Goodput Improvement for Multipath TCP in Cooperative Relay-Based Wireless Networks

Dizhi Zhou, Wei Song*, and Peijian Ju
 Faculty of Computer Science
 University of New Brunswick, Fredericton, Canada
 Emails: {q5frc, wsong, i1689}@unb.ca

Abstract

Multipath transport control protocol (MPTCP) standardized by Internet Engineering Task Force (IETF) offers a promising solution to support simultaneous delivery of transport control protocol (TCP) packets over multiple interfaces of multi-radio mobile devices. Even when there is no multiple access coverage, it is still possible to enable multipath transmission by involving mobile devices in vicinity as cooperative relays. However, the available bandwidth provided by relays can be highly varying due to wireless channel fading or dynamic local traffic load. Although MPTCP can pool the available bandwidths of multiple paths, it is challenging to address the dynamics of the cooperative wireless network and ensure a stable performance. In this paper, we extend MPTCP for the wireless network with cooperative relays by an integrated solution in three aspects, namely, subset sum-based relay selection (SSRS), adaptive congestion control (ACC) and differentiated packet forwarding (DPF). Extensive simulations with the Long Term Evolution (LTE) network in NS-3 show that the proposed modules can achieve a stable aggregate throughput by engaging a small number of relays and significantly improve the goodput in various scenarios.

Index Terms

MPTCP, LTE, relay selection, subset sum, congestion control, cooperative wireless networks.

I. INTRODUCTION

The mainstream multi-radio mobile devices usually have at least one built-in wireless wide area network (WWAN) interface, such as Long Term Evolution (LTE), as well as short-range wireless network interfaces such as Wi-Fi. The multi-homed capability offers a good opportunity to explore multiple interfaces for

*Correspondence author. This work was supported by Natural Sciences and Engineering Research Council (NSERC) of Canada. It was partially reported in PIMRC 2012 and CCNC 2013 (received Best Student Paper Award).
multipath transmission. The multipath transport control protocol (MPTCP) [1] standardized by Internet Engineering Task Force (IETF) can run in multi-homed mobile devices to simultaneously deliver transport control protocol (TCP) packets over multiple interfaces. Nonetheless, it is challenging to support a stable quality of service (QoS) in the wireless environment, since the LTE link can be highly varying within the coverage of the Evolved Node B (eNB). Especially, the mobile devices at the border of a cell may have a low-quality LTE connection. A promising technique addressing this problem is to pool mobile devices in vicinity together as a cooperative community [2], so that multipath transmission can still be enabled when the direct LTE link is not available or in a poor condition.

Fig. 1 shows a user cooperation scenario with three user equipment (UE) associated with an eNB. In this case, UE 3 is the destination that receives data from the application server (source). Two nearby relays (UE 1 and UE 2) can receive packets on behalf of the destination (UE 3) via their own LTE links and then forward the packets toward the destination via Wi-Fi links. By this means, the destination can run multiple paths with MPTCP and each path corresponds to a relay. Since each MPTCP subflow is identified by a pair of source and destination IP addresses, the destination needs to have multiple IP addresses for its Wi-Fi interface to establish multiple subflow connections with the source. Fortunately, a virtual interface technique [3] can be used to configure multiple IP addresses to one Wi-Fi interface. Henec, it is not required for the destination to be equipped with multiple Wi-Fi interfaces.

Due to wireless channel fading or varying local traffic load, the available bandwidth provided by a relay is highly dynamic. Here, available bandwidth refers to the bandwidth that a relay provides to the destination. Then, there comes up an essential problem that, given distinct and varying available bandwidths
of relays, how MPTCP guarantees a stable aggregate throughput to the application layer of the destination. The aggregate throughput is the aggregated bandwidth across multiple paths that is made available and achieved by the destination. The effective throughput of in-order packets provided by MPTCP, referred to as goodput, reflects the actual application-layer perceived throughput. As the end-to-end delay of each relay path may be varying greatly, a substantial amount of out-of-order packets can be received at the destination [4]. It is challenging for the cooperative relay-based LTE network to guarantee not only a stable aggregate throughput but also an improved goodput. In this paper, we propose an integrated solution and address this problem in three aspects, namely, subset sum-based relay selection (SSRS), adaptive congestion control (ACC) and differentiated packet forwarding (DPF). The contribution of this paper is several-fold:

- First, we investigate MPTCP in the cooperative relay-based LTE network, which is subject to a more dynamic environment than traditional wired and wireless networks. Two key performance metrics, aggregate throughput and goodput, are considered.
- Second, we propose an integrated solution to enhance MPTCP in three aspects so that a stable aggregate throughput is guaranteed and goodput is improved for the destination. The three modules can work independently and are also complementary to enhance MPTCP in a broad range of scenarios.
- Third, we conduct extensive simulations with various relay conditions to evaluate the proposed solution. The simulations follow a complete LTE and Wi-Fi protocol stack and an adapted MPTCP implementation in NS-3.

The remainder of this paper is organized as follows. In Section II, we introduce the related work and MPTCP background. The proposed MPTCP extensions are presented in Section III. Simulation results are provided in Section IV to demonstrate the improved performance in terms of aggregate throughput and goodput. Conclusions and future work are discussed in Section V.

II. RELATED WORK

Several TCP-based multipath transmission protocols have been proposed, such as pTCP [5] and mTCP [6], aiming to aggregate the bandwidths of multiple paths. Due to incompatibility with the dominant TCP, these protocols are not widely deployed. MPTCP [1,7] is an IETF solution standardized in 2011. It extends regular TCP to add the capability of using multiple paths simultaneously for an end-to-end
connection. Following a layered structure similar to the TCP protocol stack, MPTCP can be easily deployed within the existing networks. MPTCP loosely splits the transport layer into two sublayers, namely, MPTCP and subflow TCP. Subflow TCP runs on each path independently and reuses most functions of regular TCP. Although each subflow TCP maintains a congestion window at the source, a coupled congestion control function [8] is delegated to MPTCP sublayer. To reassemble data before delivering to the application, MPTCP uses two levels of sequence numbers. First, subflow sequence number (SSN) for subflow TCP works independently within each subflow and ensures that data packets of each subflow are successfully transmitted to the destination in order. Second, data sequence number (DSN) at the MPTCP level is unique for each packet received at the destination. The destination can sequence and reassemble packets from different paths by DSN. Moreover, MPTCP sublayer is responsible for path management that discovers, adds and deletes subflows for the multipath connection. MPTCP supports such operations by defining new options in the header [7], such as Add Address for announcing additional addresses to reach a host, and Remove Address for removing invalid addresses and instigating closure of the affected subflows.

As MPTCP is designed as a generic multipath transport-layer protocol, it may not perform well in cooperative wireless networks given some unique problems such as wireless channel variation and dynamics of relay paths. Relay selection also becomes a critical problem to optimize MPTCP performance. Although relay selection has been extensively studied from the physical-layer and link-layer perspectives for cooperative wireless networks [9]–[12], such solutions may not guarantee a stable end-to-end performance with MPTCP. The best relays that provide the highest throughput may have disparate end-to-end path delays. As a result, out-of-order packets can cause a large reordering delay that jeopardizes the goodput perceived at the application layer. Obviously, such relays may not always be the best relays for the transport layer.

Although MPTCP uses a data sequence number at the connection level to re-order the received packets at the destination, MPTCP does not provide any mechanism to reduce the number of out-of-order packets in a small data block. There are some recent studies that address this problem to improve the goodput of MPTCP [13]–[15]. In [13], MPTCP goodput is enhanced by appropriately selecting the path for packet retransmission. When the slow path is blocked by a full receive buffer with too many out-of-order packets, the source retransmits packets over the fast path. As this scheme is only triggered when the receive buffer
is full, it cannot handle goodput degradation in normal transmission stages. The solutions in [13] and [14] use network coding to recover packet loss at the destination and in turn increase the goodput. In such coding-based schemes, the source transmits the original data in one subflow and the linear combinations of such data in the other subflow. Thus, the data redundancy of network coding is exploited to recover lost and delayed packets. However, these schemes require the support of network coding in both endpoints. Moreover, these schemes only improve average goodput over a long term. A stable goodput with minimal variation is more favorable for real-time applications.

III. PROPOSED MPTCP EXTENSIONS FOR COOPERATIVE RELAY-BASED LTE NETWORKS

In this section, we present our proposed extensions to MPTCP for the cooperative relay-based LTE network in Fig. 1, which effectively aggregate the resources of paths to guarantee a stable throughput and improve the application-layer perceived goodput. First, subset sum-based relay selection (SSRS) running at the destination dynamically updates the relay set to ensure that the aggregate throughput satisfies an application-layer target bit rate (TBR) requirement.Second, adaptive congestion control (ACC) is proposed to extend the coupled congestion control algorithm of MPTCP, so that the end-to-end delay variation of multiple paths is mitigated to enhance the goodput. Third, differentiated packet forwarding (DPF) is designed to complement ACC, since the path delay variation cannot be eliminated merely by adapting the traffic load of each path via ACC. It temporarily buffers out-of-order packets in the relays and selectively forwards packets to the destination. In a nutshell, SSRS can be seen as the foundation to select qualified relay paths, while ACC and DPF work on top of SSRS to ensure a stable aggregate throughput and maximize the achievable goodput. ACC and DPF can work independently or complementarily.

A. Subset sum-based relay selection

In Fig. 2, the SSRS module functions with the following path management procedure. At the beginning of an application session, the relay selection component in SSRS acquires the desired TBR from the application client, and the relay manager retrieves the available bandwidths of relays from their periodical broadcasts at a frequency $F$. Each relay client estimates the available bandwidth based on its average local traffic throughput. As illustrated in Fig. 1, the relays forward packets from
the source to the destination via Wi-Fi links, which have a higher transmission rate than the LTE links between the eNB and the relays. Hence, we assume that the LTE links are the bottleneck links of the paths. The achievable aggregate throughput at the destination is then limited by the LTE bandwidths available to the relays. Moreover, it is worth mentioning that one relay can provide the relaying service for multiple destinations. In this case, the relay will receive more than one bandwidth request message from these destinations. Hence, the relay needs to decide how to allocate the available bandwidth to each destination.

Once the relay manager obtains the available bandwidths of relays, it can determine all feasible relay sets whose total available bandwidths are within an acceptable TBR range $[(1 - \sigma)\text{TBR}, (1 + \sigma)\text{TBR}]$, where $\sigma$ (0 < $\sigma$ < 1) is a small ratio such as 0.1. Due to the wireless fading channel and varying local traffic load at relays, the aggregate throughput achieved at the destination can fluctuate dynamically. It is challenging to provide a deterministic guarantee on the aggregate throughput. Hence, we aim to minimize the outage probability that the aggregate throughput falls beyond the acceptable range.

Based on the retrieved available bandwidths of relays, a best relay set can be selected according to certain criteria and configured as the *active set*. The active set should be forwarded to the *path manager* at the application layer, which further calls MPTCP socket APIs to add the corresponding subflows for...
the destination [16]. According to updates of available bandwidths, the relay selection component needs to refresh feasible relay sets. If the new best relay set is different from the current active set, it is chosen as the backup set. When the total available bandwidth of the active set is found out of the TBR range, the path manager is triggered to migrate the current active set to this backup set. Specifically, new subflows are added first, whereas the deletion of old subflows starts after a maximum round-trip-time (RTT) of the active paths. The RTT monitor measures the RTT of each active path and provides such information to the path manager. The main consideration of postponing path deleting operations is to ensure that the destination waits to receive the packets on the fly over the paths to be deleted.

Given the available bandwidths of \( K \) relays, the determination of feasible subsets is the well-known subset-sum problem, which is proved to be NP-complete. Given a set of \( K \) elements, there are totally \( 2^K \) possible subsets so that the searching scale is exponential. Fortunately, a polynomial-time approximation algorithm is available to “trim” subsets that have sums sufficiently close to neighbouring subsets [17]. In Algorithm 1, we adapt the original approximation to find relay subsets whose total available bandwidths fall into the TBR range.

Given \( K \) relays, we use \( r_i \) to denote the available bandwidth of relay \( i \) (\( 1 \leq i \leq K \)) and define \( R = \{r_1, r_2, ..., r_K\} \). All total available bandwidths no greater than \((1 + \sigma)\text{TBR}\) of relays \( \{1, 2, ..., i\} \) are denoted by \( L_i = \{S_{i1}^i, S_{i2}^i, ..., S_{|L_i|}^i\} \), where \( S_{ij}^i \leq (1 + \sigma)\text{TBR} \), \( 1 \leq j \leq |L_i| \). All subsets of relays \( \{1, 2, ..., i\} \) that have a total available bandwidth \( S_{ij}^i \) are denoted by

\[
\mathcal{U}_j(S_{ij}^i) = \{X = \{n_1, n_2, ...\}| \sum_{k \in X} r_k = S_{ij}^i\}
\]  

where the subset \( X \) contains the relays \( \{n_1, n_2, ...\} \). Algorithm 1 shows the iterative procedure to obtain feasible relay subsets. In each round, a new bandwidth set \( L_i' \) is defined by adding the available bandwidth \( r_i \) of a new relay \( i \) to each element of the previous available bandwidth set \( L_{i-1} \). As given in Line 4 of Algorithm 1, \( L_i' = \{S'^i_1, S'^i_2, ..., S'^i_{|L_i'|}\} \), where \( S'^i_j = S'^i_{j-1} + r_i \) for any \( 1 \leq j \leq |L'_i| \). That means, \( L_i' \) lists the total available bandwidths of subsets of relays \( \{1, 2, ..., i\} \), and those subsets must include relay \( i \). The corresponding subsets of relays \( \mathcal{U}_j'(S'^i_j) \) are also updated by adding the new relay \( i \) to each subset. Next, we merge the previous set \( L_{i-1} \) and the new set \( L_i' \), and then sort the combined set in a descending order. All elements greater than the TBR upper bound are removed because they are definitely greater than the
**Algorithm 1** Subset sum-based relay selection.

1: \( L_f = \{\text{null}\} \)
2: \( L_0 = \{0\} \)
3: for \( i = 1 \) to \( K \) do
4: \( L'_i = \{S'_i, S'_2, \ldots, S'_{|L'_i|}\} \), where \( S'_j = S_{j-1}^i + r_i \) // Consider a new relay \( i \) of available bandwidth \( r_i \)
5: for \( j = 0 \) to \( |L_i| \) do
6: \( \bigcup_j^i (S'_j) \leftarrow \{X = \{n_1, \ldots, i\} | \sum_{k \in X \setminus i} r_k = S_{j-1}^i\} \)
7: end for
8: \( L_i = \text{MergeSort}(L_{i-1}, L'_i) = \{S_0^i, S_1^i, \ldots, S_{|L_i|}^i\} \) // Merge \( L_{i-1} \) and \( L'_i \), and sort combined set in descending order
9: for \( j = 0 \) to \( |L_i| \) do // Remove all elements greater than TBR upper bound
10: if \( S_j^i > (1 + \sigma)\text{TBR} \) then
11: Remove \( S_j^i \) from \( L_i \)
12: else if \( (1 - \sigma)\text{TBR} \leq S_j^i \leq (1 + \sigma)\text{TBR} \) then
13: if \( S_j^i \notin L_f \) then
14: \( L_f \leftarrow L_f \cup S_j^i \)
15: end if
16: else
17: break
18: end if
19: end for
20: Trim\(\left(L_i, \varepsilon/(2K)\right)\) // Remove elements sufficiently close to reduce search space
21: end for
22: Return \( L_f = \{S_1^K, S_2^K, \ldots, S_{|L_f|}^K\} \) and \( \bigcup_j^K (S_j^K) \) for all \( 1 \leq j \leq |L_f| \)

TBR upper bound in the next round. All elements that fall into the TBR range are further trimmed by introducing an approximation parameter \( \varepsilon \) (\( 0 < \varepsilon < 1 \)). Given two neighbouring elements \( S_j^i \) and \( S_{j+1}^i \), if they are sufficiently close, i.e., they satisfy

\[
\frac{S_{j+1}^i}{1 + \varepsilon/(2K)} \leq S_j^i \leq S_{j+1}^i
\]

then \( S_{j+1}^i \) is removed from \( L_i \). Actually, \( \varepsilon \) is an indicator of the variance of the approximation result from the optimal solution. To ensure a total bandwidth in the range \([(1 - \sigma)\text{TBR}, (1 + \sigma)\text{TBR}]\), it is natural to set \( \varepsilon = \sigma \). Compared with the original algorithm in [17], Algorithm 1 does not increase the search space, so that the running time remains polynomial in both \( 1/\varepsilon \) and \( K \).

Based on the feasible relay subsets obtained from Algorithm 1, a best relay set is selected according to two main factors, i.e., the difference between TBR and the total available bandwidth of a relay subset
(denoted by $\Delta B$), and the number of relays (denoted by $K_r$). The two factors are normalized with the max-min method to $\alpha_B$ and $\alpha_K$. Then, a priority index $\gamma = \frac{1}{\alpha_B + \alpha_K}$ is defined to select the relay subset with the highest index as the active set.

**B. Adaptive congestion control**

SSRS can achieve a stable aggregate throughput by properly selecting and updating the relay sets. Out-of-order packets are still possible since the RTTs of subflows can be significantly different. Let us first examine two special cases to find out the primary factor affecting the goodput. Given two available paths, let $\tau_i$ denote the packet sending interval at the source for path $i$, $i = 1, 2$. Denoting the end-to-end delay of path $i$ by $d_i$, we assume that $d_1 < d_2$. Consider a block of $N$ packets with continuous DSNs, among which $N - 1$ packets are received on path 1 and only 1 packet is from path 2. Such a block of data packets is referred to as an in-order block. Let $S$ and $T$ denote the total size in the unit of maximum segment size (MSS) and the total receiving time of an in-order block, respectively. We define the goodput as the data throughput of in-order packets forwarded by MPTCP to the application layer, given by

$$Goodput = \frac{\text{Size of } N \text{ in-order packets}}{\text{Total receiving time of } N \text{ packets}}.$$  \hspace{1cm} (3)

Thus, we have the goodput $G = S/T$.

It is worth noting that the in-order block size is important for the calculation of goodput. Intuitively, a larger block size results in a higher goodput. Nonetheless, a longer reassembling delay is also involved with re-ordering the packets in the block. Many applications, especially real-time applications, fetch packets from the receive buffer in a certain frequency to maintain a smooth QoS. In [18], the authors consider a video streaming application and study how to retrieve packets from the TCP receive buffer in a certain rate in order to avoid video interruption. Hence, a reasonable block size should be used to evaluate goodput.

Consider two special cases in Fig. 3. Suppose that packets 1 and 2 are sent at the same time to paths 1 and 2, respectively. Fig. 3(a) shows the case with $\Delta D \triangleq |d_2 - d_1| > \tau_1$. We can easily obtain $T = \Delta D$ and the goodput, given by

$$G = \frac{S}{T} = \frac{\tau_2/\tau_1 + 1}{\Delta D}. \hspace{1cm} (4)$$

In this case, there is only one packet received from the slow path within a block, while the number of in-order packets received over the fast path is $\tau_2/\tau_1$. Fig. 3(b) shows another case with $\Delta D \leq \tau_1$, in
which the destination needs less time to receive all packets within the in-order block. In this case, we have $T = \tau_2$ and the goodput is obtained as

$$G = \frac{S}{T} = \frac{\tau_2}{\tau_1} + \frac{1}{\tau_2}. \quad (5)$$

Actually, Eq. (5) is also the aggregate throughput (denoted by $\Upsilon$) over two paths. That is,

$$\Upsilon = \frac{1}{\tau_1} + \frac{1}{\tau_2}. \quad (6)$$

This observation implies that goodput is inversely proportional to the end-to-end path delay difference $\Delta D$. Therefore, we need to minimize the end-to-end delay difference among end-to-end paths.

It is known that a high traffic load or congestion can lead to a long path delay due to packet queueing and retransmissions. A natural idea to mitigate delay difference is to reduce the traffic load over a slow path by shrinking its congestion window. Hence, we complement MPTCP with adaptive congestion control (ACC) in Algorithm 2 so as to enhance the goodput. Here, we monitor the ratio of the maximum path delay over the minimum path delay, referred to as delay ratio. The source periodically measures the RTT of each path at a frequency $F$. ACC is triggered only when a large delay ratio is detected, i.e., $\theta$ falls into a certain range $[\theta_{\min}, \theta_{\max}]$. As seen in Line 6 of Algorithm 2, the congestion window (denoted by $cwnd_i$) of path $i$ with the maximum delay is decreased proportionally to the delay ratio $\theta$. Since a larger delay ratio indicates that the high-delay path is overloaded, its congestion window needs to be decreased to relieve traffic and reduce path delay. Here, $\theta_{\max}$ is used to avoid over-blocking slow path and severely jeopardizing aggregate throughput. Meanwhile, the TCP slow start threshold ($ssthresh_i$) is updated with

Fig. 3. MPTCP examples with two transmission paths for goodput analysis.
Algorithm 2 Adaptive congestion control.

1: if $\theta_{\text{min}} \leq \theta \leq \theta_{\text{max}}$ then \hspace{1em} // High delay ratio detected
2: \hspace{1em} $i = \arg \max_p (\text{end-to-end delay of path } p)$ \hspace{1em} // Select path $i$ of maximum delay for $\text{cwnd}$ adaptation
3: \hspace{1em} if $\text{count}_i < m$ then \hspace{1em} // Adaptation counter does not exceed maximum limit
4: \hspace{2em} $\text{elapse}_i = \text{Elapsed time duration since creation of subflow}$
5: \hspace{2em} if $\text{elapse}_i > \Gamma$ then \hspace{1em} // Adjust $\text{cwnd}$ for subflow running for a duration greater than $\Gamma$
6: \hspace{3em} $\text{cwnd}_i \leftarrow \frac{\text{cwnd}_i}{\theta}$ \hspace{1em} // Decrease congestion window of path $i$
7: \hspace{3em} if $\text{ssthresh}_i > \text{cwnd}_i$ then
8: \hspace{4em} $\text{ssthresh}_i = \text{cwnd}_i$
9: \hspace{3em} end if
10: \hspace{3em} $\text{count}_i \leftarrow \text{count}_i + 1$
11: \hspace{2em} end if
12: \hspace{1em} else
13: \hspace{2em} $\text{count}_i = 0$ \hspace{1em} // Reset adaptation counter
14: \hspace{1em} end if
15: end if

the new $\text{cwnd}_i$ if $\text{ssthresh}_i > \text{cwnd}_i$. Otherwise, $\text{cwnd}_i$ of the slow path would be recovered quickly with the slow start procedure and not suppressed for enough time to reduce the delay.

Since a variety of factors in addition to traffic load contribute to the path delay, adjusting $\text{cwnd}$ itself cannot eliminate the delay variation. Thus, we introduce $\text{count}_i$ to restrict the number of continuous reductions of congestion window for a single path $i$ by $m$. As such, we can avoid severe throughput degradation on an individual path. Moreover, ACC is not activated before a subflow has run for a period $\Gamma$, because the end-to-end delay measurements for a new path cannot accurately reflect the real congestion situation immediately. In practice, $\Gamma$ can be set to several multiples of the maximum RTT to make sure that the subflow to adjust is already in the congestion avoidance phase.

C. Differentiated packet forwarding

Since it is not possible to avoid out-of-order packets proactively via ACC, differentiated packet forwarding (DPF) takes advantage of the role of relays to cope with packets that are not arriving in order. As given in Algorithms 3 and 4, DPF is implemented in a distributed fashion at the destination and relays. On the destination side, the $\text{DPF manager}$ sends an expected DSN range of $N$ packets, $(\text{MIN}_\text{DSN}, \text{MAX}_\text{DSN})$, for the coming data packets to selected relays. When DPF is working with
Algorithm 3 Differentiated packet forwarding in destination.

1: Send DSN range, \((\text{MIN\_DSN}, \text{MAX\_DSN})\), to relays

2: New packet is received

3: if \(\text{MIN\_DSN} \leq \text{DSN} \leq \text{MAX\_DSN}\) 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: \(\text{MIN\_DSN} \leftarrow \text{MIN\_DSN} + N\)

9: \(\text{MAX\_DSN} \leftarrow \text{MAX\_DSN} + N\)

10: Send update messages for DSN range to relays

11: end if

Algorithm 4 Differentiated packet forwarding in MPTCP relay.

1: Receive DSN range, \((\text{MIN\_DSN}, \text{MAX\_DSN})\), from destination

2: New packet is received

3: if \(\text{MIN\_DSN} \leq \text{DSN} \leq \text{MAX\_DSN}\) or

4: \(\text{DSN} < \text{MIN\_DSN}\) then

5: Forward new packet toward destination

6: else

7: if \(\text{DSN} > \text{MAX\_DSN}\) then

8: if Queue length \(K \geq \text{MAX\_BF\_SIZE}\) then

9: Forward head packet of receive buffer to destination

10: end if

11: Append new packet to the end of receive buffer

12: end if

13: end if

SSRS, a stable aggregate throughput around TBR can be achieved by selecting and switching relay sets. Further considering the periodical monitoring frequency \(F\), we set \(N = \text{TBR}/F\). If any packet within this DSN range is received, the destination accepts and buffers the packet. Meanwhile, the destination counts the number of received packets of continuous DSNs in the expected range, denoted by \(M\). The ratio \(M/N\) indicates the occupancy level of receive buffer and the completeness of an in-order block. Given a threshold \(\rho\) \((0 < \rho \leq 1)\), when \(M/N \geq \rho\), the destination sends update messages for the next DSN range to relays. The threshold \(\rho\) can be less than 1 so that certain margin time is reserved for the relays to forward buffered packets to the destination.
Algorithm 4 defines DPF at the DPF client on the relay side. Once a packet within the expected DSN range is received at a relay, it forwards the packet toward the destination. If the packet DSN is smaller than the lower bound \( \text{MIN}_\text{DSN} \), it indicates an urgent out-of-order packet belonging to a previous DSN range. The relay also forwards the packet immediately. In contrast, if the packet DSN is larger than the upper bound \( \text{MAX}_\text{DSN} \), the relay buffers and retains the packet on behalf of the destination. Once a DSN range update is received, the relay forwards all buffered packets within the new range to the destination.

There are two main reasons to consider buffering packets at the relays as in Algorithm 4. It is known that the bandwidth resources in wireless networks are rather limited and unstable. Due to a large number of users within a cell, e.g., in urban areas, the LTE link often has an even lower rate than the Wi-Fi link and is thus considered as the bottleneck of the end-to-end path. Hence, the source should deliver packets to the relays as soon as possible so that the relays can make use of the LTE bandwidth opportunistically when the LTE link is in good conditions. In addition, the MPTCP source in many cases is a public application server, which serves thousands of connections. Hence, it may consume a large amount of memory space if the packets are always buffered at the source. Therefore, it is a promising solution to take advantage of the buffer space of the relays to deal with the highly dynamic wireless environment and balance the service requirement for the source.

As seen in Algorithm 4, the relay is required to support a packet forwarding function. It can be extended from the personal hotspot feature, which is supported in mainstream mobile systems such as iOS and Android. Because the relay forwards the packets to the destination via the contention-based Wi-Fi link, collisions can happen and result in packet loss. As a consequence, the source may reduce the congestion window unnecessarily. In addition, although DPF handles out-of-order packets via selective forwarding, one side effect is that the ACK messages of buffered packets are delayed, which may also cause subflow TCP to unnecessarily decrease the congestion window. Hence, DPF migrates one subflow endpoint from the destination to the relays. After the first subflow is established between the source and the destination, the destination sends an address advertisement to the source with an Add Address option [7], which contains the addresses and the port numbers of one or multiple relays. Then, the source initiates the handshaking to set up one or multiple subflow connections with the relays. Once receiving packets from the relays, the destination sends a message to the source, which contains a Remove Address option [7] to
trigger the closure of the original subflow between the source and destination. As such, the relays become visible to the source so that the endpoints at the relays can take the responsibility of returning ACKs on behalf of the destination.

The above procedure exploits the signalling of MPTCP to add the relays. Nonetheless, the security mechanism of MPTCP poses another technical issue that needs to be handled properly. According to MPTCP [7], the connection initiation begins with a SYN, SYN/ACK, ACK exchange on a single path. This MPTCP handshake negotiates the cryptographic algorithm and declares the sender’s and receiver’s 64-bit keys, which are used to authenticate the addition of future subflows. The cryptographic hash of the key is a 32-bit token as a local unique connection identifier. When a new subflow is started, the SYN, SYN/ACK and ACK packets also carry an MP_JOIN option, which contains the receiver’s token to identify the MPTCP connection it is joining, and the sender’s 64-bit truncated message authentication code (MAC) for authentication. As seen, the relays joining the MPTCP connection need to know the tokens of the source and the destination, which are generated during the first subflow establishment. Also, the destination should obtain the addresses and port numbers of the relays before sending an address advertisement to the source with an Add Address option. Hence, it is required that the destination and relays exchange address information and tokens in advance via additional signalling.

IV. SIMULATION RESULTS AND DISCUSSIONS

To evaluate the performance of the proposed modules, we implement SSRS, ACC and DPF in the network simulator NS-3 [19] for the scenario of Fig. 1. Our contribution to the LTE module in NS-3 has been validated [20] and incorporated in the official release. We also implement the core functions of MPTCP such as socket APIs, coupled congestion control, path management, and packet scheduler. The detailed simulation parameters are given in Table I, which are selected by referring to the LTE standard [21] and IEEE 802.11a. Specifically, we consider an eNB connected to 11 UEs, among which there are 1 destination and 10 relays. These UEs are uniformly distributed within a rectangle area of a distance 800 – 1000 meters to eNB. A pedestrian-speed fading channel trace with 60 seconds is used to simulate the varying bandwidth of the wireless channels to UEs. The simulation time is sufficiently long to ensure that the results are collected when the system enters a stable state.
TABLE I
SYSTEM PARAMETERS FOR SIMULATIONS.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit power</td>
<td>eNB: 30 dBm, UE: 5dB</td>
<td>Noise figure</td>
<td>eNB: 5 dB, UE: 5dB</td>
</tr>
<tr>
<td>eNB scheduler</td>
<td>Blind equal throughput</td>
<td>Transmission time interval (TTI)</td>
<td>1 ms</td>
</tr>
<tr>
<td>Radio link control (RLC) mode</td>
<td>Acknowledge mode (AM)</td>
<td>Adaptive modulation &amp; coding (AMC)</td>
<td>PiroEW2010 [22]</td>
</tr>
<tr>
<td>Number of resource blocks (RBs)</td>
<td>50</td>
<td>Fading channel trace</td>
<td>Pedestrian at 3 km/h</td>
</tr>
<tr>
<td>Wi-Fi link</td>
<td>IEEE 802.11a</td>
<td>Wi-Fi transmission rate</td>
<td>54 Mbit/s</td>
</tr>
<tr>
<td>Wi-Fi Fragmentation threshold</td>
<td>2200 bytes</td>
<td>Wi-Fi RTS/CTS threshold</td>
<td>2200 bytes</td>
</tr>
<tr>
<td>TBR</td>
<td>3 Mbit/s</td>
<td>Simulation time</td>
<td>50 seconds</td>
</tr>
<tr>
<td>SSRS parameters</td>
<td>$\varepsilon = 0.1, \sigma = 0.1$</td>
<td>ACC parameters</td>
<td>$\theta_{\text{min}} = 1.2, \theta_{\text{max}} = 3$</td>
</tr>
<tr>
<td>DPF parameter</td>
<td>$\rho = 0.95$</td>
<td>Background traffic type</td>
<td>UDP</td>
</tr>
</tbody>
</table>

Each UE is equipped with both LTE and Wi-Fi interfaces. The relays and destination can use their Wi-Fi interfaces to directly communicate in an ad hoc mode. The destination receives packets from the application server through relays. The relay manager at the destination monitors the available bandwidths of relays and end-to-end path delays per 2 seconds. Given a TBR requirement of 3 Mbit/s, the maximum number of packets received in 2 seconds is 733. Hence, we set $N = 733$, which is the length of DSN range for DPF and also used in Eq. (3) for calculating goodput. In addition, we simulate varying available bandwidths of relays by adjusting the UDP background traffic loads at relays.

A. SSRS performance evaluation

As a benchmark, we consider a greedy relay selection scheme, in which the destination simply adds relays one by one until their total available bandwidth falls into the acceptable TBR range. When the total available bandwidth becomes higher than the TBR upper bound, it deletes relays one by one until their total available bandwidth falls back to the TBR range.

Fig. 4 compares the aggregate throughput and goodput of SSRS and Greedy in the static available bandwidth pattern. As seen, both SSRS and Greedy can guarantee a stable aggregate throughput in the long term. After the 15th second, both algorithms achieve a throughput around TBR with a variation less than 5%. In the beginning, SSRS has less variation (6–8%) than Greedy (13–23%). This is because
SSRS efficiently scans most feasible subsets of relays so as to select a relay set that provides a total bandwidth closest to TBR. Another observation is that the goodput of SSRS and Greedy is much lower than their aggregate throughput. The average goodput of SSRS is only 70% of its aggregate throughput, while this ratio is 53% for Greedy. One major reason is the number of subflows in use, which is not shown due to space limitation. When the number of subflows is growing, the end-to-end delay variation is also increased so that more out-of-order packets are received at the destination. Since SSRS favors less subflows in its priority index, SSRS employs 1 less subflow than Greedy, and achieves an average goodput 25% higher than that of Greedy.

Fig. 5 shows the aggregate throughput and goodput of SSRS and Greedy with the dynamic available bandwidth patterns. The UDP traffic load at each relay is linearly increased by 20% at the 7th, 13th and 19th second to simulate decreasing available bandwidths, and then linearly decreased by 20% at the 31st, 37th and 47th second to simulate increasing available bandwidths. As seen, although the aggregate throughput of SSRS goes down because of less available bandwidths, SSRS adapts more smoothly with less throughput variation and outage. For example, in Fig. 5(a), SSRS only has 1 outage among the 10 monitoring points, as opposed to 3 for Greedy. SSRS is also observed to achieve a much higher goodput than Greedy in the dynamic scenario because SSRS engages less subflows.

B. ACC and DPF performance evaluation

Fig. 6 shows the aggregate throughput and goodput of ACC or DPF working together with SSRS in the static available bandwidth pattern. It is clearly seen that both ACC and DPF significantly improve the
goodput as compared to SSRS. The average goodput of ACC upon SSRS is 1.42 times higher than that of SSRS alone, while the average goodput of DPF upon SSRS is 1.36 times higher than that of SSRS.

Fig. 7 shows the aggregate throughput and goodput of ACC or DPF with SSRS in the dynamic available bandwidth patterns. Similarly, both modules are observed to achieve a much higher goodput than SSRS. Moreover, DPF is found to keep a more stable goodput than ACC. The goodput of ACC in the bandwidth decreasing case is obviously smaller than that of increasing available bandwidth. When the available bandwidth is decreasing, the delay ratio becomes so large to exceed $\theta_{\text{max}}$ (set to 3 in the simulation). For example, the maximum delay ratio between the 5th second and 25th second is found to be 5. As a result, ACC itself cannot reduce much the delay difference, but slightly degrades the achieved throughput by
C. Overall performance

The results in the previous section show that, combined with SSRS, both ACC and DPF can achieve a much higher goodput than SSRS itself. When the available bandwidth is decreasing due to higher traffic load, ACC cannot provide a stable goodput while DPF needs more buffer space to accommodate out-of-order packets. Fig. 8 shows the goodput of the three combined modules (SSRS + ACC + DPF) with static and dynamic bandwidths. As seen in Fig. 8(a), DPF well compensates the goodput degradation of ACC in the beginning when the decreasing available bandwidths result in a large end-to-end delay ratio. In
Fig. 8(b) for the dynamic case, the three combined modules outperform ACC plus SSRS for all the time, and perform better than DPF plus SSRS for 75% of the time. When the available bandwidth is decreasing in the first stage, the large delay ratio triggers ACC to shrink the congestion window intensely, which leads to a slightly lower aggregate throughput seen in Fig. 7(a). Since the goodput is upper bounded by the aggregate throughput, the goodput with ACC is smaller at some points when compared with DPF. Nonetheless, the overall goodput of the three combined modules is more stable in all cases.

V. CONCLUSIONS AND FUTURE WORK

Although MPTCP offers a promising solution to multipath transmission, further extensions to MPTCP are required to address the new challenges of the cooperative relay-based LTE network. In this paper, we propose three modules, namely, subset sum-based relay selection (SSRS), adaptive congestion control (ACC) and differentiated packet forwarding (DPF), to achieve a stable aggregate throughput with MPTCP and improve the goodput perceived at the application layer. We conducted extensive simulations in NS-3 for both static and dynamic scenarios to evaluate the performance in terms of aggregate throughput and goodput. It is observed that SSRS achieves a stable aggregate throughput while ACC and DPF further greatly improve the goodput. In the future work, we are interested in studying the fairness between the extended MPTCP solution and regular TCP, which is another key problem to implement a multipath transmission protocol at the transport layer.

REFERENCES


