Receiver-centric Buffer Blocking-aware Multipath Data Distribution in MPTCP-based Heterogeneous Wireless Networks

Yuanlong Cao[1], Qinghua Liu[1], Yi Zuo[2], Fenfen Ke[1], Hao Wang[1] and Minghe Huang[1]

[1] School of Software, Jiangxi Normal University, Nanchang, 330022 - China
[e-mail: {ylcao, 004799, coffee, wanghao}@jxnu.edu.cn, huangmh0093@sina.com]
[2] Jiangxi innovation funds management center for small and medium-sized enterprises
Nanchang, 330046 – China
[e-mail: long-1984@163.com]
*Corresponding author: Yuanlong Cao

Received April 27, 2016; revised August 22, 2016; accepted September 20, 2016; published October 31, 2016

Abstract

One major concern of applying Multipath TCP (MPTCP) to data delivery in heterogeneous wireless networks is that the utilization of asymmetric paths with diverse networking-related parameters may cause severe packet reordering and receive buffer blocking (RB^{2}LOC). Although many efforts are devoting to addressing MPTCP's packet reordering problems, their *sender-controlled* solutions do not consider balancing overhead between an MPTCP sender and receiver, and their *fully MPTCP* mode cannot make MPTCP achieve a desired performance. This paper proposes a novel receiver-centric buffer blocking-aware data scheduling strategy for MPTCP (dubbed MPTCP-*rec*) necessitating the following aims: (1) alleviating MPTCP's packet reordering and RB^{2}LOC problems, (2) improving the MPTCP performance, and (3) balancing load between the MPTCP sender and receiver. Simulation results show that the proposed MPTCP-*rec* solution outperforms the existing MPTCP solutions in terms of data delivery performance in heterogeneous wireless networks.

Keywords: Heterogeneous wireless network, cooperative computing, packet reordering, buffer blocking, multipath TCP

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61562044, 61262014, 71663037; the Natural Science Foundation of Jiangxi Province under Grant No. 20161BAB212046; the Project of Soft Science Research Plan of JiangXi Province under Grant No. 20161BBA10010; the Science and Technology Research Project of Jiangxi Provincial Department of Education (GJJ150319); and the Higher School Teaching Reform Research Subject of Jiangxi Province (JXJG-15-2-35).

http://dx.doi.org/10.3837/tiis.2016.10.002 **ISSN : 1976-7277**

1. Introduction

Over the past few years, wireless access technologies (e.g., WiFi, 3G, and LTE) have undergone a period of extremely rapid growth and developments. These significant achievements provide mobile device users ubiquitous wireless Internet access, e.g., wireless local area networks, cellular data networks, and mobile broadband wireless networks [1]. Spurred by the latest advancements in these wireless communications and networking technologies, a growing number of modern mobile devices are now equipped with two or more network interfaces [2] (e.g., the Samsung Galaxy S5 phones now enables "Download Booster" feature which combines WiFi and LTE networks simultaneously to boost data speed [3]). Such multi-homed mobile devices provide multiple heterogeneous access ability and can increase the goodput performance by aggregating bandwidth of multiple network interfaces, supported by the Multipath TCP (MPTCP) [4-5], a promising transport layer technology standardized by the Internet Engineering Task Force (IETF).

The MPTCP is an extension to TCP, allowing the use of multiple interfaces for concurrently scheduling packets across multiple independent end-to-end paths, without requiring any modification or addition to the applications and still being compatible on today's Internet [6]. Due to its concurrent transmission and bandwidth aggregation features, MPTCP provide a multi-homed host with numerous and attractive benefits including goodput improvement, latency reduction, robustness enhancement, and high-quality service provisioning [7]. **Fig. 1** illustrates a basic MPTCP usage in a heterogeneous wireless network condition. It shows how an MPTCP-based mobile device uses three paths (Path *A*, Path *B*, and Path *C*) simultaneously to communicate with the server. Such a multi-path communication way is beneficial to improve the transmission efficiency, maximize the network utilization, and protect against the TCP-like single-path failure [8-9]. Therefore, MPTCP has been recognized as the desired transport layer protocol for parallel data transmission in the heterogeneous wireless network conditions.

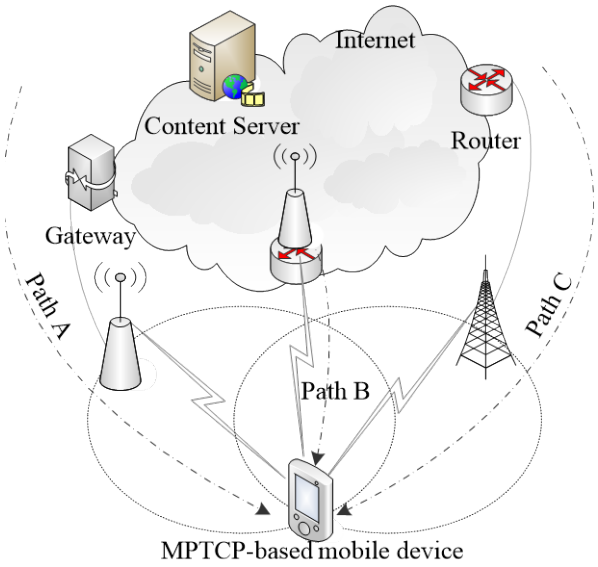


Fig. 1. MPTCP-based data delivery in a heterogeneous wireless network

Although the benefits of applying MPTCP to data transmission have been demonstrated to be extremely useful, there is still significant ongoing work addressing several remaining gaps and challenges. The primary concern of MPTCP-based data transmission is related to handling packet reordering. The regular MPTCP's scheduler splits packets over all available paths without considering the fact that asymmetric paths with diverse networking-related parameters may cause large numbers of out-of-order (O^3) packet arrivals at the receiver [10]. These O^3 packets need to be reassembled in-order at receive buffer before being delivered to the application. Therefore, in the regular MPTCP context, it is recommended that the receive buffer size should be large enough to contain all packets until the O^3 or missing packets arrive. However, the typical mobile devices used in multi-homed wireless mobile networks have in general very limited memory size and little free space for the receive buffer. Correspondingly, severe packet reordering in the constraint receive buffer will lead to a "hot potato" receive buffer blocking (RB^2LOC) problem and further leads to serious application-level throughput degradation.

Recent MPTCP efforts are devoted to addressing the packet reordering and optimizing the MPTCP performance [11-28], however, they still have some remaining challenges on this topic: 1) their *sender-controlled* operations (e.g., flow/congestion control, packet scheduling, and path management) do not consider balancing overhead between an MPTCP sender and receiver, and 2) they mostly use a traditional *full-MPTCP* bandwidth aggregation mode, in which all paths are used for data delivery, may fail to achieve a desired performance when the paths within the MPTCP session are with highly dissimilar characteristics. Our previous work [29-30] moves some operations, such as sending rate control, and path evaluation and selection, from the sender onto receiver in order to achieve a desired load balancing between the sender and receiver, and the supporting performance study convinces the advantages of using a receiver-driven Stream Control Transmission Protocol (SCTP) / Concurrent Multipath Transmission (CMT) [31-32] for multi-homed mobile device in wireless heterogeneous environment. However, like SCTP, all above SCTP extensions do not take into consideration any of the retro-compatibility benefits brought by MPTCP, as we discuss in the next section.

This paper jointly considers the benefits of MPTCP standard and our previous receiver-driven SCTP extensions [29-30] to propose a novel receiver-centric buffer blocking-aware data scheduling strategy for MPTCP (MPTCP-*rec*). The goals of MPTCP-*rec* are (i) to make MPTCP aware of RB^2LOC timely, (ii) to ensure possible in-order packet arrivals and improve the MPTCP performance, and (iii) to possible balance overhead between an MPTCP sender and receiver. Our MPTCP-*rec* solution makes fundamental contributions against the state of art in the literature, in the following aspects:

- It moves the responsibility for performing path quality estimation and congestion control from the sender to the receiver to possibly help MPTCP alleviate the workload of the sender.
- It includes a RB^2LOC-aware *full*/*partial* MPTCP switching strategy to reduce packet reordering and alleviate the RB^2LOC problem.
- It performs a comparative study to show the performance gain of the "sender-receiver" cooperation-oriented MPTCP solution compared with the regular MPTCP over a heterogeneous wireless condition.

2. Related Work

In the recent years, the growing interest in MPTCP has gained variety of attentions. Khalili *et al.* [6] proposed an opportunistic linked-increases algorithm (OLIA) to satisfy the

Pareto-optimal design goals of MPTCP. Pearce *et al*. [7] investigated the implications for network security both in the transitional state and in a future, where MPTCP is partially supported and every device supports MPTCP, respectively. Zhou *et al*. [8] extended MPTCP's congestion control mechanism for a user cooperation scenario in a LTE network environment. Peng *et al*. [9] designed a fluid model for MPTCP congestion control algorithm and identified design principles that guarantee the uniqueness and stability of system equilibrium. Arzani *et al*. [11] investigated the impact of the initial sub-path selection on the MPTCP performance. Park *et al*. [12] designed a greedy traffic scheduler in order to minimize the cost while satisfying the delay constraints. Li *et al*. [13], Cao *et al*. [14], and Raiciu *et al*. [15] extended the MPTCP's congestion control and/or scheduling algorithms in order to mitigate the incast problem and improve goodput in the Data Center Networks (DCNs).

Apart from the transport layer information-dependent MPTCP extensions, there has been extensive interest in and research activity on cross-layer MPTCP designs. Corbillon *et al*. [16] developed a cross-layer MPTCP scheduler which utilizes information from both application-layer and transport-layer in order to re-order the transmission of packet and prioritize the most significant parts of the video content. Sinky *et al*. [17] presented a handoff-aware cross-layer assisted MPTCP (CLA-MPTCP) congestion control algorithm to alleviate handoff induced issues. Their simulation results showed that CLA-MPTCP outperforms the regular MPTCP technology in terms of handoff performance in heterogeneous wireless networks. Lim *et al*. [18] proposed a cross-layer path management mechanism for MPTCP, dubbed as MPTCP-MA, which uses MAC-Layer information to locally estimate path quality and connectivity. Their experimental results showed that the proposed MPTCP-MA solution can efficiently utilize an intermittently available path based on the associated link status.

Lately, there has been a growing interest in the research on MPTCP-based multimedia distribution. Wu *et al*. [19] presented an analytical framework to model the MPTCP-based video delivery performance, and developed a novel quality-driven MPTCP approach which integrates the Forward Error Correction (FEC) coding and rate allocation to minimize the end-to-end video distortion. Xu *et al*. [20] introduced PR-MPTCP, a partially reliable extension of MPTCP which offers a flexible QoS trade-off between timeliness and reliability for real-time multimedia applications. Wu *et al*. [21] developed a mathematical framework to analyze the frame-level energy-quality tradeoff for delay-constrained multihomed multimedia system, and extended MPTCP scheduler for prioritized frame scheduling and unequal loss protection to achieve high-quality video streaming while minimizing device energy consumption. Diop *et al*. [22] investigated and analyzed the QoS benefits induced by applying the "partial reliability" concept to MPTCP for interactive video applications. Our previous work PR-MPTCP^{+} [23] introduced a 'prioritized reliable service' and a 'timed reliable service' concept to an MPTCP implementation.

More recently, an increasing number of researchers have concentrated their efforts to address the packet reordering issue. Oh *et al*. [24] presented a novel feedback-based path failure detection and a new buffer blocking protection method to alleviate the packet reordering and buffer blocking problems. Alheid *et al*. [25] investigated the effect of out-of-order packets on the performance of MPTCP, and recommended a proper packet reordering algorithm for MPTCP in order to achieve a better throughput performance. Li *et al*. [26] extended MPTCP and proposed a SC-MPTCP solution, by applying linear systematic coding to MPTCP, to reduce packet reordering and improve system performance with bounded receive buffers. Cao *et al*. [27] designed a delay-based congestion controller for MPTCP, by making use of packet queuing delay as congestion signals, to achieve fine-grained

load balancing while occupying fewer link buffers. Yang *et al*. [28] presented Non-Renegable Selective Acknowledgments (NR-SACKs) for MPTCP, and analyzed the impact of NR-SACKs in situations where an MPTCP receiver never discards received out-of-order packet from the receive buffer.

As discussed earlier, all above MPTCP solutions use a traditional sender-controlled mode. Our previous work [29-30] provids SCTP and CMT with a novel receiver-driven traffic schduling and path management algorithm in order to achieve a better load balancing between an SCTP sender and receiver. However, SCTP is not compatible with the current network infrastructure primarily due to: 1) it presents an entirely new API to applications and cannot support the socket API (Hosts cannot establish a SCTP connection by using the existing socket APIs); 2) it is not widely integrated in today's TCP/IP stacks and has difficulty traversing Network Address Translators (NATs) and any middleboxes. The limitations constraint the success of SCTP in today's and future architectures. This paper considers the benefits of our previous receiver-driven SCTP extensions and introduces a receiver-centric multipathing solution to the promising MPTCP.

3. The Motivation and Overview of MPTCP-*rec*

In regular MPTCP, given a flow with a small amount of subflows, each of them using an individual path, the sender tries to schedule traffic across those subflows (paths). However, because the latency and other networking-related characteristics of asymmetric paths can be significantly different and sensitive to variations, there is a high probability that a packet with lower sequence number sent over a low-quality path (e.g., a slower path) arrive at the receiver later than a packet with higher sequence number sent over a high-quality path (e.g., a faster path). As a result, the receiver needs to buffer the O^3 packet until the packets with higher sequence number are received successfully. Once a great number of O^3 packets held by the constrained receiver buffer for reordering, MPTCP undoubtedly suffers from significant receiver buffer blocking problems.

In order to ensure packets possible in-order arriving, the current MPTCP extensions generically optimize the operations at the sender side in order to evaluate each path's real-time transmission efficiency and schedule packets over these paths accordingly. However, such sender-centric and receiver-passive approaches fail to consider the following aspects including:

1) The sender need to perform a majority of work (e.g., flow/congestion control, loss recovery, path management, etc.), and may easily become bottleneck in terms of calculating and processing once having a significantly large amount of connections and receivers (mobile devices) communicated.
2) When the receiver informs its obtained first-hand knowledge of wireless links to the sender, the feedback information may suffer from some unexpected problems (e.g., prolonged delay, loss, etc.), which will cause the sender to make an undesired decision.

In practice, a receiver may be adjacent to the wireless last-hop, thus it is more aware of the wireless link condition than its corresponding sender. Moreover, shifting some operations from the sender onto receiver can possibly balance the overhead between the sender and receiver. Motivated by these facts, we introduce MPTCP-*rec*, a receiver-centric backwards-compatible MPTCP extension. In the MPTCP-*rec*, the receiver is in charge of flow/congestion control operations and controls the data sending rate, while the sender merely responds based on the receiver's feedback. In this context, the receiver do not need to give feedback to the sender the network parameters obtained in its forward paths but rather

immediately use the first-hand knowledge to determine its desired sending rate (DSR). This feature makes MPTCP-*rec* support the real-time processing of network information.

Fig. 2 illustrates the architecture of MPTCP-*rec* system. As an extension to MPTCP, the MPTCP-*rec* is constructed by two major modules, which are *Desired Sending Rate Estimator* (DSRor) that runs at the receiver side, and *Path Collector & Data Scheduler* (PCDer) that runs at the sender side. The main functions of the two modules are detailed below:

- DSRor: it is devoted to controlling *how much data can be sent*, which means it determines per-path's sending rate (for flow/congestion control), and prevents sthe injection of bursty packets on the paths.
- PCDer: it contributes to cognizing the RB^{2}LOC event timely and deciding *which paths can be used*, which means it chooses a subset of paths for multipath transmission when serious RB^{2}LOC is detected.

As shown in **Fig. 2**, there are seven states running at the MPTCP-*rec* receiver for flow/congestion control, which are Slow Start (SS), Slow Start Ready (SR), Congestion Avoidance (CA), Congestion Avoidance Ready (CR), Timeout (TO), GAP and Fast Recovery (FR). The general behavior of the seven states (e.g. state transitions among the seven states) is inherited from the TEAR solution [33], which is a well-known receiver-centric TCP extension. Running the flow/congestion control at the receiver side can possibly help the sender to reduce its overhead and share the workload between the sender and receiver.

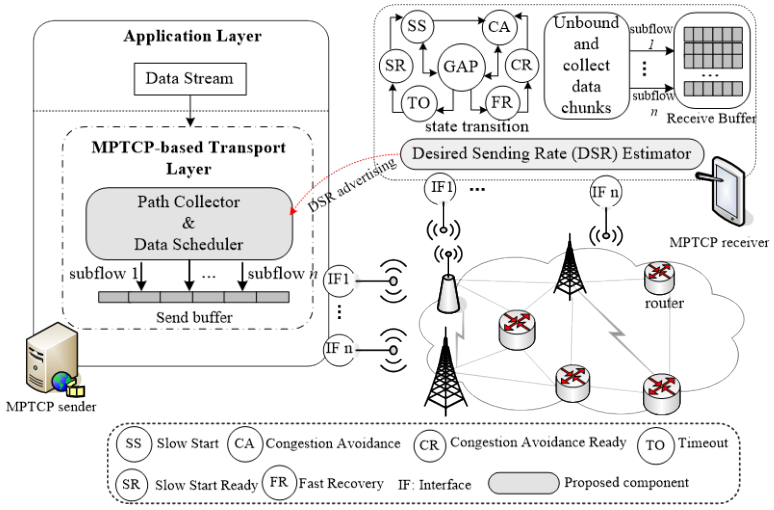


Fig. 2. MPTCP-*rec* architecture

4. MPTCP-*rec* Detail Design

4.1 Desired Sending Rate Estimator (DSRor)

As mentioned previously, the responsibility of DSRor is that performing congestion control and sending rate estimation at the MPTCP-*rec* receiver side. To this end, we draw on idea in the design of our previous SCTP-*Rev*'s SRE-*rev* (receiver-based sending rate estimator) [29] to develop the DSRor module, more precisely, the DSR estimation function of DSRor is a clone of the general behavior of the SCTP-*Rev*'s SRE-*rev* module. In order to make the paper self-contained, this subsection introduces the procedures how the MPTCP-*rec*'s DSRor, also SCTP-*Rev*'s SRE-*rev* module runs the DSR calculation at receiver.

In order to estimate the DSR value at receiver, MPTCP-*rec*'s DSRor employs a *round* information [33] to help the receiver to calculates the *round-trip time* (*rtt*) and *retransmission timeout* (*rto*). We introduce the definition and usage of *round* as following. Next the *rtt* calculation, also the major design of DSRor is detailed, respectively.

- A *round* begins when a selective acknowledgment (SACK) with the DSR information arrives at a sender and the sender adjusts the sending rate according to the advertised DSR value for data delivery. Current *round* ends and a new *round* begins once having a new SACK chunk receiving by the sender.
- Each packet will carry the current *round ID* (rID) to help the MPTCP-*rec* receiver identify the current round information.

Based on the *round* information, the MPTCP-*rec*, like our SCTP-*Rev* solution [29], runs the *rtt* and *rto* calculation at receiver following the steps below:

a) The MPTCP-*rec* receiver records the timestamp t_i when it sends a SACK chunk S_i back to the corresponding sender.

b) Once the sender receives the SACK S_i successfully, it adjusts the data sending rate according to the advertised DSR value carried in S_i (which means a new round $round_x$ starts).

c) When the first packet with the rID $round_x$ arriving, the MPTCP-*rec* receiver records the timestamp t_{i+1} and runs the *rtt* calculation by

$$\left.\begin{aligned} rtt = (\xi,\eta)\begin{bmatrix} \overrightarrow{rtt} \\ (t_{i+1}-t_i) \end{bmatrix} &= \xi \times \overrightarrow{rtt} + \eta \times (t_{i+1}-t_i); \\ s.t. \qquad \xi + \eta &= 1, \end{aligned}\right\} \tag{1}$$

which $\overrightarrow{rtt}$ is the current round trip time, the weighting factors ξ and η use default values of $\frac{1}{8}$ and $\frac{7}{8}$, respectively. The *rto* calculation at MPTCP-*rec* receiver is same as that does at the standard MPTCP sender.

Based on the *rtt* and *rto* information, MPTCP-*rec* receiver can further determine its desired sending rate for each path, and inform the sender of the DSR value. For per-path's DSR calculation, let us suppose that there are δ paths $(d_1, d_2, \cdots, d_\delta)$ within the MPTCP session, and taking path d_ℓ $(1 \le \ell \le \delta)$ for example, each time a packet is sent on d_ℓ and is received successfully, the receiver calculates the DSR value for d_ℓ by

$$rate_i^{d_\ell} \leftarrow \begin{cases} \varphi \times rate_i^{d_\ell} + (1-\varphi) \times \dfrac{\mathrm{P}_{size}}{n \times RTT}, & \text{if PLR} < thresh; \\ (1-\vartheta) \times rate_i^{d_\ell} + \vartheta \times \dfrac{\mathrm{P}_{size}}{n \times RTT}, & \text{otherwise,} \end{cases} \tag{2}$$

where n is the number of packets received by receiver within one *round*. P_{size} is the packet size. The weighting factors φ and ϑ use a default value $\frac{1}{2}$ for fairness. As analyzed in our SCTP-*rev* solution [29], using the above DSR calculation equation can help the transport-layer protocols with AIMD-like congestion control mechanism (e.g., SCTP, MPTCP) to avoid bursty transmission fluctuation in lossy wireless transmission while maximizing the wireless resource utilization.

Before informing the sender of the estimated DSR value, the receiver further smoothes the value by using the following equation,

$$Rate_e^{d_\ell} = (\overline{\mathrm{F}} \times \Omega_\varphi) \times \frac{1}{\sum_{s=1}^{\sigma} \mathrm{F}_\psi}, \tag{3}$$

where Ω_φ ($\Omega_\varphi = [rate_{\varphi,1}^{d_\ell}, rate_{\varphi,2}^{d_\ell}, ..., rate_{\varphi,\sigma}^{d_\ell}]$) is a matrix which is used to store the σ historical values of DSR for path d_ℓ. $\overline{\mathrm{F}}$ ($\overline{\mathrm{F}} = [\mathrm{F}_1, ..., \mathrm{F}_\psi, ..., \mathrm{F}_\sigma]$) is a weighting coefficient vector, where F_ψ ($\psi \in [1, \sigma]$) can be obtained by

$$\mathrm{F}_\psi = \begin{cases} 1, & \text{if } 1 \le \psi \le \dfrac{\sigma}{2}; \\ 1 - \dfrac{\psi - \sigma/2}{\sigma/2 + 1}, & \text{if } \dfrac{\sigma}{2} \le \psi \le \sigma, \end{cases} \tag{4}$$

Like our SCTP-*Rev* solution [29], the MPTCP-*rec* receiver will launch a timer with 1 *rtt* in length for each path once having a new round starts. When a timeout event occurs, or the sending rate the sender used is not equal to the advertised $Rate_e^{d_\ell}$ value, the receiver informs the sender of the current $Rate_e^{d_\ell}$ by using an extended MPTCP NR-SACK option [28], shown in **Fig. 3**. The extended NR-SACK chunk includes three additional parameters, which are *timestamp* that is used to order the NR-SACKs received from the different paths, *pid* (namely *path identifier*) that is used to identify paths between the sender and receiver [34], and *Desired Sending Rate* (DSR) that is devoted to informing the sender of the advertised DSR value for each path. The extended NR-SACK not only reports the block containing the most recently received data, but also provides a MPTCP sender with the most up-to-date information about per-path's DSR value and the state of a MPTCP receive buffer.

Kind	Length	Subtype	(reserved)	C	F	m	M	a	A
Data ACK (4 or 8 octets, depending on flags)									
Data Sequence Number (4 or 8 octets, depending on flags)									
Subflow Sequence Number (4 octets)									
Data-level Length (2 octets)			Checksum (2 octets)						
Left Edge of NR-SACK Block #1				Right Edge of					
NR-SACK Block #1			Left Edge of NR-SACK						
Block #2	Right Edge of NR-SACK Block #2								
Left Edge of NR-SACK Block #3				Right Edge of					
NR-SACK Block #3			Padding						

(a) Format of MPTCP NR-SACK chunk

Kind	Length	Subtype	(reserved)	C	F	m	M	a	A
Data ACK (4 or 8 octets, depending on flags)									
Data Sequence Number (4 or 8 octets, depending on flags)									
Subflow Sequence Number (4 octets)									
Timestamp									
Pid # 1	Desired Sending Rate ($Rate_e^{d_1}$)								
Pid # δ	Desired Sending Rate ($Rate_e^{d_\delta}$)								
Data-level Length (2 octets)			Checksum (2 octets)						
Left Edge of NR-SACK Block #1				Right Edge of					
NR-SACK Block #1			Left Edge of NR-SACK						
Block #2	Right Edge of NR-SACK Block #2								
Left Edge of NR-SACK Block #3				Right Edge of					
NR-SACK Block #3			Padding						

(b) Extension for MPTCP NR-SACK chunk

Fig. 3. Formats of (a) MPTCP NR-SACK chunk, and (b) extended MPTCP NR-SACK chunk.

4.2 Path Collector & Data Scheduler (PCDer)

As mentioned previously, because underlying heterogeneity of Internet, asymmetric paths with highly dissimilar characteristics in terms of propagation delay and other network-related parameters are more common. In such a heterogeneous environment, the traditional full-MPTCP mode, in which all paths are used for data delivery, is bound to buffer blocking, with a great number of out-of-order packet arrivals and severe data reordering in the constrained receive buffer.

In view of the above problem, the MPTCP-*rec* includes a *Path Collector & Data Scheduler*

(PCDer) aiming to mitigate the RB^{2}LOC problem while maximizing the network resource utilization, in which the sender adaptively chooses all or a subset of its available paths to construct an optimal candidate path list (denoted as P_{list}) for multipath data delivery and bandwidth aggregation. To find a suitable group of paths, the MPTCP-*rec* sender needs to: 1) calculate and distinguish per-path's transmission quality, and 2) monitor and predict a RB^{2}LOC event timely.

For path quality calculation and distinguishing, different from existing MPTCP extensions, in MPTCP-*rec*, the sender does not need to calculate the transmission quality for each path; it just only receives per-path's DSR value from the receiver and sorts its paths according to their DSR value. This receiver-assisted sending rate control feature possibly helps MPTCP reduce the sender's computation overhead and share the load between the sender and receiver.

For the RB^{2}LOC prediction, the MPTCP-*rec* sender monitors the jitter indicator of the advertised receiver window (*awnd*) [35], $awnd_{\Delta}$, in order to recognize the RB^{2}LOC event timely. Let $awnd_{curr}$, $awnd_{\max}$, $awnd_{\min}$ and $awnd_{avg}$ be the current value, the maximum value, the minimum value, and the average value of *awnd*, respectively. The $awnd_{\Delta}$ can be calculated by using the following equation,

$$awnd_{\Delta} = \frac{awnd_{curr} - awnd_{avg}}{awnd_{\max} - awnd_{\min}}, \tag{5}$$

where the average value of *awnd*, $awnd_{avg}$, can be calculated by

$$awnd_{avg} = \frac{1}{m} \times \sum_{j=1}^{m} awnd_j. \tag{6}$$

Meanwhile, assuming the observed values of $awnd$ are $(awnd_1, awnd_2, ..., awnd_m)$, the values of $awnd_{\max}$ and $awnd_{\min}$ can be computed by $awnd_{\max} = \underbrace{(awnd_1, awnd_2, ..., awnd_m)}_{\max}$ and $awnd_{\min} = \underbrace{(awnd_1, awnd_2, ..., awnd_m)}_{\min}$, respectively.

Then, the MPTCP-*rec* sender simply monitors the variation of $awnd_{\Delta}$ to predict whether or not a RB^{2}LOC event is upcoming. We determine a RB^{2}LOC is to be occurred if the below condition is detected,

$$\begin{cases} awnd_{\Delta} < 0, \\ (awnd_{\Delta})_{t_\kappa} - (awnd_{\Delta})_{t_{\kappa-1}} < 0. \end{cases} \tag{7}$$

which t_κ and $t_{\kappa-1}$ ($t_\kappa > t_{\kappa-1}$) are the observed time. When $awnd_{\Delta} < 0$, the continuous decrease of $awnd_{\Delta}$ indicates a RB^{2}LOC event is to be occurred.

In the case of an upcoming RB^{2}LOC event is detected, in order to reduce the RB^{2}LOC problem, the MPTCP-*rec* sender removes the path d_γ^* that minimizes the objective function from the candidate path list P_{list} and then marks the state of d_γ^* as UNUSED for data transmission,

$$d_\gamma^* = \arg\min \left(Rate_e^{d_\gamma} \right), \tag{8}$$

subject to

$$1 \le \gamma \le \delta. \tag{9}$$

While in the case of non-RB^{2}LOC event is detected, in order to maximize the network resource utilization, the MPTCP-*rec* needs to switch to *Full-MPTCP* mode, which refers to the regular MPTCP operations where the sender uses all of its available paths to construct the P_{list} for multipath data scheduling. This approach helps MPTCP-*rec* fully utilize all the available bandwidth and increase the packet delivery speeds up. The PCDer-based data delivery strategy is as follow, and the corresponding pseudo code is presented in Algorithm 1. We also present a basic operation of MPTCP-*rec* (shown in **Fig. 4**) to make the reader easily understand the formula's step process and MPTCP-*rec* algorithm's operation.

1) The MPTCP-*rec* sender monitors the jitter indicator $awnd_{\Delta}$ periodically (per-*round*) by using Eqs. (5), (6) and (7) in order to identify whether or not a RB^{2}LOC event is to be occurred;
2) If some kind of RB^{2}LOC event is supposed to be not happen, the MPTCP-*rec* switches to *Full-MPTCP* mode, in which all available paths are put into the P_{list} for multipath scheduling;
3) If an upcoming RB^{2}LOC event is recognized (namely Eq. (7) is met), the MPTCP-*rec* sender sorts the paths within P_{list} in an ascending order (according to their own DSR values), and removes the first path from P_{list} and marks the state of the path as UNUSED;
4) If any one UNUSED path has a larger DSR value than that of the paths within P_{list}, the MPTCP-*rec* sender puts it into P_{list} again for multipath data scheduling.

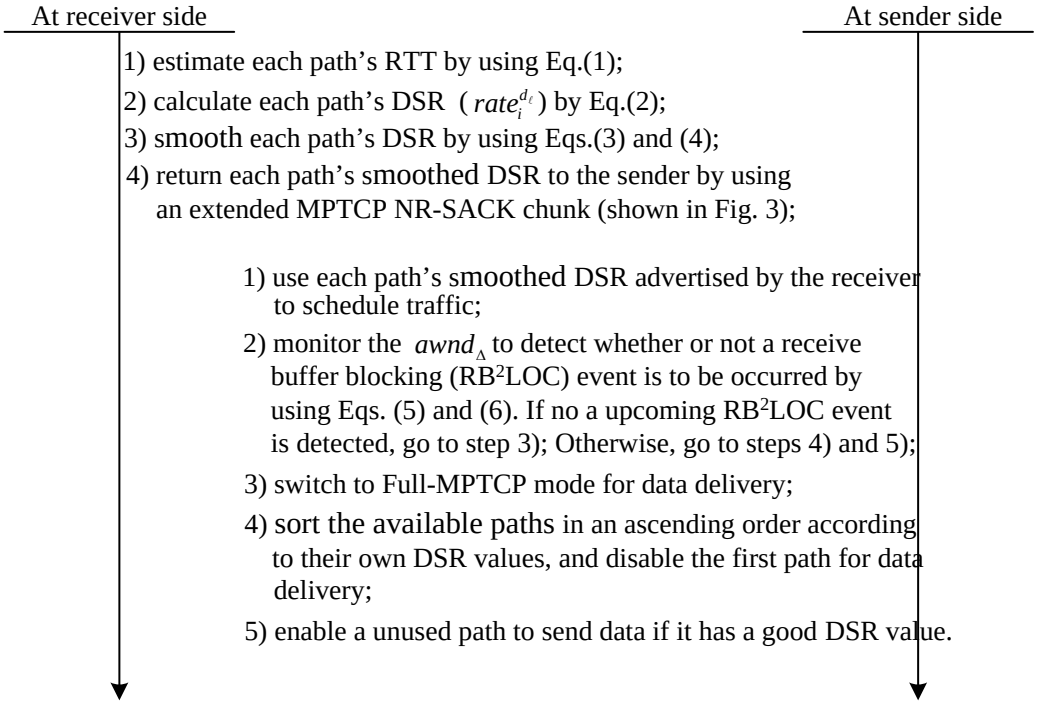


Fig. 4. A basic operation of MPTCP-*rec*

Algorithm 1: PCDer-based multipath scheduling algorithm

Definition:

d_i: the i^{th} path within the MPTCP session

$d_i^{Rate_e}$ the DSR value of the i^{th} path

P_{list}: the candidate path list within the session

$P_{list(0)}$ the first path within the P_{list}

$\overline{P_{list}}$: the UNUSED path list within the session

$\overline{d_j}$: the j^{th} path within $\overline{P_{list}}$

When an upcoming RB2LOC event is recognized,

1: **for** ($(i=1, i \le \text{count}(P_{list}), i++)$ **do**

2: **if** ($d_i^{Rate_e} < d_{list(0)}^{Rate_e}$) **then**

3: set $list(0)=i$; set $d_{list(0)}^{Rate_e} = d_i^{Rate_e}$;

4: mark the status of path $P_{list(0)}$ as UNUSED;

5: update the P_{list} info;

6: **end if**

7: **end for**

8: **for** $(j=1, j \le \text{count}(\overline{P_{list}}), j++)$ **do**

9: compare the DSR value of $\overline{d_j}$ with $P_{list(0)}$;

10: **if** $(\overline{d_j}^{Rate_e} > P_{list(0)}^{Rate_e})$ **then**

11: put the path $\overline{d_j}$ into P_{list};

12: **end if**

13: **end for**

14: use the paths within P_{list} for traffic scheduling.

5. Simulations and Analysis

5.1 Simulation topology

A comprehensive performance evaluation has been carried out on the NS-2.35 (Network Simulator version 2.35) [36], in which the MPTCP patch [37] for NS-2 has been extended and embedded. The simulations considered a MPTCP-based heterogeneous wireless network environment shown in **Fig. 5**. Both MPTCP endpoints have three asymmetric paths (denoted Path *A*, Path *B*, and Path *C*) with different network-related parameters. Path *A*'s bandwidth is set to 384Kbps and 20-50 ms propagation delay, which corresponds to a 3G/UMTS link. Path *B* experiences 11Mbps bandwidth with 10-20 ms propagation delay which is encountered in WiFi/IEEE 802.11b standard. Path *C*'s bandwidth is set to 10 Mbps and 20-50 ms propagation delay, which is representative for a WiMax/IEEE 802.16 link. The main configurations of the three paths are illustrated in Table 1. The other MPTCP parameters just use the default values provided in the NS-2 MPTCP patch.

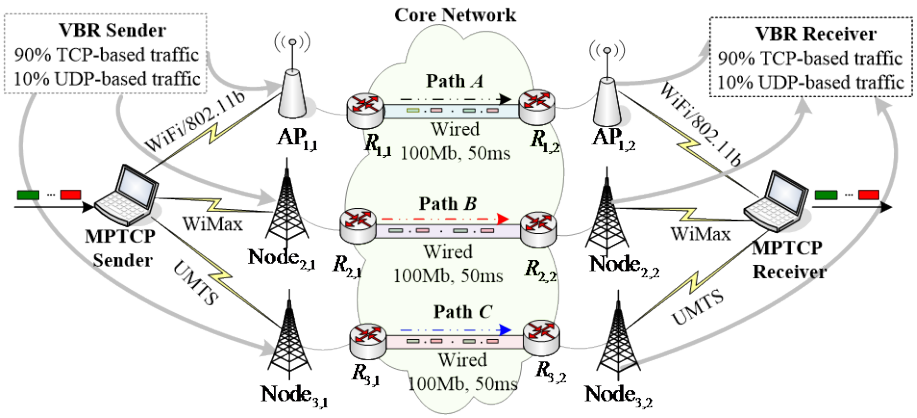


Fig. 5. Simulation topology

In order to simulate the data-link layer's frame loss, we attach two loss models for each wireless link, which are the *Uniform loss mode*l that represents distributed loss caused by random contention or wireless interference, and the *Gilbert loss model* (also refer to as the *two-state Markov loss model*) that represents infrequent continuous loss caused by signal fading or stream bursty injection. To consume per-path's available bandwidth, we attach each path with a Variable Bit Rate (VBR) generator in order to send VBR-based Internet background traffic to its corresponding VBR receiver. Like [38], the packet size (in bytes) used for the VBR background traffic are chosen as follows: 50% are 44-byte, 25% are 576-byte, and the rest 25% have 1500-byte. 90% of the total bytes are carried by TCP connection and the other 10% are over UDP connection. The aggregate VBR background traffic on each path varies randomly between 0-50% of the access-link's bandwidth. The total simulation run time is 120 seconds.

Table 1. Path configuration used in the simulation

Network Parameters	Path *A*	Path *B*	Path *C*
Wireless technology	IEEE 802.11b	IEEE 802.16	UMTS
Access link bandwidth	11Mbps	10Mbps	384Kbps
Access link propagation delay	5-15ms	1-20ms	1-20ms
Access link queue type	Droptail	Droptail	Droptail
Uniform loss rate	0-5%	2-3%	2-3%
Markov loss rate	1%	1%	1%
Core network propagation delay	100ms	100ms	200ms

5.2 Simulation results

We note that applying some promising technologies (i.e., cross-layer activities, network coding) to MPTCP is a widely-researched topic. However, the proposed MPTCP-*rec* solution is devoted to improving MPTCP protocol itself depended solely upon the information provided by the transport layer. Therefore, in this subsection, we only present the performance evaluation and comparison between the standard MPTCP and the proposed MPTCP-*rec*. To make it convenient, we portray the results of the standard MPTCP as 'MPTCP' in the test

result figures, and the results with the proposed solution are portrayed as 'MPTCP-rec', respectively.

1) *Data sending and receiving times*

Fig. 6 illustrates the sending and arrival times of several data when the standard MPTCP and MPTCP-*rec* are used, respectively. We can observe that MPTCP-*rec* attains a greater number of sending and receiving *Data Sequence Number* (DSN) than the standard MPTCP. That is because the standard MPTCP allocates traffic fairly over all its available paths, ignoring the fact that in a heterogeneous network environment different paths have different transmission capacities. Such "blind" scheduling strategy is bound to buffer blocking, with a very large amount of out-of-order data chunks arrivals and severe packet reordering in the overloaded receive buffer, which constrains the sender from sending any new data chunk. In contrast, the MPTCP-*rec* adaptively selects a set of good paths for each MPTCP flow, by making MPTCP aware of path sending rate and RB^{2}LOC event. Thereby, MPTCP-*rec* achieves higher sending and receiving DSN than the standard MPTCP.

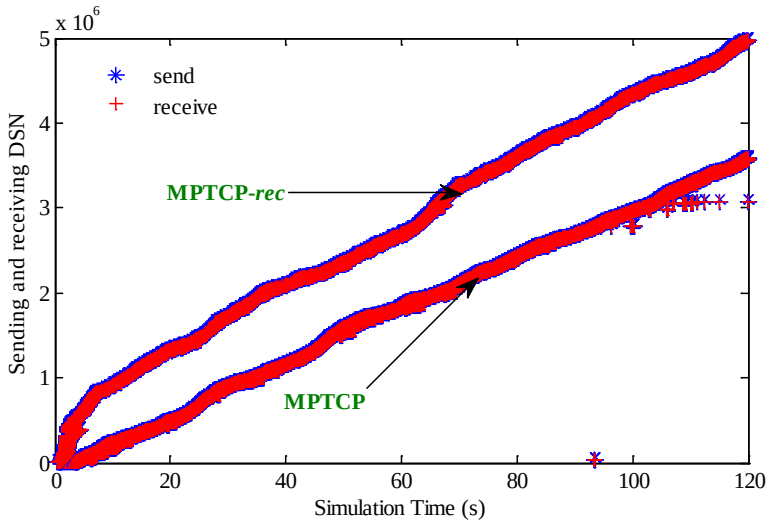


Fig. 6. Comparison of sending and receiving DSN

2) Out-of-order data chunks

The out-of-order DSN (O^3-DSN) metric, which is measured by the offset between the DSNs of two consecutively received data chunks, is a good metric selection used to reflect the performance of multipath data delivery in a heterogeneous wireless network environment. **Fig. 7** illustrates the O^3-DSN metric variation between simulation time *t=0s* and *t=120s*. As the figures shown, the standard MPTCP generates more out-of-order data chunks and requires increased packet reordering than the MPTCP-*rec*. This is because in the standard MPTCP, the sender schedules packets over all paths, without considering the fact that huge quality differences among these paths will result in a large number of outstanding data chunks. In contrast, MPTCP-*rec* selects a set of path accordingly and schedules packets over the preselected paths. This way helps MPTCP-*rec* reduce the out-of-order data arrival and consequently performs better than the standard MPTCP. When comparing the two solutions, it can be seen that the peak out-of-order data reception at the receiver is close to 3×10^5 using the standard MPTCP, while it is approximately 1.8×10^5 when using the proposed MPTCP-*rec*

solution.

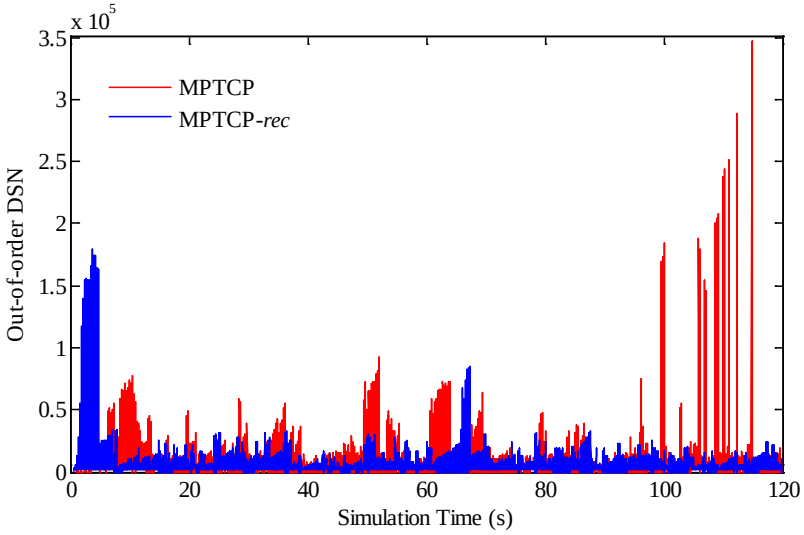


Fig. 7. Comparison of out-of-order DSN

3) Average throughput

Fig. 8 shows the comparison of average throughput when using the standard MPTCP and MPTCP-*rec*, respectively. Since MPTCP-*rec* jointly considers the delay variation, packet loss, and RB^{2}LOC probability to estimate per-path's sending rate. This feature makes MPTCP-*rec* reduce the packet loss, RB^{2}LOC probability, increase the packet sending and receiving times, and improve the goodput performance. While the standard MPTCP simply splits packets over all its asymmetric paths, regardless of the fact that some paths with low quality will cause the overall throughput degradation of an receive buffer-constrained MPTCP session. Compared with the standard MPTCP, the MPTCP-*rec* throughput is about 40.93% higher than that of the standard MPTCP scheme.

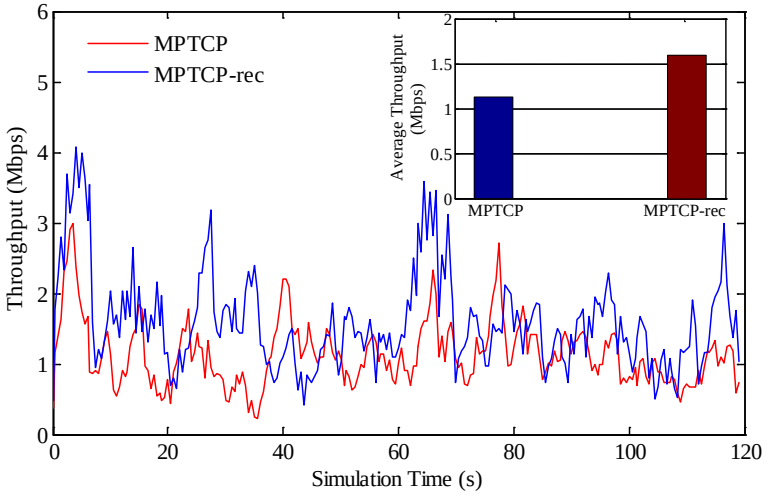


Fig. 8. Comparison of throughput performance

4) Comparisons of end-to-end delay and Jitter

Fig. 9 shows the end-to-end delay comparison when using the standard MPTCP and MPTCP-*rec*, respectively. As mentioned above, MPTCP-*rec* takes into consideration the delay variations during per-path's sending rate estimation and path selection. This feature helps MPTCP-*rec* flows avoid passing the more congested links in a heterogeneous network environment. As a result, MPTCP-*rec* achieves significantly lower end-to-end delay performance than the standard MPTCP. Compared with the standard MPTCP, the MPTCP-*rec*'s end-to-end delay is about 48.73% lower than that of the standard MPTCP scheme. Jitter is a variation in packet transport delay caused by link transient failure, stream burst, and other networking-related factors effects on transmission performance. Lower levels of jitter are more likely to occur on satisfactory transport solution and vice verse. Therefore, jitter has been recognized as a very useful metric to convince the performance of transport protocols. **Fig. 10** illustrates the jitter comparison results when the standard MPTCP and the proposed MPTCP-*rec* are used, respectively. Since the MPTCP-*rec* solution can identify per-path's sending rate, it further enables a sending rate-driven RB^{2}LOC-aware algorithm to select a set of good paths for data delivery. Thereby, it outperforms the standard MPTCP in terms of jitter performance.

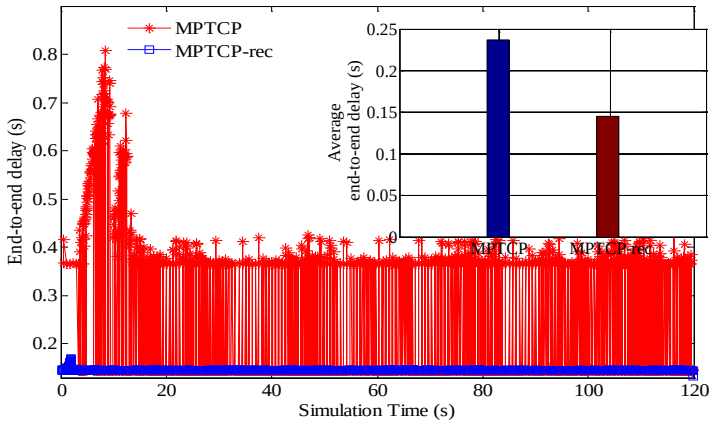


Fig. 9. Comparison of end-to-end delay

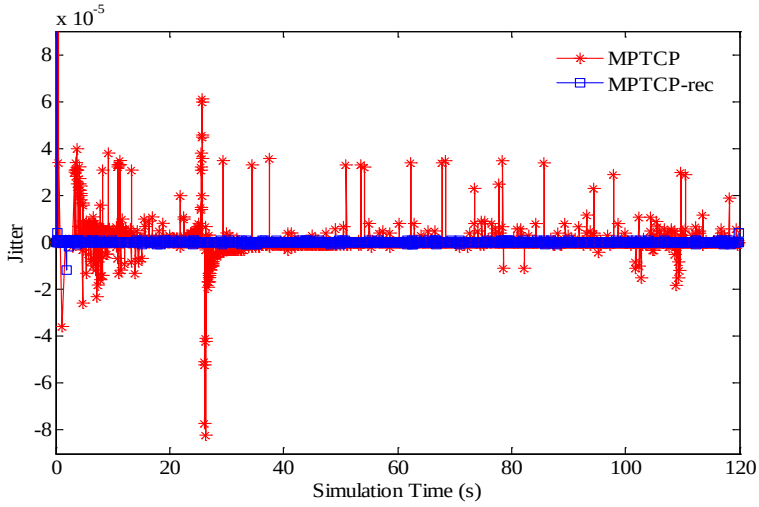


Fig. 10. Comparison of jitter

6. Conclusions

Motivated by the facts that a receiver may be adjacent to the wireless last-hop and it can be more aware of the wireless link condition than its corresponding sender, this paper presents a novel receiver-centric buffer blocking-aware data scheduling strategy for MPTCP (dubbed as MPTCP-*rec*), by jointly considering the benefits of MPTCP technology and our previous receiver-driven SCTP-based data delivery solution. In MPTCP-*rec*, the receiver is devoted to controlling *how much data can be sent*, while the sender is responsible for determining *which paths can be used*. Such a *sender-receiver* cooperation-based multipath data transmission is beneficial for MPTCP to: (1) alleviate the packet reordering and RB2LOC problems, (2) improve the performance, and (3) balance load between the sender and receiver. Simulation results show that the proposed MPTCP-*rec* solution outperforms the existing MPTCP solutions in terms of data delivery performance in heterogeneous wireless networks.

References

[1] C. Xu, J. Zhao, G.-M. Muntean, "Congestion Control Design for Multipath Transport Protocols: A Survey," *IEEE Communications Surveys & Tutorials*, vol.PP, no.99, 2016. Article (CrossRef Link).

[2] Q. D. Coninck, M. Baerts, B. Hesmans, O. Bonaventure, "Observing real smartphone applications over multipath TCP," *IEEE Communications Magazine*, vol.54, no.3, pp.88-93, 2016. Article (CrossRef Link).

[3] http://galaxys5guide.com/samsung-galaxy-s5-features-explained/galaxy-s5-download-booster/. Article (CrossRef Link).

[4] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," *IETF RFC 6824*, Jan. 2013. Article (CrossRef Link).

[5] A. Ford, C. Raiciu, M. Handley, S. Barre and J. Iyengar, "Architectural Guidelines for Multipath TCP Development," *IETF RFC 6182*, 2011. Article (CrossRef Link).

[6] R. Khalili, N. Gast, M. Popovic and J. Boudec, "MPTCP Is Not Pareto-Optimal: Performance Issues and a Possible Solution," *IEEE/ACM Transactions on Networking*, vol.21, no.5, pp.1651-1665, Oct. 2013. Article (CrossRef Link).

[7] C. Pearce, S. Zeadally, "Ancillary Impacts of Multipath TCP on Current and Future Network Security," *IEEE Internet Computing*, vol.19, no.5, pp.58-65, 2015. Article (CrossRef Link).

[8] D. Zhou, W. Song, P. Wang, W. Zhuang, "Multipath TCP for User Cooperation in LTE Networks," *IEEE Network*, vol.29, no.1, pp.18-24, Jan. 2015. Article (CrossRef Link).

[9] Q. Peng, A. Walid and S. Low, "Multipath TCP Algorithms: Theory and Design," in *Proc. of ACM SIGMETRICS*, pp.1-12, Jun. 2013. Article (CrossRef Link).

[10] Y. Cui, L. Wang, X. Wang, Y. Wang, "FMTCP: A Fountain Code-Based Multipath Transmission Control Protocol," *IEEE/ACM Transactions on Networking*, vol.23, no.2, pp.465-478, 2015. Article (CrossRef Link).

[11] B. Arzani, A. Gurney, S. Cheng, R. Guerin, B. Loo, "Deconstructing MPTCP Performance," in *Proc. of IEEE ICNP*, pp.269-274, Oct. 2014. Article (CrossRef Link).

[12] S. Park, C. Joo, Y. Park, S. Bank, "Impact of traffic splitting on the delay performance of MPTCP," in *Proc. of IEEE ICC*, pp.1204-1209, 2014. Article (CrossRef Link).

[13] M. Li, A. Lukyanenko, S. Tarkoma, A. Ylä-Jääski, "MPTCP Incast in Data Center Networks," *China Communications*, vol.11, no.4, pp.25-37, 2014. Article (CrossRef Link).

[14] Y. Cao, M. Xu, X. Fu, E. Dong, "Explicit Multipath Congestion Control for Data Center Networks," in *Proc. of ACM CoNEXT*, pp.73-84, Dec. 2013. Article (CrossRef Link).

[15] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, M. Handley, "Improving datacenter performance and robustness with multipath TCP," in *Proc. of ACM SIGCOMM*, pp.266-277, 2011. Article (CrossRef Link).

[16] X. Corbillon, R. Aparicio-Pardo, N. Kuhn, G. Texier, G. Simon, "Cross-layer scheduler for video streaming over MPTCP," in *Proc. of ACM MMSys*, 2016. Article (CrossRef Link).

[17] H. Sinky, B. Hamdaoui, M. Guizani, "Proactive Multi-Path TCP for Seamless Handoff in Heterogeneous Wireless Access Networks," *IEEE Transactions on Wireless Communications*, vol.PP, no.99, 2016. Article (CrossRef Link).

[18] Y. Lim, Y. Chen, E.M. Nahum, D. Towsley, K. Lee, "Cross-layer path management in multi-path transport protocol for mobile devices," in *Proc. of IEEE INFOCOM*, pp.1815-1823, 2014. Article (CrossRef Link).

[19] J. Wu, C. Yuen, B. Cheng, M. Wang, J. Chen, "Streaming High-Quality Mobile Video with Multipath TCP in Heterogeneous Wireless Networks," *IEEE Transactions on Mobile Computing*, vol.15, no.9, pp.2345-2361, 2016. Article (CrossRef Link).

[20] C. Xu, H. Huang, H. Zhang, *et al*., "Multipath Transmission Control Protocol (MPTCP) Partial Reliability Extension," *IETF draft-xu-mptcpprmp-02.txt*, Apr. 2016. Article (CrossRef Link).

[21] J. Wu, C. Yuen, B. Cheng, M. Wang, J. Chen, "Energy-Minimized Multipath Video Transport to Mobile Devices in Heterogeneous Wireless Network," *IEEE Journal on Selected Areas in Communications*, vol.34, no.5, pp.1160-1178, May 2016. Article (CrossRef Link).

[22] C. Diop, G. Dugu, C. Chassot, E. Exposito, "QoS-oriented MPTCP extensions for multimedia multi-homed systems," in *Proc. of IEEE AINA*, 2012. Article (CrossRef Link).

[23] Y. Cao, Q. Liu, G. Luo, Y. Yi, M. Huang, "PR-MPTCP^{+}: Context-aware QoE-oriented Multipath TCP Partial Reliability Extension for Real-time Multimedia Applications," in *Proc. of IEEE VCIP*, 2016. Article (CrossRef Link).

[24] B. H. Oh, J. Lee, "Feedback-Based Path Failure Detection and Buffer Blocking Protection for MPTCP," *IEEE/ACM Transactions on Networking*, vol.PP, no.99, pp.1-12, 2016. Article (CrossRef Link).

[25] A. Alheid, A. Doufexi, D. Kaleshi, "A study on MPTCP for tolerating packet reordering and path heterogeneity in wireless networks," in *Proc. of IFIP Wireless Days conference, pp.1-7, 2016*. Article (CrossRef Link).

[26] M. Li, A. Lukyanenko, S. Tarkoma, Y. Cui, Yla-Jaaski, "Tolerating path heterogeneity in multipath TCP with bounded receive buffers," *Computer Networks*, vol.64, pp. 1-14, 2014. Article (CrossRef Link).

[27] Y. Cao, M. Xu, X. Fu, "Delay-based Congestion Control for Multipath TCP," in *Proc. of IEEE ICNP*, pp.1-10, Austin, USA, Oct. 2012. Article (CrossRef Link).

[28] F. Yang, P. Amer, "Non-renegable Selective Acknowledgments (NR-SACKs) for MPTCP," in *Proc. of IEEE AINA*, pp.1113-1118, 2013. Article (CrossRef Link).

[29] Y. Cao, C. Xu, J. Guan, H. Zhang, "Receiver-driven SCTP-based multimedia streaming services in heterogeneous wireless networks," in *Proc. of IEEE ICME*, 2014. Article (CrossRef Link).

[30] Y. Cao, Q. Liu, Y. Zuo, M. Huang, "Receiver-driven Cooperation-based Concurrent Multipath Transfer over Heterogeneous Wireless Networks," *KSII Transactions on Internet and Information Systems*, vol.9, no.7, pp. 2354-2370, 2015. Article (CrossRef Link).

[31] R. Stewart, "Stream Control Transmission Protocol," *IETF RFC 4960* (Proposed Standard), Sep. 2007. Article (CrossRef Link).

[32] J. R. Iyengar, P. Amer and R. Stewart, "Concurrent Multipath Transfer Using SCTP Multihoming over Independent End-to-end Paths," *IEEE/ACM Transactions on Networking*, vol.14, no.5, pp.951-964, Oct. 2006. Article (CrossRef Link).

[33] I. Rhee, V. Ozdemir, Y. Yi, "TEAR: TCP Emulation at Receivers-Flow Control for Multimedia Streaming," *Technical Report*, North Carolina State University. Available from: http://netsrv.csc.ncsu.edu/export/tcpemul.htm. Article (CrossRef Link).

[34] J. Liao, J. Wang, X. Zhu, "cmpSCTP: An Extension of SCTP to Support Concurrent Multi-Path Transfer," in *Proc. of IEEE International Conference on Communications* (ICC), pp.5762-5766, May 2008. Article (CrossRef Link).

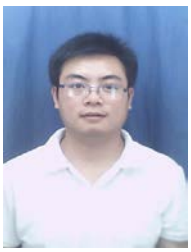
[35] K. Xue, J. Han, H. Zhang, K. Chen, P. Hong, "Migrating Unfairness among Subflows in MPTCP with Network Coding for Wired-Wireless Networks," *IEEE Transactions on Vehicular Technology*, vol.PP, no.99, 2016. Article (CrossRef Link).

[36] UC Berkeley, LBL, USC/ISI and Xerox Parc, NS-2 documentation and software, version 2.35

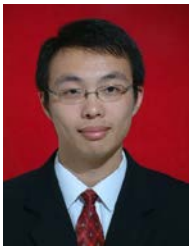
[37] Google Code Project, "Multipath-TCP: Implement multipath TCP on NS-2," http://code.google.com/p/multipath-tcp/. Article (CrossRef Link).

[38] C. Xu, T. Liu, J. Guan, H. Zhang, G. Muntean, "CMT-QA: Quality-aware Adaptive Concurrent Multipath Data Transfer in Heterogeneous Wireless Networks," *IEEE Transactions on Mobile Computing*, vol.12, no.11, pp.2193-2205, Nov. 2013. Article (CrossRef Link).

Yuanlong Cao received his Ph.D. degree from Institute of Network Technology, Beijing University of Posts and Telecommunications (BUPT) in Sep 2014, received his MS degree in software engineering from BUPT in July 2008, and received his BS degree in computer science and technology from Nanchang University, China, in July 2006. He worked as an intern/engineer in BEA TTC, IBM CDL, and DT Research (Beijing) from 2007 to 2011, respectively. He is a Lecturer with the School of Software at Jiangxi Normal University, China. His research interests include multimedia communications and next generation Internet technology.



Qinhua Liu is now working as an experimentalist in the Software of School, JiangXi Normal University (JXNU). He received the B.S. degree in software engineering from JXNU, China, in 2007, and his Master degree in management science and engineering in 2011, from JXNU. His research interests include multimedia networking and next generation Internet technology.



Yi Zuo is now working as a project manager in the Innovation Fund for Small Technology-based Firms (STF) of Jiangxi province. He received the B.S. degree from Jiangxi University of Finance and Economics, China, in July 2007. He received the Master Degree from Technische Universität Dresden, German, in May 2009. His research interests include next generation service creation technology.

Hao Wang is a professor and the chairman of the Software of School at JiangXi Normal University. He was elected as a “University Young and Middle-aged Academic Leaders in Jiangxi Province” and “Distinguished Teacher of Jiangxi Province”. His major research interests include computer network congestion control, computer network management.

Minghe Huang is a professor of the Software of School at JiangXi Normal University (JXNU), and the Jiangxi provincial government counselor. He has published more than 50 research papers in the areas of computer networks, communications, and information theory. He was the chairman of Software of School at JXNU, and the standing director of the Association of Fundamental Computing Education in Chinese Universities (AFCEC). Prof. Huang was elected as a “University Young and Middle-aged Academic Leaders in Jiangxi Province”, and “Distinguished Teacher of Jiangxi Province”.