MPMTP-AR: Multipath Message Transport Protocol Based on Application-Level Relay

Shaowei Liu, Weimin Lei, Wei Zhang and Xiaoshi Song
School of Computer Science and Engineering, Northeastern University
Shen Yang, Liao Ning 110819 – China
[e-mail: shaoweiliu@stumail.neu.edu.cn, leiweimin@mail.neu.edu.cn]
*Corresponding author: Weimin Lei

Received July 7, 2016; revised October 17, 2016; revised November 17, 2016; accepted January 11, 2017; published March 31, 2017

Abstract

Recent advancements in network infrastructures provide increased opportunities to support data delivery over multiple paths. Compared with multi-homing scenario, overlay network is regarded as an effective way to construct multiple paths between end devices without any change on the underlying network. Exploiting multipath characteristics has been explored for TCP with multi-homing device, but the corresponding exploration with overlay network has not been studied in detail yet. Motivated by improving quality of experience (QoE) for reliable data delivery, we propose a multipath message transport protocol based on application level relay (MPMTP-AR). MPMTP-AR proposes mechanisms and algorithms to support basic operations of multipath transmission. Dynamic feedback provides a foundation to distribute reasonable load to each path. Common source decrease (CSD) takes the load weight of the path with congestion into consideration to adjust congestion window. MPMTP-AR uses two-level sending buffer to ensure independence between paths and utilizes two-level receiving buffer to improve queuing performance. Finally, the MPMTP-AR is implemented on the Linux platform and evaluated by comprehensive experiments.

Keywords: Transport protocol, multipath transport, overlay network, reliable data delivery, congestion control

https://doi.org/10.3837/tiis.2017.03.009 **ISSN : 1976-7277**

1. Introduction

Currently, the majority of data transmissions go through TCP. However, the performance of TCP degrades significantly due to large amount of retransmission caused by packet loss or error. In addition, it is either not possible or cost-effective to deliver a traffic flow with a certain bandwidth demand over a single path. Motivated by optimizing transport performance, the user may want to simultaneously transmit data over multiple paths.

Multipath transport is an effective way to aggregate transport capacity of multiple paths and protect transmission from path failure. During the past few years, multipath transmission has attracted extensive research interests. Multi-homing scenario is one of the most popular approaches to construct multiple paths between end devices. However, it requires at least one of end devices has multiple network interfaces, which is hard to be satisfied in practical applications. An alternative approach for multipath transport is to provide support mechanisms in the application layer while retaining the best-effort network layer. Overlay network is regarded as an effective way to establish multiple paths between end devices by application layer routing, as well as protocols without any changes in the underlying network layer. In our previous work, a multipath transport system based on the application level relay (MPTS-AR) was proposed to provide multipath transport service by using overlay network technique [1, 2]. In MPTS-AR, end-to-end path between end devices is constructed by controlling the sequence of overlay nodes that the packet will travel before reaching its destination. Overlay node on the end-to-end path provides UDP-based data forwarding service for the sender.

Conventional transport protocols such as TCP and UDP cannot support the using of multiple paths concurrently. To solve this problem, some research has been devoted to multipath transmission. Exploiting multipath characteristics has been explored for multi-homing scenario. For instance, the Internet Engineering Task Force (IETF) has proposed multipath TCP (MPTCP) [3] and stream control transport protocol (SCTP) [4]. MPTCP is a TCP extension that allows sender to send data over different paths. It establishes one TCP connection, called subflow, between each IP address pair [5]. Data can be sent over any subflow according to data distribution policy. SCTP is an alternative transport protocol that supports multiple IP addresses pairs. The previous versions of SCTP only use multiple addresses in failover scenario, but recent evolutions enable SCTP to use several paths simultaneously [6]. As of this writing, no detailed solution of exploring multipath characteristics for overlay network scenario has been published.

Multipath message transport protocol based on application level relay (MPMTP-AR) that provides reliable transport service is proposed to fill this gap. The requirements for overlay network scenario differ notably from that of multi-homing scenario. For instance, forwarding data at the application layer may destruct original end-to-end connection which has a significant impact on transmission efficiency. A connection between the sender and receiver consists of multiple connections between adjacent overlay network nodes. If packet is lost, the lost packet can be retransmitted from either the sender or the nearest overlay network node. Comparing with retransmission from the sender, retransmission from the nearest overlay network node decreases transmission efficiency. As overlay network node only sends data from one overlay network node to the other overlay network node, and there is no guarantee that the data reach the receiver.

The main contributions of this paper are summarized as below.

- Dynamic feedback mechanism provides a foundation to optimize data distribution.
- Congestion control scheme takes load weight into consideration to adjust the congestion window, which ensures fairness between available paths.
- Two-level sending buffer ensures independent transmission of each path.
- Two-level receiving buffer improves queuing performance.

The rest of the paper is organized as follows. Section 2 briefly reviews related work about multipath transport. Section 3 discusses the detail of MPMTP-AR. Section 4 presents experimental results. Finally, Section 5 concludes this work.

2. Related Work

This section briefly reviews MPTS-AR and multipath transport protocols.

2.1 MPTS-AR

MPTS-AR is a multipath transport solution that establishes a new routing system on overlay network. The MPTS-AR architecture provides multipath transport related functions. The functional architecture consists of three kinds of functional entities (FEs), i.e., user agent, relay controller, and relay server, as shown in **Fig. 1**. Although MPTS-AR has no control over how the packet is routed in the underlying network between two FEs, it controls the sequence of FEs that the packet traverses before reaching its destination.

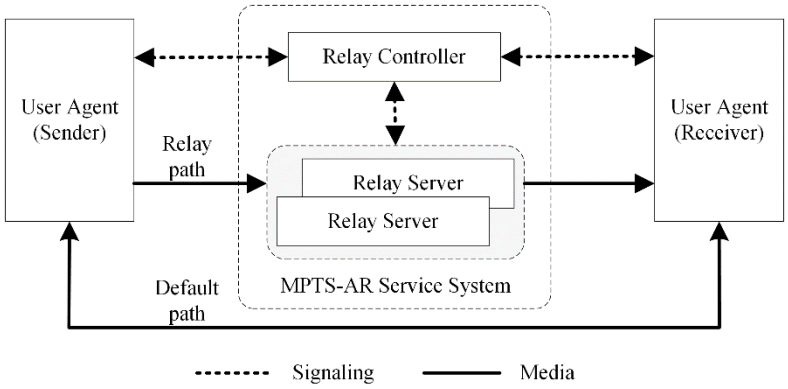


Fig. 1. The general framework of multipath transport system based on application-level relay.

Terms to be used in MPTS-AR are defined as below.

- Flow: A set of data packets along a specific path.
- Default path: A path between the sender and receiver does not pass through any relay server.
- Relay path: A path between the sender and receiver passes through one or more relay servers.
- Session: An association of one or more flows used to transmit data between sender and receiver.

Relay controller and relay server compose of MPTS-AR service system that provides multipath data forwarding service to user agent. Relay controller gathers location and state

information of available relay servers, constructs a topology map of the overlay network, and manages relay path for MPTS-AR service system.

Relay server serves as routing module that accepts routing instructions from relay controller. It maintains a flow table and forwards data flow to next hop FE. Massive deployment of relay server is regarded as an efficient way to provide a better multipath condition for MPTS-AR service system. Meanwhile, MPTS-AR service system subscriber can provide data forwarding service for each other.

User agent is responsible for sending and receiving data packet over multiple paths. At the sender side, user agent maintains a path list including a default path and one or more relay paths. Before data transmission, user agent requests relay controller to allocate relay path for the transmission. User agent selects a path for each data packet from the path list. At the receiver side, user agent reorders the received data packets. After data transmission termination, user agent requests relay controller to release relay path.

Relay path from the sender to the receiver is composed of a set of continuous links that between two FEs. Relay paths are distinguished by a unique path identity (PID) assigned by relay controller. Each item in flow table includes a PID and the address of next hop FE. PID carried by packet header is used to look for the address of next hop FE. In addition to PID, data packet header also includes a transmission sequence number (TSN) and a flow sequence number (FSN) to collect transfer status and reorder the received packet.

MPTS-AR also defines a general multipath transport protocol (MPTP) to encapsulate data from the upper application. UDP is a fast and lightweight transport layer protocol. In order to be capable for various applications, MPTP adopts UDP as transport layer protocol. Each UDP packet carries one MPTP packet. MPTS-AR adopts a set of MPTPs, each of which extends from MPTP and aims to support a specific category of upper applications. MPMTP-AR discussed in this paper is one of MPTPs.

2.2 Multipath Transport Protocols

MPTCP is a TCP-based multipath transport protocol. It supports the simultaneous use of multiple paths for an end-to-end connection. MPTCP establishes one connection for each IP address pair and provides a single interface to application. Transmission on each path reuses most functions of TCP. However, when multiple paths have significantly different path characteristics, MPTCP suffers from a performance degradation. This is because a path with low quality affects the performance of other paths, which may lead to head-of-line blocking problem. To address this problem, Zhou et al. introduced congestion window adaption algorithm for the MPTCP (CWA-MPTCP) [7]. CWA-MPTCP dynamically adjusts the congestion window for each flow so as to mitigate the variation of end-to-end delivery delay. It reduces out-of-order packets by predicting the arrival sequence. Y. Cui et al. proposed a solution that takes advantage of the random nature of the fountain code to flexibly transmit encoded symbols from the same or different data blocks over different paths to mitigate the negative impact of the heterogeneity of different paths [8]. D. Zhou et al. proposed extension to the congestion control of MPTCP to promote user cooperation [9].

SCTP is initially designed for transmitting signaling reliably, such as SS7 for VoIP session establishment and teardown. SCTP application submits data to the SCTP transport layer. Then SCTP controls message delivery with a granularity of chunk. It is capable of supporting several IP address pairs per connection. In the primary version of SCTP, only the primary IP address pair is used and the other IP address pairs are used as backups. But recent version enables it to support the use of multiple paths simultaneously. Dreibholz et al. introduced the

SCTP and gave an overview of activities and challenges in the areas of security and concurrent multipath transfer [10]. Iyengar et al. proposed concurrent multipath transfer using SCTP and handled with three negative side-effects introduced by concurrent multipath transfer [6, 11]: (1) unnecessary fast retransmissions by a sender; (2) increased acknowledgment traffic; and (3) overly conservative congestion window growth. Wallace et al. presented a comprehensive survey of SCTP and discussed three main research topics: handover management, concurrent multipath transfer, and cross-layer activities [12].

S. Mao et al. proposed a solution for real-time multimedia delivery and gave an overview of basic elements of multipath transport [13]. Huang et al. proposed fast retransmission made by the relay gateway for concurrent multipath transfer (RG-CMT) in vehicular networks [14]. When packets are lost due to error loss or handoff loss in the wireless link, RG-CMT is able to fast retransmit lost packets from the relay gateway to the vehicle, which saves transfer time and bandwidth. However, RG-CMT requires relay gateway to store data, which is hard to be implemented in backbone network. C. Xu et al. proposed a quality-aware adaptive concurrent multipath transfer solution (CMT-QA) to provides real-time media delivery and reliable data transmission in wireless networks [15]. CMT-QA monitors and analyzes each path's data handling capability and makes data delivery adaptation decisions to select the qualified paths for concurrent data transfer. O. C. Kwon et al. improved video quality and to alleviated head-of-line blocking problem by using systematic raptor codes in multipath transport scenario [16].

3. MPMTP-AR

3.1 Overview

Fig. 2 shows the relationship of MPMTP-AR to other protocol in TCP/IP protocol suite. MPMTP-AR works over UDP and serves as the intermediary between the application programs and the transport layer operations. MPMTP-AR provides process-to-process communication that simultaneously uses multiple paths to deliver data. Not only does MPMTP-AR provide reliable delivery service, but also it offers full duplex communication. Hence, two user agents have to establish/terminate a virtual connection, maintain its own sending and receiving buffer, and use acknowledgement mechanism to ensure correctly arrival of data.

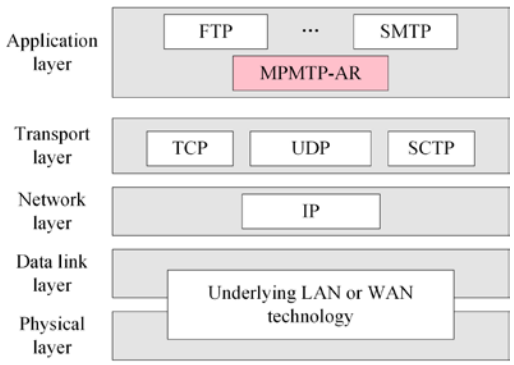


Fig. 2. TCP/IP protocol suite.

3.2 Features

To provide multipath reliable transport services, MPMTP-AR has six features that are briefly introduced in this section and discussed later. Although parts of the features are well-known in conventional single path transport scenario, MPMTP-AR redefines them.

1) Numbering system: Numbering system assists user agent to record delivery status of each path. Each packet associates with a TSN and a FSN.

MPMTP-AR uses two-level number mechanism to number all data bytes. When MPMTP-AR receives data bytes from the upper application, it stores them in session-level sending buffer and numbers them by TSN. The TSN starts from an arbitrary number negotiated in session establishment. Then MPMTP-AR distributes data bytes to path-level sending buffers. Within each path-level sending buffer, MPMTP-AR numbers data bytes by FSN. FSN must start from 0.

At the receiver side. MPMTP-AR uses acknowledgment number to confirm the TSN of the last continuous byte received successively and discontinuous blocks of packet which are received correctly.

2) Feedback: User agent collects status of each path. Feedback message which carries transmission status provides a foundation to optimize transmission performance.

3) Flow control: Flow control mechanism regulates the amount of data a sender can send without overwhelming the receiver. Receiver sends feedback to sender to inform the sending permission. The numbering system supports MPMTP-AR to do a byte-oriented flow control.

4) Error control: Error control mechanism is used to recover the lost data packet. At the receiver side, MPMTP-AR detects and discards the corrupted or duplicated data packet, and acknowledges the correctly received data packets. At the sender side, MPMTP-AR retransmits the unacknowledged data packets.

5) Congestion control: The purpose of congestion control is to ensure the traffic in the network not to exceed the capacity of the network. In a real network, data transmission rate is not only controlled by the receiver, but also determined by the level of congestion.

6) Data distribution and data reassembling: Data packets are transmitted from the sender to receiver over multiple paths. At the sender side, MPMTP-AR distributes data packets to multiple paths. At the receiver side, MPMTP-AR reorders the received data packets and recoveries the packet loss.

3.3 Packet Format

MPMTP-AR uses header extensions to support additional functions not defined by MPTP. It defines two kinds of packets, i.e., MPMTP-AR data packet and MPMTP-AR control packet. This section introduces parts of their header field while their usage will be presented in basic operations.

MPMTP-AR data packet consists of a header of 16 bytes. Header field of the data packet includes type of service (TOS), checksum, PID, TSN, and FSN. TOS, similar to that in Internet packet header, represents its priority. If relay server meets congestion, it will discard data packet with the lowest priority first.

MPMTP-AR control packet consists of a header of 8 bytes. Header field of the control packet includes control packet type (CT), length, and PID. CT indicates different types of control packet. MPMTP-AR defines three classes of control packets, i.e., session management packet, feedback packet, and enhanced selective acknowledgment packet (ESACK).

3.4 MPMTP-AR Session

MPMTP-AR is a connection-oriented protocol. MPMTP-AR establishes a virtual connection that is composed of multiple paths between the sender and receiver. All packets that belong to a session are sent over this virtual connection.

MPMTP-AR supports full-duplex communication. Before starting data transmission, User agent uses typical three-way handshaking to establish a virtual connection. User agent negotiates initial value of TSN, receive window size, and the number of path to be used.

MPMTP-AR, unlike TCP, does not allow a “half-close” session. If one user agent closes the session, the other user agent must stop receiving data from the upper application, send leftover data and close the session.

3.5 State Transition Diagram

To keep track of all the different events happening during session establishment, data transfer, and session termination, MPMTP-AR, like TCP and SCTP, is specified as finite state machine (FSM).

Fig. 3 shows the two FSMs used by the MPMTP-AR sender and receiver combined in one diagram. The ovals represent the states. The transition from one state to another is shown using directed lines. The solid lines of the figure represent normal transition for the sender; the dotted lines of the figure represent normal transition for the receiver, and the chain dotted lines represent unusual situation. **Table 1** shows the state of MPMTP-AR. This paper does not intend to illustrate the MPMTP-AR state and transition in detail. FSM of TCP and SCTP can help us to understand the state of MPMTP-AR.

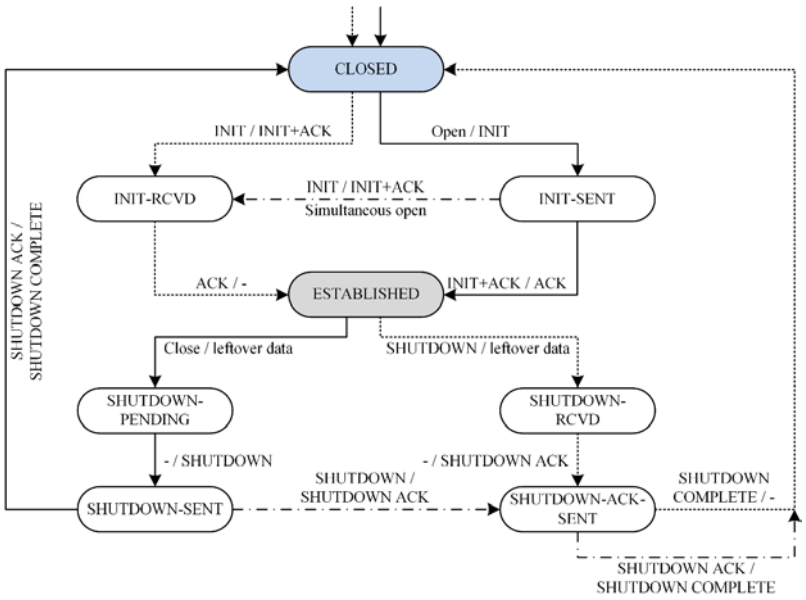


Fig. 3. State transition diagram.

Table 1. State for MPMTP-AR

State	Description
CLOSED	No session (starting state)
INIT SENT	INIT sent; waiting for ACK
INIT RECV	INIT+ACK sent; waiting for ACK
ESTABLISHED	Session established; data transfer take place
SHUTDOWN-PENDING	Sending data after receiving close from upper application
SHUTDOWN-SENT	SHUTDOWN sent; waiting for SHUTDOWN ACK
SHUTDOWN-RECV	Sending data after receiving SHUTDOWN
SHUTDOWN-ACK-SENT	Waiting for close completion

3.6 Basic Operations

In the following, we discuss the basic operations of MPMTP-AR.

1) Window in MPMTP-AR: User agent maintains a shared session-level send window and a shared session-level receive window for all paths. The size of session-level send window is determined by flow control and congestion control. The session-level send window in MPMTP-AR is similar to send window in TCP, but with a difference in the number of timer. TCP maintains only one timer, while MPMTP-AR maintains a timer for each path-level send window.

2) Feedback: Receiver sends receiver report (RR) to inform the sender of the transmission status of each path. RR carries the bytes of correctly received bytes, and the maximum TSN/FSN of the last correctly received packet. To ensure correct arrival, RR are transmitted over the path with the best performance.

In a stable network environment, feedback is sent at a relatively constant rate. In a highly dynamic network environment, timely feedback is necessary for the sender to quickly adapt to transmission errors.

3) Flow control: MPMTP-AR separates flow control from congestion control. We assume that the virtual connection between the sender and receiver is congestion free when we discuss flow control.

The ESACK messages are used to inform the sender about receive window size. Aggregation of receive window of available paths is one of the factors that determine the size of session-level send window.

4) Error control: Error control mechanism is an essential way to ensure the reliability of data delivery. Error control mechanism retransmits corrupted/lost packet and discards duplicated packet. Three simple tools, i.e., checksum, ESACK, and timer, help MPMTP-AR to do error control.

The checksum, similar to that in TCP, is used to check for corrupted packet. If a packet has an invalid checksum, the packet is discarded by receiver and considered as packet loss.

Sender retransmits the lost packet. When sender sends a packet, it stores the packet in a session-level sending buffer until the packet is acknowledged. Retransmission in MPMTP-AR is more complexity than that in TCP. On the one hand, multiple paths have different path characteristics. Sender sets a retransmission timer for each path. On the other hand, a packet is retransmitted when the retransmission timer expires or when the sender receives multiple duplicate ACKs from the path on which the packet was sent. The number of duplicate ACKs of the path with a smaller round trip time (RTT) is smaller than that of the path with a bigger RTT.

5) Congestion control: Congestion control mechanism helps sender to avoid congestion. Sender maintains a session-level receive window and a shared congestion window for all paths. The actual size of sending permission is the minimum value of these two windows. Typical congestion control includes three phases: slow start, congestion avoid, and congestion detection. When congestion happens, sender multiplicatively decreases congestion window. In multipath transport scenario, congestion of one path is not equal to congestion of all paths. Hence, congestion control mechanism has to evaluate the level of congestion. Then sender decreases congestion window and redistributes the load that originally assigned to the path with congestion to the other paths.

6) Data distribution: To improve the transmission performance, sender distributes data to multiple paths in an optimal way. A simple distribution scheme is round-robin algorithm which equally assigns data to multiple paths. Although the algorithm is easy to be implemented, it cannot significantly improve the transmission efficiency even reduces throughput. Referring to [15, 17], data distribution should consider the path quality.

MPMTP-AR keeps a shared session-level sending buffer for all paths and a path-level sending buffer for each path, as shown in **Fig. 4**. Sender distributes traffic to path-level sending buffer based on path quality. Each path transmits packet independently which alleviates head-of-line blocking problem.

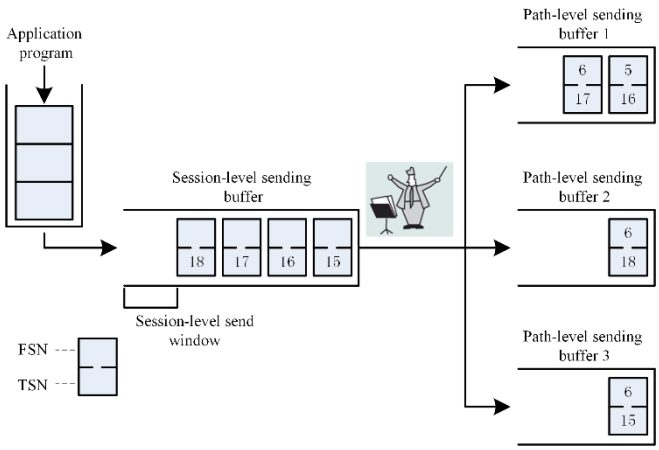


Fig. 4. Data distribution.

7) Data reassembling: At the receiver sider, MPMTP-AR reassembles received packets, as shown in **Fig. 5**. The received packets experience two types of jitter including jitter within path and jitter across multiple paths. The receiver uses two-level receiving buffer to absorb jitter and reorder the received packets. First, the receiver uses path-level receiving buffer to store received packet from each path. The path-level receiving buffer collects receiving status and orders the received packets by FSN. The session-level receiving buffer publishes reassembling requirement to each path-level receiving buffer. Then, the path-level receiving buffer pushes packets to session-level receiving buffer accordingly. As shown in **Fig. 5**, session-level receiving buffer is waiting for the packet with TSN of 16. Path-level receiving buffer of path 1 immediately sends the packet with TSN of 16 to session-level receiving buffer after receiving it. The session-level receiving buffer orders received packets by TSN.

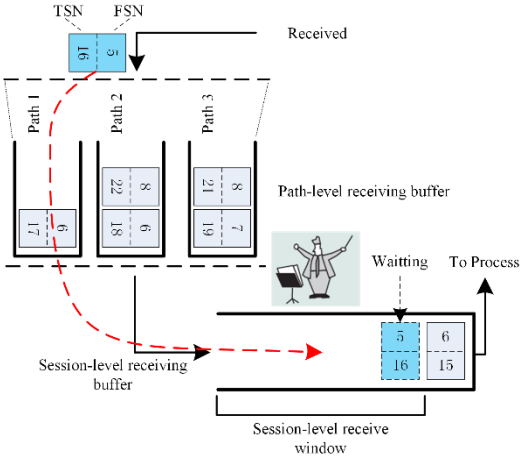


Fig. 5. Data reassembling.

Some studies focus on data reassembling [18-20]. These studies state that buffer requirement and computational complexity are moderate if the data distribution scheme is adaptive to path quality.

3.7 Implementation Policy

The previous section introduces MPMTP-AR. This section proposes several policies for MPMTP-AR deployment.

1) Feedback and evaluation policy: RR carries transmission status of each path. The sender evaluates path quality by using local sending status and long-term receiver status. In this work, path quality mainly refers to reliability.

At the beginning of data transfer, sender assumes that all paths have an equal reliability. When a subsequent statistic is collected, sender updates path *i*'s reliability (R_i) as following

$$R_i = \alpha R_i + (1-\alpha)\frac{er_i}{es_i} \tag{1}$$

where er_i is the amount of packet in bytes correctly received in current evaluation period, and es_i represents the amount of packet in bytes sent in current evaluation period. α indicates the impact of previous result on the following result.

Sender distributes more load to the path with higher reliability and distributes less load to the path with lower reliability. After obtaining a new value of reliability, the sender adjusts the load weight by a ratio between the reliability of path and the average reliability of available paths as following:

$$R = \alpha R + (1-\alpha)\frac{\sum_{i=1}^{N} er_i}{\sum_{i=1}^{N} es_i} \tag{2}$$

$$w_i = w_i \times \frac{R_i}{R}$$

where N is the number of available paths, R represents the average reliability of available paths, and w_i is the load weight of path *i*.

Timely and accurate feedback is very important to evaluate path quality. In default configuration, feedback consumes 2.5% of session bandwidth. The bandwidth fraction for feedback should be dynamically adjusted. If path performance fluctuates rapidly, the MPMTP-AR increases the bandwidth fraction for feedback. Otherwise, the MPMTP-AR decreases the bandwidth fraction for feedback. The sender sets a variable (d_R) to calculate the change of R, as shown in (3).

$$d_R = R_c - R_l \tag{3}$$

R_c and R_l are the current and last value of R, respectively.

The bandwidth fraction for feedback (bff) should be kept within a reasonable range from 1% to 5% . It can be dynamically adjusted as following:

$$bff = \begin{cases} \max(0.01,(1-\beta)bff), d_R \in [-0.05,0.05]; \\ \min((1+\beta)bff,0.05), d_R \notin [-0.05,0.05]. \end{cases} \tag{4}$$

where β is a factor.

How long should we update path reliability and bff is an issue to be addressed. If time interval is too short, it may not correctly reflect the reliability when data packet transmitted in network encounters burst congestion. However, if time interval is too long, it may not accurately reflect variation trend of path reliability. Consequently, the time interval should be dynamically adjusted based on transmission status.

In MPMTP-AR, time interval adjustment is considered as a cognitive closed loop process. The sender calculates reliability and bff when it receives every three new RRs. The new value of bff affects the frequency of sending RR. The time interval between RRs serves as a control variable to determinate the time interval of updating the value of path reliability. Section 4.2 will illustrate how to select appropriate value of α and β.

2) Congestion control policy: With TCP, congestion window decreases to the half of its value when the sender detects a congestion. In multipath transport, congestion control should ensure fairness between available paths. Common source decrease (CSD) scheme is proposed to address this problem. CSD focuses on multiplicative decrease phase. Briefly, CSD detects congestion on each path and tries to decrease the traffic of the path with congestion. Then, the congestion window of the session is adjusted based on the level of congestion to ensure it does not decrease faster than regular TCP connection. For instance, when sender detects a congestion on path 1, CSD acts as below:

a) It decreases the load weight of path 1 to half of its value, marking as $w_1^{'}$.

b) It distributes the difference of load weight of path 1 to the other paths by

$$w_j = w_j + w_1^{'} \times \frac{w_j}{\sum_{m=2}^{N} w_m} \tag{5}$$

where N is the number of available paths, w_j is the load weight of path j.

c) It decreases the congestion window of the session to $(1-w_1^{'})$ of its value.

3) Data distribution policy: The sender selects path for each packet to ensure sequential arrival as possible. The amount of traffic corresponding to load weight should be evenly

distributed. Hence, the sender selects the path with the lowest utilization ratio to transmit the packet. Utilization ratio is calculated as following:

$$u = s / w \tag{6}$$

where s is the number of bytes of packet has been sent, w is load weight, u represents utilization.

3.8 Usage Scenarios

We assume that there are a lot of resources on the server. When a client wants to download a big file from the server or to update a big file to the server, MPMTP-AR will probably appear here.

When a MPMTP-AR session is established between the client and server, the session can simultaneously use multiple paths to deliver data. Thus the session has an opportunity to obtain a higher bandwidth and a better reliability.

4. Performance Study

During experiments, the proposed MPMTP-AR is fully implemented on simulation and actual platform. These experiments first verify the performance of MPMTP-AR by simulations. Then, MPMTP-AR is compared with MPTCP by actual tests. All test results displayed are the average value of the results of 30 runs in order to avoid being influenced by any stochastic factors.

4.1 Performance Verification

In this section, performance verification of MPMTP-AR is presented.

1) The impact of congestion control: From the viewpoint of congestion control, using multiple paths for one session leads to an important and attractive problem. MPMTP-AR session uses multiple paths, and these paths will typically experience different levels of congestion. A naive solution to the congestion problem in MPMTP-AR would be to use the standard TCP congestion-control scheme. This can be easily implemented but leads to unfairness. If a session uses two paths, then the sender experiences the congestion from the two paths. When only one path occurs congestion, it is unfair, if the sender uses classical congestion window management mechanism called Reno [21], to decrease half of its congestion window.

In this experiment, there are multiple available paths between the sender and receiver. Each path has the same path characteristics. When the sender detects a congestion on a path, it uses Reno and CSD to adjust congestion window, respectively. **Fig. 6** shows the normalized congestion window when the number of available paths increases from 1 to 4. In the case of Reno, we can observe that the congestion window decreases to 50 percent regardless of the number of available paths. In the case of CSD, the congestion window is reduced decreases to a reasonable size according to the load weight of the congested path. Transmission on the other paths will not be affected by the congested path. Hence, CSD ensures the fairness between available paths.

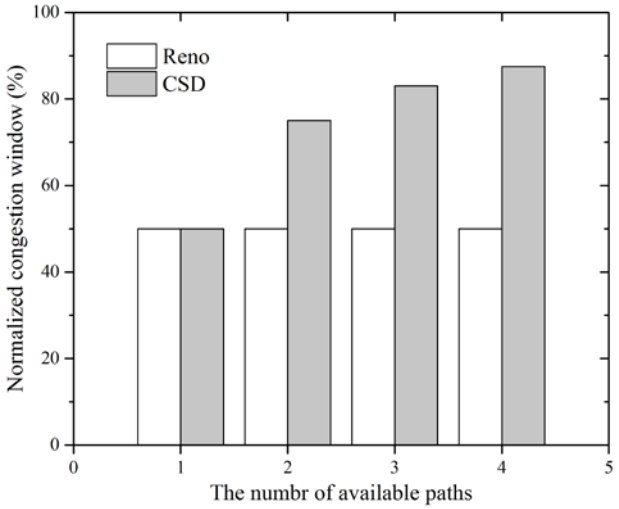


Fig. 6. Comparison of normalized congestion window between Reno and CSD.

2) The impact of error control: This part presents a study of error control through a hardware-in-the-loop simulation on OMNeT++ platform. The simulation environment is described as **Fig. 7**. Relay server 2 is implemented on a real server. There are two relay paths for each directional of data transfer. In each direction, one relay path passes relay server 1 and the other relay path passes real relay server 2. During transmission, the performance of relay server 2 fluctuates rapidly.

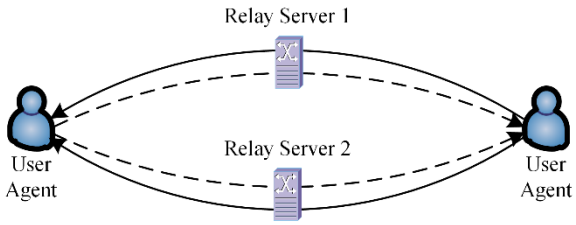


Fig. 7. The environment of simulation. There are two alternative paths on each direction between two user agents.

Fig. 8 shows a unidirectional data transfer trace between these two user agents. The x-axis shows the time since the start of the data transfer, while the y-axis displays the evolution of the sequence number of outgoing packets. The triangle curve represents the evolution of FSN on path 2 which passes relay server 2. At the beginning, path 2 transmits data at a regular rate. Between the 3 s and the 4 s, the relay server 2 turns down its forwarding service for user agent, and even drops packets indicated by cross symbol. Sender detects the packet loss on path 2 at the 4.5 s. Sender decreases load weight of path 2 and redistributes the load originally assigned to the path 2 to path 1. At the 7 s, relay server 2 recovers from the poor status. Sender increases load weight of path 2. The circle curve displays the statistic of FSN of path 1 which passes relay server 1, and the plus symbol indicates the retransmitted data dropped by relay server 2 and redistributed to path 1.

The upper square curve describes the evolution of TSN of the session. It represents the sum of the throughput of two paths. When the performance of relay server 2 becomes poor, the session's connection is not affected. It recovers the losses by retransmission and adjusts the data distribution quickly. **Fig. 8** displays that the MPMTP-AR is an effective way to recover the packet loss and protect the session from path failure.

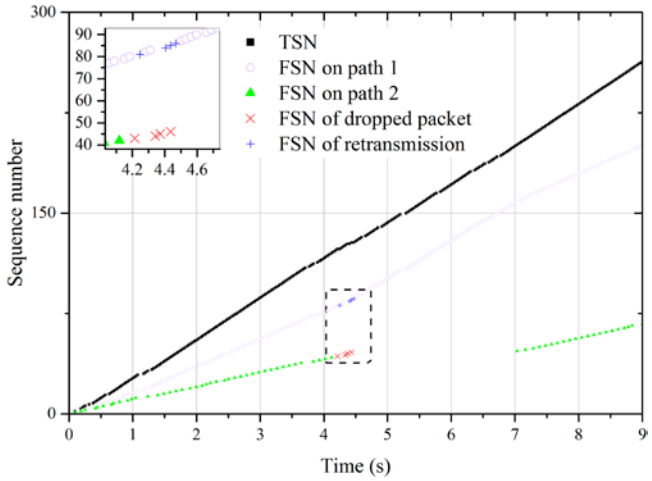


Fig. 8. Error control of MPMTP-AR. (Sender redistributes traffic when it detects packet loss.)

3) Packet overhead and packet duplication: Packet overhead (P_o) and duplicated packet are very important measure for a transport protocol. P_o is defined as follow:

$$P_o = \frac{n_{add} + n_{control}}{n_{data}} \tag{6}$$

where $n_{control}$ is the amount of control packets in bytes, i.e., feedback message and reception acknowledgement. n_{add} is the number of bytes of retransmitted data packets, and n_{data} is the amount of data in bytes. Duplicated packet is generated by unnecessary retransmission.

The performance of packet overhead and duplicated packet are studied by OMNeT++ simulation. The simulation topology is described as **Fig. 7**. **Table 2** displays packet overhead and duplicated packet with different packet loss rate (PLR). When PLR is 1 percent, packet overhead is 2.13%. When PLR increases, the packet overhead increases faster than PLR. This is mainly because MPMTP-AR generates more feedback message when path reliability becomes worse. **Table 2** also presents the ratio between duplicated packet and data packet. Almost the same phenomenon as packet overhead is observed. Because it becomes more difficult to accurately determine packet loss when path reliability becomes worse, which will lead to unnecessary retransmission.

Table 2. Packet overhead with different packet loss rate

Packet loss rate (%)	Packet overhead (%)	Duplicated packet (%)
1	2.130	0.03
2	3.475	0.08
3	4.985	0.15
4	6.649	0.24
5	8.511	0.35

4.2 Comparison with MPTCP

Both MPMTP-AR and MPTCP are designed to provide reliable data delivery service over multiple paths. This section compares MPMTP-AR with MPTCP in terms of throughput, packet loss rate, receiving buffer size, and reassembling delay. The proposed MPMTP-AR is fully implemented on the Linux platform. MPTCP is implemented according to http://multipath-tcp.org.

In a real network, a client uses MPMTP-AR and MPTCP to download a file from a server, respectively. According to maximum transfer unit, the file to be delivered is divided into equal length packets. In the case of MPTCP, client uses the Reno to manage congestion window and a shared receiving buffer to reorder packets. In the case of MPMTP-AR, client uses CSD to adjust congestion window and uses two-level receiving buffer to reorder packets. Bottleneck link of each path has 1 Mbps bandwidth, and the loss rate of each path obeys uniform distribution ranging from 0.75% to 1.25%. **Fig. 9** and **Fig. 10** show trace collected by MPMTP-AR and MPTCP. It is obvious that MPMTP-AR outperforms MPTCP in all tests.

In **Fig. 9 (a)**, the number of available paths changes from 1 to 5. As the number of available paths increases, the throughput becomes larger and the improvement becomes smaller. Both MPMTP-AR and MPTCP cannot maximize utilization of multiple paths. This is mainly because protocols have to pay for the management of multiple paths. A reasonable value of α contributes to the accurate evaluation of path quality. Experimental results show that MPMTP-AR obtains the highest throughput when $\alpha = 0.8$ regardless of the number of path.

In **Fig. 9 (b)**, as the number of available paths increases, the loss rate of MPMTP-AR is quite stable with no significant changes, and the loss rate of MPTCP becomes bigger. This is mainly caused by the relevance between available paths. Multiple paths provided by overlay network have a higher irrelevance than that provided by multiple IP address pairs. In the case of MPMTP-AR, the loss rate is the lowest when $\beta = 0.07$. As the value of β adjusts the bandwidth fraction for feedback in a moderate step, which timely reflects the variation trend of path condition and avoids being influenced by any stochastic factors.

MPMTP-AR sets $\alpha = 0.8$ and $\beta = 0.07$ to evaluate the performance of data reassembling. During the transmission, there are two paths between the sender and receiver. **Fig. 10 (a)** displays the occupancy probability of receiving buffer. The average occupancy of receiving buffer decreases from 19.7 packets of MPTCP to 16.6 packets of MPMTP-AR. As two-level receiving buffer mechanism helps the receiver to reduce the buffer size that is occupied by discontinuous blocks. For instance, the receiver receives two packets. One packet has a TSN of 1 and a FSN of 1 from path 1, and the other packet has a TSN of 3 and a FSN of 2 from path 2. In the case of only one receiving buffer, these two packets would occupy the space of three packets. Two-level receiving buffer mechanism only requires two packets space by storing these two packets in corresponding path-level receiving buffer.

Fig. 10 (b) displays the delay which is the duration from the packet enters the receiving buffer to the time when it can be delivered to the upper application. The average delay decreases from 98 ms of MPTCP to 85 ms of MPMTP-AR. As two-level receiving buffer accelerates packet loss detection and retransmission. In addition, both MPMTP-AR and MPTCP have multi-modal property in probability distribution. This is mainly caused by time interval of reassembling procedure. For example, in the case of MPMTP-AR, most of the durations obeys uniform distribution ranging from 10 ms to 160 ms. The time interval of reassembling procedure is about 5 ms which leads to many peaks.

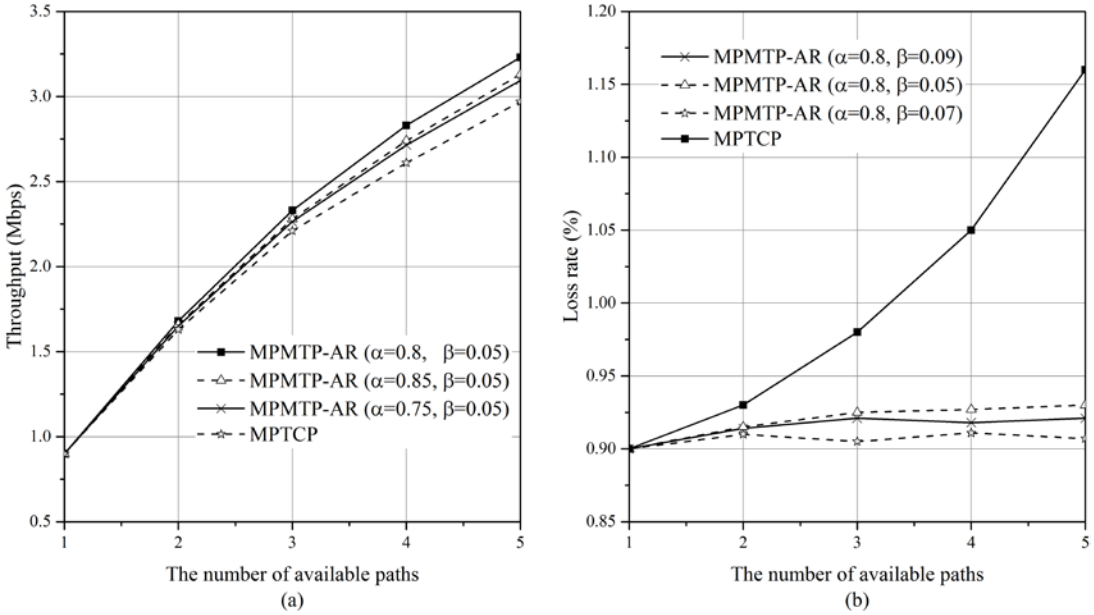


Fig. 9. Comparison of throughput and loss rate between MPMTP-AR and MPTCP. MPMTP-AR with different value of α and β. (a) Throughput (b) Loss rate.

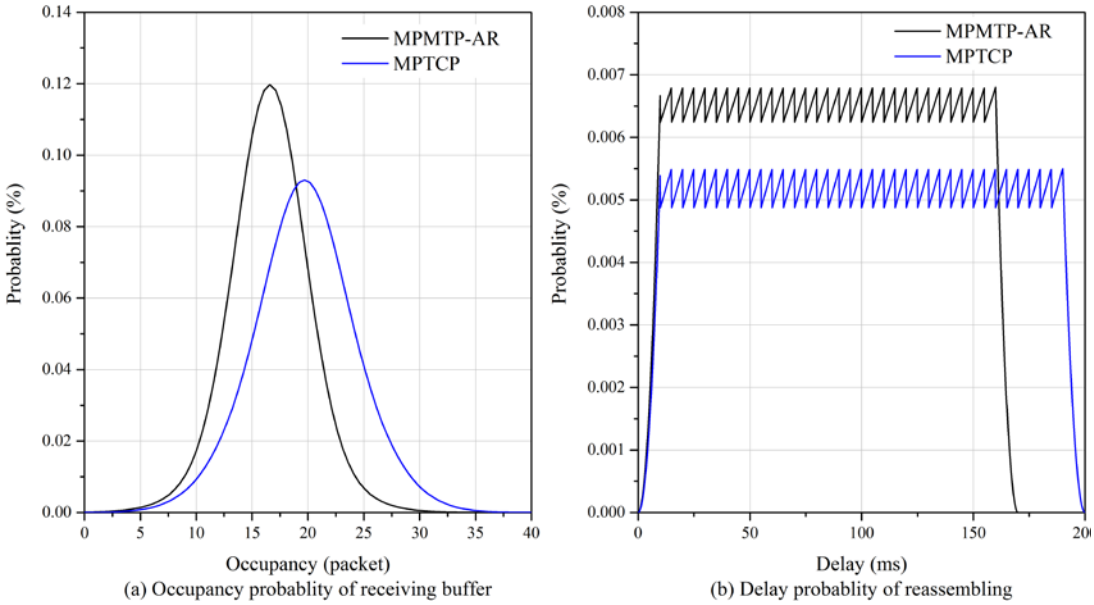


Fig. 10. Comparison of receiving buffer and reassembling delay between MPMTP-AR and MPTCP. (a) Occupancy of receiving buffer (b) Reassembling delay.

5. Conclusion

In this paper, we propose MPMTP-AR to provide reliable delivery service, which fills the gap of exploring multipath characteristics for overlay network. MPMTP-AR uses three-way handshaking mechanism to establish a virtual connection at application layer. Data packet is

transmitted from the sender to receiver by UDP-based application layer forwarding. Retransmission mechanism is used to recover the packet loss. Taking advantage of feedback, MPMTP-AR accurately evaluates path quality which serves as a reference to select an appropriate path for each packet. Independent sending buffer of each path alleviates negative impact on other paths. Two-level receiving buffer improves the efficiency of data reassembling. MPMTP-AR takes load weight of the congested path into consideration to deal with congestion. Experimental results demonstrate that MPMTP-AR outperforms MPTCP.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (No. 61401081), the Fundamental Research Funds for the Central Universities (No. N150404005), and Ministry of Education and China Mobile Scientific Research Fund (No. MCM20150103).

References

[1] S. Liu, W. Zhang, and L. Weimin, "A Framework of Multipath Transport System Based on Application-Level Relay (MPTS-AR)," *Internet Engineering Task Force, Internet-Draft*, January, work in Progress, 2016. Article (CrossRef Link)

[2] W. Zhang, W. Lei, S. Liu, and G. Li, "A general framework of multipath transport system based on application-level relay," *Computer Communications*, vol. 51, pp. 70–80, September, 2014. Article (CrossRef Link)

[3] J. Iyengar, C. Raiciu, S. Barre, M. J. Handley, and A. Ford, "Architectural Guidelines for Multipath TCP Development," *RFC 6182*, October, 2015. Article (CrossRef Link)

[4] R. R. Stewart, "Stream Control Transmission Protocol," *RFC 4960*, October, 2015. Article (CrossRef Link)

[5] A. Ford, C. Raiciu, M. J. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses," *RFC 6824*, October, 2015. Article (CrossRef Link)

[6] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE/ACM Transactions on Networking*, vol. 14, no. 5, pp. 951–964, October, 2006. Article (CrossRef Link)

[7] D. Zhou, W. Song, and M. Shi, "Goodput improvement for multipath TCP by congestion window adaptation in multi-radio devices," in *Proc. of IEEE 10th Consumer Communications and Networking Conference (CCNC)*, pp. 508–514, January, 2013. Article (CrossRef Link)

[8] Y. Cui, L. Wang, X. Wang, H. Wang, and Y. Wang, "FMTCP: A fountain code-based multipath transmission control protocol," *IEEE/ACM Transactions on Networking*, vol. 23, no. 2, pp. 465–478, April, 2015. Article (CrossRef Link)

[9] D. Zhou, W. Song, P. Wang, and W. Zhuang, "Multipath TCP for user cooperation in LTE networks," *IEEE Network*, vol. 29, no. 1, pp. 18–24, January, 2015. Article (CrossRef Link)

[10] T. D. Wallace and A. Shami, "A review of multihoming issues using the stream control transmission protocol," *IEEE Communications Surveys Tutorials*, vol. 14, no. 2, pp. 565–578, 2012. Article (CrossRef Link)

[11] J. R. Iyengar, P. D. Amer, and R. Stewart, "Receive buffer blocking in concurrent multipath transfer," in *Proc. of IEEE Conf. on Global Telecommunications*, vol. 1, pp.121-126, November, 2005. Article (CrossRef Link)

[12] T. Dreibholz, E. P. Rathgeb, I. Rungeler, R. Seggelmann, M. Tuxen, and R. R. Stewart, "Stream control transmission protocol: Past, current, and future standardization activities," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 82–88, April, 2011. Article (CrossRef Link)

[13] S. Mao, D. Bushmitch, S. Narayanan, and S. S. Panwar, "MRTP: a multiflow real-time transport protocol for ad hoc networks," *IEEE Transactions on Multimedia*, vol. 8, no. 2, pp. 356–369, April, 2006. Article (CrossRef Link)

[14] C. M. Huang and M. S. Lin, "Fast retransmission for concurrent multipath transfer (CMT) over vehicular networks," *IEEE Communications Letters*, vol. 15, no. 4, pp. 386–388, April, 2011. Article (CrossRef Link)

[15] C. Xu, T. Liu, J. Guan, H. Zhang, and G.-M. Muntean, "CMT-QA: Quality-aware adaptive concurrent multipath data transfer in heterogeneous wireless networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 11, pp. 2193–2205, November, 2013. Article (CrossRef Link)

[16] O. C. Kwon, Y. Go, Y. Park, and H. Song, "MPMTP: Multipath multimedia transport protocol using systematic raptor codes over wireless networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 9, pp.1903–1916, September, 2015. Article (CrossRef Link)

[17] S. Mao, S. S. Panwar, and Y. T. Hou, "On optimal partitioning of realtime traffic over multiple paths," in *Proc. of IEEE Conf. on Computer Communications (INFOCOM)*, vol. 4, pp. 2325–2336, March, 2005. Article (CrossRef Link)

[18] Y. Nebat and M. Sidi, "Resequencing considerations in parallel downloads," in *Proc. of IEEE Conf. on Computer Communications (INFOCOM)*, vol. 3, pp. 1326–1335, 2002.
Article (CrossRef Link)

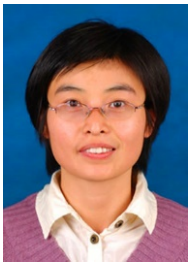
[19] Y. Li, S. S. Panwar, S. Mao, S. Burugupalli, and J. ha Lee, "A mobile ad hoc bio-sensor network," in *Proc. of IEEE Conf. on International Conference on Communications (ICC)*, vol. 2, pp. 1241–1245, May, 2005. Article (CrossRef Link)

[20] K. Zheng, X. Jiao, M. Liu, and Z. Li, "An analysis of resequencing delay of reliable transmission protocols over multipath," in *Proc. of IEEE Conf. on International Conference on Communications (ICC)*, pp. 1–5, May, 2010. Article (CrossRef Link)

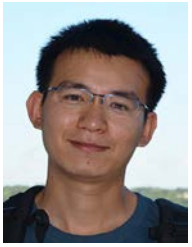
[21] A. Gurtov, T. Henderson, S. Floyd, and Y. Nishida, "The NewReno Modification to TCP's Fast Recovery Algorithm," *RFC 6582*, October, 2015. Article(CrossRefLink)

Shaowei Liu is a Ph.D. candidate with School of Computer Science and Engineering, Northeastern University, China. His current research interests include media transmission, network architecture, and network protocol.

Weimin Lei is a professor and a supervisor with Institute of Communication and Electronic Engineering, Northeastern University, China. His current research interests include protocols and services in IP multimedia system, multipath transport, overlay network, SDN and adhoc network.



Wei Zhang is a lecturer with Institute of Communication and Electronic Engineering, Northeastern University, China. Her current research interests include multimedia communication, protocols and services in next generation network, and network architecture of cloud computing



Xiaoshi Song is a lecturer with School of Computer Science and Engineering, Northeastern University, His research interests include stochastic geometry and its applications in large-scale wireless networks, multiuser information theory, wireless communication and multipath transport