Receive Buffer based Path Management for MPTCP in Heterogeneous Networks

Jinhwan Kim
Department of Electrical and Electronic Engineering,
Yonsei University
Seoul, Korea
11591@yonsei.ac.kr

Bong-Hwan Oh, Jaiyong Lee
Department of Electrical and Electronic Engineering,
Yonsei University
Seoul, Korea
{crusader27, jyl}@yonsei.ac.kr

***Abstract*— Multipath Transport Control Protocol (MPTCP) is a promising solution to support simultaneous transmission of packets through multiple paths. With bounded receive buffer and heterogeneous networks, MPTCP could suffer from degradation of its performances, undermining the advantage from multiple path transmission. In this paper, we propose a simple and effective method to manage the multiple paths of MPTCP, called Receive Buffer based Path Management (RBPM), which operates based on the available receive buffer size and dissimilar characteristics of multiple paths. The RBPM scheme estimates out-of-ordered packets, predicts the buffer blocking problem in advance and stops transferring over bad performance paths. We implement RBPM in the Linux kernel and evaluate its performances over a virtual network environment using NS-3 Direct Code Execution. The results show that the proposed scheme significantly improves the throughput and network utilization with bounded receive buffer in heterogeneous networks.**

Keywords—MPTCP; Path management; buffer blocking problem;

I. INTRODUCTION

In today's Internet, datacenters have a number of paths between compute and storage nodes, current portable devices are equipped with multiple radio interfaces, and the number of hosts that are multi-homed has been increased. But current dominant protocols cannot transfer data through these multiple paths concurrently. These protocols select only one best path and send data through it. Therefore, they cannot utilize the existing redundant paths.

As one promising solution, Multipath Transport Control Protocol (MPTCP) [1], which is standardized by Internet Engineering Task Force (IETF), is a TCP extension version that allows to transfer data packets over multiple paths simultaneously and utilize multiple interfaces.

One of major concern when using MPTCP is the buffer blocking problem. A buffer blocking problem occurs because of shared finite receive buffer and dissimilar characteristics among multiple paths. Depending on Internet Service Provider (ISP), communication technic, background traffic and etc., multiple paths have different properties. Although the packets are transmitted later through fast paths than packets through slow paths, they can arrive at destination earlier. Then out-of-ordered packets fill receive buffer. If receive buffer is not large enough, these out-of-ordered packets fill all of receive buffer and source cannot transfer packets until packets transmitted through slow paths are arrived to destination. This phenomenon is called the buffer blocking problem.

There are some of research which studied the buffer blocking problem [2]-[5]. But these works showed results only when network topology has two paths between source and destination. On the contrary, there are some of works that consider more than two paths to maximize their performances of MPTCP [6]-[8]. But they assumed infinite receive buffer, so the buffer blocking problem doesn't occur in this case.

In this paper, we propose a new and simple path management method for MPTCP which is called Receive Buffer based Path Management (RBPM). RBPM considers that more than 2 paths are available between the source and destination and receive buffer is not infinite. The proposed scheme predicts buffer blocking phenomenon and stops transferring through bad performance paths to avoid buffer blocking problem in advance and maximize performances of MPTCP.

The remainder of this paper is organized as follows. In Section II, we specify the overall description of the proposed RBPM scheme. Section III presents the performance measurements of the proposed scheme and compare to performance of original MPTCP. Finally, conclusions are presented in section IV.

II. RECEIVE BUFFER BASED PATH SELECTION(RBPS)

In this section, we describe our proposed scheme, RBPM. If there is enough large receive buffer and no buffer blocking problem, MPTCP with RBPM operates identically to original MPTCP. To make MPTCP with RBPM operate as original MPTCP, the size of shared receive buffer is recommended [1] as

$$Buffer = 2 * \sum_{i=1}^{N} BW_i * RTT_{max} \qquad (1)$$

where $Buffer$ represents the required buffer size, N is the number of subflows, BW_i indicates the available bandwidth of the ith subflow, RTT_{max} refers to the largest RTT across all subflows. But according to dissimilar characteristics of multiple paths, this recommended buffer size can be too large. In real networks, every destination cannot afford to have the enough

Algorithm 1 Operation of RBPM

```
for all subflows j do
    if cwnd in best performance subflow is not filled
        transfer though best performance subflow
    else
recalculate:
        calculate B_active
        if B_active > available buffer then
            make worst performance path in R(N) inactive
            goto recalculate;
        else
            if available buffer > 2*B_active
                use all the subflows
            end if
            transfer though subflow j
        end if
    end if
end for
```

required receive buffer size for MPTCP. The original MPTCP always tries to use all available paths regardless of characteristics of the paths and available buffer size. But, due to this feature, original MPTCP could experience buffer blocking problem. And this buffer blocking problem causes significant degradation of throughput and underutilization of the network capacity. Thus, we propose RBPM which don't deteriorate the performances in case there is no enough receive buffer and dissimilar characteristics of multiple paths.

RBPM predicts buffer blocking in advance to prevent experiencing performance degradation of MPTCP. When required buffer exceeds the available receive buffer at the destination, a buffer blocking occurs. To predict buffer blocking phenomenon, the number of out-of-ordered packets and the required buffer size have to be estimated. We assume that all subflows are sorted in ascending order according to the RTT for simplicity. A method to estimate required buffer size is referenced from [2]. In [2], they define the parameter $L_{i,j}$ that is the estimated number of out-of-ordered packets in subflow i during RTT of subflow j.

$$L_{i,j} = \text{Floor}\left(\frac{RTT_j}{RTT_i}\right) - 1, \quad when\ \frac{RTT_j}{RTT_i} \geq 2$$

$$L_{i,j} = 0, \quad when\ \frac{RTT_j}{RTT_i} < 2 \qquad (2)$$

when $j > i$. For simple estimation, the number of transmitted packets during RTT is assumed to be the same as outstanding packets. Because we estimate out-of-ordered packets every packet transmission, the number of outstanding packets become equal to congestion window (*cwnd*) of subflows during RTT. Accordingly, $L_{i,j}$ represents how many times subflow i can transfer data packets as much as its *cwnd*. On the basis of $L_{i,j}$, the required buffer size when MPTCP uses active paths concurrently is

$$B_{active} = \sum_{R(N)} P_i * L_{i,N} * MSS_i \qquad (3)$$

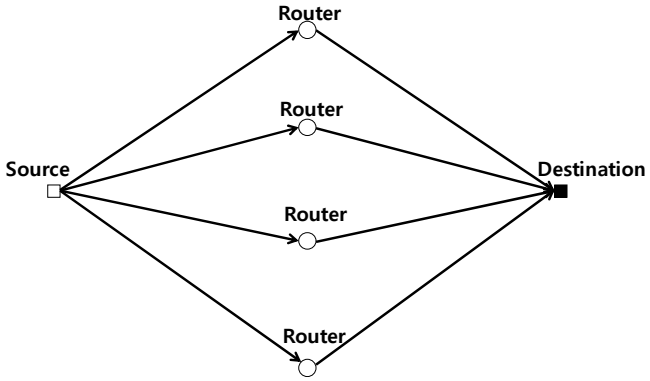

Fig. 1. Virtual network topology

where

$$R(N) = \{i \in integer \mid 1 \leq i \leq N, \; subflow\ i\ is\ active\} \qquad (4)$$

, P_i is the outstanding packets in subflow i, MSS_i is the maximum segment size on subflow i and N is the number of subflows. The *cwnd* could be changed every RTT of subflow according to congestion control algorithm. But, because it is hard to predict increase and decrease of *cwnd* perfectly, we assume *cwnd* of subflows is fixed during RTT of subflows.

From estimation of the required buffer size above, RBPM can predict buffer blocking phenomenon. If RBPM predicts buffer blocking phenomenon, it finds out bad performance subflows and set those subflows inactive. To define bad performance, we use bandwidth and RTT as parameters and Max-Min normalization as method.

$$Score_i = \frac{BW_i - BW_{min}}{BW_{max} - BW_{min}} - \frac{RTT_i - RTT_{min}}{RTT_{max} - RTT_{min}} \qquad (5)$$

where BW_{min} is the smallest bandwidth, BW_{max} is the largest bandwidth and RTT_{min} is the smallest RTT across all subflows and RTT_i is RTT of subflow *i*. The higher score it is, the better performance path it has. After finding bad performance flow influencing buffer blocking, RBPM stops transferring data packet through that path of subflow. Because of network dynamics, some of useful paths can be stopped. To utilize these paths, if the available buffer size is large enough, RBPM starts to send data through all stopped paths. We define the available buffer size is large enough, when the available buffer size at destination is over twice than the required buffer size. Algorithm I shows the overall algorithm of RBPM.

III. Performance Evaluation

In this section, the performances of the proposed RBPM scheme are presented and compared to those of the original MPTCP using measurements over a virtual network environment. We implement RBPM in the Multipath TCP - Linux kernel implementation version 0.89 [9]. To generate a virtual network environment, we use NS-3 Direct Code Execution, which is a framework that provides facilities to execute real user-space and kernel-space network protocol within NS-3 [10]. We set parameters of MPTCP for performance evaluation. The bounded receiver buffer size is set as 200K. The MPTCP path manager is 'fullmesh', the congestion control

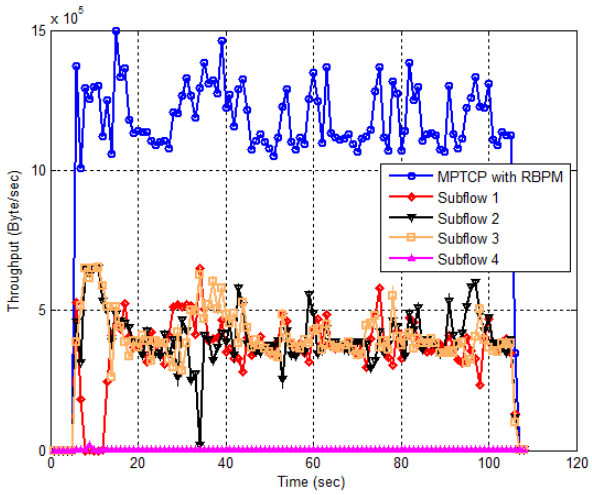

Fig. 2. Throughput of MPTCP with RBPM over three 5Mbps-10ms and a 1Mbps-100ms paths.

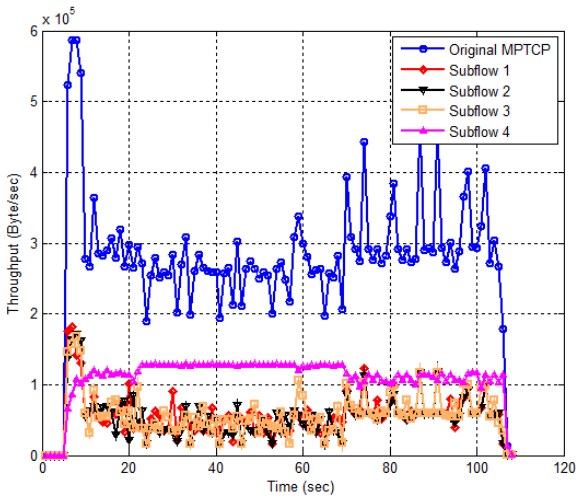

Fig. 3. Throughput of original MPTCP over three 5Mbps-10ms and a 1Mbps-100ms paths.

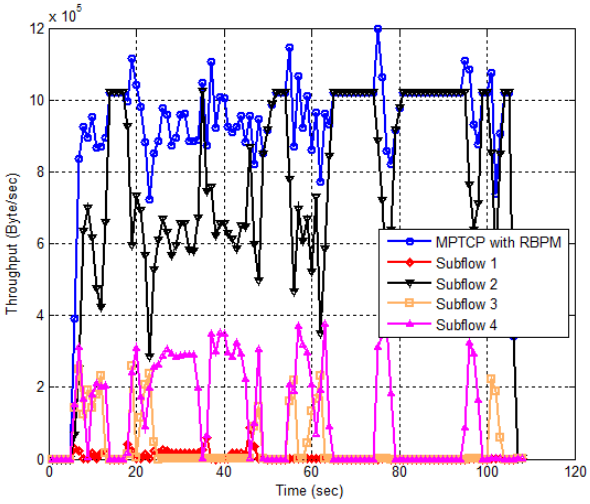

Fig. 4. Throughput of MPTCP with RBPM over 1Mbps-100ms, 8Mbps-50ms, 3Mbps-30ms, and 2Mbps-10ms paths.

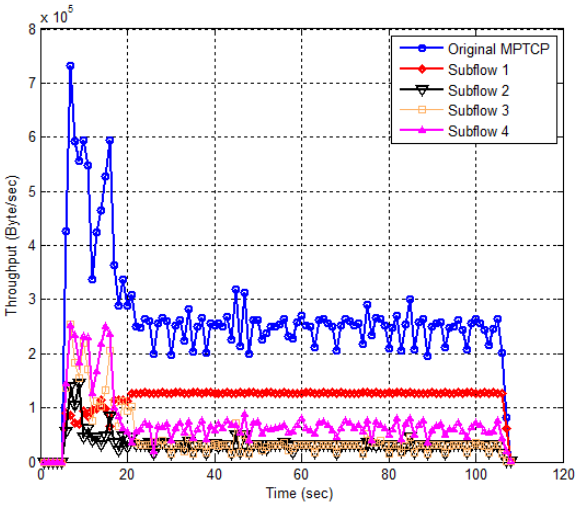

Fig. 5. Throughput of original MPTCP over 1Mbps-100ms, 8Mbps-50ms, 3Mbps-30ms, and 2Mbps-10ms paths.

algorithm is 'reno', and the opportunistic retransmission and penalization method proposed by [4] are 'disable'. The opportunistic retransmission and penalization are effective methods that reduce the performance degradation of MPTCP with the bounded buffer and dissimilar paths. But they don't prevent buffer blocking problem perfectly and a lot of packets transmitted over slow paths retransmit over fast paths when they are 'enable'. This duplicated retransmission with no error wastes bandwidth of slow paths. Thus, to show the buffer blocking clearly and prevent to waste bandwidth of slow paths, we don't use the opportunistic retransmission and penalization for performance evaluation.

We use 4-path topology, which is shown in Figure 1, to evaluate performance of MPTCP with RBPM and original MPTCP where characteristics of multiple paths are diverse. The source and destination are connected through four disjoint routers, and the links between routers and the destination have sufficient performance not to effect overall throughput (Bandwidth : 100Mbps, Delay : 1ns). And the links between routers and the source have various performances according to each path and environment of performance measurement. The queue type and size in each link are Drop-Tail and 100 packets, respectively.

Figure 2 and 3 show the performances of MPTCP with RBPM and original MPTCP where 3-path have good performance (Bandwidth : 5Mbps, Delay : 10ms) and 1-path is bottleneck (Bandwidth : 1Mbps, Delay : 100ms). We measure their performance for 100 seconds. Figure 2 shows the throughput of MPTCP with RBPM. Subflow 1, 2, and 3 pass the paths that have good performance and Subflow 4 pass the bottleneck path. When measurement starts, every path transfers data packets through their paths and evaluates the required buffer, B_j. After some transmission of all subflow, RBPM predicts buffer blocking problem and perceives that problematic subflow is Subflow 4. Then, MPTCP with RBPM stops transferring over the path of Subflow 4 until the available receive buffer size is large enough. Other three subflows send data over their paths with fully utilizing bandwidth of paths as if they use original MPTCP where there are only three paths that

have good performance without performance degradation. With these paths, MPTCP with RBPM transfers average 1299.757 packets per second and average 1114986.038 byte per second.

Figure 3 shows the throughput of original MPTCP with same paths in Figure 2. With these paths, original MPTCP transfers average 323.129 packets per second and average 275563.174 byte per second. In same 4-path topology where there is a bottleneck path, the throughput of original MPTCP is about four times lower than that of MPTCP with RBPM. In Figure 3, every path transfers data well through their paths at first. But the blocking problem occurs in a few seconds after measurement starts. Because a bottleneck path of Subflow 4 is slower than other paths, packets transmitted over other paths become out-of-ordered and start to fill the shared receive buffer during RTT of the bottleneck path. If out-of-ordered packets fully fill the receive buffer, no subflow can transfer data packets to destination. Thus, once the buffer blocking occurs, other three subflows cannot transfer data and underutilize their bandwidth during RTT of the bottleneck path. As a result, the throughput of three subflows that have good performance paths could be lower than that of bottleneck subflow.

Figure 4 shows the throughput of MPTCP with RBPM with 4-path that have various characteristics. The characteristics of paths are bandwidth : 1Mbps - delay : 100ms, bandwidth : 8Mbps - delay : 50ms, bandwidth : 3Mbps - delay : 30ms, and bandwidth : 2Mbps - delay : 10ms. We measure its performance for 100 seconds. In Figure 4, there is only one subflow that transfers data packet over their paths with fully utilizing bandwidth of path and other three subflows that are restricted to send data. Because using all subflow causes the buffer blocking problem, RBPM doesn't assign data packets to path of Subflow 1 and Subflow 3 that have bad performance. When amount of data packets to path of Subflow 4 is increased, the buffer blocking occurs. Then, MPTCP with RBPM stops transferring over the path of Subflow 4 and only Subflow 2 transfers packets. The transmission of only one subflow makes the available receive buffer size large enough and MPTCP with RBPM starts to send data packets through the other three stopped paths. With these various multiple paths, MPTCP with RBPM transfers average 964.032 packets per second and average 898704.71 byte per second.

Figure 5 shows the throughput of original MPTCP with 4-path that have various bandwidth and delay. We also measure performance for 100 seconds. The original MPTCP transfers average 306.143 packets per second and average 264422.111 byte per second with 4-paths that are same to Figure 4. The performance of original MPTCP with these paths is about three times lower than that of MPTCP with RBPM. As explained in Figure 3, the throughput is significantly degraded due to buffer blocking problem. When the source starts to transmit data to destination, every subflow transfers data packets well over their paths. But after a few seconds, Subflow 2, Subflow 3, and Subflow 4 are restricted to transfer data packets. Subflow 1 that has the worst path (Bandwidth : 1Mbps, Delay : 100ms) fully utilize its bandwidth because no subflow make packets transmitted over the worst path blocked. But Subflow 2 (Bandwidth : 8Mbps, Delay : 50ms) and Subflow 3 (Bandwidth : 3Mbps, Delay : 30ms) almost don't utilize their bandwidth because the MPTCP path manager is 'fullmesh', which creates full-mesh of subflows among all available subflows. If the packets from slowest path arrive at the receive buffer during buffer blocking phenomenon, buffer blocking problem is resolved. With resolving buffer blocking, available space in receive buffer is made. Then, Subflow 4 which has fastest RTT and smallest bandwidth fills space in the receive buffer with packets transmitted through path of itself by 'fullmesh' prior to Subflow 2 and Sublfow 3. That is why throughput of Subflow 4 has better performance than Subflow 2 and Sublfow 3.

IV. Conclusions

In this paper, we proposed RBPM, a simple and effective method to manage the multiple paths of MPTCP in order to improve the throughput and network utilization with bounded receive buffer in heterogeneous networks. RBPM estimates the number of out-of-ordered packets and the required buffer for transmission and predicts buffer blocking problem in advance. If it predicts buffer blocking, it finds out the bad performance subflows using Max-Min normalization which consider bandwidth and RTT delay as parameters and stops transmitting data packet to those paths. That is, RBPM prevents to hinder packet transmission of the other subflows from bad performance paths and suffer from throughput degradation. Consequentially, MPTCP with RBPM could have better performance than original MPTCP in most cases. Furthermore, it is easy to implement in real MPTCP Linux kernel because Algorithm of RBPM is simple as shown Algorithm 1.

Acknowledgment

This work was supported by the ICT R&D program of MSIP/IITP. [B0101-16-1276, Access Network Control Techniques for Various IoT Services].

References

[1] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural guidelines for multipath TCP development," IETF RFC 6182, Mar. 2011

[2] B. Oh, and J. Lee. "Constraint-based proactive scheduling for MPTCP in wireless networks." *Computer Networks*, vol. 91, pp. 548-563, Nov. 2015.

[3] S. Barr´e, C. Paasch, and O. Bonaventure, "Multipath TCP: From theory to practice," in *Proc. IFIP Networking*, May 2011.

[4] C. Raiciu, C. Paasch, S. Barr´e, A. Ford, M. Honda, F. Duchˆene, O. Bonaventure, and M. Handley, "How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP," in *Proc. 9th USENIX NSDI*, Apr. 2012.

[5] N. Kuhn, E. Lochin, A. Mifdaoui, G. Sarwar, O. Mehani, and R. Boreli, " DAPS: intelligent delay-aware packet scheduling for multipath transport" In *Proc. IEEE ICC*, Jun. 2014.

[6] L. Li, N. Hu, K. Liu, B. Fu, M. Chen, and L. Zhang, "Amtcp: an adaptive multi-path transmission control protocol," in *Proc. ACM CF*, May 2015.

[7] C. Raiciu, S. Barr´e, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath tcp," in *Proc. ACM SIGCOMM*, Aug. 2011,

[8] L. Kou, S. R. Chen and R. Wang, "A MPTCP path selection strategy based on improved grey relational analysis", in *Proc. 3rd ICFMM*, Jul. 2013.

[9] C. Paasch, S. Barre, et al., Multipath TCP in the Linux Kernel, available from http://www.multipath-tcp.org.

[10] NS-3 Project, Direct code execution, http://www.nsnam.org/overview/projects/direct-code-execution/.