congestion window, which further decreases the subflow TCP throughput, which is shown in Fig. 10.

V. Related Work

To enable multipath transmission with cooperative relays, there have been some existing solutions at different layers. COMBINE [6] is an application-layer scheme at the receiver side for multipath file downloading via relays based on hypertext transmission protocol (HTTP). The ultimate receiver deploys a downloading session at each relay based on how much wireless bandwidth is available and the service cost. Each relay buffers received packets and forwards them to the receiver until the downloading session finishes with all required data. PRISM [5] is a sender-side multipath transport protocol with generic routing encapsulation (GRE) tunnel to transmit packets from the sender to relays. Different from the layered protocol stack of MPTCP, PRISM deploys a TCP proxy for a regular TCP source. The TCP proxy is responsible for all operations related to multipath transmission, such as scheduling packets based on the link states of wireless paths, and detecting duplicate ACKs for out-of-order packets at the receiver. MOPED [4] is a network-layer solution focused on the impact of a collaborative community on network routing.

VI. Conclusions and Future Work

We propose an extension *Co-MPTCP* for MPTCP to enable multipath transmission in a collaborative community, consisting of a root node and multiple relays in vicinity. To emulate a virtual multi-homed node, we develop fast ACK and receive buffer sharing at relays. Fast ACK resolves packet collisions locally in the collaborative community to mitigate source blocking. The buffer sharing algorithms take advantage of relays to buffer out-of-order packets and only forward packets in the expected DSN range to the sink. Our experiments show that Co-MPTCP approaches the performance of normal MPTCP running in a single multi-homed node in terms of subflow TCP throughput and goodput. Co-MPTCP also greatly mitigates the buffer constraint for the root node.

As a key factor for the Co-MPTCP performance, the threshold ρ for the DSN range update depends on the buffer size of each relay, the free space in the root's receive buffer, and the scheduler at the MPTCP source. In our simulations, we determine the threshold experimentally for high performance. In the future, we are interested in exploring its impact in various scenarios and developing an analytical framework to determine the optimal value systematically.

References

[1] W. Song and W. Zhuang, "Performance analysis of probabilistic multipath transmission of video streaming traffic over multi-radio wireless devices," *IEEE Transactions on Wireless Communications*, to appear.

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

[3] P. Sharma, S. Lee, J. Brassil, and K. Shin, "Aggregating bandwidth for multihomed mobile collaborative communities," *IEEE Trans. Mobile Comput.*, vol. 6, no. 3, pp. 1536–1233, Jan. 2007.

[4] C. Carter and R. Kravets, "User devices cooperating to support resource aggregation," in *IEEE Workshop. Mobile Computing Syst. and Appl.*, Aug. 2002.

[5] K. Kim and K. Shin, "Improving TCP performance over wireless networks with collaborative multi-homed mobile hosts," in *Proc. ACM MOBISYS*, Jun. 2005.

[6] G. Ananthanarayanan, V. Padmanabhan, L. Ravindranath, and C. Thekkath, "COMBINE: Leveraging the power of wireless peers through collaborative downloading," in *Proc. ACM MOBISYS*, Jun. 2007.

[7] A. Raniwala and T. Chiueh, "Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network," in *Proc. IEEE INFOCOM*, Mar. 2005.

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

[9] K. Leung, V. Li, and D. Yang, "An overview of packet reordering in transmission control protocol (TCP): Problems, solutions, and challenges," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 4, pp. 522 – 535, Mar. 2007.

[10] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," IETF draft-ietf-mptcp-multiaddressed-08, May. 2012.