Loss-Aware Throughput Estimation Scheduler for Multi-Path TCP in Heterogeneous Wireless Networks

Wenjun Yang, *Graduate Student Member, IEEE*, Pingping Dong, Lin Cai, *Fellow, IEEE*, and Wensheng Tang

***Abstract*— Multi-path TCP (MPTCP) is increasingly popular with the widespread usage of multihomed devices. MPTCP allows data streams to be delivered across multiple simultaneous connections, providing higher bandwidth aggregation and throughput in comparison with single-path TCP. However, due to the path heterogeneity and packet losses, the occurrence of Out-of-Order (OFO) packets is inevitable for MPTCP. Although many approaches have been proposed to mitigate OFO, most of them focused on compensating path delay differences but not considered the impact of packet loss. In this paper, we take the first step towards analyzing the impact of packet loss on OFO, and propose Loss-Aware Throughput Estimation scheduler, LATE. LATE comprehensively considers each subflow's path characteristics and protocol parameters including Round Trip Time (RTT), congestion window (cwnd), and loss rate, to predict the data amount that can be sent over each subflow at a given time and determine wisely which segments should be allocated to which subflows. Experimental results show that LATE achieves a gain of 5.13% in mean goodput with long-lasting flows while reducing the completion time of short flows by about 26.68% compared to the state-of-the-art scheduler for MPTCP.**

***Index Terms*— Multipath TCP, lossy network, out-of-order, scheduling policy, RTO.**

I. Introduction

MOBILE devices gain popularity, while the majority of them are equipped with multiple interfaces (e.g., 4G and Wi-Fi), preferring multipath transport protocols that can leverage multiple wireless paths to transfer data concurrently. Multi-path TCP (MPTCP) [1], a representative multipath transport protocol, has been proposed and adopted by IETF to not only enhance reliability in dealing with path failures but also achieve higher end-to-end throughput by concurrent multipath transfer (CMT) [2].

However, out-of-order packets at the receiver is a common problem for MPTCP [3]–[5]. Given the different characteristics of different paths in terms of path delay, bandwidth, and loss probability, packets transmitted over different paths may experience a different end-to-end delay, so a packet with a higher sequence number may arrive at the receiver earlier than that with a lower sequence number, namely, the Out-of-Order (OFO) problem [6]–[8]. The high transmission error rate in wireless networks is another cause of OFO [9]–[11]. If a packet with a lower sequence is lost and cannot arrive at the receiver timely, a receiver may have to store a large number of out-of-order packets while waiting for the lost one.

When the receiver buffer is limited, the OFO packets may eventually occupy the entire buffer, stalling the sender's transmission, resulting in the Head-of-Line Blocking (HoL-Blocking) problem [12]–[15]. It indicates that path heterogeneity and packet losses largely impair the performance of MPTCP, which is sensitive to OFO in heterogeneous lossy networks.

Recently, several solutions have been proposed to mitigate these problems. First, changing the receiver buffer, e.g., expanding it [14], [16] or splitting it [17], [18], is useful to store more out-of-order packets. However, using a larger receive buffer may lead to longer delay, not desirable for real-time data transmission [8].

Another category of solutions focused on the design of a packet pre-allocation policy that intentionally sends packets out of order while ensuring they arrive at the receiver in order. Sarwar *et al.* [19] presented DAPS to carefully choose and send packets based on the delay of the associated paths to receive packets in order. The principle of OTIAS [20] is the same as that in [19]. Ferlin *et al.* [12] showed, both analytically and experimentally that DAPS and OTIAS were unable to react upon network changes promptly due to high heterogeneity in subflow delays, and they presented BLEST to estimate whether a path will cause HoL-Blocking and dynamically adapt scheduling to prevent blocking. As these proposals did not fully address the influence of the lossy

Manuscript received June 8, 2020; revised October 17, 2020; accepted December 28, 2020. Date of publication January 12, 2021; date of current version May 10, 2021. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), in part by the Compute Canada, in part by the China Scholarship Council (CSC), in part by the National Natural Science Foundation of China under Grant 61602171, and in part by the Hunan Province's Strategic and Emerging Industrial Projects under Grant 2018GK4035. The associate editor coordinating the review of this article and approving it for publication was L. Zhao. *(Corresponding author: Pingping Dong.)*

Wenjun Yang is with the Hunnan Provincial Key Laboratory of Intelligent Computing and Language Information Processing, Hunan Normal University, Changsha 410081, China, and also with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8P 5C2, Canada (e-mail: wenjunyang@uvic.ca).

Pingping Dong and Wensheng Tang are with the Hunnan Provincial Key Laboratory of Intelligent Computing and Language Information Processing, Hunan Normal University, Changsha 410081, China, also with the College of Information Science and Engineering, Hunan Normal University, Changsha 410081, China, and also with the Hunan Xiangjiang Artificial Intelligence Academy, Changsha 410081, China (e-mail: ppdong@csu.edu.cn; tangws@hunnu.edu.cn).

Lin Cai is with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC V8P 5C2, Canada (e-mail: cai@ece.uvic.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2021.3049300.

Digital Object Identifier 10.1109/TWC.2021.3049300

network, Xue *et al.* [21] proposed DPSAF, which considers packet loss and uses the information in SACK to detect and prevent out-of-order packets. However, with severe packet losses, MPTCP may suffer from retransmission timeout (RTO) especially when transmitting short flows [22], [23], which is not considered by DPSAF. J. Padhye *et al.* [24] observed that there are more timeout events than fast retransmit events in almost all of their experimental traces, and the majority of window decreases are due to time-outs, rather than fast retransmits.

In this paper, we first analyze the impact of packet losses on in-order arrival. Then we develop a novel Loss-Aware Data Estimation scheduler for MPTCP, LATE, by considering subflow's characteristics including round trip time (RTT), congestion window (cwnd), and loss rate, to predict the data amount that can be sent out over each subflow at a given time. Therefore, we schedule different numbers and sequences of packets into each subflow ensuring in-order arrivals.

The main contributions of this paper are three-fold.

- We first theoretically and experimentally study the existing MPTCP performance, which indicates that the loss rate plays an important role in predicting each subflow's transmission capacity within a certain time.
- To improve prediction accuracy, we propose a transmission model that comprehensively considers the situation of fast retransmission and RTO to calculate the data amount that can be sent out. Besides, all the transmission parameters, including time offset, RTT, cwnd, slow start threshold (sst) and loss rate, are also taken into consideration in the proposed estimation model.
- We not only implement LATE scheduler based on MPTCP-enabled Linux kernel to validate its performance in heterogeneous wireless networks consisting of both 4G and WiFi channels in the real world, but implement LATE on basis of Network Simulator 3 (NS3) [25] to gain some insights about how LATE behaves whenever the number of subflows is greater than 2. The experimental results show that LATE outperforms the state-of-the-art DPSAF in reducing the completion time of short flows by 26.68% and increasing the mean goodput of long flows by 5.13% respectively.

The rest of this paper is organized as follows. Section II introduces the related work. Section 3 analyzes the impact of packet losses on in-order arrival and introduces our motivation for this paper. Section 4 proposed the LATE scheduler. Section 5 presents the evaluation of LATE under different experimental scenes. Finally, section 6 concludes this paper.

II. MPTCP AND RELATED WORK

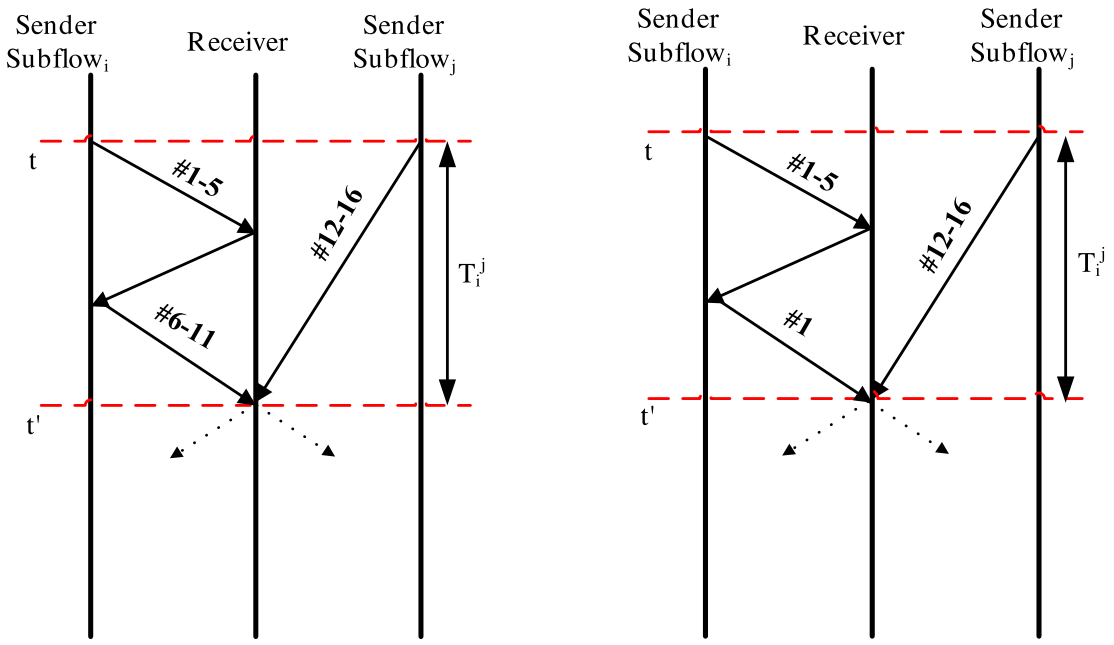
MPTCP has been proposed for supporting multi-homing, which can offer high bandwidth and reliability [26]. In practice, its performance is impaired by several factors [27]. To mitigate the impairment, various multipath algorithms have been proposed recently in two main categories: congestion control and path scheduling.

The congestion control algorithms of MPTCP mainly focuses on adjusting the transmission rate of each subflow and shifting traffic from more congested paths to less congested ones, thereby improving the throughput and link utilization. A large number of congestion control algorithms have been developed such as Linked Increases Algorithm (LIA) [28], Opportunistic Linked-Increases Algorithm (OLIA) [29], Balanced Linked Adaptation (Balia) [30], Weighted Vegas (wVegas) [31], mVeno [32]. These algorithms rely on an arithmetic model to control the increase of each subflow's cwnd, to balance the congestion level of them.

On the other hand, the path scheduling policy is designed to rationally split data packets over multiple paths to improve MPTCP performance [22]. There are many scheduling algorithms, such as the Lowest-RTT-First (minRTT) scheduler, Constraint-based proactive scheduling (CP) [33], Highest Sending Rate (HSR), Largest Window Space (LWS), and Lowest Time/Space (LTS) [34].

Although the MPTCP performance can be enhanced with these proposals, it still suffers from serious OFO problems, especially in lossy networks. To mitigate the impact of OFO packets on throughput, an intuitive solution is buffer management, namely, expanding the receiver buffer [14], [16], splitting the buffer space of receiver [17], [18] or predicting the appropriate buffer size over time [35] to accommodate much more OFO packets, so the receiver cache will not be fully occupied and stalled by a large amount of OFO packets. These solutions consider the trade-off between large buffer size and fixed capacity with limited buffer size. However, using a larger receive buffer may lead to longer delay, not desirable for real-time data transmission [8].

Then, researchers start to focus on designing a pre-allocate packet policy that intentionally sends packets out of order to arrive in order. Delay-Aware Packet Scheduler (DAPS) [19] determines which segments should be sent over which subflows by considering both forward delay and cwnd of each subflow, to make segments arrive in order. OTIAS [20] evaluated and extended the idea of DAPS to schedule more segments on a fast subflow. Queues may build up at each subflow under the assumption that these segments will be sent as soon as there is space in the cwnd for the subflow. However, no consideration is given to segment reinjection if a certain path is blocking the connection [12]. This situation motivates Ferlin *et al.* [12] to present BLEST to resolve the problem of HoL-Blocking caused by OFO packets. Rather than stopping the slow subflows, BLEST estimates whether the slow path will cause HoL-blocking and dynamically schedules the packets over certain subflows to prevent blocking. These existing algorithms can reduce OFO number and improve network throughput, yet they suffer from significant performance degradation when the path heterogeneity is too large or losses occur frequently. Thus, Xue *et al.* [21] proposed DPSAF, which combines a loss-based throughput estimation model and SACK information to detect and prevent out-of-order packets. However, DPSAF considers only two cases: no packet loss and fast retransmission but not RTO, where RTO is typically set to several times of RTT, leading to severe performance degradation. On the other hand, DPSAF chooses the maximum probability of different cases to conduct the estimation, which cannot appropriately show the expected value of data



(a) No packet loss and no OFO

(b) Packet 1 is lost and then packets #12-16 are disordered

Fig. 1. The impact of packet loss on OFO.

amount and lose some accuracy accordingly. Although the analysis on TCP's average throughput has been heavily investigated [36]–[38], for MPTCP, the analytical model should capture the instantaneous rate and latency, which remains an open issue and motivates this work. In this paper, we develop a new prediction model that comprehensively considers slow start, congestion avoidance, lost recovery phases to assist the scheduler to ensure in-order arrival at the receiver side.

III. Motivation

We first study the performance of the state-of-the-art scheduling algorithm BLEST, to analyze the problems of those prediction algorithms which do not consider packet loss.

BLEST mainly pursues the goal of making segments arrive in order, thus mitigating the problem of HOL-Blocking caused by OFO packets. BLEST is formulated for a scenario with two classes of subflows, i.e., fast subflow ($subflow_i$) and slow subflow ($subflow_j$). The basic idea of BLEST is to prevent the amount of data being sent from the sender from surpassing the available space of the receiver buffer, so it introduces a variable named total MPTCP send window ($MPTCP_{sw}$), which equals to the receiver window ($rwnd$). BLEST specifies that the sum of cwnd of $subflow_i$ and that of $subflow_j$ cannot exceed $MPTCP_{sw}$.

See Fig. 1a for an example, assume that the available send window $MPTCP_{sw}$ is 20 packets at packet scheduling time t, the RTT of $subflow_i$ and $subflow_j$ are 10 ms and 20 ms respectively, and both of the cwnds are 5 packets. According to the estimation of BLEST, $subflow_i$'s cwnd is increased by 1 for every RTT_i as it is in congestion avoidance, this will last for two rounds within RTT_j as the value of $\text{RTT}_j/\text{RTT}_i$ is two, and the amount of data X_i that will be sent on $subflow_i$ within RTT_j would be 11 packets (5 packets in the first round and 6 packets in the second round). Since $MPTCP_{sw}$ is greater than 11, $subflow_j$ is allowed to be allocated X_j packets, where X_j is calculated by $min\{MPTCP_{sw} - X_i, cwnd_j\}$. Consequently, segments from 1 to 11 will be in flight on $subflow_i$, and segments from 12 to 16 will be in flight on $subflow_j$. Ideally, at time t' when packets 1-11 arrive at the receiver, new packets 12-16 also arrive at the same time, so there is no disordered packet at the receiver at time t'.

However, in the lossy heterogeneous network, this ideal situation discussed above may not hold. We assume that a packet (e.g., packet 1) is lost during the transmission over $subflow_i$, MPTCP sender has to recover the lost one in the next round through the recovery mechanism, namely fast retransmission or RTO. No matter which recovery mechanism is used, packets 6-11 cannot all reach the receiver earlier than packets 12-16, as described in Fig. 1b, as the MPTCP sender has to retransmit the lost packet but not to send new packets 6-11 in the second round. Therefore, these estimation models are not suitable for lossy networks.

We further conduct some experiments based on the Linux testbed to verify the impact of loss rate on BLEST. Here the file size of the transmitted data is set to 1MB, and the loss rates are set to 0, 0.5%, 1%, 3%, 5% respectively.

As shown in Fig. 2, with the increase of packet loss rate, the mean goodput of both TCP and BLEST decreases, while BLEST achieves a higher average goodput than TCP as it can leverage multiple paths to transmit data concurrently. However, when the loss rate goes up to a high value, the average goodput of BLEST decreases more drastically than TCP. We can observe that the multipath transmission protocol is more susceptible to packet loss.

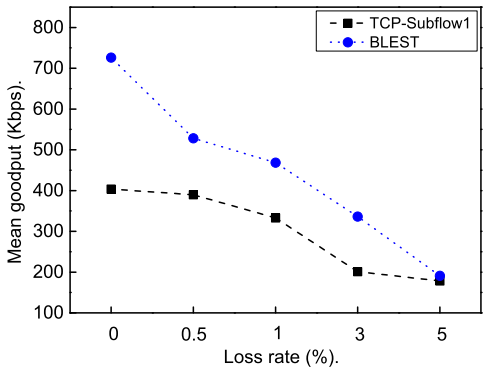


Fig. 2. The mean goodput of each algorithm under different loss rates.

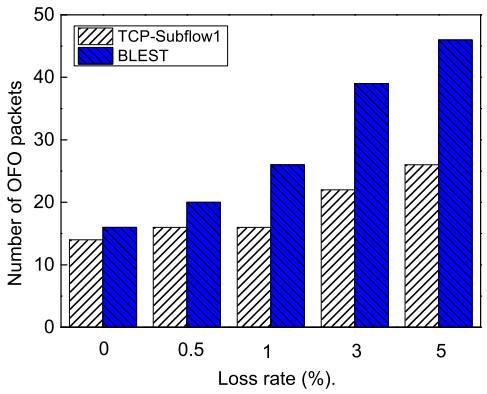


Fig. 3. The number of OFO packets of each algorithm under different loss rates.

To reveal this cause of BLEST's performance degradation, we count the number of OFO packets at the receiver to make a comparison between TCP and BLEST. The results are shown in Fig. 3. Compared with BLEST algorithm, the number of OFO packets of TCP is always lower. When the loss rate is small (e.g., 0 or 0.5%), the number of OFO packets is 20 at most, it is not enough to occupy all of the receiver caches to block transmission progress as the default receiver buffer size is 65536 bytes in Linux kernel, so it has little influence on BLEST in terms of the mean goodput in this situation. However, when the packet loss rate exceeds 1%, the number of OFO packets increases significantly, and a large number of OFO packets occupy the cache of the receiving side. The sender window, which is given by $min\{rwnd, cwnd\}$, consequently becomes extremely small, and the mean goodput of BLEST decreases largely.

Based on the above analysis, we conclude that utilizing RTT only to schedule packets over each subflow is insufficient. It motivates us to design a more accurate model to pre-allocate data packets out of order to minimize OFO arrival.

IV. The Proposed Algorithm LATE

To effectively reduce the OFO number at the receiver within a certain period T_i^j which is visually reflected in Fig. 1, i.e., the elapsed time from time t to t' that can be roughly set as $RTT_j/2$, in this section, we first utilize MPTCP with two subflows, i.e., $subflow_i$ and $subflow_j$, to formulate a loss-based transmission model that accurately estimates the amount of data transmitted over fast subflow ($subflow_i$). Without loss of generality, we then elaborate on how LATE works for MPTCP with more than two subflows, as well as how it selects packets from the sending pool for each subflow based on the former prediction.

A. Packet Loss vs Round Trip Time

During MPTCP transmission process, four intertwined phases, i.e., slow start, coupled congestion avoidance, fast retransmit, and RTO, involve in congestion control and data recovery. To estimate the data amount of $subflow_i$ at a given time, here we first give a discussion about these phases.

According to the TCP transmission model, if there is no packet loss in one round, all packets of this round would arrive at the receiver successfully after $RTT_i/2$, and the next round will step into the slow start or congestion avoidance phase accordingly. This situation is shown in Fig. 4a. However, if one packet loss happens, the MPTCP sender has to recover from the lost one through fast retransmit or RTO. Once the MPTCP sender receives a certain number (i.e., three) of duplicated ACKs (dupACKs), it reacts quickly to retransmit the lost packets, namely fast retransmit, which spends almost $RTT_i + RTT_i/2$ to deliver all the packets of the current round to the receiver successfully. Fig. 4b briefly illustrates this process. Last but not least, when transmitting a small amount of data, the more the number of paths being utilized, the fewer the number of packets being scheduled into each path. If any packet is lost, there may not have enough dupACKs, so the MPTCP sender has to rely on RTO to retransmits all un-ACKed packets after the timeout. In this situation, it takes $RTO_i + RTT_i/2$ to transmit all packets successfully. This situation is shown in Fig. 4c.

Based on the above analysis, we redefine the complete round r is the process that includes new packet transmission and lost retransmission if possible, ensuring that all new packets are delivered successfully to the receiver within this round. Therefore, the duration of one round varies in a different loss situation.

B. The Proposed LATE Estimation Model

The throughput estimation model is at the heart of Loss-Aware Throughput Estimation scheduler (LATE). The variables used in this model are listed in Table I where w and sst represent the congestion window and slow start threshold, respectively.

Given specific path characteristics such as cwnd, RTT and loss rate, this model aims at calculating the number of data packets ($N(T_i^j)$) that can be delivered to the receiver over the subflow (i.e., $subflow_i$) with a smaller RTT during the period of T_i^j. For instance, there might be multiple rounds the sender can run in time T_i^j, we initially count the time t_1 the sender will cost to make all packets of the first round arrive at the receiver successfully. After that, if there is still time left (i.e., $T_i^j - t_1 > 0$), we further calculate the second round of data amount $N(T_i^j - t_1)$ by obtaining parameters from the

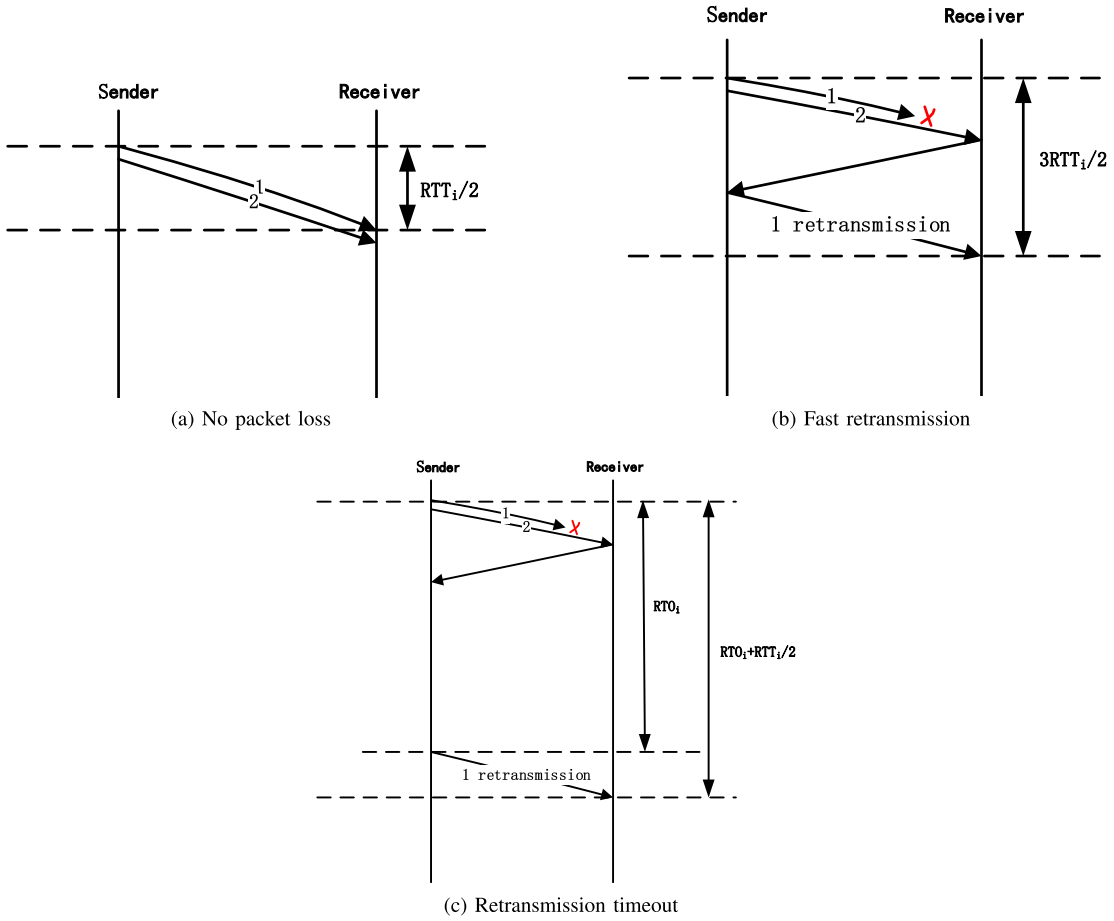


Fig. 4. The different situations during the transmission process.

TABLE I
THE PARAMETERS AND THEIR PHYSICAL SIGNIFICANCE IN THE LATE TRANSMISSION MODEL

Parameters	Physical Significance
$subflow_j$	A slow subflow with larger RTT
$subflow_i$	The fast subflow whose RTT is smaller than that of $subflow_j$
T_i^j	The largest elapsed time from time t when packets start to be transmitted over $subflow_i$ and $subflow_j$ to time t' when the last packet arrives at the receiver.
$T(r)$	The remaining time of the current round r
$w(r)$	The congestion window of $subflow_i$ at the current round r
$sst(r)$	The slow start threshold of $subflow_i$ at the current round r
p	The loss rate of $subflow_i$
$N^{(r)}(w(r))$	The number of packets can be delivered successfully to the receiver within period of $T(r)$ given the initial value of $w(r)$
N_j	The start number of the segments being scheduled to $subflow_j$
$SEG_{P_i'}$	Packet segments being scheduled to subflow P_i'
$SEG_{P_j'}$	Packet segments being scheduled to subflow P_j'
S_{min}	The minimum sequence number of those segments waiting to be sent

former round and repeat this cycle until the total time T_i^j runs out. Therefore, the calculation of the total number of packets N can be written as

$$N(T_i^j) = n_1 + N(T_i^j - t_1). \tag{1}$$

where the n_1 is the number of packets that reach the receiver during the first round.

According to (1), the calculation of $N(T_i^j)$ is a recursive process. Without loss of generality, we use variable $T(r)$ to denote the remaining time when the r-th ($r = 1, 2, 3, \ldots$) round starts, and $N^{(r)}(w(r))$ to denote the number of packets can be delivered successfully to the receiver within $T(r)$, where $w(r)$ is the cwnd when the r-th round transmission started. For different values of $T(r)$, there are four conditions as follows.

1) $T(r) < RTT_i/2$*:* The time is too limited to transmit a new packet. In other words, no packet can reach the

receiver. Then we have

$$N^{(r)}(w(r)) = 0. \tag{2}$$

2) $RTT_i/2 \leq T(r) < 3 \cdot RTT_i/2$: The time is enough to transmit new packets in the r-th round, but not enough to recover the lost packets through retransmission or start the $(r+1)$-th round of transmission. Denoted by $P(x|w(r))$ the probability of x packets being lost when the current cwnd is $w(r)$. As the loss rate of $subflow_i$ is l and packet losses are independent of each other [38]–[40], the value of $P(x|w(r))$ obeys the Bernoulli formula shown in (3).

$$P(x|w(r)) = \binom{w(r)}{x} * l^x * (1-l)^{w(r)-x}. \tag{3}$$

We can calculate the number of packets that can be successfully transmitted. That is,

$$N^{(r)}(w(r)) = w(r) - \sum_{x=0}^{w(r)} P(x|w(r)) * x. \tag{4}$$

3) $3 \cdot RTT_i/2 \leq T(r) < RTO_i + RTT_i/2$: This means that the time is enough for the sender to complete the current round of transmission and even start the next round of transmission. However, if there are lost packets, it can be recovered through fast retransmission but not RTO retransmission. The cwnd change behavior of this situation is illustrated by means of the TCP-based Markov chain diagram [39], [40] shown in Fig. 5, in which every state consists of two elements $(w(r), sst)$, but for ease of reading, only $w(r)$ is shown in the circles, sst is labeled at the bottom of each column and the states in the same column have the same sst. Note that the end nodes with W notation are assumed to model the largest cwnd, whose value depends on the bandwidth-delay product (BDP) and the buffer size of the network bottleneck [41]. Given certain $(cwnd, sst)$, there could be three state transition directions as follows:

- ***Slow Start (SS) or Congestion Avoidance (CA)*:** All packets are transmitted successfully and no loss occurs in the r-th round. The number of packets arrived at the receiver during the r-th round is $w(r)$ and the duration of the r-th round is RTT_i. The remaining time $T_1(r+1)$ is abundant to start the $(r+1)$-th round in the SS or CA state, and thus the state changes from $(w(r), sst)$ to $(2 * w(r), sst)$ or $(w(r)+1, sst)$ with a transition rate of $p_1(r)$. We can obtain $p_1(r)$ and other parameters as follows.

$$p_1(r) = \binom{w(r)}{0} * l^0 * (1-l)^{w(r)}, \tag{5}$$

$$T_1(r+1) = T(r) - RTT_i, \tag{6}$$

$$w_1(r+1) = \begin{cases} 2 * w(r), & w(r) < sst(r) \\ w(r)+1, & w(r) \geq sst(r), \end{cases} \tag{7}$$

$$sst_1(r+1) = sst(r). \tag{8}$$

Note that $w_1(r+1)$ and $sst_1(r+1)$ are the definitions of $cwnd$ and sst when the $(r+1)$-th round starts, respectively. Then we have

$$\hat{N}_1^{(r)}(w(r)) = p_1(r) * w(r) + \hat{N}_1^{(r+1)}(w_1(r+1)). \tag{9}$$

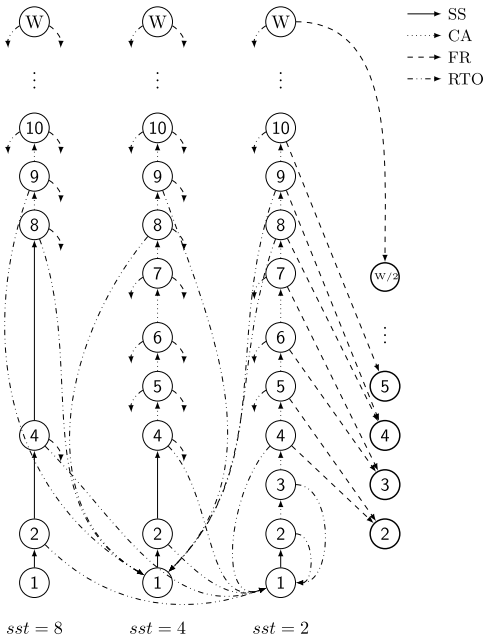


Fig. 5. State transitions of each MPTCP subflow.

where $\hat{N}_1^{(r)}(w(r))$ is the equivalent of $N^{(r)}(w(r))$ in the first category.

- ***Fast Retransmission (FR)*:** m lost packets in the r-th round would have the chance to be recovered through FR if m is less than or equal to $w(r)-3$. It implies that the triggering of FR requires a current cwnd of at least 4, as any loss in states $(2,2)$ and $(3,2)$ shown in Fig. 5 leads to RTO only, not FR. In Fig. 5, the rightmost column with thick circles refers to states undergoing fast retransmission, the states change from $(w(r), sst)$ to $(\lfloor w(r)/2 \rfloor, \lfloor w(r)/2 \rfloor)$. Since the $sst(r+1)$ has nothing to do with $sst(r)$, this rightmost column should be identical for $sst = 2, 4, 8$. To keep the figure readable, only the case for $sst = 2$ is shown. Since this situation takes $3 \cdot RTT_i/2$ to ensure lost packets arriving the receiver end and the next round will take place after $2 \cdot TT_i$, we have the probability $p_2(r)$ of this state transition and the next round parameters as below

$$p_2(r) = \sum_{m=1}^{w(r)-3} \binom{w(r)}{m} * l^m * (1-l)^{w(r)-m}, \tag{10}$$

$$T_2(r+1) = T(r) - 2 * RTT_i, \tag{11}$$

$$w_2(r+1) = \lfloor w(r)/2 \rfloor, \tag{12}$$

$$sst_2(r+1) = \lfloor w(r)/2 \rfloor, \tag{13}$$

$$\hat{N}_2^{(r)}(w(r)) = p_2(r) * w(r) + \hat{N}_2^{(r+1)}(w_2(r+1)). \tag{14}$$

- ***RTO*:** The triggering condition of FR is as opposed RTO which happens when there are more than $w(r)$ - 3 lost packets in the r-th round, as instantiated in Fig. 5, every state is possible to jump into RTO phase with a huge $cwnd$ reduction and consequently performance

degradation. The importance of RTO consideration is also stressed by J. Padhye *et al.* [24], as they verified that timeout events happen more frequently than fast retransmits events, and majority of window decreases are due to timeouts, rather than fast retransmits. The probability $p_3(r)$ of RTO occurrence is yield by

$$p_3(r) = \sum_{m=w(r)-2}^{w(r)} \binom{w(r)}{m} * l^m * (1-l)^{w(r)-m}. \quad (15)$$

However, in this case, $T(r)$ is not large enough to trigger RTO and the next round $(r+1)$ would not start up. $T_3(r+1)$ is zero and the total number of packets $\hat{N}_3^{(r)}(w(r))$ is

$$\hat{N}_3^{(r)}(w(r)) = \sum_{m=w(r)-2}^{w(r)} \binom{w(r)}{m} * l^m * (1-l)^{w(r)-m} * (w(r)-m). \quad (16)$$

Based on the above analysis, we can conclude that, given the specific $T(r)$, the recursive process is going to go through all three situations with corresponding probability. After several iterations, the value of $N^{(r)}(w(r))$ can be obtained by

$$N^{(r)}(w(r)) = \sum_{i=1}^{3} \hat{N}_i^{(r)}(w(r)). \quad (17)$$

4) T(r) $\geq$ $RTO_i + RTT_i/2$: In this condition, $T(r)$ is large enough to finish the r-th round transmission, as well as retransmit the lost packets through fast retransmission or RTO. Apparently, the transmission process of this condition includes the similar three categories as condition 3), while the ***RTO*** situation here differs from that of condition 3). Below we only have the distinct RTO situation discussed.

Because the time is enough to recover the packets through RTO, $w(r)$ packets are able to reach the receiver within $RTO_i + RTT_i/2$. In comparison of condition 3 where $T(r) < RTO_i + RTT_i/2$, the RTO probability $p_3(r)$ is the same, but sender can deliver $w(r)$ packets in r-th round rather than $w(r) - m$ $(m = w(r) - 3, \ldots, w(r))$, and the next new round $(r+1)$ will start with the following parameters:

$$p_3(r) = \sum_{m=w(r)-2}^{w(r)} \binom{w(r)}{m} * l^m * (1-l)^{w(r)-m}, \quad (18)$$

$$T_3(r+1) = T(r) - (RTO_i + RTT_i/2), \quad (19)$$

$$w_3(r+1) = 1, \quad (20)$$

$$sst_3(r+1) = \lfloor w(r)/2 \rfloor. \quad (21)$$

The value of $\hat{N}_3^{(r)}(w(r))$ has changed accordingly, i.e.,

$$\hat{N}_3^{(r)}(w(r)) = p_3(r) * w(r) + \hat{N}_3^{(r+1)}(w_3(r+1)). \quad (22)$$

By merging equations (2) (4) (17), we re-formulate $N^{(r)}(w(r))$ calculation as below. Consequently, all behaviors caused by packet loss are involved in (23), shown at the bottom of the next page, leading to a more accurate estimation on data amount $subflow_i$ can process within a given period.

C. Scheduling Policy of LATE

The key idea of LATE is to first employ the transmission model proposed above to estimate the data amount N that each $subflow_i$ can deliver, given the parameters set $P = \{T_i^j, RTT, cwnd, ssthresh, l\}$ of each round. Then, it will schedule packets adaptively into different subflow based on their transmission capacity to make in-order arrival.

The pseudo-code of the LATE algorithm is given in **Algorithm 1**. To generalize LATE to the scenario where there are n $(n \geq 2)$ subflows in networks, two concepts, i.e., master-subflow and slave-subflow, are introduced. Master-subflow stands for the subflow with the largest RTT among n subflows, the rest of the subflows with smaller RTT constitute its slave-subflows set. To make sure the packets ran over all slave-subflows arrive at the receiver no later than those over master-subflow, LATE algorithm operates in two folds.

The first step is to determine the subflow order of data reception and initialize the modeling time T_i^j for all slave-subflows. As illustrated in line 2 of **Algorithm 1**, given the initial subflow set $P = \{P_1, P_2, \ldots, P_k, \ldots, P_n\}$, LATE sorts subflows based on RTT of each subflow to obtain the subflow set $P' = \{P_1{}', P_2{}', \ldots, P_k{}', \ldots, P_n{}'\}$ in the ascending order based on RTT. Therefore, for master-subflow $P_n{}'$, its slave-subflow set includes $P_1{}'$ up to $P_{n-1}{}'$. By inheriting the setting of T_i^j in two-subflows scenario which is $RTT_j/2$, the T_i^j for $n-1$ slave-subflows can be set to $RTT_n{}'/2$, where $RTT_n{}'$ is RTT of $P_n{}'$, this process is done by line 8 of **Algorithm 1**.

The next step is to derive packet sequence number to transmit per path using expected reception order. By employing Eq. 23 to recursively calculate $N^{(r)}(w(r))$, LATE obtains how many packets (N_i) each $subflow_i$ $(i \in [1, \text{n-1}])$ can deliver within T_i^j. Note that $t(r_i)$ located at the line 13 of **Algorithm 1** stands for the expected duration of round r_i. Afterwards, as shown in line 15 of **Algorithm 1**, LATE will select N_i packets from the start point of the sending buffer and append those packets sequence to $SEG_{P_i{}'}$ to transmit. After traversing through all $subflow_i$, LATE returns the sum of each N_i, namely S_{min} whose physical meaning is the last sequence number of $SEG_{P_{n-1}{}'}$, the segment allocated for master-subflow would start from $S_{min}+1$ and end with $S_{min} + w_n$. Eventually, LATE obtains the whole segment set $S = \{SEG_{P_1{}'}, SEG_{P_2{}'}, \ldots, SEG_{P_n{}'}\}$, and schedules corresponding segments from the sending buffer to fill the cwnd of each subflow, thereby ensuring the arrival of packets in sequence.

LATE scheduler is repeatedly launched after receiving ACKs with master-subflow, i.e., after around the $RTT_n{}'$ interval, LATE will refresh parameters of n subflows and obtain the new modeling time T_i^j to start the next scheduling decision. In this way, LATE can dynamically respond to the change of path characteristics, effectively offsetting the estimation error in the previous round.

V. Evaluation

In this section, we conduct a series of experiments to validate the performance of the proposed LATE. To reveal

Algorithm 1 The Proposed LATE Algorithm

Input : A set of established paths $P = \{P_1, P_2, \ldots, P_k, \ldots, P_n\}$ with given path parameters.

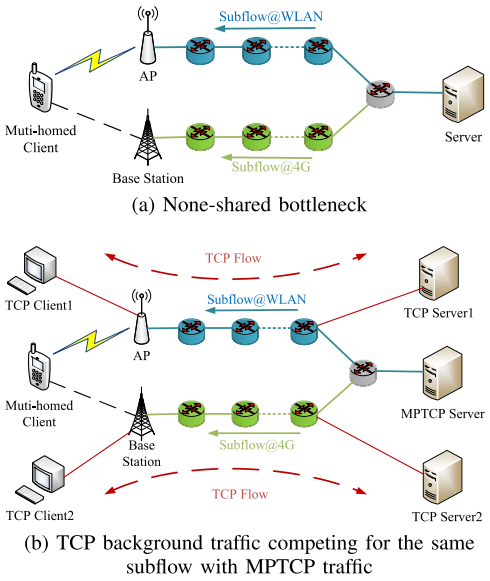
Output: A set of segment sequences $S = \{SEG_{P_1'}, SEG_{P_2'}, \ldots, SEG_{P_n'}\}$ that selected for delivery.

Initialization at time t
  Generate $P' = \{P_1', P_2', \ldots, P_k', \ldots, P_n'\}$ *by sorting RTT of* P *in the ascending order*
  $S_{min} = 0$
  $T_j \leftarrow maxRTT(P')/2$
  for each $P_i' \in P'(i \in [1, n-1])$ **do**
    $SEG_{P_i'} \leftarrow InitializeSeg()$
    $r_i = 0$
    $T_i^j(r_i) = T_j$
  $SEG_{P_n'} \leftarrow InitializeSeg()$
for each $P_i' \in P'$ $(i \in [1, n-1])$ **do**
  while $T_i^j(r_i) \geq RTT_i/2$ **do**
    $N_i += N^{(r_i)}(w(r_i))$
    $T_i^j(r_i + 1) -= t(r_i)$
    $r_i \leftarrow r_i + 1$
  $SEG_{P_i'} \leftarrow Append(SEG_{P_i'}, [S_{min} + 1, S_{min} + N_i])$
  $S_{min} \leftarrow S_{min} + N_i$
$SEG_{P_n'} \leftarrow Append(SEG_{P_n'}, [S_{min} + 1, S_{min} + w_n])$
Return $S = \{SEG_{P_1'}, SEG_{P_2'}, \ldots, SEG_{P_n'}\}$

the effect of each algorithm in terms of OFO reduction, we modify the Linux kernel to implement BLEST, DPSAF, and LATE, to compare these algorithms with the default MPTCP scheduler minRTT in different scenarios. In addition, to observe LATE's behaviors under networks where there are more than 2 subflows, we implemented LATE based on NS3 to make some further analyses.

A. Testbed Construction and Experimental Methodology

In this subsection, we first demonstrate how we construct the testbed to match real networks. As shown in Fig. 6a, we deploy a typical multi-homing scenario (i.e., a client having two access network like WLAN/4G) [12], [42] where we can set different loss rate and RTT for different paths, and then examine whether LATE can alleviate the impairment of path loss and path heterogeneity to reduce OFO packets at the receiver. Another scenario is that TCP and MPTCP controlled flows coexist and share the same bottleneck as



(a) None-shared bottleneck

(b) TCP background traffic competing for the same subflow with MPTCP traffic

Fig. 6. Experimental topology for the real-world traffic emulations.

shown in Fig. 6b, probably causing severe buffer overflow and packet loss problems at the bottleneck link if the traffic is bursty or heavy. Therefore, we conduct a series of experiments over the scenario shown in Fig. 6b to validate the behavior of LATE.

To enable multi-path communication between client and server shown in Fig. 6, these machines are running Linux Ubuntu 12.10 OS with kernel version 3.14.33 that have already applied the MPTCP-enabled protocol patches. The client uses a stock version of MPTCP kernel and runs on DELL Optiplex 745, equipped with Intel PentiumD 3.4G processor, 512MB RAM and 160 GB hard disk. The server uses a modified version including the BLEST, DPSAF, LATE, as well as default minRTT schedulers, running on Dell T1500, equipped with Intel Xeon E5620 (2.4GHz/12M), 16 GB RAM, and a 600 GB Hard Disk. The two subflows (i.e., $subflow_{wlan}$ and $subflow_{4g}$) are established between MPTCP client and MPTCP server by equipping with two Gigabit network interface cards. The multiple connected routers shown in Fig. 6 run WANem to construct a two-way bottleneck link, where WANem is a wide area network emulator that supports various wide area network characteristics such as bandwidth limitation, latency, packet loss, network disconnection, and so on.

Next, to evaluate the impact of the number of subflows, we implemented LATE scheduler by employing NS3 modeling and simulations, based on the NS3 open source MPTCP implementation [43]. Fig. 7 shows that M applications run between

$$N^{(r)}(w(r)) = \begin{cases} 0, & T(r) < RTT_i/2 \\ w(r) - \sum_{x=0}^{w(r)} P(x|w(r)) * x, & RTT_i/2 \leq T(r) < 3 \cdot RTT_i/2 \\ \sum_{i=1}^{3} \hat{N}_i^{(r)}(w(r)), & T(r) \geq 3 \cdot RTT_i/2. \end{cases} \tag{23}$$

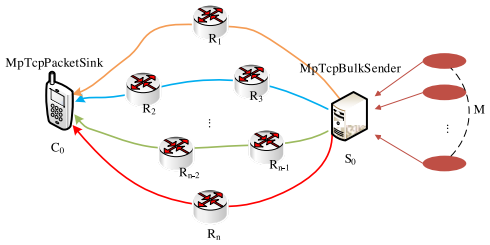


Fig. 7. The NS3-based topology to investigate the LATE performance when there are more than 2 MPTCP subflows between sender and receiver.

C_0 and S_0 which are connected through n routers, where C_0 and S_0 are equipped with MpTcpPacketSink and MpTcpBulkSender applications respectively, thus forming $N(N = 1, 2, 3, 4 \ldots)$ independent MPTCP subflows between them. The communication uses a point-to-point model and all nodes are connected using links with configurable data rate and delay.

B. Experimental Results Based on Real-Network Emulations

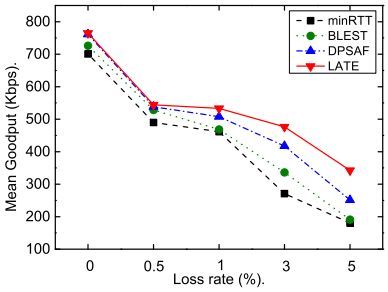
We investigate the behavior of different schedulers with real-network experiments in the following two cases: 1) None-shared Bottleneck. As shown in Fig. 6a, this is the typical scenario where the mobile client is connected to the server by two disjoint subflows without a shared bottleneck. 2) Competing traffic. As shown in Fig. 6b, the regular TCP background traffic competes with MPTCP traffic for path resources. We compare the performance of each algorithm in terms of goodput, flow completion time, the RTO rate as well as the number of OFO packets at the receiver.

1) None-Shared Bottleneck: In this scenario where the link bandwidth is configured to 100 Mbps and RTT ranges from 10 ms to 100 ms according to prior work [44], we explore the performance of each algorithm under the impact of different metrics including loss rate, RTT difference and the size of transmitted file.

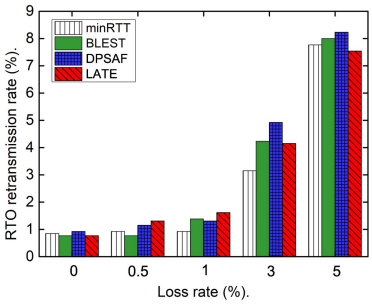
First of all, the impact of the loss rate is evaluated. Fig. 8 describes the behavior of each algorithm with different loss rate during the transmission of 20 concurrent 1MB files. Fig. 8a shows the change of goodput each algorithm achieved, and Fig. 8b counts the corresponding number of OFO packets at the receiver.

From Fig. 8a, we observe that the larger the loss rate, the smaller the goodput and the more OFO packets exist. This is because as the packet loss rate increases, the network becomes more dynamic and the accuracy of any prediction scheduling policy will be reduced. However, DPSAF and LATE perform better than BLEST and minRTT in the presence of packet loss. On the other hand, compared with DPSAF, LATE is more accurate in terms of accuracy of throughput estimation within each time slot, thus making a reasonable way of packet allocation. This illustrates the truth that LATE outperforms DPSAF.

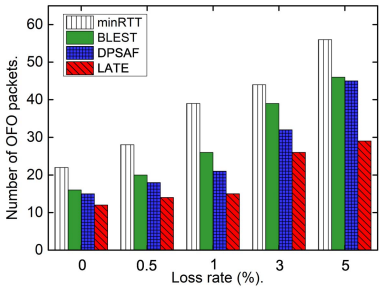
To further explore the reason why LATE appears such performance enhancement compared to DPSAF, especially to BLEST and minRTT, we measure the ratio of RTO retransmission times to the total number of packets, as well as the number of OFO packets at the receiver, the results are shown



(a) Goodput under different loss rate



(b) RTO rate under different loss rate



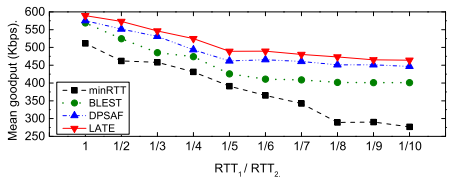
(c) OFO number under different loss rate

Fig. 8. The impact of loss rate over long-lasting flows.

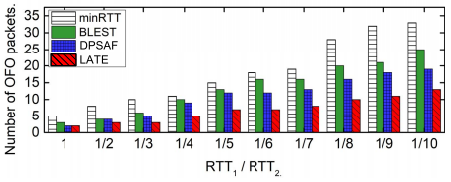
in Fig. 8b and Fig. 8c respectively. According to Fig. 8b, the RTO rate of each algorithm is similar because none of these schedulers considers how to mitigate this problem, and their value goes up quickly when the packet loss rate is larger than 3%. Consequently, the existing schedulers (i.e., minRTT, BLEST, DPSAF) have large estimation errors, leading to more OFO packets, as shown in Fig. 8c. According to Fig. 8c, the number of OFO packets of LATE is the least, because LATE comprehensively considers both loss rate, fast retransmission, RTO retransmission to guarantee in-order arrival.

The RTT difference among subflows is another direct factor that leads to the occurrence of OFO packets. The estimation model of LATE dynamically measures the ratio of fast subflow's RTT to slow subflow's RTT to determine which sequence number of packets should be scheduled over the slow path and the fast path. During the RTT test, the RTT of router1 shown in Fig. 6 is fixed at 10 ms, while the RTT value of router2 ranges from 10 ms to 100 ms. Fig. 9 shows the performance of each algorithm with varying RTT ratio of fast subflow to slow subflow.

As shown in Fig. 9a, with the RTT difference among subflows increases, namely, the ratio of RTT1 to RTT2 increases, and the corresponding goodput of each algorithm decreases.



(a) The impact of RTT ration on goodput



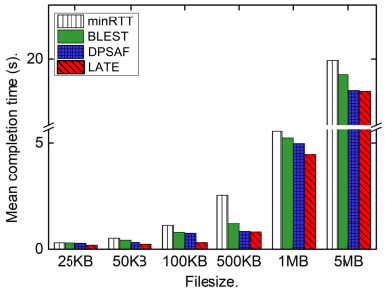
(b) The impact of RTT ration on OFO number

Fig. 9. The impact of RTT over long-lasting flows.

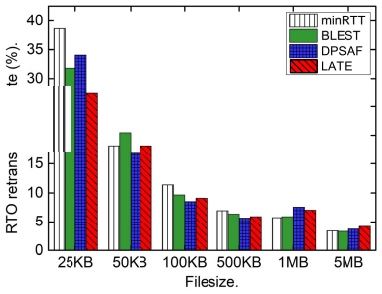
Specifically, minRTT, performs worst, followed by BLEST, while the goodput achieved by LATE is always the largest. According to Fig. 9b, the larger the RTT difference, the more OFO packets the minRTT corresponds to, as minRTT does not consider the RTT ratio when scheduling data. Although the RTT difference is considered in BLEST as that in DPSAF and LATE, the OFO packets of BLEST are more than that of DPSAF and LATE due to the lack of loss rate consideration.

Finally, we track how each algorithm performs when transmitting different sizes of files. It is known that using MPTCP to transmit short flow is prone to RTO, which aggravates the occurrence of OFO packets at the receiving end. To validate that LATE performs better for both short and long flows, we test the different sizes of files ranging from 25KB to 5MB. Fig. 10 shows the comparison of each algorithm under different traffic sizes, where the loss rate is set to 0.5%.

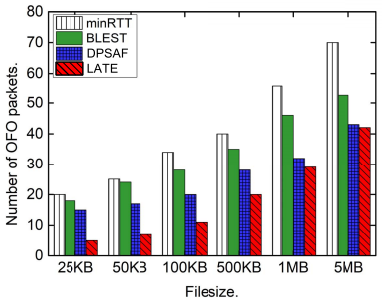
According to Fig. 10a which compares the average completion time of each algorithm with different traffic sizes, LATE possesses the least completion time both for short flows and long flows compared to the other three scheduling algorithms. Specifically, LATE reduces the completion time of short flow (around tens of KB) by about 37.36% against minRTT, and improves goodput of long flow (larger than 1MB) by about 19.84%. In addition, LATE outperforms DPSAF in reducing the completion time of short flows by 26.68% and improving the goodput of long flows by 5.13%. We further take the RTO retransmission rate and OFO number as metrics to compare, where RTO retransmission rate is given by the ratio of the total number of RTO occurrences to the total number of packets at the sender. As shown in Fig. 10b, with the high packet loss rate, the RTO retransmission rate remains high, especially when transmitting short flows, because the lost packets cannot be recovered through fast retransmission when using multiple paths to transmit short flows [22], [23]. Therefore, it is important for an estimation-based algorithm to consider the RTO case, which results in the superior performance of LATE. Fig. 10c shows the number of OFO at the receiver with varying traffic size. If the traffic size is small, the corresponding OFO number of LATE is far less than the other three algorithms. However, when traffic size is larger than 1MB, the performance gap between DPSAF and



(a) The completion time under different filesize



(b) RTO rate under different filesize



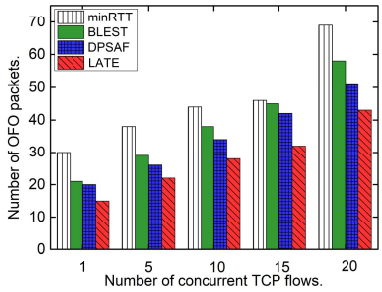
(c) OFO number under different filesize

Fig. 10. The performance of each algorithm under the influence of different flow size.

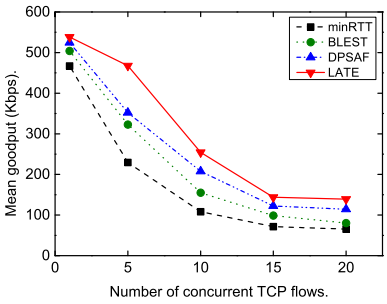
LATE gradually decreases, that is, both DPSAF and LATE perform well in terms of reducing OFO number as the RTO rate decreases.

2) Competing Traffic: In the real network, it is common for TCP background traffic and MPTCP traffic to share the same bottleneck to compete for resources, as described in Fig. 6b. In this case, when there is a large number of competing TCP flows, the cache of the bottleneck router will overflow, leading to packet loss and an increased number of OFO packets. To verify LATE's robustness in this condition, we conduct a series of experiments in this scenario where the bandwidth capacity and path delay are fixed at 10 Mbps and 20 ms respectively. By configuring WANem, the setting of the packet loss rate is 0.5% by default unless specifically stated.

Firstly, we analyze the performance of each algorithm with varying data amount of TCP traffic by changing the number of concurrent TCP flows. Fig. 11 depicts the mean goodput of MPTCP flows and the number of OFO packets at the receiver when 20 concurrent 1MB MPTCP flows compete with varying numbers of 1MB TCP flows through two subflows. As shown in Fig. 11a, with the increasing number of TCP flows, the goodput of each algorithm decreases, because the



(a) The OFO number under different number of coexisted TCP flows



(b) Mean goodput under different number of coexisted TCP flows

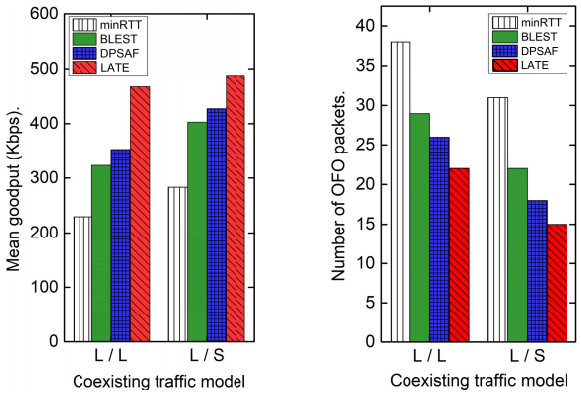
Fig. 11. The performance of each algorithm under the influence of different numbers of concurrent TCP flows.

more TCP traffic that competes with MPTCP flows, the more congested the bottleneck router will be and the higher frequency of packet losses. LATE outperforms the other three algorithms. From Fig. 11b, we observe that OFO number of each algorithm increases quickly especially when the number of concurrent TCP flows is 20, the OFO number of LATE is the least, leading to the highest goodput.

We further conduct experiments to explore the impact of traffic models on each algorithm. There are four competing traffic models, i.e., both MPTCP and TCP servers with long flows (L/L model), MPTCP server with long flows while TCP server with short ones (L/S model), MPTCP server with short flows while TCP server with long ones (S/L model), both MPTCP and TCP servers with short flows (S/S model). In these experiments, the number of MPTCP flows and that of TCP competing flows are set to 20 and 10 respectively, and we select the 1MB files as long flow and 57KB file as short flow according to the real-world Web traffic model in [45].

Fig. 12 describes the performance of each algorithm when MPTCP transmits long flows competing with long TCP flows and short TCP flows respectively. It is clear that the long TCP flows have a greater impact on MPTCP flows as the total goodput of L/L model is smaller than that of L/S model as shown in Fig. 12a. However, LATE performs the best compared with the other algorithms in terms of improved goodput and reduced OFO packets as shown in Fig. 12b.

Similarly, Fig. 13 describes the completion time of each algorithm when transmitting short flows with TCP background traffic. Since the long TCP flows take up most of the bottleneck buffer, the bottleneck router gets congested and short flows



(a) (b)

Fig. 12. The performance of each algorithm in L/L and L/S model: (a) mean goodput, (b) number of OFO packets.

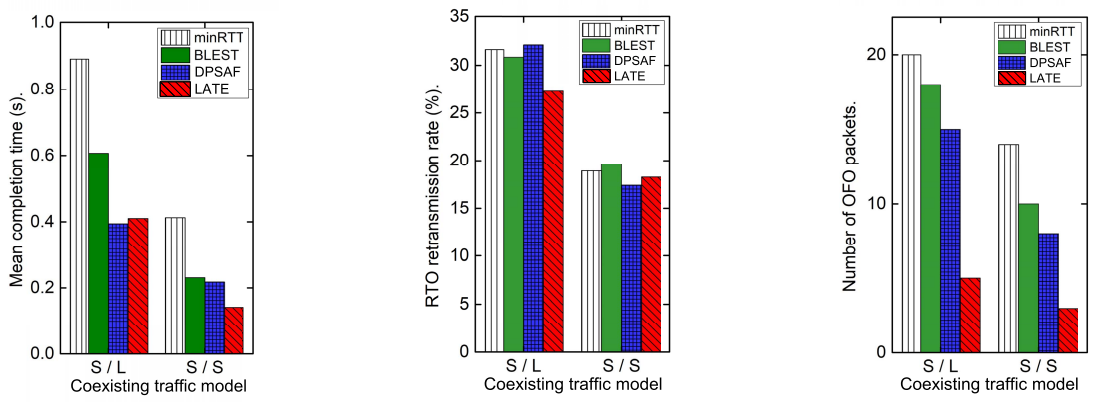
of MPTCP are prone to packet loss consequently, leading to the higher RTO rate of S/L model shown in Fig. 11(b), and consequently increased number of OFO shown in Fig. 11(c). It explains the fact shown in Fig. 11(a) that S/S model consumes less time than S/L. Meanwhile, we observe that LATE has the lowest OFO packets both under S/L and S/S models due to its RTO consideration.

Small buffer size will induce more packet losses due to buffer overflow [46], we finally test how each scheduler works with the changing of the buffer size of the bottleneck router. In our emulations, the BDP, i.e., the product between the bandwidth capacity of the bottleneck link and the average round trip time of each path, is about 18 packets when bandwidth is set to 10 Mbps, RTT is 20 ms and loss rate is 0.5%, and the rule of thumb for the choice of router buffer size was at least BDP according to [47]. Therefore, we set the bottleneck buffer size ranging from 10 packets to 60 packets to evaluate our algorithm. Fig. 14 shows the goodput of each scheduler when 20 concurrent MPTCP flows (1MB per-flow) compete with 10 TCP background flows (1MB per-flow) under different buffer sizes of the bottleneck router.

As shown in Fig. 14, the goodput of LATE is always the largest. In addition, as the buffer size increases, the corresponding goodput of each algorithm gradually increases. But when the buffer size exceeds 40 packets, their goodput does not change much. We further count the number of OFO packets at the receiver under different buffer size, it turns out that when the buffer size is larger than 2 times of BDP (around 36 packets), the number of OFO packets of each algorithm remains stable. Moreover, the results show that when the buffer is small enough, the OFO number of LATE is significantly less than the other algorithms, indicating that the LATE estimation model is effective.

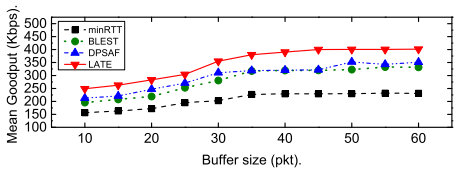
C. The Impact of the Number of Subflows

In this subsection, to validate the performance of LATE subjected to the impact of number of subflows, we compare LATE against minRTT based on the scenario in Fig. 7, where we set M, the number of applications to 20. Specifically, each application on host S_0 transfers binary documents ranging

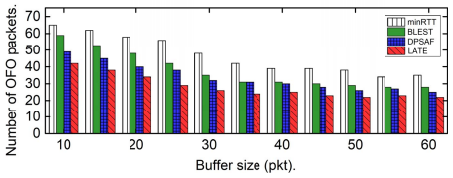


(a) (b) (c)

Fig. 13. The performance of each algorithm in S/L and S/S model: (a) mean completion time, (b) RTO rate, (c) number of OFO packets.



(a) Goodput under different buffer size



(b) OFO number under different buffer size

Fig. 14. The performance of each algorithm under the influence of different sizes of the bottleneck buffer.

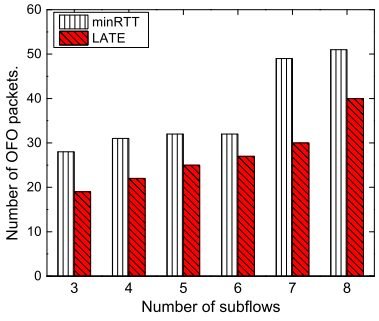


Fig. 15. The performance of each algorithm in L/L and L/S model.

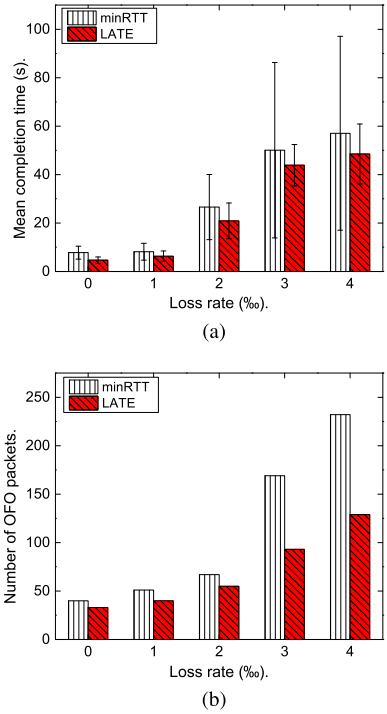


Fig. 16. The performance of each algorithm in S/L and S/S model: (a) mean completion time, (b) number of OFO packets.

from 50KB to 1MB to client C_0. For simplicity, the data rate of all subflows is set to 1Mbps while the delay of each subflow varies from 20 ms to 100 ms. On basis of these configurations, Some brief comparison results are discussed as follows.

First of all, the impact of the number N of subflows on OFO packets is investigated and then we have Fig. 15 where shows the results for $N \in [3, 8]$, the more the number of subflows MPTCP scheduler uses, the more the number of OFO packets at the receiver, because the multipath transfer is characterized by OFO. Meanwhile, compared to minRTT, the results appear an overall reduction of OFO number for LATE, revealing the truth that LATE outperforms minRTT especially when the number of subflows is high.

Then, we fix the number of subflows N at 8 and evaluate network performance with different loss rates and file sizes.

Fig. 16 is associated with a situation where 20 applications generate 100KB traffic and transfer to C_0 concurrently, Fig. 16a records the average completion time and standard deviation under different loss rate, Fig. 16b refers to the corresponding OFO records. Based on the average value of completion time in Fig. 16a, we observe that LATE outperforms minRTT in each case, the reason can be further realized by combining the standard deviation, as well as the OFO results

in Fig. 16b. Since minRTT scheduler allocates data to the subflow with lowest RTT all the time, it's good for the packets that arrive first, whereas others who arrive late will be blocked if the best path gets congested under big traffic load, thus increasing the deviation of arriving time and probability of packet loss and the number of OFO packets. However, LATE differs from minRTT in that LATE considers both RTT and loss rate to make an accurate estimation, spreading packets into multiple paths out-of-order for the in-order arrival.

VI. Conclusion

Packet loss and disorder packets are rather rule than exception with MPTCP. In this paper, we investigate the relationship between packet loss and disorder packet and go further to analyze the state-of-the-art algorithms aiming at mitigating this issue. We found neither performs well in lossy network scenarios. Therefore, we present LATE, a new scheduler that consists of a loss-based transmission model and packet segments scheduling policy. By comparing LATE with minRTT, as well as the alternative BLEST, LATE has better performance in the lossy network as it directly leverages the loss rate as a metric to minimize. Compared with the loss-based prediction algorithm DPSAF, LATE achieves higher prediction accuracy due to the RTO consideration. Theoretical analysis backed up by experimental validation shows that our approach is adaptive and good enough to reduce OFO packets in networks with varied loss rates, and those advantages can be sustained and even highlighted in the scenario where there are more than 2 subflows.

References

[1] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, *TCP Extensions for Multipath Operation With Multiple Addresses*, document RFC 6824, Jan. 2013. [Online]. Available: https://tools.ietf.org/html/rfc6824

[2] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, pp. 951–964, Oct. 2006.

[3] B.-H. Oh and J. Lee, "Feedback-based path failure detection and buffer blocking protection for MPTCP," *IEEE/ACM Trans. Netw.*, vol. 24, no. 6, pp. 3450–3461, Dec. 2016.

[4] Y.-S. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "ECF: An MPTCP path scheduler to manage heterogeneous paths," in *Proc. 13th Int. Conf. Emerg. Netw. Exp. Technol.*, 2017, pp. 147–159.

[5] S. Ferlin, S. Kucera, H. Claussen, and O. Alay, "MPTCP meets FEC: Supporting latency-sensitive applications over heterogeneous networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 5, pp. 2005–2018, Oct. 2018.

[6] J. Hu, J. Huang, W. Lv, Y. Zhou, J. Wang, and T. He, "CAPS: Coding-based adaptive packet spraying to reduce flow completion time in data center," in *Proc. IEEE INFOCOM - Conf. Comput. Commun.*, Apr. 2018, pp. 2294–2302.

[7] K. Gao, C. Xu, J. Qin, S. Yang, L. Zhong, and G.-M. Muntean, "QoS-driven path selection for MPTCP: A scalable SDN-assisted approach," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–6.

[8] W. Wei, K. Xue, J. Han, D. S. L. Wei, and P. Hong, "Shared bottleneck-based congestion control and packet scheduling for multipath TCP," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 653–666, Apr. 2020.

[9] Y. Cui, L. Wang, X. Wang, H. Wang, and Y. Wang, "FMTCP: A fountain code-based multipath transmission control protocol," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 465–478, Apr. 2015.

[10] Q. Liu, F. Ke, Z. Liu, and J. Zeng, "Loss-aware CMT-based multipathing scheme for efficient data delivery to heterogeneous wireless networks," *Int. J. Digit. Multimedia Broadcast.*, vol. 2019, pp. 1–8, Feb. 2019.

[11] E. Dong, M. Xu, X. Fu, and Y. Cao, "A loss aware MPTCP scheduler for highly lossy networks," *Comput. Netw.*, vol. 157, pp. 146–158, Jul. 2019.

[12] S. Ferlin, O. Alay, O. Mehani, and R. Boreli, "BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks," in *Proc. IFIP Netw. Conf. (IFIP Netw.) Workshops*, May 2016, pp. 431–439.

[13] K. W. Choi, Y. S. Cho, Aneta, J. W. Lee, S. M. Cho, and J. Choi, "Optimal load balancing scheduler for MPTCP-based bandwidth aggregation in heterogeneous wireless environments," *Comput. Commun.*, vol. 112, pp. 116–130, Nov. 2017.

[14] K. Xue, J. Han, H. Zhang, K. Chen, and P. Hong, "Migrating unfairness among subflows in MPTCP with network coding for wired–wireless networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 1, pp. 798–809, Jan. 2017.

[15] B. Y. L. Kimura, D. C. S. F. Lima, and A. A. F. Loureiro, "Packet scheduling in multipath TCP: Fundamentals, lessons, and opportunities," *IEEE Syst. J.*, early access, Jan. 27, 2020, doi: 10.1109/JSYST.2020.2965471.

[16] C. Raiciu *et al.*, "How hard can it be? Designing and implementing a deployable multipath TCP," in *Proc. 9th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2012, pp. 399–412.

[17] H. Adhari, T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tüxen, "Evaluation of concurrent multipath transfer over dissimilar paths," in *Proc. IEEE Workshops Int. Conf. Adv. Inf. Netw. Appl.*, Mar. 2011, pp. 708–714.

[18] T. Dreibholz, M. Becke, E. P. Rathgeb, and M. Tuxen, "On the use of concurrent multipath transfer over asymmetric paths," in *Proc. IEEE Global Telecommun. Conf. GLOBECOM*, Dec. 2010, pp. 1–6.

[19] G. Sarwar, R. Boreli, E. Lochin, A. Mifdaoui, and G. Smith, "Mitigating Receiver's buffer blocking by delay aware packet scheduling in multipath data transfer," in *Proc. 27th Int. Conf. Adv. Inf. Netw. Appl. Workshops*, Mar. 2013, pp. 1119–1124.

[20] F. Yang, Q. Wang, and P. D. Amer, "Out-of-order transmission for in-order arrival scheduling for multipath TCP," in *Proc. 28th Int. Conf. Adv. Inf. Netw. Appl. Workshops*, May 2014, pp. 749–752.

[21] K. Xue *et al.*, "DPSAF: Forward prediction based dynamic packet scheduling and adjusting with feedback for multipath TCP in lossy heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1521–1534, Feb. 2018.

[22] P. Dong *et al.*, "Reducing transport latency for short flows with multipath TCP," *J. Netw. Comput. Appl.*, vol. 108, pp. 20–36, Apr. 2018.

[23] M. Kheirkhah, I. Wakeman, and G. Parisis, "MMPTCP: A multipath transport protocol for data centers," in *Proc. IEEE INFOCOM - 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.

[24] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 1998, pp. 303–314.

[25] *NS3 Simulator*. Accessed: 2020. [Online]. Available: https://www.nsnam.org

[26] Q. Peng, A. Walid, and S. H. Low, "Multipath TCP algorithms: Theory and design," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 41, no. 1, pp. 305–316, 2013.

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

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

[29] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "MPTCP is not Pareto-optimal: Performance issues and a possible solution," *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1651–1665, Oct. 2013.

[30] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, design, and implementation," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 596–609, Feb. 2016.

[31] W. Guo *et al.*, "Delay-based congestion control for multipath TCP," in *Proc. Adv. Multimedia, Commun. Netw.*, Dec. 2013, pp. 1–10.

[32] P. Dong, J. Wang, J. Huang, H. Wang, and G. Min, "Performance enhancement of multipath TCP for wireless communications with multiple radio interfaces," *IEEE Trans. Commun.*, vol. 64, no. 8, pp. 3456–3466, Aug. 2016.

[33] B.-H. Oh and J. Lee, "Constraint-based proactive scheduling for MPTCP in wireless networks," *Comput. Netw.*, vol. 91, pp. 548–563, Nov. 2015.

[34] B. Y. L. Kimura, D. C. S. F. Lima, and A. A. F. Loureiro, "Alternative scheduling decisions for multipath TCP," *IEEE Commun. Lett.*, vol. 21, no. 11, pp. 2412–2415, Nov. 2017.

[35] Q. Tan, X. Yang, L. Zhao, X. Zhou, and T. Dreibholz, "A statistic procedure to find formulae for buffer size in MPTCP," in *Proc. IEEE 3rd Adv. Inf. Technol., Electron. Autom. Control Conf. (IAEAC)*, Oct. 2018, pp. 900–907.
[36] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 43–56, Oct. 2000.
[37] L. Cai, X. Shen, J. Pan, and J. W. Mark, "Performance analysis of TCP-friendly AIMD algorithms for multimedia applications," *IEEE Trans. Multimedia*, vol. 7, no. 2, pp. 339–355, Apr. 2005.
[38] L. Cai, X. Shen, J. Mark, and J. Pan, "Performance modeling and analysis of window-controlled multimedia flows in wireless/wired networks," *IEEE Trans. Wireless Commun.*, vol. 6, no. 4, pp. 1356–1365, Apr. 2007.
[39] C. Casetti and M. Meo, "A new approach to model the stationary behavior of TCP connections," in *Proc. IEEE INFOCOM . Conf. Comput. Communications. 19th Annu. Joint Conf. IEEE Comput. Commun. Societies*, Mar. 2000, pp. 367–375.
[40] S. Fu and M. Atiquzzaman, "Performance modeling of SCTP multihoming," in *Proc. GLOBECOM IEEE Global Telecommun. Conf.*, Mar. 2005, p. 6
[41] N. Dukkipati *et al.*, "An argument for increasing TCP's initial congestion window," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 3, pp. 26–33, Jun. 2010.
[42] R. Barik, M. Welzl, S. Ferlin, and O. Alay, "LISA: A linked slow-start algorithm for MPTCP," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2016, pp. 1–7.
[43] *MPTCP NS3 Code*. Accessed: 2020. [Online]. Available: https://code.google.com/archive/p/mptcp-ns3/
[44] S. Barré, C. Paasch, and O. Bonaventure, "Multipath TCP: From theory to practice," in *Proc. Int. Conf. Res. Netw.* Berlin, Germany: Springer, 2011, pp. 444–457.
[45] C. I. N. I. Centre. (2017). *China Statistical Report on Internet Development*. Accessed: Jul. 3, 2017. [Online]. Available: http://cnnic.cn/hlwfzyj/hlwxzbg/hlwtjbg/201701/P020170123364672657408.pdf
[46] S. Pack, X. Shen, J. W. Mark, and L. Cai, "Throughput analysis of TCP-friendly rate control in mobile hotspots," *IEEE Trans. Wireless Commun.*, vol. 7, no. 1, pp. 193–203, Jan. 2008.
[47] J. Sommers, P. Barford, A. Greenberg, and W. Willinger, "An SLA perspective on the router buffer sizing problem," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 4, pp. 40–51, Mar. 2008.

Wenjun Yang (Graduate Student Member, IEEE) received the M.S. degree in information science and engineering from Hunan Normal University, Changsha, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada. His current research interests include next generation of network architecture and related issues, such as multipath TCP, multihoming, and mobility.

Pingping Dong received the B.S., M.S., and Ph.D. degrees from the School of Information Science and Engineering, Central South University, China. She is currently an Associate Professor with the College of Information Science and Engineering, Hunan Normal University, Changsha, China. Her research interests include protocol optimization and protocol design in wide area networks (WANs) and wireless local area networks (WLANs).

Lin Cai (Fellow, IEEE) received the M.A.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2002 and 2005, respectively. Since 2005, she has been with the Department of Electrical and Computer Engineering, University of Victoria. She is currently a Professor. Her research interests include several areas in communications and networking, with a focus on network protocol and architecture design supporting emerging multimedia traffic and the Internet of Things. She is also an NSERC E.W.R. Steacie Memorial Fellow. In 2020, she was elected as a member of the Royal Society of Canada's College of New Scholars, Artists, and Scientists. She was also elected as a 2020 "Star in Computer Networking and Communications" by N2Women. She was a recipient of the NSERC Discovery Accelerator Supplement (DAS) Grants in 2010 and 2015, respectively, and the best paper awards of IEEE ICC 2008 and IEEE WCNC 2011. She awarded Outstanding Achievement in Graduate Studies. She has co-founded and chaired the IEEE Victoria Section Vehicular Technology and Communications Joint Societies Chapter. She has been elected to serve the IEEE Vehicular Technology Society Board of Governors since 2019. She has served as an Area Editor for IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, a member of the Steering Committee for the IEEE TRANSACTIONS ON BIG DATA (TBD) and IEEE TRANSACTIONS ON CLOUD COMPUTING (TCC), an Associate Editor for the IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON COMMUNICATIONS, *EURASIP Journal on Wireless Communications and Networking*, *International Journal of Sensor Networks*, and *Journal of Communications and Networks* (JCN), and as the Distinguished Lecturer for the IEEE VTS Society. She has also served as a TPC Co-Chair for IEEE VTC2020-Fall, and a TPC Symposium Co-Chair for IEEE Globecom'10 and Globecom'13. She is also a Registered Professional Engineer in British Columbia, Canada.

Wensheng Tang received the B.S. degree from Hunan Normal University, Changsha, China, in 1992, and the M.S. and Ph.D. degrees from the National University of Defense Technology, Changsha, in 1997 and 2009, respectively. He is currently a Professor with Hunan Normal University. His research interests include protocol optimization, cloud computing, information security, and quantum cryptography.