Multipath Congestion Control for Shared Bottleneck

Michio Honda*, Yoshifumi Nishida*, Lars Eggert†, Pasi Sarolahti†, Hideyuki Tokuda*‡
Keio University*, Nokia Research Center†, JST-CREST‡
{micchie,nishida}@sfc.wide.ad.jp, {lars.eggert,pasi.sarolahti}@nokia.com, hxt@ht.sfc.keio.ac.jp

ABSTRACT

Multipath transport protocols, which transmit data over multiple distinct paths in an end-to-end connection are introduced. However, they have a problem in terms of fairness. When the transmissions along several paths share the same bottleneck link, the multipath connection receives higher throughput than a competing regular TCP flow, because it executes congestion control per path with the same algorithm as TCP. We investigate a congestion control scheme that addresses this problem with the weighted congestion control approach. In our scheme, an end-to-end connection that uses flows along multiple paths can fairly compete with TCP flows at the shared bottleneck. Our scheme also maximizes the utilization of different path characteristics, such as bandwidth capacity and RTT. Our simulation results show that a bundle of multiple flows based on our scheme fairly competes with TCP flows at the shared bottleneck.

1. INTRODUCTION

In 2006, P2P traffic was 37%, and HTTP traffic containing video and audio (e.g., YouTube) was 19% of the total Internet traffic [22]. These applications require high bandwidth allocations for a long time. TCP traffic still comprises a major share of the total Internet traffic [13]. Hence, we assume that high-speed and long-lived TCP connections are becoming more frequent. Such TCP connections remain in the congestion-avoidance phase, and hence compete against each other at the shared bottlenecks. This means that users or applications require more network capacity.

Simultaneous multipath utilization is a promising way in which end hosts can increase available network capacity. As the market for networking technology is evolving, it is becoming more common that the end hosts are equipped with multiple network interfaces (e.g., WLAN, GPRS and 3G). They have multiple links connected to the Internet, which results in availability of multiple paths between a source and a destination end host. Such multi-homed hosts can use more available network capacity if they aggregate the available bandwidth of multiple paths.

We assume that transport protocols will have capabilities to utilize multiple paths simultaneously between a source and a destination host. We call the entity over which applications communicate a **multipath connection**. For example, a multipath connection looks like a TCP connection to the application and provides a reliable and ordered byte stream. The endpoint of the multipath connection stripes user data across multiple distinct paths, using one **subflow** along each path.

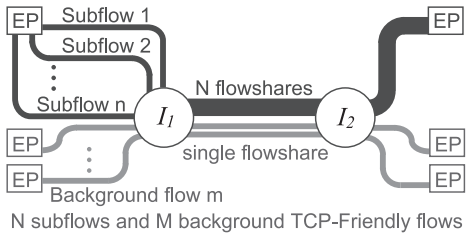

Figure 1: Unfair share at the shared bottleneck

Current connection-oriented transport protocols (e.g., TCP, SCTP and DCCP) transmit data only over a single path between a source and a destination hosts at any given time. Although SCTP supports multi-homing, standard SCTP does not transmit data over multiple paths simultaneously. Several proposals extend transport protocols to simultaneously utilize multiple paths [11, 24, 20, 12], achieving higher throughput than the base protocols.

However, multipath connections of these extensions are not compatible with TCP friendliness, which are overly aggressive at the shared bottleneck, because each subflow independently performs congestion control with an algorithm similar to TCP. When N subflows in a multipath connection compete against TCP flows at the same bottleneck, the multipath connection is approximately N times as aggressive as each of the TCP flows. In Fig. 1, one multipath connection that contains N subflows competes with M background TCP flows at the shared bottleneck between two intermediate nodes I_1 and I_2. While each of background flows receives a $1/(N+M)$ share of the bottleneck, a bundle of N subflows receives a $N/(N+M)$ share.

In Internet congestion control, a congestion-controlled flow between two transport endpoints (e.g., a single TCP connection) uses a single flowshare [16], which equally shares the bottleneck bandwidth with each other. N flowshares receive throughput N times a single flowshare at the same bottleneck, which are called multiple flowshares [16]. Multiple flowshares are mainly utilized to achieve weighted proportional fairness [6] between aggregation points that bundle multiple flows transmitted by multiple users. For example, distributed-multimedia applications [15] and wireless TCP proxies [5] use aggregation points. Some applications leverage parallel TCP connections between the same hosts to obtain more bandwidth or avoid head-of-line blocking. Such use of multiple TCP connections is unfair to the other traffic sharing the path. Congestion control of these connections

should reduce their bandwidth consumption to that of a single flowshare [8], e.g., as with E-TCP [7] and CM [3]. Even if an endpoint utilizes multiple paths for transmission, the multipath connection is essentially a single connection in the communication primitive to applications, such as a reliable and ordered byte stream. Consequently, the endpoint should use a combined single flowshare at the shared bottleneck for all subflows.

Sharing links among multiple end-to-end paths exist in a wide range of contexts in multi-homed environments. The most straightforward scenario is duplicate use of a source or a destination address, for example, when two subflows transmit data from different source addresses to the same destination address, or vice versa. This situation always occurs in multipath connections between a multi-homed host and a single-homed host. When both hosts are multi-homed, they can use multiple paths without duplicate use of addresses. However, even if the endpoint uses only paths where both the source and destination addresses are different, some intermediate routes can be shared. In addition, it is hard that routing systems ensure disjoint bottlenecks between end systems.

End-to-end multipath congestion control requires two properties, which are:

- **Fairness**
  TCP-friendliness is the most common fairness metric in the Internet. A multipath connection has the same communication primitive as a single TCP connection or the other end-to-end connection, as far as the application is concerned. Multipath connections should be TCP-friendly at the shared bottleneck regardless of the number of subflows.

- **Utilization**
  Distinct paths have different characteristics, such as RTT and spare bandwidth. In order to maximize the performance of the whole multipath connection, effective utilization of the shared bottleneck and spare bandwidth of distinct paths is desired.

In this paper, we investigate congestion control for multipath transport protocols. The main contribution of this paper is a new end-to-end congestion control scheme for multi-homed environment: Bidimensional-Probe Multipath Congestion Control (BMC). Using BMC, a bundle of subflows in a multipath connection fairly competes with background TCP-friendly flows at the shared bottleneck. BMC also maximizes the utilization of resources that along multiple paths with different characteristics. The remainder of this paper is organized as follows: In Section 2, we design a congestion control algorithm for multipath-enabled transport protocols. Section 3 evaluates our algorithm through simulations. The paper concludes with Section 4.

2. DESIGNING BIDIMENSIONAL-PROBE MULTIPATH CONGESTION CONTROL

In order to satisfy the fairness and utilization properties for multipath connections, we can consider three approaches. The first one is congestion window sharing between subflows that belong to the same multipath connection. This approach is implemented in E-TCP [7], CM [3] and MPAT [21] for aggregate congestion control between parallel TCP connections along the same path. The aggregated congestion window is increased by one packet within one RTT when there is no packet loss in this period. It is reduced by half when there is a packet loss in any of the connections that share the aggregated congestion window. In order to apply this approach to multipath connections, we could allocate the portion of the aggregated congestion window to subflows based on the spare bandwidth along each path. However, the congestion window is the allowable amount of data to be transmitted in an RTT, which varies with the interval of ACK. Subflows with different RTTs cannot share the same congestion window, because they experience events to increase or decrease the window size with different interval. In addition, since the aggregated congestion window is reduced by packet loss on any subflow, one packet loss in a subflow affects the other subflows. This degrades performance of the whole multipath connection. This approach is thus insufficient for multipath congestion control not only in terms of fairness, but also in terms of utilization.

The second approach is shared bottleneck detection. mTCP [24] performs TCP congestion control for each subflow, however, it implements shared bottleneck detection [19] to avoid the use of multiple subflows on the same bottleneck. mTCP suppresses subflows traversing the same bottleneck. However, mTCP takes maximum 15 seconds to detect the shared bottleneck. This slow response can be significant.

The third approach is to apply the weight to individual congestion control of each subflow so that a bundle of subflows can have the same aggressiveness as one TCP flow. In this approach, each subflow independently performs congestion control by adjusting its own congestion window. Hence, each subflow is not affected by the loss events on the other subflows. It can also work when the RTTs of the subflows are different, because subflows do not need to share one congestion window. For this reason, we design BMC based on this approach. We introduce two components: the *Aggressiveness Manager* and the *Proportion Manager* in BMC to achieve fairness and utilization properties.

2.1 Aggressiveness Manager

The aggressiveness manager maintains a constant aggressiveness for the overall multipath connection. It is critical for a bundle of subflows to compete fairly with background TCP-friendly flows which share the same bottleneck. Each of the individual subflows must not be more aggressive than a TCP-friendly flow, because the bundle of subflows must be as aggressive as a single TCP-friendly flow. We achieve this by using the weighted TCP congestion control algorithm for each subflow. This achieves throughput in proportion to its weight compared to a single TCP flow. The aggressiveness manager applies the weight to each subflow, while it maintains the sum of the weight so that a bundle of subflows in the multipath connection is kept as aggressive as one TCP flow.

We define the weight of a standard TCP connection as 1, and denote the weight of a subflow$_n$ as D_n $(0 < D_n \leq 1)$. A subflow$_n$ with the weight D_n should receive D_n times as much throughput as a competing TCP flow. When a bundle of N subflows receives the same throughput as one TCP flow, the following equilibrium is satisfied:

$$\sum_{n=1}^{N} D_n = 1 \tag{1}$$

The aggressiveness manager maintains this equilibrium across

the subflows in a multipath connection.

The aggressiveness manager applies the modified TCP congestion control to each subflow based on its weight. TCP's standard congestion control in the congestion-avoidance phase increases the window size by one MTU when packets are acknowledged without any packet loss within one RTT, and otherwise halves the window size [2]. In the aggressiveness manager, an individual subflow increases the window size in order to achieve proportional throughput to its weight. We refer to the number of packets by which the window size is increased within one RTT as "increase parameter".

We derive the increase parameter of the TCP congestion control that achieves the throughput in proportion to the weight from TCP modeling. In contrast to MulTCP [6], each subflow has the weight less than one, hence we need to apply different modeling from MulTCP to such weighted TCP congestion control. Although many researchers have modeled TCP, we investigate the weighted TCP congestion control based on [17], because it is useful for the weight less than one. [17] models throughput of TCP T as a function of RTT R, the retransmission timeout value t, packet size s, packet loss rate p, and the increase parameter a:

$$T = \frac{s}{R\sqrt{\frac{2p}{3a}} + t(3\sqrt{\frac{3p}{8a}})p(1+32p^2)} \tag{2}$$

The throughput of the standard TCP, T_{tcp} is given by applying $a = 1$ to Equation (2):

$$T_{tcp} = \frac{s}{R\sqrt{\frac{2p}{3}} + t(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{3}$$

When we want D times of TCP throughput for an individual subflow, the desired throughput T_d is:

$$T_d = D\frac{s}{R\sqrt{\frac{2p}{3}} + t(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{4}$$

In order to acquire the increase parameter a that achieves throughput in proportion to the weight D, we obtain equilibrium between T_d in Equation (4) and T in Equation (2):

$$a = D^2 \tag{5}$$

The aggressiveness manager thus applies the increase parameter D^2 to the subflow with the weight D. The individual subflow with the weight D consequently increases the window size by D^2 within one RTT when packets are acknowledged without any packet loss, and otherwise halves the window size.

[9] reduces the aggressiveness of parallel TCP flows along the same path by reducing the increase parameter of TCP, emulating TCP behavior along the longer RTT. However, although the TCP response function requires $(1/N)^2$ of the increase parameter for $1/N$ times throughput, [9] applies $1/N$ to the increase parameter. This results in over-aggressiveness of the bundle of parallel TCP flows to compete fairly with the other standard TCP flows.

Slow-start behavior of subflows is important on injecting new subflows to the network, or after the retransmission timeout. Parallel slow-starting subflows augment the transient effect on the other flows as well as parallel TCP connections, if each slow-starting subflow is as aggressive as TCP (i.e., increasing window size by a factor of two within one RTT). In addition, not only steady-state behavior but also slow-start behavior affects the overall throughput of each flow on the congested bottleneck, because such flows often cause retransmission timeouts, and then go into slow-start [17].

We mitigate the aggressiveness of slow-starting subflows in terms of "transient effect" on the network based on the principle of the slow-start in [8]. This means how quickly the slow-starting subflow increases the sending rate. One slow-starting TCP connection increases the window size by one packet per reception of an acknowledgment. Hence, one slow-starting subflow with the weight D increases the window size by D packets per reception of an acknowledgment. In other words, while slow-starting TCP increases the window size from W to $2W$ after one RTT, a slow-starting subflow with the weight D increases the window size from W_d to $W_d + W_d \times D$ after one RTT.

At the beginning of the slow-start, we apply the same window size as TCP to each subflow (i.e., two or four packets at the beginning of the transmission, and one packet after the retransmission timeout [2, 1]). If we use the conservative initial window size based on the weight and double the window size within an RTT, a multipath connection could achieve the same "throughput" as TCP at any point during the slow-start phase. This, however, makes the initial window size to be less than one MTU for subflows that have the weight less than one. It could be problematic, because it enforces transmission of a packet smaller than one MTU.

2.2 Proportion Manager

The proportion manager optimizes the combination of the weight parameters for the subflows, resulting in the maximized chances of full utilization of disjoint links and the shared bottleneck. If some subflows are constrained by the limitation of the spare bandwidth of disjoint links, the multipath connection cannot achieve equal throughput to a TCP flow which competes at the shared bottleneck. Consequently, the multipath connection cannot maximize the utilization of the shared bottleneck. This is because, based on the aggressiveness manager, the bundle of subflows receives the equal throughput with one TCP flow when all subflows are constrained by the shared bottleneck. Hence, we have to apply the weight to subflows so that the desired throughput of each subflow (i.e., T_d in Equation (4)) does not exceed the spare bandwidth of the disjoint links.

The proportion manager detects subflows that are constrained by the spare bandwidth of disjoint links. According to Equation (4), when subflows are constrained by the shared bottleneck, throughput of each subflow should be proportional to its weight, and inversely proportional to the RTT. We denote the value which has deducted the effect of the weight and RTT from the throughput of a subflow as T_{wr}:

$$T_{wr} = \frac{R}{D}T \tag{6}$$

where T is the throughput of a subflow. D and R are the weight and RTT of that subflow, respectively. When subflows are constrained by the shared bottleneck, each of them could have the same T_{wr}. Subflows which have T_{wr} less than that of the other subflows could be constrained by the spare bandwidth of disjoint links. The proportion manager aims at the convergence to the same T_{wr} on all subflows by adjusting the weight of each subflow in the multipath connection.

In order to obtain T_{wr} in Equation (6), the proportion manager measures the throughput of each subflow during a

certain period. Subsequently, it reduces the weight of the subflow that has experienced the smallest T_{wr}, at the same time increases the weight of the subflow that has experienced the largest T_{wr} that would have higher spare bandwidth. The proportion manager performs these two steps repeatedly. In this way, we pass over the spare bandwidth limitation of the disjoint link to increase the total throughput of the multipath connection.

Throughput measurement on each subflow has to be done at least during one congestion epoch of the subflow, due to the window adjustment mechanism of the TCP congestion control. One congestion epoch consists of $(W/2a) + 1$ round trip times, where W is the window size at the end of the congestion epoch. a is the number of packets by which the window size is increased within an RTT. The longer the measurement is conducted, the more exact the average throughput is measured. For example, lossy wireless links would require longer measurement, because the window size at the end of each congestion epoch could not be uniform. However, the convergence time to the optimal proportion becomes slower in proportion to the measurement duration. On the other hand, excessively short measurement and frequent change of the weight might lead to network instability. Details of the measurement mechanism is currently being investigated.

The changing factor of the weight is important for the quick convergence to the optimal proportion and the stability. If we drastically reduce the weight of subflows that achieve higher throughput, the throughput after the reduction is seriously reduced. This could lead to performance degradation of the whole multipath connection. The proportion manager maintains subflows with the larger weight more conservatively about decreasing the weight. This allows the subflows with the larger weight to maintain aggressiveness, hence the higher throughput of them are maintained. It also allows the subflows with the smaller weight to maintain conservativeness, hence it can mitigate the effect on the higher-throughput subflows.

We denote the weight before and after the reduction as D^{dec}_{cur} and D^{dec}_{new}. D^{dec}_{new} is calculated by:

$$D^{dec}_{new} = (D^{dec}_{cur})^2 \quad (7)$$

This means that the reduction factor of the weight is smaller when the original weight is larger. Based on the reduction of the weight of a subflow, the proportion manager increases the weight of another subflow that has achieved the maximum value of T_{wr} in Equation (6) with satisfying Equation (1).

2.3 Behavior on Disjoint Bottlenecks

BMC can achieve not only a fair resource allocation at the shared bottleneck, but also resource pooling [14, 23] along the disjoint bottlenecks. In the resource pooling, multiple bottlenecks are treated as a set of resources, resulting in a fair resource allocation between flows across the distinct bottlenecks. Fig. 2 illustrates such a resource allocation. s_1, s_2 and s_3 are communicating with d_1, d_2 and d_3, respectively. There are two paths between s_2 and d_2. s_1 and s_3 transmit standard TCP flows. The sum of the available bandwidth of the two bottlenecks is 30 Mbps, hence each source should send at 10 Mbps to share it equally. In order to achieve the resource pooling in Fig. 2, s_2 has to use 2 Mbps of 12 Mbps bottleneck and 8 Mbps of 18 Mbps bottleneck. If s_2 performs

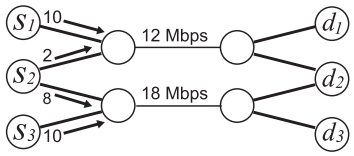

Figure 2: Resource pooling

per-subflow TCP's congestion control, s_1, s_2 and s_3 receive 6, 15 and 9 Mbps of throughput, respectively. Hence, per-subflow TCP's congestion control is inappropriate not only for the shared bottleneck, but also for the resource pooling.

BMC converges to the resource utilization based on the resource pooling principle between disjoint bottlenecks along the same RTT paths. In Fig. 2, we denote the weight for the subflow transmitted from s_2 to the 12 Mbps bottleneck as D_1, and that to the 18 Mbps bottleneck as D_2. We define RTT of each end-to-end path as 1. When s_2 shares 2 Mbps at 12 Mbps bottleneck, and 8 Mbps at 18 Mbps bottleneck, both flows transmitted by s_2 have equal T_{wr} (i.e., $T_{wr} = 10$) in Equation (6). Then, the weight parameters for these subflows are $D_1 = 1/5$ and $D_2 = 4/5$. If these weight parameters are different, an equal resource allocation between s_1, s_2 and s_3 is not achieved. When $D_1 > 1/5$, T_{wr} at the 12 Mbps bottleneck becomes less than T_{wr} at the 18 Mbps bottleneck. When $D_1 < 1/5$, T_{wr} at the 12 Mbps bottleneck becomes more than T_{wr} at the 18 Mbps bottleneck. For example, if both of these subflows have the weight $1/2$, T_{wr} for each subflow is 2 and 3, respectively. Our proportion manager then reduces D_1 and increases D_2.

3. EXPERIMENTAL RESULTS

We evaluate the aggressiveness manager of BMC with the ns-2 network simulator. We substitute TCP connections with the weighted TCP congestion control for subflows. In this section, we refer to such modified TCP connections as the weighted AIMD (WAIMD) flows. We also refer to a WAIMD flow with the weight D as the WAIMD(D) flow. Based on the design of the aggressiveness manager, a WAIMD(D) flow increases the window size by D^2 packets when packets are successfully acknowledged within one RTT, and otherwise halves the window size in the steady-state. Slow-start behavior of the WAIMD(D) flow is also modified to increase the window size by D packets per reception of an acknowledgment. Substituting the weighted TCP connections for subflows eliminates the effect of receiver-buffer blocking and handling packets that are received out-of-order, which could influence the behavior of the multipath transport protocols in the experiment. This is a deliberate decision, as the paper focuses on congestion control for multipath transport protocols. Protocol performance caused by the receiver-buffer blocking or out-of-ordered packet handling is separate area.

Fig. 3 illustrates the simulation setup. In this simulation, the WAIMD flows and the same number of standard TCP flows compete at the 60 Mbps bottleneck link with RED queue management. The TCP version is Reno in both WAIMD and standard TCP flows. The delay of the bottleneck link is set to 20 ms. Each sender endpoint is connected to a 100 Mbps link with 2 ms delay before the bottleneck. Each receiver endpoint is connected to a 100 Mbps link at

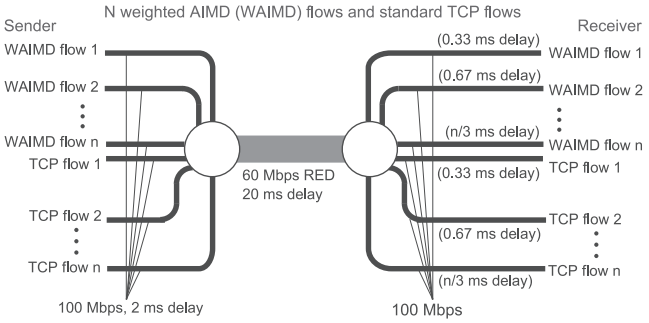

Figure 3: Simulation setup

the other side of the bottleneck link. The delay of the receiver link is set to the value of the index of the flow divided by 3. This means that nth standard TCP flow and WAIMD flow have $n/3$ ms delay at their receiver links.

First, for the most fundamental experiment, we observe the throughput of the weighted TCP congestion control with the weight less than one. Fig. 4 plots the throughput ratio of the WAIMD flows to standard TCP flows during 100 second simulation. In each trial, N WAIMD flows with the same weight and N standard TCP flows compete at the 60 Mbps bottleneck. We conduct such trials for the weight 4/5, 3/4, 2/3, 3/5, 1/2, 1/3, 1/4, and 1/5. For each weight, we conduct trials for 2, 4, 8, 16, 32, 48 and 64 WAIMD and standard TCP flows. As a result, we observe that WAIMD flows obeying our weighted TCP congestion control with the weight less than one can approximately achieve throughput with regard to the weight in the wide rage of the weight value and the number of flows.

When the number of flows is larger, hence the loss-event rate is higher, the WAIMD flows are a little more aggressive compared to the weight parameters. The possible reason is retransmission timeouts on the WAIMD flows. When the loss-event rate is higher, probability of the timeouts becomes higher. These results would occur due to the slow-starting subflow behavior. As mentioned in Sec. 2.1, the throughput of the slow-start behavior of subflows is larger compared to the weight. In addition, when the number of the WAIMD flows and standard TCP flows are 1, 2 and 4, the throughput ratio of WAIMD flows is a little off from the ideal one. We consider that the extremely low loss-event rate (0.029 - 0.164%) affects this behavior, because when the loss-event rate is lower, the influence to the throughput caused by one loss event is larger.

Second, we confirm that the throughput of a bundle of subflows is constant regardless of the combination of the weight as long as the sum is one. We observe throughput of bundles of two or four WAIMD flows. The sum of the weight parameters in each bundle is one. To observe the behavior of the bundles of two subflows, we make $N/2$ WAIMD(D), $N/2$ WAIMD($1-D$) and N standard TCP flows compete at the 60 Mbps bottleneck in each trial. We conduct the trials for $N = 2$, 4, 8, 16, 32, 48 and 64 for each combination of the weight parameters. To observe the behavior of the bundles of four subflows, we make $N/4$ WAIMD(1/10) and $3N/4$ WAIMD(3/10) flows (i.e., $N/4$ WAIMD(3/10) $\times 3$), or $N/4$ WAIMD(3/18), $N/4$ WAIMD(4/18, 1/2), $N/4$ WAIMD(5/18) and $N/4$ WAIMD(6/18) flows compete with

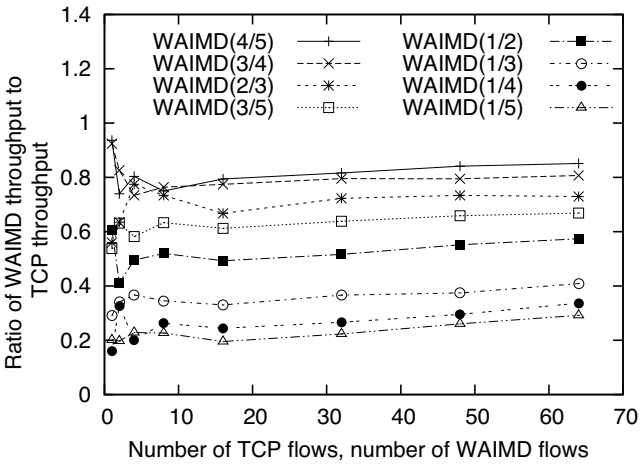

Figure 4: WAIMD flows vs. TCP flows

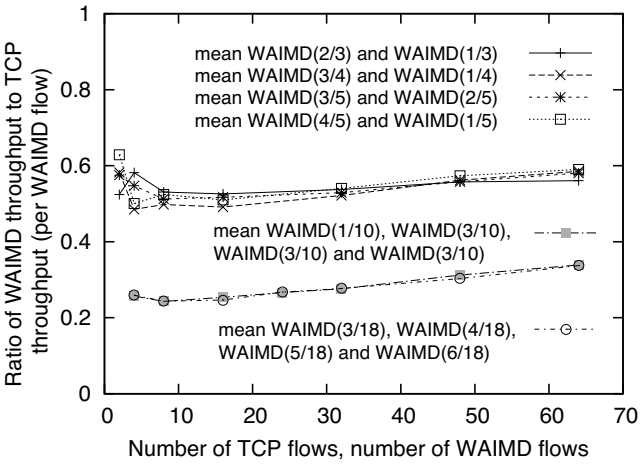

Figure 5: Bundles of WAIMD flows vs. TCP flows

N standard TCP flows at the 60 Mbps bottleneck in each trial. We conduct the trials for $N = 4$, 8, 16, 24, 32, 48 and 64 for each combination of the weight parameters. Fig. 5 plots the throughput ratio of bundles of WAIMD flows to the standard TCP flows during 100 second simulation. The throughput ratio is averaged out per WAIMD flow, hence the ideal ratio on the behavior of two subflows is 0.5, and that of four subflows is 0.25. As a result, the bundle of subflows of which the sum of the weight is one can achieve approximately equal throughput with one TCP flow at the shared bottleneck. Thus, we confirm that our aggressiveness manager can certainly improve TCP-friendliness of multipath connections at the shared bottleneck.

When the number of the WAIMD or standard TCP flows is 2 and 4, the throughput ratio of the bundles of the WAIMD flows to the standard TCP flows is a little off from the ideal one in some cases. In addition, when the number of the WAIMD and standard TCP flows is larger, the throughput ratio of the bundles of the WAIMD flows is a little higher. These results would occur due to the same reason as the first experiments in Fig. 4.

4. CONCLUSION AND FUTURE WORK

In this paper, we presented a congestion control scheme for multipath transport protocols, which is TCP-friendly at the shared bottleneck, and effectively uses spare bandwidth of distinct paths. Our simulation results showed that a bundle of the subflows can receive approximately TCP-friendly throughput, resulting in the improved TCP-friendliness of multipath connections.

We are currently investigating details of the proportion manager. At the same time, investigating the principle of TCP friendliness in multipath congestion control is interesting, for example, how the multipath connection should receive RTT bias on multiple subflows. Dealing with different fairness criterion, such as cost fairness [4] would also be challenging. We believe that multipath transport protocols will become common due to the significant benefits of increased available network capacity, improved reliability and traffic engineering. Hence, we are also developing a multipath transport protocol that implements BMC, because none of existing multipath transport protocols is practical to the current commercial Internet [10].

According to [18], when the sending rate of flows are significantly different, the higher sending rate flow experiences lower loss-event rate. Since each subflow in a multipath connection obeying BMC is less aggressive than TCP, it might give higher loss-event rate on subflows. We explore the impact on BMC with further experiments and analysis. In addition, in this paper, we investigated weighted AIMD-based design of BMC. On the other hand, it is interesting to use other congestion control variants for BMC. It will result in adaptation to more various scenarios, such as high-speed and long-distance networks, and lossy wireless networks.

Acknowledgment

We thank Damon Wischik for his helpful comments. We also thank Kenji Yonekawa for improving quality of the paper. Lars Eggert and Pasi Sarolahti were partly funded by *Trilogy*, a research project supported by the European Commission under its Seventh Framework Program.

5. REFERENCES

[1] M. Allman, S. Floyd, and C. Partridge. Increasing TCP's Initial Window. *RFC 3390*, Oct. 2002.

[2] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. *RFC 2581*, Oct. 1999.

[3] H. Balakrishnan, H. Rahul, and S. Seshan. An Integrated Congestion Management Architecture for Internet Hosts. In *Proc. ACM SIGCOMM*, pages 175–187, 1999.

[4] B. Briscoe. Flow Rate Fairness: Dismantling a Religion. *ACM CCR*, 37(2):63–74, 2007.

[5] R. Chakravorty, S. Katti, J. Crowcroft, and I. Pratt. Flow Aggregation for Enhanced TCP over Wide-Area Wireless. In *Proc. IEEE INFOCOM*, pages 1754–1764, 2003.

[6] J. Crowcroft and P. Oechslin. Differentiated End-to-End Internet Services using a Weighted Proportional Fair Sharing TCP. *ACM CCR*, 28(3):53–69, 1998.

[7] L. Eggert, J. Heidemann, and J. Touch. Effects of Ensemble-TCP. *ACM CCR*, 30(1):15–29, 2000.

[8] S. Floyd. Congestion Control Principles. *RFC 2914*, Sep. 2000.

[9] T.J. Hacker, B.D. Noble, and B.D. Athey. Improving Throughput and Maintaining Fairness using Parallel TCP. In *Proc. IEEE INFOCOM*, pages 2480–2489, Mar. 2004.

[10] M. Honda, E. Balandina, P. Sarolahti, and L. Eggert. Designing a Resource Pooling Transport Protocol. In *Proc. IEEE GIS*, Apr. 2009.

[11] H.-Y. Hsieh and R. Sivakumar. A Transport Layer Approach for Achieving Aggregate Bandwidths on Multi-homed Mobile Hosts. In *Proc. ACM MobiCom*, pages 83–94, 2002.

[12] J. R. Iyengar, P. D. Amer, and R. Stewart. Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths. *IEEE/ACM ToN*, 14(5):951–964, 2006.

[13] W. John and S. Tafvelin. Analysis of Internet Backbone Traffic and Header Anomalies Observed. In *Proc. ACM IMC*, pages 111–116, 2007.

[14] F. P. Kelly and T. Voice. Stability of End-to-End Algorithms for Joint Routing and Rate Control. *ACM CCR*, 35(2):5–12, Apr. 2005.

[15] D. Ott and K. Mayer-Patel. An Open Architecture for Transport-Level Protocol Coordination in Distributed Multimedia Applications. *ACM TOMCCAP*, 3(3):17, 2007.

[16] D. Ott, T. Sparks, and K. Mayer-Patel. Aggregate Congestion Control for Distributed Multimedia Applications. In *Proc. IEEE INFOCOM*, pages 13–23, 2004.

[17] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. In *Proc. ACM SIGCOMM*, pages 303–314, Aug. 1998.

[18] I. Rhee and L. Xu. Limitations of Equation-based Congestion Control. *IEEE/ACM ToN*, 15(4):852–865, 2007.

[19] D. Rubenstein, J. Kurose, and D. Towsley. Detecting Shared Congestion of Flows Via End-to-End Measurement. *IEEE/ACM ToN*, 10(3):381–395, 2002.

[20] S. Saito, Y. Tanaka, M. Kunishi, Y. Nishida, and F. Teraoka. AMS: An Adaptive TCP Bandwidth Aggregation Mechanism for Multi-homed Mobile Hosts. *IEICE ToIS*, 89:2838–2847, 2006.

[21] M. Singh, P. Pradhan, and P. Francis. MPAT: Aggregate TCP Congestion Management as a Building Block for Internet QoS. In *Proc. ICNP*, pages 129–138, 2004.

[22] M. Sullivan. Surveys: Internet Traffic Touched by YouTube. URL http://www.lightreading.com/, Jan. 2007.

[23] D. Wischik, M. Handley, and M. Bagnulo. The Resource Pooling Principle. *ACM CCR*, 38(5):47–52, 2008.

[24] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang. A Transport Layer Approach for Improving End-to-End Performance and Robustness Using Redundant Paths. In *Proc. USENIX ATC*, pages 99–112, 2004.