DPSAF: Forward Prediction Based Dynamic Packet Scheduling and Adjusting With Feedback for Multipath TCP in Lossy Heterogeneous Networks

Kaiping Xue, *Senior Member, IEEE*, Jiangping Han, Dan Ni, Wenjia Wei, Ying Cai, *Member, IEEE*, Qing Xu, and Peilin Hong

***Abstract*—As multihomed terminals are equipped with multiple interfaces and allowed to access heterogeneous networks, transferring data simultaneously through all the available paths becomes possible and also brings many benefits. Multipath TCP (MPTCP) has been proposed to distribute an application stream over different TCP connections. However, due to the disparate latencies of different paths, the problem of existing out-of-order packets usually occurs at the receiver. Large number of these packets exhaust the limited receiving buffer and make the receive window be stalled, which greatly degrade the throughput. Thus, an efficient scheduling mechanism will play an important role to keep in-order delivery. However, almost all of the previous intelligent scheduling mechanisms ignored packet losses, and didn't consider window changes of the congestion control algorithm and utilize the feedback information, which cannot perform well in the lossy heterogeneous networks. In this paper we propose a new scheduling algorithm: Forward Prediction based Dynamic Packet Scheduling and Adjusting with Feedback (DPSAF). DPSAF first utilizes maximum likelihood estimation in TCP modeling to estimate the data amount sent on other paths simultaneously, which takes packet loss rate and time offset into consideration, then gets feedback information from SACK options and fixes the scheduling value. From the simulation, we can see that our mechanism obviously improves throughput and reduces cache occupancy at receiver in lossy heterogeneous networks.**

***Index Terms*—Forward prediction, Multipath TCP, multiple interfaces, packet scheduling, SACK.**

I. Introduction

As VARIOUS radio access technologies (RATs) overlap and form heterogeneous networks, it is common that mobile terminals equipped with multiple network interfaces, which make data transmission through different interfaces simultaneously become possible. Unfortunately, currently used protocols only utilize one single interface at a time even though multiple interfaces are connected. Theoretically, reasonable data stripping solutions to exploit multiple interfaces can aggregate the available network resources of different RATs [1]–[3] to provide better service, such as higher bandwidth, better connectivity and so on. However, simultaneous transmission over different paths for one application stream leads to packet reordering [4], [5] due to the dissimilar path characteristics, i.e. latency, bandwidth, packet loss rate, etc. It can adversely affect the performance of any real-time applications.

In recent years, there are solutions of data stripping [6] on multi-interface terminals to enhance concurrent transfer across multiple paths for different purposes, such as load sharing, in-order delivery, fairness and so on. These solutions can be implemented at different layers to efficiently schedule packets, such as [7]–[10] in link layer, [11] in network layer, [12], [13] in application layer. Consequently, transport layer solution better suits multi-interface terminals because it is the appropriate layer to not only no-modification based use services provided in lower layers, but also provide transparency for applications. Moreover, TCP can react to the congestion on different paths instantly while compared with any other transport layer protocols, besides having good features of providing reliable delivery, guaranteeing fairness, and so on.

In TCP-based multipath transmission, a connection can be composed of multiple TCP flows and the scheduling is packet-oriented, that is to say, packets of a connection are scheduled individually and sent over different TCP flows. It can exploit multiple paths simultaneously if each TCP flow is relative to a different path. Since the latency of each path differs, there is a high probability that the packets with lower sequence numbers sent over a slower path arrive at the receiver later than the packets with higher sequence numbers sent over a faster

Manuscript received February 10, 2017; revised June 14, 2017 and August 26, 2017; accepted September 12, 2017. Date of publication September 18, 2017; date of current version February 12, 2018. This work was supported in part by the National Key Research and Development Plan of China under Grant 2016YFB0800301, in part by the National Natural Science Foundation of China under Grant 61379129, Grant 61671420, and Grant 61672106, in part by the Fund of Science and Technology on Communication Networks Laboratory under Grant KX162600024, in part by the Youth Innovation Promotion Association CAS under Grant 2016394, and in part by the Fundamental Research Funds for the Central Universities. The review of this paper was coordinated by Dr. Z. Cai. *(Corresponding author: Kaiping Xue.)*

K. Xue, J. Han, D. Ni, W. Wei, and P. Hong are with Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China. K. Xue is also with Science and Technology on Communication Networks Laboratory, Shijiazhuang, Hebei 050081, China (e-mail: kpxue@ustc.edu.cn; jphang@mail.ustc.edu.cn; nidan@mail.ustc.edu.cn; wwj2014@mail.ustc.edu.cn; plhong@ustc.edu.cn).

Y. Cai is with the Department of Computer Science and Technology, Beijing Information Science and Technology University, Beijing 100101, China (e-mail: ycai@bistu.edu.cn).

Q. Xu is with Huawei Shanghai Research Institute, Shanghai 201206, China (e-mail: xuqing7@huawei.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2017.2753398

path. Thus, there exists holes in sequence numbers at receiver. The receiver has to store large number of out-of-order packets, which will exhaust the limited receiving buffer and occupy the receive window. Employing a large buffer at receiver is always a solution, but it wastes the memory. Instead, the problem can be resolved more intelligently at the sender with a little computing by implementing a scheduling mechanism.

Till now, a large number of multipath reliable transmission protocols based on TCP have been studied [14]–[17] and some are further specified by IETF WG, in which, MPTCP is one of the most typical schemes. It adds a MPTCP layer above TCP by a TCP option, which help manage multiple paths between endpoints. Moreover, a scheduling function should be also implemented to keep in-order delivery. An original data stream is then divided into a series of segments and sent over multiple TCP connections, which is named as "TCP subflows" in MPTCP.

Furthermore, although the intelligent scheduling algorithms at sender have been proposed to minimize the chance of out-of-order, they all ignore packet losses [18] and provide less robustness in lossy heterogeneous networks. Wu *et al.* [19]–[22] have considered the factors of packet loss, streaming coding and Energy consumption, but their work only focuses on mobile video transmission. Network Coding [23]–[26] is a good idea to reduce the influence of packet loss, but it could not solve the problem of out-of-order packets. Meanwhile, they also ignore the feedback information carried in the acknowledgments, which have clues about the accuracy of the previous scheduling. Thus, the sender has no prior knowledge to timely correct scheduling of the next round and has to do the scheduling each round independently, where errors will be accumulated in the following rounds.

In this paper we first propose a new scheduling algorithm for MPTCP: Fine-grained Forward Prediction based Dynamic Packet Scheduling (FPDPS). It allocates a number of packets to each under-scheduling TCP subflow by estimating the amount of packets that will be transmitted on all other TCP subflows simultaneously. The estimation is done by utilizing TCP characteristics of each TCP subflow within a MPTCP connection. Meanwhile, it adopts the idea of TCP modeling and takes account of packet loss, which is more adaptive and suitable in lossy heterogeneous networks. However, note that the scheduling result is estimated in a statistical sense, so FPDPS is still a not-entirely-accurate solution, which cannot instantly react to the varying of the real path condition. Therefore, we further propose a Dynamic Adjustment with the Feedback of SACK (DAF) to make an offset to eliminate the previous scheduling deviation for this round of scheduling. The sender gets feedback information over SACK options, which can help the sender distinguish the scheduling deviation in the last round, and then the sender modifies the value in the next round accordingly. By this way, it will further eliminate the accumulated estimation error brought out by the not-entirely-accurate FPDPS.

The main contribution can be summarized as follows:

1) We first put forward a fine-grained forward prediction based dynamic packet scheduling scheme for Multipath TCP. For each subflow, we utilize maximum likelihood estimation in TCP modelling to estimate the data amount to be transmitted on all other subflows other than the under-scheduling path, which takes both packet loss rate and time offset into consideration.
2) Based on the above mechanism, we further propose a enhanced offset compensation scheme, which is based on utilizing the feedback information carried on SACK options. This scheme can further eliminate the accumulated estimation error offset brought out by the not-entirely-accurate scheduling result for the future rounds.

This paper inherits the basic idea of our conference papers [27] and [28]. They differ in the following aspects: 1) In this paper, we utilize maximum likelihood estimation to replace the original algorithm in TCP modelling to conduct the estimation, which can provide a more accurate estimation and reduce the algorithm complexity compared with the algorithm in [27]. 2) We re-design the dynamic adjustment algorithm, which makes a more fine adjustment compared with the algorithm in [28]. 3) In performance analysis part, we realize the updated algorithms and re-design the simulation sceneries, which include random wireless loss scenario and mobile scenario.

The rest of the paper is structured as follows. In Section II, we present a literature review of some schedule algorithms used in multipath transmission, especially the schemes for transport layer. The detail of the our proposed schedule scheme (DPSAF) is given in Section III. Simulation results and performance comparisons are given in Section IV. Section V concludes the paper.

II. Related Work

Many proposals have been proposed to be implemented at different layers to efficiently schedule packets on different paths. Firstly, we talk about some typical scheduling schemes in link layer and network layer. A link layer solution of traffic distribution over multi-radio networks has been studied in [29]. It is based on the MAC-level measurement, dispersing traffic across the links in proportion to their available capacities. The most notable network layer solution is the earliest delivery path first (EDPF) scheduling [11]. It dynamically estimates the delivery time of the next packet on each link, then transmits the packet through the path that delivers it the earliest. However, these lower layer scheduling solutions are transparent to transport layer, which means transport layer doesn't aware of the multiple paths. Regarding to TCP, it cannot distinguish between out-of-order and packet loss, which leads to lots of unnecessary retransmission. Application layer solutions [12], [30] are also proposed, which can be packet-oriented or connection-oriented. [12] adopts an approach that works in case of using HTTP, which issues a set of range queries each using a separate connection on a different interface. MuniSocket [30] describes the design and implementation of an UDP-based socket that utilizes multiple network interfaces connected through heterogeneous networks. However, the applications need modifications to be aware of multiple interfaces.

Consequently, transport layer solution better suits multi-interface terminals because it is the appropriate layer to not only use services provided without any modification in lower layers,

but also provide transparency for applications. Since TCP has good features, there are several multipath transmission protocols based on TCP, such as LS-SCTP (improved on SCTP) and MPTCP. They support to establish a connection with multiple TCP flows through different paths, and scheduling algorithm is essential to schedule data efficiently over multiple TCP flows.

Stream Control Transmission Protocol (SCTP) [31] is a transport layer protocol and supports multi-homing, serving in a similar role to TCP and UDP. Multi-homed terminal with SCTP implemented assigns a different IP address for each interface and uses them in a single "association", which is similar to "connection". SCTP supports the sender to establishes an association with a primary path and reserves alternative paths for retransmission or back-up. If the primary path fails, the alternative paths can be used.

Concurrent Multipath Transfer using SCTP (CMT-SCTP) [32] extends SCTP to support simultaneously using multiple paths within a SCTP association. The sender simply adopts round robin (RR) manner without intelligence, where it just schedules data from the sending buffer in sequence to the available congestion window space of the next path. However, it cannot alleviate the effects brought up by heterogeneous path characteristics, such as packet reordering. Packets with larger sequence number may arrive at receiver earlier than expectation, and have to wait until the sequence numbers are continuous. The number of out-of-order packets aggregated from multiple paths arises due to the dissimilar and timely changed path characteristics (e.g, latency, bandwidth, packet loss rate). A large receiving buffer is required to cache out-of-order packets, which leads to large waiting delay and heavily degrades the throughput.

Just as mentioned above, in-order delivery in a single connection over multiple paths is important. Load Sharing for SCTP (LS-SCTP) [33] supports weighted round robin and distributes data to each path in proportion to the ratio cwnd/RTT (congestion window/round trip time). However, it is coarse-grained and cannot ensure in-order delivery for each packet. WestwoodSCTP [34] performs a more intelligent bandwidth aware scheduling at sender, which is named as BAS. It scores for each path, and the path with the lowest score has the highest priority to transmit packets. It tries to provide in-order delivery but still suffers from serious performance degradation if the paths in an association have significantly different latencies.

Forward Prediction Packet Scheduling (FPS) for multi-interface terminals with disparate latencies is introduced in [35], which is verified in SCTP. When a path under scheduling frees congestion window space to pull new data from the sending buffer, it estimates the duration of this new transmission. Then it estimates the number of packets (N) that can be delivered simultaneously in other paths during this given duration. The estimation is based on the assumption that no loss will occur, thus the congestion window size will increase for every RTT. Then the under-scheduling path chooses the $(N+1)$-th packet and the following ones from the sending buffer to fill its congestion window. If the packets experience no loss, the estimation is precise and FPS can keep windows sliding smoothly at both sides and enhances throughput. However, in lossy networks where losses should be taken into consideration, it becomes fragile.

For SCTP isn't compatible with regular TCP, it is difficult to implement SCTP in the current network. Recently, Multipath TCP (MPTCP) [36] is raised to conquer the problems in SCTP. MPTCP can establish a MPTCP connection with multiple regular TCP subflows, each may be on a different path, and it can just fall back to regular TCP when there is only a TCP subflow. In MPTCP layer, it divides data of a connection into several portions and schedules them on parallel TCP subflows. Since the packets are strictly ordered by sequence number, scheduling algorithm at MPTCP layer is essential and needs more intelligence to improve the chances of in-order delivery at connection level. In current MPTCP specification [36], it simply uses RR manner to schedule data. MPTCP employs a large receiving buffer shared by all the subflows to hold out-of-order packets. Linux-MPTCP scheduler [37] is an intelligent scheduler implemented in Linux MPTCP kernel [38]. The amount of data scheduled on each TCP subflow is in proportion to the estimated bandwidth of the path, calculated by $BW = cwnd/RTT$. Besides, it has the intelligence to choose which packet to allocate from the shared sending buffer. Nonetheless, it doesn't utilize the TCP characteristics of each subflow and becomes fragile in lossy networks.

III. Forward Prediction Based Dynamic Packet Scheduling and Adjusting With Feedback (DPSAF)

Previous forward prediction based packet scheduling algorithms for multi-path transmission only consider the difference of RTT between different paths, without taking packet loss and the dynamic effect of path changing into consideration. In this section, we propose a new intelligent scheduling mechanism for MPTCP in lossy networks, named DPSAF (forward prediction based Dynamic Packet Scheduling and Adjusting with Feedback). DPSAF can be mainly divided into two parts: the basic Forward Prediction based Dynamic Packet Scheduling mechanism (FPDPS) and Dynamic Adjustment with the Feedback of SACK (DAF). On the basis of the existing scheduling schemes, the former one considers both packet loss and time delay, which can be more adapted to wireless packet loss and improve the estimation accuracy. The latter one further takes dynamic variation of the path performance into consideration, and uses the feedback information carried in SACK to make a further correction of the predicted scheduling value for the next round, which better adapts to the dynamic network environment changing.

A. Forward Prediction Based Dynamic Packet Scheduling Mechanism (FPDPS)

Firstly, we propose the basic Forward Prediction based Dynamic Packet Scheduling mechanism (FPDPS), which is implemented at sender to allocate packets over multiple TCP subflows in a connection. FPDPS is close to FPS, but is more fine-grained and more robust in lossy heterogeneous networks. When a subflow is under-scheduling, the sender predicts the size varying of TCP's sending window for each faster subflow in the same connection and estimates the data amount (N) sent on them during one successful delivery time on the under-scheduling subflow. The under-scheduling subflow then selects the $(N+1)$-th

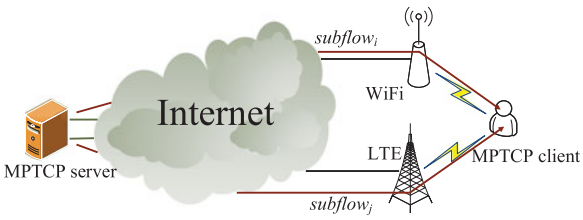


Fig. 1. An example of two-path MPTCP scenario.

packet and the following ones to fill its congestion window. The estimation model is the key issue to be solved. Different from FPS, FPDPS takes packet loss into consideration. FPDPS adopts the idea of TCP modelling, which models TCP's behavior for each faster TCP subflow by considering all possible packet loss events. In this section, we yield an estimation of N, which is a function of RTT, cwnd and packet loss rate.

We adopt most of our terminology from TCP modeling [39], [40], which develops the characteristics of steady-state throughput and latency. We assume that each subflow adopts TCP Reno and we model TCP Reno in terms of "round". A round starts with the transmission of packets in current congestion window. The first ACK reception marks the end of the current round and the beginning of the next round. Without loss of generality, the duration of a round is equal to RTT. Furthermore, We make some other assumptions as follows:

1) For every subflow, the packets arrive at the receiver side in order if they haven't be lost in transmission.
2) The modelling of TCP Reno adopts independent packet loss model, which means the lost rate of each packet is independent of any other packets.
3) The time required to send all packet in a round is smaller than RTT.
4) The state of each link is stable within a short period of time, which means RTT and packet loss rate remain unchanged in principle.

We elaborate the main idea of FPDPS through a two-path MPTCP scenario illustrated in Fig. 1. Both endpoints (e.g., client and server) support MPTCP and a MPTCP connection with two TCP subflows (e.g., $subflow_i$, $subflow_j$) has been established between them. These two subflows are transmitted on different paths, where $path_j$ experiences a larger delay than $path_i$, which means $RTT_j > RTT_i$. The packet loss rates of these two paths are p_i and p_j separately. Besides, it is allowed that more subflows through different paths join the connection. In this paper, we just use the two-path scenario for the simplification of description and analysis.

Fig. 2 shows an example of packet transmission process of the two-path MPTCP scenario. The first packet transmitted on $subflow_j$ starts at time t, and is received by receiver at time t'. During $[t, t']$, sender predicts there could be 5 packets sent on $subflow_i$. Therefore, when $subflow_j$ starts to send data, sender should keep the packets of $\#1-5$ for $subflow_i$, and $subflow_j$ starts at packet $\#6$, thus all these packets will arrive at receiver in order. This is the main idea of packet allocation for MPTCP in the existing schemes. However, the example in Fig. 2 does not consider the situation of packet loss. As shown

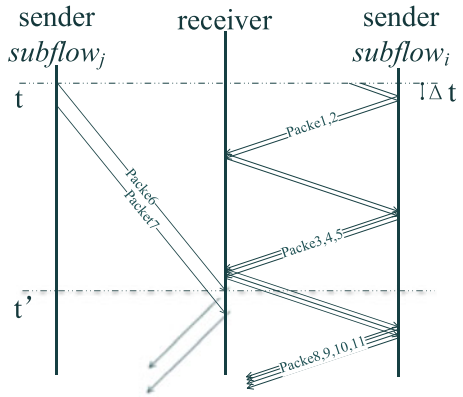


Fig. 2. Packet transmission process of two-path MPTCP scenario.

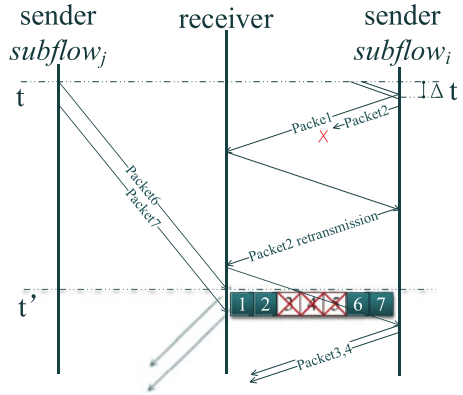


Fig. 3. Packet transmission process of two-path MPTCP scenario in lossy situation.

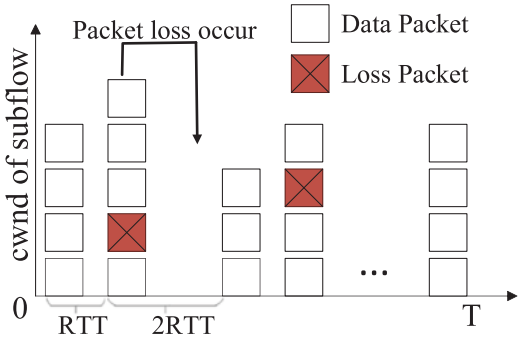


Fig. 4. Modeling TCP's behavior on $subflow_i$ during transmission time T.

in Fig. 3, if packet loss occurs during transmission, sender will make a wrong prediction and packets will not arrive at receiver in order, which could still lead to a reduction of transmission rate. In the actual situation for MPTCP, we need to model the TCP behaviour considering packet loss.

Therefore, in Fig. 4, we model $subflow_i$'s behaviour during T_j (the time between $[t, t']$) in details. Finally, we can get the estimated value of N on $subflow_i$. We use the maximum probability to conduct the estimation, referring to the maximum likelihood estimation, which can greatly improve the accuracy of estimation. We use some parameters in the estimation: RTT, p, $cwnd$ and ss. RTT_i, RTT_j, p_i and p_j are the round-trip time

and packet loss rate of $subflow_i$ and $subflow_j$ respectively, which are the estimated values in the sender and the accuracy of their estimated values also affect the accuracy of the final estimated N. The parameters $cwnd_i$, $cwnd_j$, ss_i and ss_j are the congestion window and the slow start threshold of $subflow_i$ and $subflow_j$ respectively, which can be directly obtained in the sender.

In the process of modelling TCP behaviour, there are two important steps: One is determining the modelling time, and the other is modelling TCP behaviour in different packet loss situations. For the first step, as shown in Fig. 2, considering that different subflows usually do not start sending packets at the same time, and there should be a rectification of modelling time. For the next step, FPDPS uses maximum likelihood estimation to estimate the scheduling value for each TCP subflow. Therefore, the detailed modelling processes can be further divided into the following two steps: 1) Correction of modelling time T_j^i; 2) Estimation of N.

1) Determination and Correction of Modelling Time T_j^i: In Fig. 2, T_j is defined as the elapsed time for the packet from sender to receiver in $path_j$. Packet loss rate of wired/wireless link is usually less than 5%, then without loss of generality, T_j can be directly set as $RTT_j/2$. However, in the modeling, we should consider that the two subflows usually do not start sending packets at the same time, as show in Fig. 2. There will be a staggered interval between two subflows. When $subflow_j$ started scheduling, the last round sent on $subflow_i$ is not over, and the next round has not started yet. At this time we do not model that $subflow_i$ starts the next round at the same time with $subflow_j$, otherwise we will allocate too more packets for $subflow_i$. Therefore, another variable Δt_j^i is introduced, which is the estimation of time offset that $subflow_i$ will start the next round compared with $subflow_j$'s starting time. We use Formula. (1) to compute Δt_j^i.

$$\Delta t_j^i = \begin{cases} RTT_i - t_i & , \quad t_i < RTT_i, \\ 0 & , \quad t_i \geq RTT_i, \end{cases} \tag{1}$$

where t_i is the packet's estimated transmission time on $subflow_i$, which can be obtained from the timeout retransmission timer and the duration of this timer is referred to as RTO (retransmission timeout), which dynamically adjusted as the same way in TCP. In this way, we can get the modelling time for $subflow_i$: T_j^i $(= T_j - \Delta t_j^i)$. Furthermore, we need to estimate the number of packets that can be transmitted on $subflow_i$ during the modelling time T_j^i.

2) Estimation of N: We use the maximum likelihood estimation to estimate the scheduling vale N, which can be obtained by Formula. (2):

$$\hat{N} = \max_N P(N|\mathbf{X}), \tag{2}$$

where $\mathbf{X}$ is the set of the parameters referred in the previous, i.e., T_j^i, RTT_i, p_i, $cwnd_i$ and ss_i. The parameters $cwnd_i$, ss_i are the congestion window and the slow start threshold of $subflow_i$ respectively, which can be directly acquired at sender. The parameter RTT_i is the Round-Trip Time of $subflow_i$, which can be measured by the sender. The parameter p_i is the random packet loss rate of $subflow_i$, which can be measured by sender. In order to simplify the description, let $\mathbf{X} = \{T, RTT, p, cwnd, ss\}$ in the following.

Because packet loss state of each packet is independent, the packet loss situation of each congestion window is independent too, then we can get:

$$P(N|\mathbf{X}) = \prod P(n^{(m)}|\mathbf{X}^{(m)}), \tag{3}$$

where $n^{(m)}$ is the number of packets sent on the m-th round. The parameter $\mathbf{X}^{(m)}$ represents the estimated parameters of the m-th round, i.e., $T^{(m)}$, $RTT^{(m)}$, $p^{(m)}$, $cwnd^{(m)}$ and $ss^{(m)}$. Assuming that the link is stable within a short period of time, we have $RTT^{(m)} = RTT$, $p^{(m)} = p$. The parameter $\mathbf{X}^{(m)}$ depends on the status of the previous $(m-1)$-th round. When $m = 1$, we have $\mathbf{X}^{(1)} = \mathbf{X}$. Since $cwnd^{(m)}$ depends on $cwnd^{(m-1)}$ and the packet lost situation in the previous round, Formula. (2) can be shown as bellow:

$$\begin{aligned} \hat{N} &= \max_{N^{(1)}} \left(P(n^{(1)}|\mathbf{X}^{(1)}) \max_{N^{(2)}} \left(P(n^{(2)}|\mathbf{X}^{(2)}) \ldots \right) \right), \\ N^{(m)} &= n^{(m)} + \hat{N}^{(m+1)}. \end{aligned} \tag{4}$$

Because the maximum probability value is taken and some small probabilities will not affect the final result, they may not be taken into account. Therefore, we ignore the timeout retransmissions and packet loss after retransmissions.

The estimation procedure in FPDPS is a recursive procedure, which starts from the first round and ends until $T^{(m)} < \frac{1}{2} \cdot RTT$. For the m-th round, we consider the following situations:

1) If $T^{(m)} < \frac{1}{2} \cdot RTT$, time is not enough to transmit new packets, and the recursive process ends. The estimation of packages is $\hat{N}^{(m)} = 0$, and the probability of $\hat{N}^{(m)}$ is 1.
2) When $\frac{1}{2} \cdot RTT \leqslant T^{(m)} < \frac{3}{2} \cdot RTT$, the time is enough to transmit new packets, but not enough to retransmit the loss packets or implement the transmission for one more round, so the recursive process ends. For this situation, the transmission number of packets is $N^{(m)} = cwnd^{(m)} - x$, where x is the number of lost packets in this round. The probability of $N^{(m)}(x)$ is:

$$P^{(m)}(x) = p^{(m)}(x) = C^x_{\text{cwnd}^{(m)}} \cdot p^x (1-p)^{\text{cwnd}^{(m)} - x}. \tag{5}$$

3) When $T^{(m)} \geqslant \frac{3}{2} \cdot RTT$, time is enough to transmit new packets and retransmit the loss packets.

If packet loss does not occur, the probability of no packet loss in $cwnd^{(m)}$ is $p^{(m)}(0) = (1-p)^{\text{cwnd}^{(m)}}$, the transfer time is $RTT/2$, the time to start the next round of transmission is RTT. Therefore, the parameters of the next round are $\mathbf{X}_0^{(m+1)}$:

$$\begin{aligned} T^{(m+1)} &= T^{(m)} - RTT, \\ ss^{(m+1)} &= ss^{(m)}, \\ cwnd^{(m+1)} &= \begin{cases} 2 \cdot cwnd^{(m)}, & cwnd^{(m)} < ss^{(m)}, \\ cwnd^{(m)} + 1, & cwnd^{(m)} \geqslant ss^{(m)}. \end{cases} \end{aligned} \tag{6}$$

Then we can get $N^{(m)}(0) = \hat{N}^{(m+1)} + cwnd^{(m)}$, and the total probability of $N^{(m)}(0)$ is:

$$P^{(m)}(0) = p^{(m)}(0) \cdot \max\left\{P^{(m+1)}(x)\right\}, \tag{7}$$

where x $(\in 1, 2, 3, \ldots)$ is the number of lost packets in the $(m+1)$-th round.

If packet loss occurs, and there are x $(x \geqslant 1)$ packets lost. The transmission time is $3 \cdot RTT/2$. The time to start next round of transmission is: $2 \cdot RTT$. The parameters of the next round are $\mathbf{X}_x^{(m+1)}$:

$$\begin{aligned} T^{(m+1)} &= T^{(m)} - 2 \cdot RTT, \\ ss^{(m+1)} &= cwnd^{(m+1)} = \max(cwnd^{(m)}/2^x, 2). \end{aligned} \tag{8}$$

Then the probability of losing x packets in $cwnd^{(m)}$ is:

$$p^{(m)}(x) = C^x_{\text{cwnd}^{(m)}} p^x (1-p)^{\text{cwnd}^{(m)} - x}. \tag{9}$$

In the same way, we can get $N^{(m)}(x) = \hat{N}^{(m+1)} + cwnd^{(m)}$ The total probability of $N^{(m)}(x)$ is $P^{(m)}(x) = p^{(m)}(x) \cdot \max\left\{P^{(m+1)}(y)\right\}$, where x and y are respectively the numbers of lost packets in the m-th and $(m+1)$-th rounds.

The estimation algorithm of FPDPS is showed in Algorithm 1, which is a recursive procedure.

To be noted, if the number of subflows is more than two, our scheme is also suitable. For each subflow (e.g., $subflow_j$), sender needs to model all the subflows whose RTT is less than its RTT and puts the values together as $N_j = \sum_{i \in S} N_j^i$. Here, as the above definition, S is the set of subflows whose RTT is less than RTT_j. In the process of scheduling data, data in the sending buffer can be seen as a data flow. When a subflow is able to send packets, the subflow with the smallest RTT fetches data at the starting position of the data flow, and each of the other subflow fetches data at the position of $N_j * MSS$. After a subflow fetching data, the rest of the data can be treated as a new complete data flow and next subflows will further fetch data from it.

FPDPS provides the estimation of scheduling value by the most likely situation of packet loss events during $[t, t']$, but the real successfully scheduled value is always bigger or smaller. If sender could get the feedback information of the true value and adjust the estimation value in the next round, the performance will be better.

Algorithm 1: Estimation function of FP-DPS.

Require: parameters $\mathbf{X} = (T_j^i, RTT_i, p_i, cwnd_i, ss_i)$
Ensure: estimation of scheduling value $\hat{N}_j^i$, and the probability of $\hat{N}_j^i$
function GETNP $(T, RTT, p, cwnd, ss)$
  if $T < RTT/2$ **then**
    return(0,1);
  else if $RTT/2 <= T < 3 * RTT/2$ **then**
    for $x = 0 \rightarrow cwnd$ **do**
      $P(x) = p(x) = C^x_{\text{cwnd}} p^x (1-p)^{\text{cwnd}-x}$
      $N(x) = cwnd - x$
    end for
    calculate $\max(P(x))$, the order number is l.
    return(N(l), P(l));
  else
    for $x = 0 \rightarrow cwnd$ **do**
      $P(x) = p(x) \cdot$ GETNP($\boldsymbol{X}_x$).P
      $N(x) = cwnd +$ GETNP($\boldsymbol{X}_x$).N
    end for
    calculate $\max P(x)$, the order number is l.
    return (N(l), P(l));
  end if
  end function

B. Dynamic Adjustment With the Feedback of SACK (DAF)

In prediction based packet scheduling algorithm for MPTCP, the scheduling value N is the key parameter. However, note that the prediction mechanisms always inevitably bring about the corresponding deviation, and the estimation of N in each round isn't always precise, which may be larger or smaller than the actually scheduling value due to the varying of the path condition. We assume that the link state is stable within a short period in the previous modeling process. However, when the path changes quickly, the prediction error will relatively increase, so as to make the performance become poor, especially in mobile networks. However, if the sender can make full use of the feedback information from the receiver and make appropriate adjustments, it will be as far as possible to minimize the effect of the deviation in future scheduling and transmission.

The accuracy of the predicted value depends on the prediction algorithm (e.g., FPS, basic FPDPS) and the accuracy of estimated parameters used in the prediction algorithm. RTT, and the loss rate are always changing over time, and they cannot be estimated so accurately at sender. Moreover, there are deviations between the predicted values and the actual current values. Therefore, if the path changing rapidly, the measurement of RTT and loss rate will not keep up with the pace of the path condition changing, which leads to the deviation. In addition, the congestion also easily produces estimation deviation.

DAF further utilizes the feedback information of SACK to get the actual scheduling value and further adjusts estimation of the future rounds. Therefore, it can have a better performance in the dynamic environment. In our enhanced scheme over FPDPS, scheduling at MPTCP layer collects the path conditions while the feedback module collects the information of previous predictions.

1) Arithmetic Statement: Fig. 5 gives the basic process of DAF algorithm. Sender keeps an offset to adjust the estimation of scheduling value which is computed by FPDPS or other schedule algorithms (e.g., FPS). Then sender sends packets to receiver and receiver sends feedback information to sender in SACK. After that, sender could get the real transmission information and adjust the offset for the next round scheduling.

2) Definition of Variables in Analysis: In order to describe DAF more clearly, we introduce two concepts, master-subflow and slave-subflow. For each subflow, any other subflow with

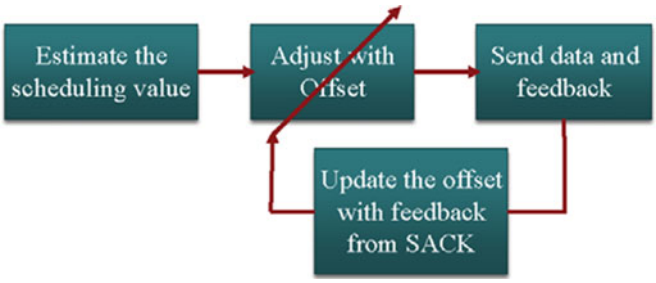


Fig. 5. DAF algorithm statement.

TABLE I
DEFINITION OF VARIABLES IN ANALYSIS

variables	meaning
$N_j(n)$	the real scheduling value of $subflow_j$ during the n-th round.
$\hat{N}_j(n)$	the estimation of scheduling value of $subflow_j$ during the n-th round.
$\hat{N}'_j(n)$	the adjusted estimation of scheduling value $subflow_j$ by offset Δ_j during the n-th round.
$\Delta_j(n)$	the offset for $subflow_j$ to modify $\hat{N}_j(n)$.
$e_j(n)$	the estimation deviation of the scheduling value of $subflow_j$.

longer RTT is its master-subflow, and other subflow with shorter RTT is its slave-subflow.

For each master-subflow $subflow_j$, we introduce 5 parameters: $N_j(n)$, $\hat{N}_j(n)$, $\hat{N}'_j(n)$, $\Delta_j(n)$, $e_j(n)$. The definition of variables in analysis is shown in Table I, where n is on behalf of the number of round of master-subflow. For each master-subflow (e.g., $subflow_j$), $\hat{N}_j(n)$ can be computed by estimating the number of the packets that can be sent simultaneously on all its slave-subflows.

3) Two Specific Situations: Based on FPDPS, the sender can compute scheduling value N and schedule the corresponding packets for each subflow. However, for any subflow, the ideal situation that the predicted scheduling value is equal to the real transmitted packets will hardly happen. If the predicted scheduling value is too large or too small, it will also cause out-of-order packets in receiving buffer and degrade the transmission performance. Based on the feedback information from SACK, sender can know the actual number of the transmitted packets in the last round and infer the elimination deviation.

MPTCP uses dual sequence numbers, connection level sequence number (DSN) and subflow level sequence number (SSN). Take the scenario in Fig. 2 for example again, there are two subflows in a MPTCP connection, $subflow_i$ and $subflow_j$. We assume that the DSN starts at 0, SSN on $subflow_i$ and $subflow_j$ start from 10831 and 566 respectively, The size of each TCP packet is equal to 1400 bytes. Next, we will illustrate the situations of "too large" and "too small".

1) *Situation1:* The estimation of N is too large, which means the predicted scheduling value is larger than the number of actually transmitted packets, and too many packets are scheduled for the subflow $subflow_i$ during the scheduling time of $subflow_j$. The transmitted packets of $subflow_j$ have to wait at the receiver.

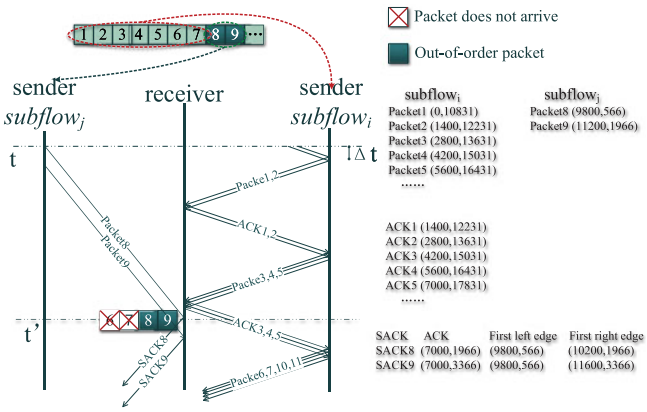


Fig. 6. Situation1: The estimation of N is too large.

Just take an example as shown in Fig. 6, assuming that N_j is estimated to be 7 at the sender, which is larger than the actual value due to the existence of the prediction deviation. Based on the predicted scheduling value, $subflow_j$ schedules 7 packets for $subflow_i$ at the scheduling time, and selects the following 8-th, 9-th packet for $subflow_j$, whose sequence numbers are (9800,566) and (10200,1966) in the form (DSN, SSN). The subflow $subflow_i$ takes the $1-7$-th packets out from the shared sending buffer and sends them successively in several rounds. After the first five packets arriving at the receiver, MPTCP can deliver these five packets to the application layer in turn and returns the ACK (ACK1-5). Later the two packets (packet 8, 9) sent on $subflow_j$ arrive at the receiver, which makes the DSN in the MPTCP connection discontinuity. Therefore, the receiving buffer has to store these two out-of-order packets, and triggers the TCP SACK.

According to ACK in the TCP subflow level, we can know that there is no packet loss in $subflow_j$ and the "missing" packets are sent on $subflow_i$, from which we can determine N_j is too large. According to the DSN value in ACK (7000) and the first left edge (9800) in SACK8 on $subflow_j$, we can know there is a hole of 2800 $(= 9800 - 7000)$ Bytes in the receiving buffer which is sent on $subflow_i$, means the scheduling value is 2 packets larger.

2) *Situation2:* The estimation of N_j is too small, which means the predicted scheduling value is smaller than the number of actually transmitted packets, and too few packets are scheduled for the subflow $subflow_i$ during the scheduling time of $subflow_j$. The transmitted packets of $subflow_i$ have to wait at the receiver.

Just take an example as shown in Fig. 7, assuming that the sender estimates $N = 3$. The subflow $subflow_j$ sets aside 3 packets for $subflow_i$ at the scheduling time of $subflow_j$, and selects the the following 4-th, 5-th packets to send over $subflow_j$. However, the actual value of the number of packets that can be transmitted over $subflow_i$ is 5.

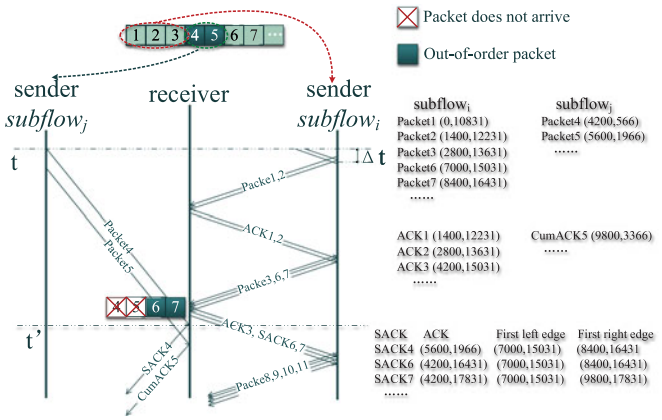


Fig. 7. Situation2: The estimation of N is too small.

The subflow $subflow_i$ takes the 1-st and the 2-nd packets out from the shared sending buffer and sends them in the first round. Then in the second round, besides the 3-rd packets, the sender also schedules the following unscheduling 6-th and 7-th packets to fill $subflow_i$'s congestion windows and sends them over $subflow_i$. After the these five packets arriving at the receiver, MPTCP can deliver the first 3 packets to the application layer in turn and returns the Cumulative Acknowledgement (CumACK), but the following arriving 2 packets (the 6-th and 7-th packets) make the DSN in the MPTCP connection discontinuity. Therefore, the receiving buffer has to store these two out-of-order packets, and triggers the TCP SACK. For SACK6, the packets between its DSN value in ACK (4200) and the first left edge (7000) are sent on $subflow_j$, so sender could know the scheduling value is too small. As there are two SACKs like this, the scheduling value is 2 packets smaller than the actual value.

4) The Dynamic Adjustment Algorithm: Every time to start the n-th round of scheduling on $subflow_j$, the sender can compute the estimation of scheduling value $\hat{N}_j(n)$ by using the basic scheduling mechanism (FPDPS), and adjusts it with the offset Δ_j as shown in Formula. (10).

$$\hat{N}'_j(n) = \hat{N}_j(n) + \Delta_j(n), \tag{10}$$

where $\hat{N}'_j(n)$ is the adjusted prediction value for the n-th round. $\Delta_j(n)$ is the dynamic adjustment value of $subflow_j$ in the n-th round, and updates at the beginning of each round by:

$$\Delta_j(n) = \Delta_j(n-1) + \alpha \cdot e_j(n-1), \tag{11}$$

where $e_j(n-1)$ is deviation of scheduling value of the $(n-1)$-th round, which can be obtained according to the feedback information in SACK from receiver. α is the parameter of update processing, we set $\alpha = 1/8$. SACK can indicate the out-of-order packets in the receiving buffer, and the sender can use these information to amend the scheduling value.

SACK is an indication about whether the prediction value deviates from the true value. Then, the sender determines whether it is in *Situation1* or *Situation2*, and computes the actual enabled scheduling value. After that, sender will compute the estimation

Algorithm 2: DAF Algorithm Description(Part I).

Require: Sender starts the n-th round scheduling of $subflow_j$
update the offset $\Delta_j(n) = \Delta_j(n-1) + \alpha \cdot e_j\ (n-1)$
calculate the scheduling value and modify
with offset $\hat{N}'_j(n) = \hat{N}_j(n) + \Delta_j(n)$

offset of the last round and amend the scheduling value of the current round. In addition, if the sender does not get SACK in the $(n-1)$-th round of $subflow_j$, which indicates that the previous predicted $\hat{N}'_j(n-1)$ is accurate. Therefore, we have $\hat{N}'_j(n-1) = N_j(n-1)$ and $e_j(n-1) = 0$, and the offset of $subflow_j$ shall not be updated in the n-th round.

In *Situation1*, the sender receives SACK from $subflow_j$ and finds out there is no loss indication within $subflow_j$. Hence, it infers that the out-of-order packets are among subflows, which is caused by wrongly estimating the scheduling value in the last round. After comparing the DSNs in the hole with the DSNs in the sending buffer, the sender finds out the packets in the hole were scheduled on $subflow_i$. Since $subflow_j$ is the master-subflow of $subflow_i$, this SACK means the estimation $\hat{N}'_j$ in the last round is too large. The sender can use SACK of the first scheduled packet on $subflow_j$ to compute the actual enabled scheduling value. The sender can get the number of packets of $subflow_j$'s slave-subflow $subflow_i$ in the hole ($Nh_{j,i}(n)$), so the actual enabled scheduling value is $N_j(n) = \hat{N}'_j(n) - Nh_{j,i}(n)$, and the deviation of $\hat{N}'_j(n)$ is:

$$e_j(n) = e_j(n) - Nh_{j,i}(n). \tag{12}$$

In *Situation2*, the sender receives SACK from $subflow_i$ and finds out there is no loss indication within $subflow_i$. Thus, it indicates the out-of-order packets are between subflows. Then the sender can find the packets in the hole were scheduled to $subflow_j$. Since $subflow_j$ is the master-subflow of $subflow_i$, this SACK means the estimated $\hat{N}'_j$ in the last round is too small. Whenever the sender receives a SACK from $subflow_i$, means the actual scheduling value $N_j(n)$ need to increase one, until the first scheduled packet on $subflow_j$ is confirmed according to a DSN. Let $sack_{j,i}(n)$ be the number of SACK received on $subflow_i$ before the first scheduled packet on $subflow_j$ is confirmed, we fend $N_j(n) = \hat{N}'_j(n) + sack_{j,i}(n)$, and the deviation of $\hat{N}'_j(n)$ is:

$$e_j(n) = e_j(n) + sack_{j,i}(n). \tag{13}$$

DAF algorithm description is showed in Algorithms 2 and 3. If the number of subflows is more than 2, the algorithm is also applicable. When the SACK is returned from $subflow_k$ and it can't handle the out-of-order problem by simply retransmitting, which means the packets in the holes are sent on other subflows, it operates in two folds:

1) *Step 1:* The sender finds whether the packets in some holes were scheduled to $subflow_k$'s slave-subflows. If so it means at $subflow_k$'s scheduling time, the sender is supposed to distribute too many packets to its slave-

Algorithm 3: DAF Algorithm Description(Part II).

Require: The sender receives $subflow_k$'s SACK and there is no unordered data of $subflow_k$ itself.
if this SACK is triggered by the first packet sent by $subflow_k$ in this round **then**
update the deviation $e_k(n) \leftarrow e_k(n) - \sum_{l \in S} Nh_{k,l}(n)$, where S is the set of $subflow_k$'s slave-subflows, $Nh_{k,l}$ is unACKed data of $subflow_l$ between ACK and first left edge in $subflow_k$'s SACK.
end if

for each m, which $subflow_m$ is $subflow_k$'s master-subflows **do**
if some holes belong to $subflow_k$'s master-subflows $subflow_m$ and the first packet sent by $subflow_k$ in this round has not been confirmed **then**
increase the deviation $e_m(n) \leftarrow e_m(n) + 1$;
end if
end for

subflows, estimated $\hat{N}'_k(n)$ of $subflow_k$ is larger than the true value. And the SACK of $\hat{N}'_k(n)$-th packet will show how many packets of $subflow_k$'s slave-subflows are in the holes. For each $subflow_k$'s slave-subflow $subflow_l$, the number of packets in the holes is $Nh_{k,l}(n)$. And the deviation of $\hat{N}'_k(n)$ is updated by: $e_k(n) \leftarrow e_k(n) - \sum_{l \in S} Nh_{k,l}(n)$, where S is the set of $subflow_k$'s slave-subflows.

2) *Step 2:* The sender checks whether there are packets in the holes were scheduled to $subflow_k$'s master-subflows. If so for $subflow_k$'s each master-subflow (e.g, $subflow_m$), the estimated $\hat{N}'_m(n)$ is smaller than the true value. Before the ACK of the first scheduled packet on $subflow_m$ is received, each SACK means the deviation of $\hat{N}'_m(n)$ is increasing one: $e_m(n) \leftarrow e_m(n) + 1$.

The initial value of $e_k(n)$ is 0. If there is no SACK during the n-th round, which means the estimation of $N_k(n)$ is exactly accurate, the deviation of $\hat{N}'_k(n)$ is zero.

IV. Performance Evaluation

In this section, we evaluate our scheduling mechanism proposed in this paper on NS3 simulator [41]. The MPTCP NS3 code is provided by google MPTCP group [42]. Another two scheduling mechanisms, RR and FPS are implemented as comparisons. The main difference between FPS and DPSAF is the algorithm to estimate scheduling value N. Compared with FPS, DPSAF is more adaptive and suitable in wireless network and mobile network, in which packet loss and feedback information are considerable.

We set up two scenarios in this section. The link state is relatively stable with random wireless packet loss in the first scenario, which is used to verify DPSAF's performance advantages in the packet loss situation. Furthermore, the link state changes more dynamically in the second scenario, which is utilized to verify the effect of mobile context.

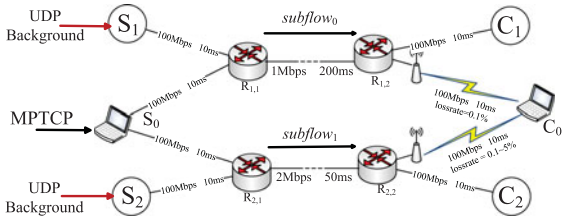


Fig. 8. Simulation setup of random wireless loss scenario.

A. Random Wireless Loss Scenario

1) *Simulation Setup:* The first scenario is showed in Fig. 8, two subflows are established between MPTCP client (C_0) and server (S_0), $subflow_0$ and $subflow_1$, and the two subflows are through different paths. UDP background flows produce UDP data flow between UDP client (C_1, C_2) and server (S_1, S_2), and compete with MPTCP flow at bottleneck. $R_{i,j}$ is the router on each path, $i = 1$ means it is the router on $subflow_0$ while $i = 2$ means it is the router on $subflow_1$. There are two routers on each path, and the link between the $R_{i,1}$ and $R_{i,2}$ is the bottleneck. The MPTCP client has two interfaces to separately access the two wireless access points, i.e., the last hop reaching MPTCP client of each subflow is wireless link, which is prone to wireless link packet loss. Random packet loss happens on the last hop of each subflow. The loss rate of $subflow_0$ is 0.1% and the loss rate of $subflow_1$ is between 0.1% and 5%. The subflow $subflow_0$'s bottleneck has the link bandwidth of 1 Mbps and the latency of 200 ms and the subflow $subflow_1$'s bottleneck has the link bandwidth of 2 Mbps and the latency of 50 ms. The total delay of two paths are 220 ms and 70 ms respectively.
The greatest queue length of bottleneck router is set to 100 packets, and abandon tail packet loss model has been used. Maximal Segment Size (MSS) in our simulation is set to 1400 bytes, which is also the packet size at TCP layer. In our simulation, the size of UDP packets is set to 1024 bytes. The UDP data streams are produced by uniformly distributed generators, and the interval between continuous packets is set to 0.01 s. Each UDP server starts transmission at 0 s, and produces 1000 packets in total.
The first part of DPSAF, i.e., FPDPS, shows the idea of taking packet loss into consideration, so it is more applicable to the packet loss scenario.

2) *Simulation results:* In order to make the result more clear, we display the simulation result focused on total throughput, aggregation benefit and number of out-of-order packets in receiving buffer.
We record the information of transfer time and out-of-ordered packets when MPTCP sender sends 10 MB data to the MPTCP receiver. Fig. 9 shows the total throughput of RR, FPS and FPDPS, aggregated throughput of two single

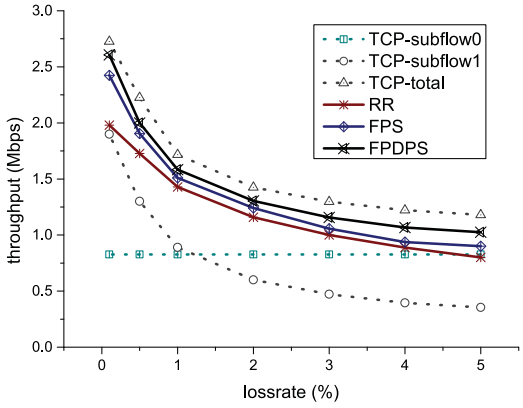


Fig. 9. the contrast of global throughput with the packet loss rate change.

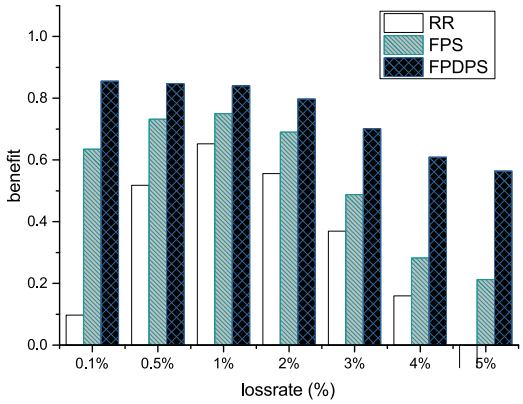


Fig. 10. the contrast of aggregation benefit with the packet loss rate change.

TCP is also showed. The abscissa gives the packet loss rate of $subflow_1$, while the packet loss rate of $subflow_0$ is 0.1%. With the increase of packet loss rate, the single TCP throughput is declined. Because the greater the packet loss rate, the worse the path quality will be. At the same time, the throughput of three scheduling policies are dropping. We use formula. (14) to calculate the aggregation benefit [43] of different scheduling policies. Here g is the global throughput of MPTCP, C_i is the throughput of the single TCP on $subflow_i$, $C_{\max}$ is the max throughput of single TCPs. Assume there are n subflows in a MPTCP connection.

$$Ben(S) = \begin{cases} \dfrac{g - C_{\max}}{\sum_{i=0}^{n-1} C_i - C_{\max}} & if\ g > C_{\max}, \\ \dfrac{g - C_{\max}}{C_{\max}} & if\ g < C_{\max}. \end{cases} \quad (14)$$

As show in Fig. 10, with the increase of packet loss rate, aggregation benefit will all fall. This is because as the packet loss rate increases, the accuracy of any prediction scheduling algorithms will be lower. Although, as show in Figs. 9 and 10, comparison results show the throughput and aggregation benefit of the basic FPDPS are always the biggest.

MPTCP stipulates that packets must be delivered to application layer after the DSN continues. Out-of-order packets need to wait in the receiving buffer. In the simulation, we keep a record of out-of-order packets every 100 ms. The average is get in Fig. 11. The abscissa is $subflow_1$'s packet loss rate. From Fig. 11 we know that out-of-order packets of basic FP-DAF are less than FPS. This means the time of receiving buffer block is less. On the one hand, reducing buffer block means that receiver could choose a smaller buffer under the same condition, which can reduce the resource usage. On the other hand, reducing buffer block can reduce the packet reordering time and data blocking time, which can improve the transmission rate. Nevertheless, with the increase of packet loss rate in $subflow_1$, out-of-order packets number of two algorithms tend to rise. Because as the packet loss rate increases the accuracy of the prediction algorithm will be discounted. But the advantage of FPDPS is more obvious.

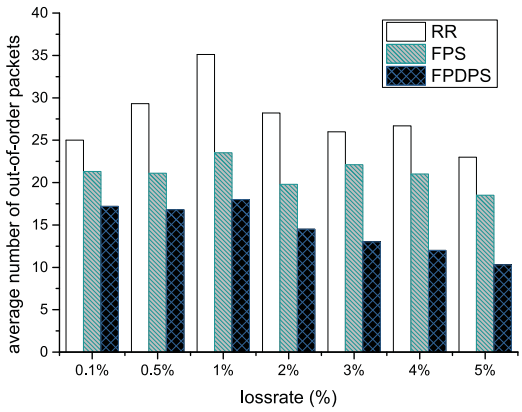


Fig. 11. the contrast of out of ordered packets with the packet loss rate change.

In addition to the packet loss rate, receiving buffer is one of the main factors affecting the throughput of MPTCP. If the receiving buffer is infinite, out-of-order packets can be temporarily stored in the receiving buffer and do not hinder to the rest of the packet to send, so that the overall throughput is not decreased, while the instantaneous throughput is still affected. In the next simulation, we control receiving buffer size as a variable, the loss rate of $subflow_0$ and $subflow_1$ are 0.5% and 1% respectively, other parameters are still same as before.

Fig. 12 shows the comparison of the globe throughput with the receiving buffer size changing, the accumulation of two separate TCP is also taken as a comparison. The throughput of TCP is a straight line without change because the buffer of TCP is always 65 536 bytes, it just displays as a benchmark. It can be seen in Fig. 12, when the receiving buffer is small, MPTCP throughput will be severely affected, even worse than the best throughput of TCP flows. As the receiving buffer increases gradually, the throughput of MPTCP also increases and tends to be the total throughput of two TCP flows. In this process, our solution is always the best.

Fig. 13 shows the aggregated benefit comparison. Note that the globe throughput of FPS and FPDPS are only equivalent to the best of a single TCP when the receiving buffer is too small. This is because scheduling algorithm will arrange almost all the data in the path which RTT is smaller than others. In addition,

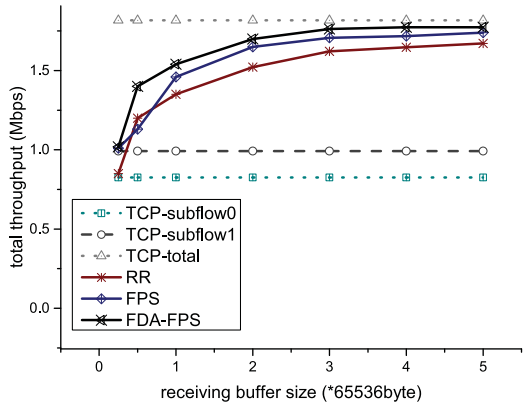


Fig. 12. the contrast of global throughput with the receiving buffer size changing.

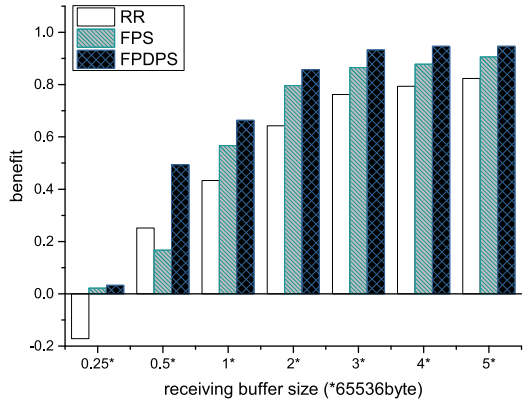


Fig. 13. the contrast of aggregation benefit with the receiving buffer size changing.

FPS makes a low utilization rate on the subflow with larger RTT because of FPS ignores the packet loss factor. When receiving buffer is only $65\,536 * 0.5$ bytes, the effect of the FPS is even worse than RR, while basic FPDPS will not have this problem. As the receiving buffer increases gradually, the benefit of these schemes also increase gradually, in which the basic FPDPS is always the best one.

Fig. 14 shows the number comparison of out-of-order packets in the three scheduling schemes. When the receiving buffer is only $65\,536 * 0.5$ bytes, the number of out-of-order packets in FPS is still very small, while the throughput of FPS is even lower than RR. Because FPS only uses $subflow_0$ to send data basically, there is no out-of-order packets caused by different factors among subflows. Removing this exception, it can see that the number of out-of-order packets in basic FPDPS is much smaller than that in RR and FPS.

B. Mobile Scenario

The second part of DPSAF, i.e., DAF, uses SACK feedback information to adjust basic scheduling value, which has a better performance experience under the condition of mobile network. In addition, mobile scenario always gives rise to the decrease of link quality. In this part, based on scenario in Fig. 8, we put up an effect of mobile link. As shown in Fig. 15, the link between

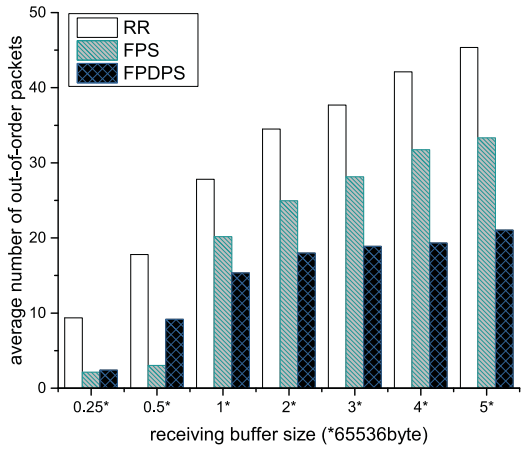


Fig. 14. the contrast of out of ordered packets with the receiving buffer size changing.

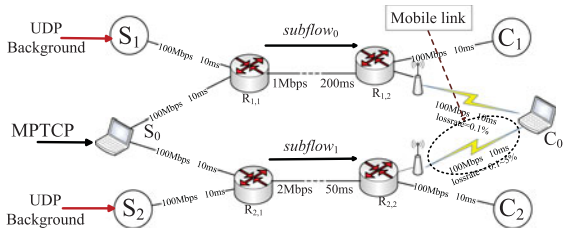


Fig. 15. Simulation setup of mobile scenario.

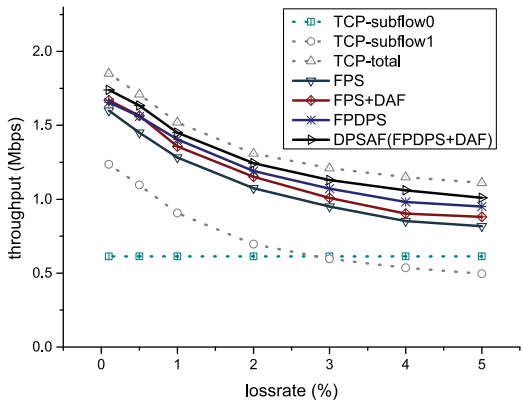


Fig. 16. global throughput changes with the packet loss rate.

MPTCP client C_0 and router $R_{2,2}$ is a mobile link. The user starts moving away from router $R_{2,2}$ at 20 s and moves back at 40 s, which means link state becomes worse at 20 s and recovers to the original state at 40 s. After 40 s, the link returns to the initial state.

2) Simulation results: Fig. 16 gives the comparison of globe throughput changing with packet loss rate of $subflow_1$. TCP-$subflow_i$ means the throughput of single TCP on path of $subflow_i$. The line of DPSAF shows the result of the complete scheme, which means FPDPS with DAF. With the increasing of packet loss, the throughput of all the schemes are falling. Because the more the packet loss rate grows, the greater the congestion window will be triggered to halve, which will make the falling throughput. However, Fig. 16 still shows the superiority of the scheduling algorithms revised based on the feedback

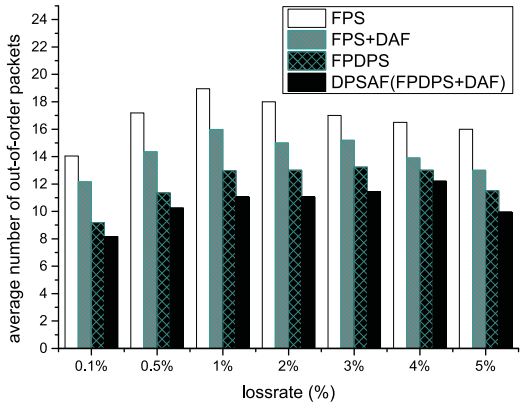


Fig. 17. unordered packets changes with the packet loss rate.

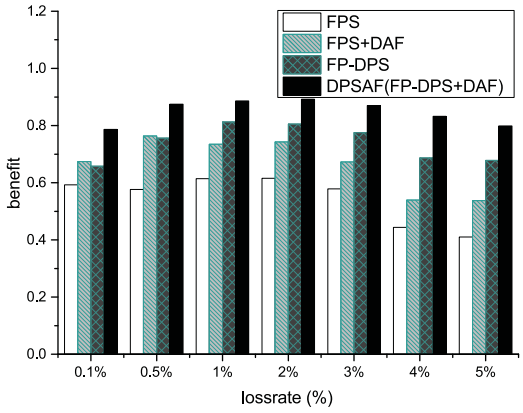


Fig. 18. MPTCP benefit changes with the packet loss rate.

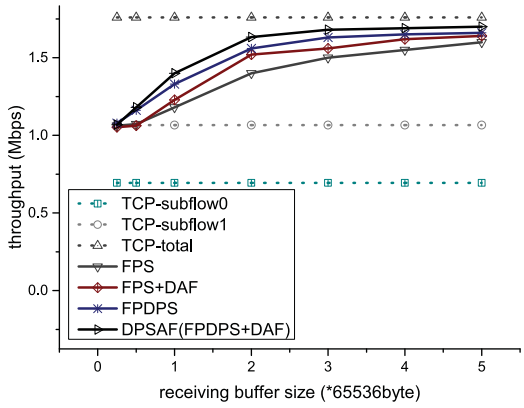


Fig. 19. global throughput changes with the receiving buffer size.

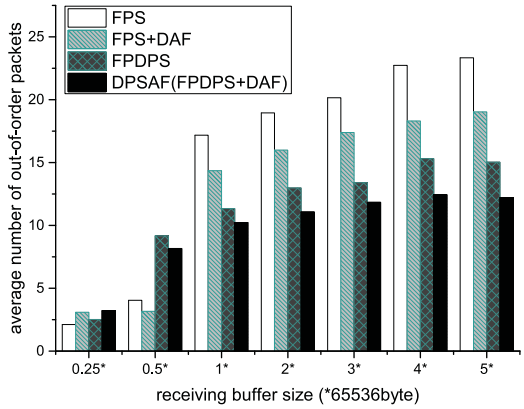


Fig. 20. unordered packets changes with the receiving buffer size.

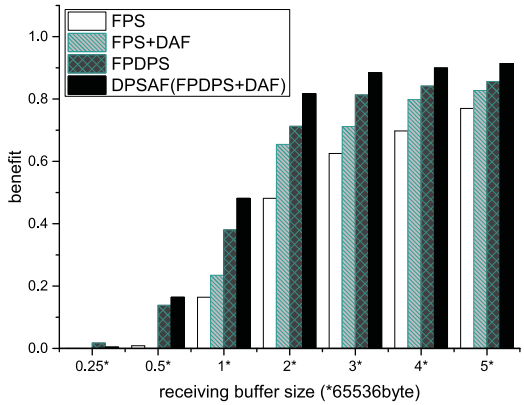


Fig. 21. MPTCP benefit changes with the receiving buffer size.

of SACK. Compared to the scheduling algorithms without the adjusting with feedback (FPS, FPDPS), the revised scheduling algorithms (FPS+DAF, DPSAF) have a certain degree of the throughput ascension. Fig. 17 shows the number comparison of out-of-ordered packets. For the performance comparison in this aspect, FPDPS is better than FPS, and the number of out-of-order packets in DPSAF (FPDPS+DAF) and FPS+DAF are obviously smaller than the original scheduling algorithms (FPDPS, FPS). The benefit comparison of FPS, FPS+DAF, FPDPS and DPSAF is shown in Fig. 18. From the figure, we can see that DAF brings benefits for the basic schedule algorithms of FPS and FPDPS. If there is less packet loss in network, the performance of FPS+DAF is even better than FPDPS. However, the benefit of DPSAF is always the best one, , which means DPSAF can efficiently aggregate both subflows and bring a best performance.

Fig. 19 shows the comparison of the globe throughput changing with buffer size, the loss rate of $subflow_1$ is 0.5% while other parameters are changeless. The simulation results show that the throughput of MPTCP increases with the receiving buffer increasing. As the receiving buffer increases, the throughput of DPSAF first approaches to a horizontal line, and the line of DPSAF is the closest to the line of the total throughout of two TCP subflows, which shows the applicability of DPSAF and it's the best among the algorithms in comparison. Fig. 20 shows the number comparison of the average out-of-order packets at MPTCP receiving buffer. Corresponding with the figure of throughput comparison, the number increment of out-of-order packets also reduced gradually with the increasing of buffer size, where DPSAF first tends to be smooth and steady. Fig. 21 shows the benefit comparison of the different schedule algorithms. Note that the globe throughput of FPS and the basic FPDPS is only equivalent to the best of two single TCP flows when the receiving buffer is enough small, and DAF will not bring benefits to them. As the receiving buffer increases gradually, in all these schemes, the throughput benefit of MPTCP also increases gradually and approximately approaches to 1, where DPSAF is always the best one, and more adapted to the situation with limited buffer.

V. Conclusion

MPTCP will inevitably encounter out-of-order packets problem caused by path asymmetry while using multiple paths for parallel data transmission. Scheduling algorithms try to guarantee the packets arrived in order, which can promote MPTCP throughput. However, traditional design principles of predictive scheduling algorithms only consider the influence of RTT in a single dimension, which cannot adapt to the wireless random packet loss environment. Moreover, the sender can not respond quickly when the network environment changes over time. In this paper, we introduce the packet loss rate into consideration, and use the feedback information from the SACK for further correction, which can further enhance the throughput of MPTCP in lossy heterogeneous networks.

Acknowledgment

The authors sincerely thank the anonymous referees for their valuable suggestions that have led to the present improved version of the original manuscript.

References

[1] S. Prabhavat, H. Nishiyama, N. Ansari, and N. Kato, "Effective delay-controlled load distribution over multipath networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 10, pp. 1730–1741, Oct. 2011.

[2] J. Wu, B. Cheng, Y. Shang, J. Huang, and J. Chen, "A novel scheduling approach to concurrent multipath transmission of high definition video in overlay networks," *J. Netw. Comput. Appl.*, vol. 44, pp. 17–29, 2014.

[3] S. Han, H. Joo, D. Lee, and H. Song, "An end-to-end virtual path construction system for stable live video streaming over heterogeneous wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 5, pp. 1032–1041, May 2011.

[4] S. Ferlin-Oliveira, T. Dreibholz, and Ö. Alay, "Tackling the challenge of bufferbloat in multi-path transport over heterogeneous wireless networks," in *Proc. 22nd IEEE Int. Symp. Qual. Serv.*, 2014, pp. 123–128.

[5] H. Im, C. Joo, T. Lee, and S. Bahk, "Receiver-side TCP countermeasure to bufferbloat in wireless access networks," *IEEE Trans. Mobile Comput.*, vol. 15, no. 8, pp. 2080–2093, Aug. 2016.

[6] K. Habak, K. A. Harras, and M. Youssef, "Bandwidth aggregation techniques in heterogeneous multi-homed devices: A survey," *Comput. Netw.*, vol. 92, no. 2015, pp. 168–188, 2015.

[7] W. Lee, J. Koo, Y. Park, and S. Choi, "Transfer time, energy, and quota-aware multi-RAT operation scheme in smartphone," *IEEE Trans. Veh. Technol.*, vol. 65, no. 1, pp. 307–317, Jan. 2016.

[8] X. Zheng, Z. Cai, J. Li, and H. Gao, "Scheduling flows with multiple service frequency constraints," *IEEE Internet of Things J.*, vol. 4, no. 2, pp. 496–504, Apr. 2017.

[9] X. Zheng, Z. Cai, J. Li, and H. Gao, "An application-aware scheduling policy for real-time traffic," in *Proc. 35th IEEE Int. Conf. Distrib. Comput. Syst..*, 2015, pp. 421–430.

[10] X. Zheng, Z. Cai, J. Li, and H. Gao, "A study on application-aware scheduling in wireless networks," *IEEE Trans. Mobile Comput.*, vol. 16, no. 7, pp. 1787–1801, Jul. 2017.

[11] K. Chebroluand and R. Rao, "Communication using multiple wireless interfaces," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2002, pp. 327–331.

[12] D. Kaspar, K. Evensen, P. Engelstad, and A. Hansen, "Using HTTP pipelining to improve progressive download over multiple heterogeneous interfaces," in *Proc. IEEE Int. Conf. Commun.*, 2010, pp. 1–5.

[13] P. Sharma, S. Lee, J. Brassil, and K. Shin, "Aggregating bandwidth for multihomed mobile collaborative communities," *IEEE Trans. Mobile Comput.*, vol. 6, no. 3, pp. 280–296, Mar. 2007.

[14] Y. Hasegawa, I. Yamaguchi, T. Hama, H. Shimonishi, and T. Murase, "Improved data distribution for multipath TCP communication," in *Proc. IEEE Glob. Telecommun. Conf.*, 2005, pp. 271–275.

[15] L. Magalhaesand and R. Kravets, "MMTP: Multimedia multiplexing transport protocol," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 2, pp. 220–243, 2001.

[16] C.-M. Huang, M.-S. Lin, and L.-H. Chang, "The design of mobile concurrent multipath transfer in multihomed wireless mobile networks," *Comput. J.*, vol. 53, no. 10, pp. 1704–1718, 2010.

[17] S. Shailendra, R. Bhattacharjee, and S. K. Bose, "MPSCTP: A simple and efficient multipath algorithm for SCTP," *IEEE Commun. Lett.*, vol. 15, no. 10, pp. 1139–1141, Oct. 2011.

[18] K. Tan, F. Jiang, Q. Zhang, and X. Shen, "Congestion control in multihop wireless networks," *IEEE Trans. Veh. Technol.*, vol. 56, no. 2, pp. 863–873, 2007.

[19] J. Wu, B. Cheng, and M. Wang, "Energy minimization for quality-constrained video with multipath TCP over heterogeneous wireless networks," in *Proc. 36th IEEE Int. Conf. Distrib. Comput. Syst.*, 2016, pp. 487–496.

[20] J. Wu, C. Yuen, B. Cheng, M. Wang, and J. Chen, "Streaming high-quality mobile video with multipath TCP in heterogeneous wireless networks," *IEEE Trans. Mobile Comput.*, vol. 15, no. 9, pp. 2345–2361, Sep. 2016.

[21] J. Wu, C. Yuen, B. Cheng, M. Wang, and J. Chen, "Energy-minimized multipath video transport to mobile devices in heterogeneous wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1160–1178, May 2016.

[22] J. Wu, C. Yuen, B. Cheng, Y. Yang, M. Wang, and J. Chen, "Bandwidth-efficient multipath transport protocol for quality-guaranteed real-time video over heterogeneous wireless networks," *IEEE Trans. Commun.*, vol. 64, no. 6, pp. 2477–2493, Jun. 2016.

[23] 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. 2016.

[24] J. Cloud, F. du Pin Calmon, W. Zeng, G. Pau, L. Zeger, and M. Medard, "Multi-path TCP with network coding for mobile devices in heterogeneous networks," in *Proc. 78th IEEE Veh. Technol. Conf.*, 2013, pp. 1–5.

[25] C. Xu, P. Wang, C. Xiong, X. Wei, and G.-M. Muntean, "Pipeline network coding-based multipath data transfer in heterogeneous wireless networks," *IEEE Trans. Broadcast.*, vol. 63, no. 2, pp. 376–390, Jun. 2017.

[26] C. Xu, Z. Li, L. Zhong, H. Zhang, and G.-M. Muntean, "CMT-NC: Improving the concurrent multipath transfer performance using network coding in wireless networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 3, pp. 1735–1751, Mar. 2016.

[27] D. Ni, K. Xue, P. Hong, and S. Shen, "Fine-grained forward prediction based dynamic packet scheduling mechanism for multipath TCP in lossy networks," in *Proc. 23rd Int. Conf. Comput. Commun. Netw.*, 2014, pp. 1–7.

[28] D. Ni, K. Xue, P. Hong, H. Zhang, and H. Lu, "OCPS: Offset compensation based packet scheduling mechanism for multipath TCP," in *Proc. IEEE Int. Conf. Commun.*, 2015, pp. 6187–6192.

[29] J.-O. Kim, T. Ueda, and S. Obana, "MAC-level measurement based traffic distribution over IEEE 802.11 multi-radio networks," *IEEE Trans. Consum. Electron.*, vol. 54, no. 3, pp. 1185–1191, Aug. 2008.

[30] N. Mohamed, J. Al-Jaroodi, H. Jiang, and D. R. Swanson, "A user-level socket layer over multiple physical network interfaces," in *Proc. Int. Conf. Parallel Distrib. Comput. Syst.*, 2002, pp. 804–810.

[31] R. Stewart *et al.*, "Stream control transmission protocol," Informational RFC, RFC2960, 2000.

[32] J. Iyengar, P. 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.

[33] P. Amer *et al.*, "Load sharing for the stream control transmission protocol (SCTP)," *IETF Personal Draft, draft-tuexen-tsvwg-sctp-multipath-13*, 2016.

[34] C. Casetti and W. Gaiotto, "Westwood SCTP: Load balancing over multipaths using bandwidth-aware source scheduling," in *Proc. 60th IEEE Veh. Technol. Conf.*, vol. 4, 2004, pp. 3025–3029.

[35] F. Mirani, N. Boukhatem, and M. Tran, "A data-scheduling mechanism for multi-homed mobile terminals with disparate link latencies," in *Proc. 72nd IEEE Veh. Technol. Conf.*, 2010, pp. 1–5.

[36] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," Experimental RFC, RFC6824, 2013.

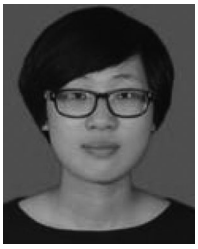
[37] S. Barré, "Implementation and assessment of modern host-based multipath solutions," Ph.D. dissertation, Louvain School Eng., Université catholique de Louvain, Louvain-la-Neuve, 2011.

[38] Linux MPTCP kernel, [Online]. Available: http://www.multipath-tcp.org/, Accessed on: 2017.

[39] B. Sikdar, S. Kalyanaraman, and K. S. Vastola, "Analytic models and comparative study of the latency and steady-state throughput of TCP Tahoe, Reno and SACK," in *Proc. IEEE Glob. Telecommun. Conf.*, 2001, pp. 1781–1787.
[40] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: A simple model and its empirical validation," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 133–145, Apr. 2000.
[41] NS3 simulator, [Online]. Available: www.nsnam.org/, Accessed on: 2017.
[42] MPTCP NS3 code, [Online]. Available: http://code.google.com/p/mptcp-ns3/, Accessed on: 2017.
[43] C. Paasch, "Improving multipath TCP," Ph.D. dissertation, Louvain School Eng., Université catholique de Louvain, Louvain-la-Neuve, 2014.

Kaiping Xue (M'09–SM'15) received the B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC), Hefei, China, in 2003 and the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. He is currently an Associate Professor in the Department of Information Security and the Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks, and network security.

Jiangping Han received the B.S. degree from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), Hefei, China, in July, 2016. She is currently a graduate student in communication and information system in the Department of EEIS, USTC. Her research interests include future Internet architecture design and MPTCP performance optimization.



Dan Ni received the Master's degree from the Department of Electrical Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), Hefei, China, in 2015. Her research interests focuses on MPTCP performance optimization.

Wenjia Wei received the B.S. degree from the school of information science and engineering, in 2013. He is currently working toward the Ph.D. degree in information and communication engineering from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China. His research interests has focused on next-generation Internet performance optimization.

Ying Cai (M'14) received the B.S. degree from Xidian University, Xi'an, China, the M.S. degree from the University of Science and Technology Beijing, Beijing, China, both in applied mathematics, and the Ph.D. degree in information security from Beijing Jiaotong University, Beijing, China, in 1989, 1992, and 2010, respectively. She is currently a Full Professor at Beijing Information Science and Technology University, Beijing, China. Her current research interests include cyber security, wireless networks, and cryptography algorithm.

Qing Xu received the B.S. and Ph.D. degrees from the Department of Physics, Zhejiang University, Hangzhou, China, in 2006 and 2012. He is currently a Research Engineering in Huawei Shanghai Research Institute, Shanghai, China. His research interests include traffic flow distribution in IP network, new packets delivery protocols, and 5G core network research.

Peilin Hong received the B.S. and M.S. degrees from the Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China, in 1983 and 1986. She is currently a Professor and Advisor for Ph.D. candidates in the Department of EEIS, USTC. Her research interests include next-generation Internet, policy control, IP QoS, and information security.