International Journal of Hybrid Information Technology
Vol. 10, No.4 (2017), pp. 1-10
http://dx.doi.org/10.14257/ijhit.2017.10.4.01

Goodput Improvement for MPTCP Based on Controlling the Difference of Delay

Yang Tao[1] and *Zhijun Yan[2]

Institute of Communication and Information Engineering, The Chong Qing University of Posts and Telecommunication, Chong Qing, China
[1]goforbeauty@163.com,[2]928351320@qq.com

Abstract

Due to the out of order problem, the goodput is usually far lower than the aggregated throughput. How to improve the goodput is a focus of researches. However, lots of them hardly consider the influence of the congestion window on out of order data. So, based on that, A goodput improvement solution based on controlling delay difference for the MPTCP (referred to as CDD-MPTCP) is put forward. By dynamically adjusting congestion window of each sub-streams, it can reduce end to end path delay, thus reducing out of order data. And, we also propose a data scheduling strategy based on the congestion degree of sub-streams to reduce the congestion during the transmission process. The NS-2 simulation results show that CDD-MPTCP can effectively decrease the amount of out of order data received at the receiver without loss of aggregate throughput.

Keywords*: goodput, CDD-MPTCP, delay difference, out of order*

1. Introduction

In recent years, as the development of Internet access technologies, more and more mobile devices support two or more network access ways. Meanwhile, users are able to access to the Internet by different ways due to heterogeneous networks which is consist of integration of different networks. In order to fully utilize multi-network access and satisfy users' demand for bandwidth, the multi-path parallel transmission technologies become a research focus. Like the pTCP [1], mTCP [2] and the multi-path TCP protocol (MPTCP) [3-5] which is proposed by IETE in 2011.

Early multi-path parallel transmission schemes such as the pTCP and mTCP, for the purpose of multi-path parallel transmission, mostly modify the traditional TCP protocol to aggregate available bandwidth of multiple paths. However, due to numerous modifications to TCP, those protocols possess not good backward compatibility and cannot coexist with traditional TCP protocols in the network, so they cannot be broadly applied.

With network technologies' development, network applications need more bandwidth. Some researchers attempt to improve the throughput stability by using multiple TCP transmission. In [6], the TCP data stream is divided into several streams at the IP layer, and then distributed to multiple paths parallel transmission. Real-time video stream transmission is improved. In [7], aggregating available bandwidth of multiple paths is proposed to improve transmission performance of video streams. The above solutions, most of them are put forward based on the assumption that aggregating large bandwidth can be able to improve the performance of video transmission. However, in the practical application environment, the receiver will receive lots of out of order packets, which can severely decrease the amount of valid data from the transport layer to the application layer.

Compared with the above multi-path parallel transmission schemes, MPTCP has better compatibility than traditional TCP protocols. So it can support existing

ISSN: 1738-9968 IJHIT

middleware(NATs, firewall, proxy, *etc*.), which can reduce the costs to upgrade and is possible for large-scale commercial deployment. But in the MPTCP protocol, the effective throughout is also lower than the theoretical throughput. In recent years, researches on how to improve the MPTCP protocol goodput have gradually become focus. In [8], appropriately choosing packet retransmission path scheme is proposed to improve goodput. However, this solution only considers the retransmission strategy when the receiving buffer is full, and halve the congestion window directly will sharply decline data throughput. In [9], congestion losses and random packet losses is distinguished, and different retransmission strategy is adopted to avoid network random packet losses which will result in unnecessary throughput degradation. However, the algorithm can't improve normal transmission throughput when packet losses do not happen. In [10] and [11], authors put forward a scheme that uses network coding scheme and recovers packets at the receiver by sending redundant encoding data packets, thus increasing goodput. However, they all need communication nodes to support encoding operation, resulting in the complexity of network transmissions. Therefore, an improved MPTCP scheme is necessary to be proposed, which can improve the MPTCP protocol throughput performance in practice and preserve MPTCP compatibility at the same time.

1. The Goodput Analysis

MPTCP can allow data to be transferred in parallel on multiple paths by integrating bandwidth resources from different paths together and unified scheduling management, thus increasing the overall connection throughput of MPTCP. However, in heterogeneous network, there are huge difference on parameters, such as round-trip delay, packet loss rate *etc*. So packets can't follow the order in sender sequentially to reach the receiver. This is the phenomenon so-called data out of order. On the other hand, as stipulated in MPTCP protocol, data blocks arrived in a sequence only can be delivered to be processed from the transport layer to the application layer. But, the received disordered data will be temporarily stored in cache in receiver, and wait to be delivered together with the arrived data packet with smaller transmission serial number(TSN). This will cause goodput in MPTCP much less than expected when network parameters are quite different between different paths.

In the paper, MPTCP goodput is defined as the packet number submitted by the transport layer to the application layer per unit time. In order to identify the factors that affect the goodput of MPTCP. We study the simplest MPTCP connection with only two sub-streams. Parameters are described below:

τ_i : The time interval to send packets from the sender, where i is the router number (i=1,2).

D_i : Transmission delay on the path i (the time when packets are sent from the sender to the receiver) and $D_1 < D_2$.

$\Delta D = | D_2 - D_1 |$: the transmission delay of two paths.

T : The time when it takes to receive the data block in order.

S : The total amount of data received in time T .

$G = \frac{S}{T}$: The MPTCP goodput.

We assume that N packets with continuous transmission series number are waiting to be sent, which is called a sequential unit. N-1 of packets are transmitted over the path 1, and another packet transmitted over the path 2.

As shown in Figure 1 and Figure 2, two special conditions are taken into account. The sequential unit consists of 4 consecutive TSN (1, 2, 3 and 4) packets. And we assume that

the data packet 1 and packet 2 are sent separately to the path 1 and path 2 at the same time.

(1) When $\Delta D > \tau_1$ and the packet which is sent through the path 2 arrives at the receiver, the data block will be delivered to the application layer. So

$$G = \frac{S}{T} = \frac{\tau_2/\tau_1 + 1}{\Delta D} \quad (1)$$

(2) When $\Delta D < \tau_1$, the total time of N data packets delivered in order is what it takes to the receiver receives N-1 data packets through the path 1. So

$$G = \frac{S}{T} = \frac{\tau_2/\tau_1 + 1}{\tau_2} \quad (2)$$

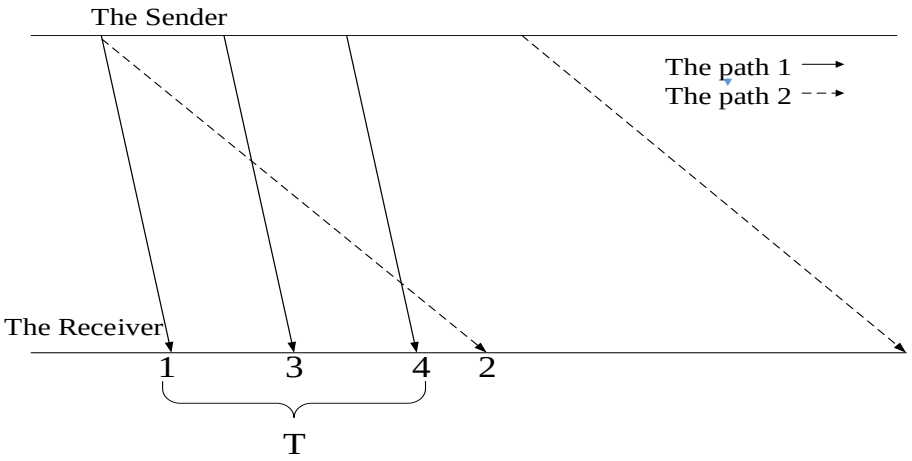

Figure 1. The General MPTCP Transmission Way

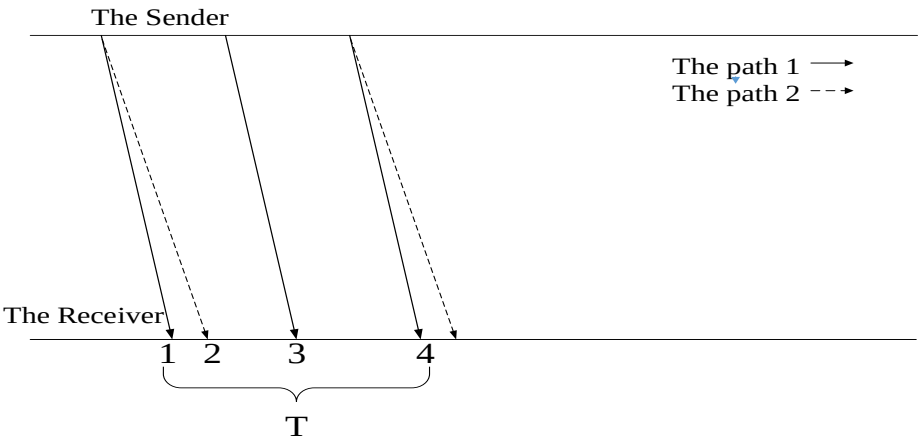

Figure 2. The Best MPTCP Transmission Way

The above analysis shows that MPTCP goodput is inversely proportional to end to end delay inequality of different paths. The more the transmission delay inequality is, the longer it will take the receiver to receive data blocks with contiguous TSN and deliver at the application layer. The goodput of MPTCP connection is also much lower.

2. MPTCP Goodput Improvement Based on Delay Control

In order to reduce the data out of sequence, and improve the MPTCP goodput, we need decrease the delay inequality as far as possible. In this section, we propose MPTCP effective improvement algorithm based on controlling the difference of delay (CDD-MPTCP).

2.1. Round-Trip Delay Estimation

For accurately estimating the MPTCP transmission forward delay, this strategy uses the forward delay prediction algorithm based on time stamp, which is similar with RTT forecast in the datagram congestion control protocol (DCCP)[12]

The specific algorithm is as follows: add two fields of the transmission time T_s and the reception time T_r into the MPTCP packet header. And add the response time T_a into the SACK packet header, as shown in Figure 2. T_s is the time which the sender spends to send the packet P_i . T_r is what the receiver spends to receive the packet P_i . T_a is the time that it takes the receiver to receives the packet P_i before SACK of the packet is sent. The forward delay D of links can be calculated:

$$D = T_r - T_e - T_s \tag{3}$$

In order to ensure smoothness and avoid packet loss leading to substantial fluctuations of the estimated value, combined with previous sample values, the formula for estimating forward transmission delay D_i is:

$$D_i = (1-\alpha)D_{new} + \alpha D_{i-1} \tag{4}$$

Where α is a scaling factor, in this paper, it equals the empirical value 0.75.

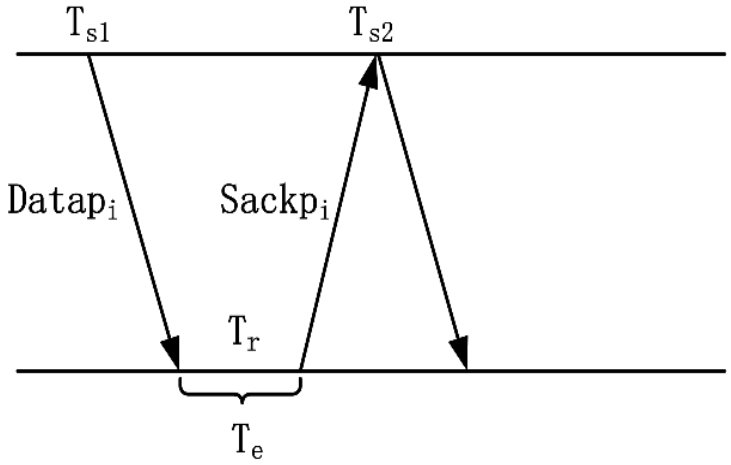

Figure 3. The Forward Delay Prediction Algorithm Based on Time Tamp

2.2. Congestion Window Adjustment Strategy

The MPTCP protocol uses the coupling congestion control strategy which is the additive increase and multiplicative decrease (AIMD). Each MPTCP sub-stream has an independent congestion window $cwnd\_i$. Whenever sub-streams receive new ACK messages which return from the receiver, $cwnd\_i$ of corresponding sub-streams increases linearly. Whenever sub-streams continuously receive three repeated ACK message, $cwnd\_i$ of corresponding sub-streams decrease proportionally. Meanwhile,

the congestion control strategy require that all sub-streams congestion windows has association. The amplitude of increasing or decreasing the congestion window of each sub-stream is determined by the size of other sub-streams' congestion windows. So, MPTCP connection will not consume too many resources and the fairness of traditional MPTCP connections can be guaranteed. However, after the strategy is being implemented for a while, $cwnd\_i$ of each sub-stream will be vastly different, which will cause huge difference of each sub-stream's transmission delay. This can lead to serious disorderly data and limit the MPTCP goodput.

In order to increase the goodput of MPTCP, this paper improves the MPTCP congestion control strategy by controlling the congestions windows' size to reduce the maximum delay difference of different sub-streams. This paper adds new control strategy based on the MPTCP original congestion control strategy and retains the MPTCP coupling congestion control which is additive increase and multiplicative decrease. Define the delay coefficient θ as the ratio of the sub-streams' maximum and minimum transmission delay. That is:

$$\theta = D_{\max} / D_{\min} \tag{5}$$

Just like round-trip delay, in order to ensure the smoothness of the delay factor, combined with previous sample values, the formula that calculates delay coefficient is:

$$Q_i = (1-\beta)\theta_{new} + \beta\theta_{i-1} \tag{6}$$

When θ exceeds a certain value $Q_{\max}$, it proves that the MPTCP delay difference of different sub-streams is too large. It may lead to the buffer congestion at the receiver, thus decreasing the MPTCP goodput. At this moment, even if the sender have not received repeated ACK messages, it will also reduce the congestion window of the sub-stream which has the maximum transmission delay. The specific process and the pseudo code in this strategy follows:

(a) When $\theta > \theta_{\max}$, the strategy starts.

(b) Find the sub-stream P_i with the maximum transmission delay, decrease the congestion window $cwnd\_i$ of P_i to:

$$cwnd_{new} = \frac{1}{2}\left(\frac{cwnd_{old}}{q} + cwnd_{old}\right) \tag{7}$$

(c) Compare the decreased congestion window $cwnd\_i$ and the slow-start threshold $SSthresh\_i$ of this sub-stream. If $cwnd\_i < SSthresh\_i$, $SSthresh\_i = cwnd\_i$.

The adjusted congestion window is the average of the original congestion window and the original congestion window divided by the delay factor θ. That is to avoid the congestion window to decrease too quickly and lead to overall MPTCP throughput decrease significantly. When the value of the adjusted $cwnd\_i$ is too small, the formula $SSthresh\_i = cwnd\_i$ is going to ensure the sub-stream remaining in the congestion avoidance period after the strategy is over.

The algorithm 1：The strategy of adjusting congestion windows based on delay cofficient

// Update delay coefficient according to the newest transmission delay.

1: $q_{new} = D_{\max} / D_{\min}$

2: $\theta \leftarrow (1-\beta)\theta_{new} + \beta\theta$

3: **if** $q > q_{\max}$ **then**

// Search the sub-stream with maximum transmission delay

4: $i = \arg\max(\text{The substreams' transmission delay})$

//Reduce the number congestion window of i

5: $cwnd_i \neg cwnd_i / q$

6: **if** $cwnd\_i < SSthresh\_i$ **then**

7: $SSthresh\_i = cwnd\_i$

8: **end if**

9: **end if**

10: **else if** $q_0 < q < q_{\max}$ **then**

11: $i = \arg\max(\text{The substreams' transmission delay})$

12: $cwnd_i \neg \left(cwnd_i / q + cwnd_i\right) / 2$

13: **if** $cwnd\_i < SSthresh\_i$ **then**

14: $SSthresh\_i = cwnd\_i$

15: **end if**

16:**end for**

In this algorithm, the choice of $q_{\max}$ has a significant impact on the performance of the congestion control strategy. If the value of $q_{\max}$ is too big, it will lead to serious end to end delay and aggravate data out of order. If the value of $q_{\max}$ is too small, it will make the congestion window of the sub-stream change too quickly within a short time and increase the link jitter. Serious data out of order will result in severe cache congestion at the receiver and it will result in deterioration of overall MPTCP throughput sharply. So, we select the cache size as the reference standard at the receiver and puts the number of packets which the cache can accommodate at the receiver as N_{Buffer}. $q_{\max}$ is calculated as followed:

$$\frac{d_{\max} - d_{\min}}{\tau} + 1 < N_{Buffer} < \frac{d_{\max} - d_{\min}}{\tau} + 2 \tag{8}$$

$$\theta_{\max} = \frac{d_{\max}}{d_{\min}} \tag{9}$$

$$\frac{(N-2)\tau}{d_{\min}} + 1 < \theta_{\max} < \frac{(N-1)\tau}{d_{\min}} + 1 \tag{10}$$

2.3. The Data Scheduling Strategy

The proposed congestion control strategy above controls the MPTCP transmission delay of each sub-stream within a smaller range and theoretically, the serious cache congestion will not exist at the receiver. However, in order to make packets arrive at the receiver as soon as possible, this paper propose a data scheduling algorithm based on congestion level of sub-streams. Define congestion coefficient γ of sub-streams as the ratio of the number of packets which have been already sent at the sender but not yet receive the ACK response accounting for congestion windows at the receiver. That is:

$$g = \frac{unacknowledged\_packets}{cwnd} \tag{11}$$

The strategy description and the pseudo code are as follows:

(a) Judge whether the next packet to be sent is retransmitted packet. If it is the retransmitted packet, remove the original transmission sub-stream of the packet from the collection of alternate sub-streams.

(b) Traverse alternate sub-streams collection, calculate the congestion coefficient of sub-streams, select the sub-stream with minimum congestion coefficient to send the packet.

(c) If there are multiple sub-streams with the same congestion coefficient, select the path with minimum forward transmission delay D_i

The algorithm 2：The data scheduling strategy based on congestion coefficient

// Initialize parameters
1: $Selected\_subflow = NULL$
2: $\gamma_{\min} = \infty$
3: $D_0 = 0$
4: **for** each alternate substream P_i **do**// Traverse the collection of the alternate substream
5: **if**(the next packet is retransmitted packets) **and** (the packet was previously transmitted over the stream P_i) **then**
6: continue
7: **end if**
8: $g_i = \frac{unacknowledged\_packets\_i}{cwnd\_i}$
//Search the substream with minimum congestion coefficient
9: **if**($\gamma_i < \gamma_{\min}$)**then**
10: $\gamma_{\min} = \gamma_i$
11: $Selected\_subflow = i$
12: $D_0 = D_i$
13: **end if**
14: **else if** ($\gamma_i = \gamma_{\min}$) and ($D_i < D_0$)**then**
15: $Selected\_subflow = i$
16: $D_0 = D_i$
17: **end if**
18: **end for**

3. Experimental Results and Analysis

In order to verify the validity of the algorithm presented in this paper, we use NS-2 to simulate the CDD-MPTCP algorithm. The sender and receiver are both equipped with two network interfaces, and connect to a typical heterogeneous wireless networks with multiple transmission channels. Use two paths for parallel transmission. The average bandwidth is 4.0Mbit/s on path 1 while it is 2.0Mbit/s on path 2. The cache size is 1.16Mbit/s(100MSS) at the receiver. For assessing the long term throughput as well as the goodput performance, we use the FTP protocol to transmit a file large enough for simulating the data transmission process. According to the congestion window control strategy, update the congestion window $cwnd$ per 0.5s. Whenever the receiver receives 100 packets with continuous TSN, record a valid throughput value and the value of current receiving buffer. This paper uses the receiving buffer to represent the storage size required for the receiver get K packets with continuous TSN, which is proportional to the level of data out of order received by the receiver through multipath parallel transmission. Simulation results are shown in Figure 3 and Figure 4, Figure 3 is for standard MPTCP goodput compared with this CDD-MPTCP, Figure 4 is standard MPTCP receiving buffer size compared to CDD-MPTCP.

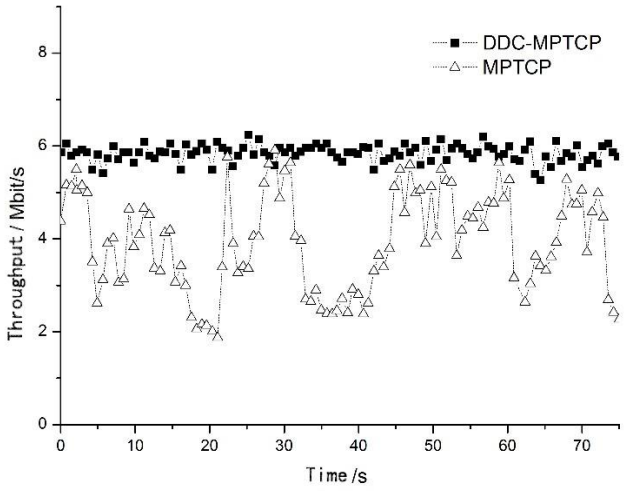

Figure 4. Throughput Comparison

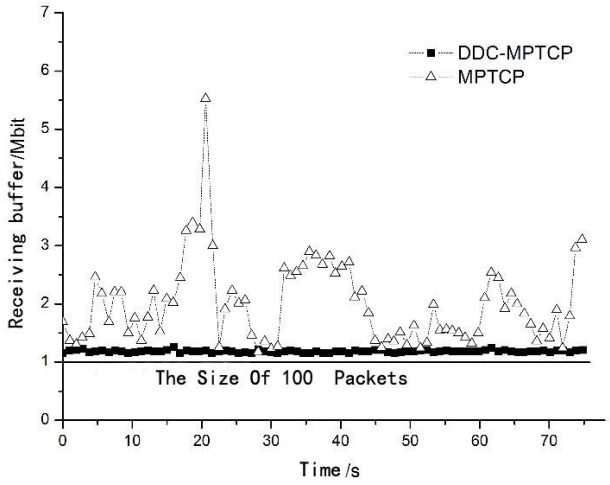

Figure 4. Receiving Buffer Size Comparison

The Figure 3 shows that the CDD-MPTCP goodput is almost equal to the aggregate bandwidth 6Mbit/s which is the maximum goodput of the multipath connection. However, the MPTCP aggregate throughput is very low, which is just equal to the throughput on the slow path at some time. By contrast, CDD-MPTCP can be able to reduce the delay difference between two paths and dramatically improve goodput and throughput stability.

From the Figure 4, CDD-MPTCP receiving buffer size is close to the number of 100 consecutive TSN packets (as shown in the diagram black line), while MPTCP receiving buffer size is much larger than the CDD-MPTCP receiving buffer size. Therefore, compared with the MPTCP, multipath parallel transmission by CDD-MPTCP can greatly reduce the data out of order. CDD-MPTCP can reduce the number of packets out of order without loss of aggregate throughput.

4. Conclusion

We propose the MPTCP goodput improvement algorithm CDD-MPTCP based on delay difference. Measure multiple each path' delay precisely and adjust the path delay by dynamically adjusting the congestion window of each sub-stream, so as to reduce the delay difference between multiple paths. In addition, we also propose the data scheduling strategy based on the congestion degree of sub-streams. It distributes packets to sub-streams with low congestion degree. By this way, it makes packets arrive at the receiver as soon as possible and reduces the possibility of the congestion during the transmission process. Simulation results show that CDD-MPTCP can reduce the number of out of order packets received by the receiver, thereby increasing aggregate goodput of multi-path connectivity.

References

[1] D. Ganz, S. Leschke and H. D. Doran, "Improving EtherCAT master-slave synchronization precision using PTCP embedded in EtherCAT frames: A proof-of-concept", Factory Communication Systems (WFCS), 2015 IEEE World Conference on, Palma de Mallorca, **(2015)**.

[2] M. Zhang, J. Lai and A. K. MURTHY, "A Transport Layer Approach for Improving End-to-End Performance and Robustness Using Redundant Paths", USENIX Annual Technical Conference, **(2004)**, pp. 99-112.

[3] A. Ford, C. Raiciu, M. Handley and J. Iyengar, "RFC 6182, Architectural guide lines for multipath TCP", IETF, **(2011)**.

[4] M. Bagnulo, "RFC 6181, Threat analysis for TCP extensions for multipath operation with multiple addresses", IETF, **(2011)**.
[5] C. Raiciu, M. Handley and D. Wischik, "RFC 6356, Coupled congestion control for multipath transport protocols", IETF, **(2011)**.
[6] M. F. Tsai, N. Chilamkurt, J. H. Park and C.-K. Shieh, "Multi-path transmission control scheme combining bandwidth aggregation and packet scheduling for real-time streaming in multi-path environment", IET Communications, vol. 4, no. 8, **(2010)**, pp. 937-945.
[7] B. Wang, W. Wei, Z. Guo and D. Towsley, "Multipath live streaming via TCP: scheme, performance and benefits", ACM Transactions on Multimedia Computing, Communications and Applications, vol. 5, no. 3, **(2009)**.
[8] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene and O. Bonaventure, "How hard can it be? Designing and implementing a deployable multipath TCP", NSDI, **(2012)**, pp. 29-29.
[9] V. Sharma, K. Kar, K. K. Ramakrishnan and S. Kalyanaraman, "A Transport Protocol to Exploit Multipath Diversity in Wireless Networks", IEEE/ACM Transactions on Networking, IEEE, **(2012)**, pp. 1024-1039.
[10] M. Li, A. Lukyanenko and Y. Cui, "Network coding based multipath TCP", Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on, Orlando, FL, **(2012)**.
[11] O. C. Kwon and H. Song, "Cross-layer optimized multipath video streaming over heterogeneous wireless networks", APSIPA, 2013 Asia-Pacific, Kaohsiung, **(2013)**.
[12] E. Kohler, M. Handley and S. Floyd, "RFC 4340, Datagram Congestion Control Protocol", IETF, **(2006)**.

Authors

Yang Tao, he was born in 1964, he was received the PhD degree in Chongqing University China. And He was received the Postdoctoral degree in the Institute of Computing Technology, Chinese Academy. Now, he is working as a Professor in the college of communication and information engineering, in the Chongqing University of Posts and Telecommunication. His main research includes the theory of communication networks, network application management.

Zhijun Yan, he was born in 1991, is now a master candidate of the college of communication and information engineering, in the Chong Qing University of Posts and Telecommunication, China. His main research includes the network application management, the theory of communication networks.

