Multipath TCP with Multiple ACKs for Heterogeneous Communication Links

Takumi Kurosaka and Masaki Bandai
Dept. of Information and Communication Sciences, Sophia University
4-7 Yombancho, Chiyoda, Tokyo, 102-0071, Japan
Email: {t_kurosaka, bandai}@sophia.ac.jp

I. INTRODUCTION

Multipath TCP (MPTCP) [1] has a great potential to improve end-to-end throughput by aggregating bandwidth of multiple communication paths. However, paths have different characteristics such as round trip time (RTT) [2]. In such a situation, data packets from paths frequently arrive at a receiver in a wrong order. This is out-of-order packet problem, which incurs throughput degradation of MPTCP. When receiving an out-of-order data packet, a receiver keeps the packet in its receive buffer until an in-order data packet arrives. If a receive buffer is filled up with out-of-order data packets, the receiver advertises its zero-window to the sender. When receiving the ACK with zero-window, the sender is blocked sending the next packet until receiving an ACK with nonzero-window. The more characteristics are different between paths, the longer the sender is blocked sending packets, which incurs severe throughput degradation.

In this paper, we propose an extended version of MPTCP with heterogeneous communication links. The proposed method solves the out-of-order packet problem and improves end-to-end throughput performance.

II. PROPOSED METHOD

The proposed method defines two types of ACK: Regular ACK and Extended ACK. When a receiver receives an in-order data packet, the receiver sends a Regular ACK to the original subflow through which the data is transmitted, and Extended ACKs to all the other subflows. The format of Extended ACK is the same that of Regular ACK. Extended ACK is used for updating Data Sequence Number (DSN) and advertising window size for a sender. On the other hand, Extended ACK is not used for measuring RTT for calculating retransmission timeout value, and for judging whether it is duplicate ACK or not by the sender. Since Extended ACKs are much smaller than data packets, the proposed method has a small overhead.

When the Extended ACK arrives at the sender earlier than the Regular ACK, the Extended ACK decreases the blocking duration of sending, and makes the sender restart sending next packets more quickly.

Algorithm 1 shows the algorithm of the proposed method. The algorithm shows how a receiver operates when it receives a data packet. $SSN_k$ and $DSN$ are the values of the sequence number input in the received packet. $N$ is the number of subflows. $k$ is the value of subflow index. A receiver keeps $nextSSN_k$ and $nextDSN$, respectively. If $SSN_k = nextSSN_k$ and $DSN = nextDSN$, the received packet is in-order data packet. Otherwise, the receiver realizes that the received packet is out-of-order one. The receiver also keeps a counter, $Count_k$, for counting the number of receiving out-of-order packet at Subflow$_k$. The counter is reset to zero when the receiver receives an in-order packet. When receiving an in-order packet, the receiver delivers it to application directly and, updates $nextDSN$. Therefore, if there is a packet whose $DSN$ is equal to the updated $nextDSN$ in a receive buffer, the packet becomes in-order packet and delivered to application directly. In this case, the receiver updates $nextDSN$ again. As a result, the receiver has a non-zero space in the receive buffer, and the receiver can advertise non-zero-window. In this case, $Count_k$ becomes zero, the receiver sends Regular ACK to Subflow$_k$ and Extended ACKs.
simultaneously to all the other Subflows. Since Extended ACK at includes awnd (advertise window), Extended ACK always advertises nonzero-window. If there is a Subflow whose RTT is shorter than Subflow1, the Extended ACK sent to the Subflow arrives earlier than Regular ACK. A sender which is blocked sending restarts sending next packets earlier when the sender receives the Extended ACK. In this way, the proposed method solves out-of-order packet problem.

Figure 1 illustrates the operation example of the proposed method. In Figure 1, there are two subflows Subflow1 and Subflow2 in a connection between Sender and Receiver. Sender transmits a data packet P1 whose DSN is 1. After that, Sender transmits three data packets P2 to P4 (DSNs are 2 to 4). We assume that the RTT of Subflow1 (RTT1) is shorter than that of Subflow2 (RTT2). In this case, out-of-order packet problem arises because Receiver receives P2 to P4 earlier than P1. At T_zero, we assume that the receive buffer at Receiver is filled up with P2 to P4. In this case, Sender is advertised zero-window at T_block and blocked sending the next packet. When receiving the in-order packet (P1) at T, the out-of-order situation is resolved. Receiver sends Regular ACK to Subflow2 and Extended ACK to Subflow1 simultaneously. Regular ACK arrives at Sender at T2. Extended ACK arrives at Sender at T1. In this case, T1 is earlier than T2 because we assume RTT1 < RTT2. We define the time duration between T_block and T2 as D, and between T_block and T1 as D_prop. D also means the blocking duration of sending in conventional MPTCP. D_prop is the blocking duration of sending in the proposed method. Consequently, as shown in Figure 1, the proposed method shortens the blocking duration to D_prop.

III. PERFORMANCE EVALUATION

We evaluate the performance of the proposed method via computer simulations on NS-3. We use [3] as conventional method for comparison, which is called conventional MPTCP.

We consider there are two subflows Subflow1 and Subflow2 between a sender and a receiver. We fix the bandwidth of each subflow to 5 Mbps. RTT1 is fixed to 50 milliseconds in every simulation. The maximum segment size is 1,400 Byte. The size of receive buffer at the receiver is 64 KByte. The sender transmits the data whose size is 10 MByte.

We evaluate the throughput performance. Throughput is the aggregated value of all subflows. Figure 2 shows the throughput performance for various RTT2s. The dashed line in Figure 2 is the throughput performance of TCP using only Subflow1. We observe that when RTT1 < RTT2, the proposed method achieves higher throughput performance than conventional MPTCP. This is because the blocking duration caused by out-of-order packet problem can be effectively reduced by adding Extended ACK in the proposed method. Moreover, when RTT2 is 400 milliseconds, the proposed method improves the throughput performance up to 72.7%. On the other hand, when RTT1 = RTT2, both methods have almost the same throughput performance. This result leads that the overhead of the transmission of Extended ACKs in the proposed method does not affect the throughput performance in this situation. In other words, the proposed method can obtain almost the same throughput performance as the conventional MPTCP in the ideal condition for MPTCP.

IV. CONCLUSION

In this paper, we have proposed an extended version of MPTCP with heterogeneous communication links to solve the out-of-order packet problem. We have confirmed that the proposed method can improve the end-to-end throughput performance up to 72.7% when RTTs of subflows are different. In addition, we have also confirmed that the proposed method does not affect the end-to-end throughput performance in the situation when RTTs of subflows are almost the same.

REFERENCES