SplitBuff: Improving the Interaction of Heterogeneous RTT Flows on the Internet

Shahida Jabeen, Muhammad Bilal Zafar, Ihsan Ayyub Qazi, Zartash Afzal Uzmi
Computer Science Department, LUMS
Email: {shahida.jabeen,12100060,ihsan.qazi,zartash}@lums.edu.pk

***Abstract*— Today router buffers are sized according to the well-known Bandwidth-Delay Product (BDP) rule, which uses the average round-trip time (RTT) of flows traversing a router. The BDP rule not only leads to large queueing delays, but also imposes "one (buffer) size fits all" philosophy for flows exhibiting a large variation in RTT. When short and long RTT flows compete at a single buffer, they may adversely affect each other in throughput and delay. We propose SplitBuff, a scheme using which short RTT flows achieve low delay and long RTT flows achieve high throughput, without requiring any protocol modifications. With SplitBuff, a router splits a buffer into multiple buffers of varying sizes and maps flows onto these buffers based on their RTTs. We describe SplitBuff and evaluate its performance using extensive ns-2 simulations to demonstrate its effectiveness.**

I. Introduction

Buffer sizing is a key issue in the design of Internet routers because it can significantly impact the Quality of Service (QoS) seen by applications [1], [2], [3]. In particular, while over-sized buffers lead to large queueing delays and high jitter, small buffers can degrade throughput and increase packet loss rate. Today, buffers are sized based on two primary objectives. First, to accommodate short-term bursts that occur when the arrival rate temporarily exceeds the link capacity. Second, to ensure that a Transmission Control Protocol (TCP) connection is able to maintain full link utilization at all times. This results in the well-known Bandwidth-Delay Product (BDP) rule for buffer sizing [1].

The BDP rule states that a router needs buffer of size equal to the average round-trip time (RTT) of passing flows multiplied by the bottleneck link capacity. The BDP rule is based on the dynamics of TCP's congestion control algorithm. In particular, the goal is to have enough buffers at a router so that when a TCP source responds to congestion, the buffer never goes empty, thus keeping the bottleneck link busy 100% of the time [2]. When flows with different RTTs co-exist, the BDP rule chooses the buffer size based on the average of their RTTs. This has important consequences on the performance seen by different RTT flows. In particular, short RTT flows observe a large queueing delay due to the presence of long RTT flows, and long RTT flows experience throughput degradation due to the presence of short RTT flows. For example, when 10 ms and 990 ms RTT flows share a bottleneck, BDP rule leads to 500 ms of buffering. If 10 ms RTT flow was the only flow present, only 10 ms worth of buffering would have been required. But the presence of a 990 ms flow can cause an extra 490 ms queueing delay for the short RTT flow. On the other hand, short RTT flows fill up the buffer quickly causing high loss rate and subsequently low throughput for long RTT flows.

In this paper, we ask, "*How can we simulateneously improve the throughput of long RTT flows and reduce the delay experienced by short RTT flows while sizing buffers according to the BDP rule?*". To this end, we propose SplitBuff, a framework for splitting buffers based on the RTT of flows. With SplitBuff, routers split a single buffer into multiple buffers of smaller sizes, each of which carry flows with a smaller RTT range than the RTT range of all flows traversing the router. The cumulative size of these buffers is chosen according to the BDP rule and the individual buffers are sized based on the average RTT of flows traversing them. We then use round-robin scheduling for transmitting packets from these buffers. Since flows with similar RTTs are grouped together, this results in smaller queueing delays. The throughput of long RTT flows improve due to lower loss rates. This is achieved by using a buffer size no larger than given by the BDP rule.

We make the following contributions in this paper: (1) We study the interaction of short and long RTT flows when buffers are sized according to the BDP rule, (2) we propose SplitBuff, a framework for isolating the impact of short and long RTT flows when they co-exist, and (3) we conduct rigorous ns-2 simulations to show the efficacy of SplitBuff.

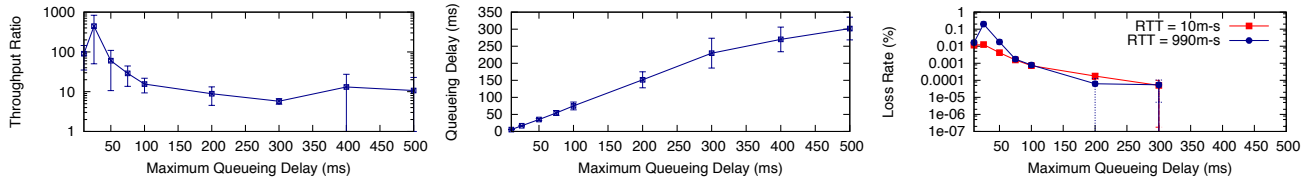
The rest of the paper is organized as follows. We study the interaction of short and long RTT flows in Section II. We present SplitBuff in Section III. SplitBuff is evaluated in Section IV. We discuss related work in Section V and finally, we offer concluding remarks in Section VI.

II. Interaction of Short and Long RTT Flows

The throughput of a TCP flow is inversely proportional to its RTT. Consequently, short RTT flows achieve a larger share of the bottleneck capacity when competing with long RTT flows as they are able to increase their sending rate faster. In particular, the TCP throughput S of a long-lived connection is related to the RTT T and the loss rate p by the following equation:

$$S = \frac{0.87MSS}{T\sqrt{p}} \quad (1)$$

where MSS is the maximum segment size [4]. The RTT $T = T_p + T_q$, where T_p and T_q are the propagation and queueing delays, respectively. The maximum value of T_q is determined by the bottleneck router's buffer size.



(a) Throughput ratio of short and long RTT flows (b) Average queueing delay (c) Loss rate for short and long RTT flows

Fig. 1. The figure shows that the throughput of long RTT flows is significantly less than that of short RTT flows. Moreover, increasing the buffer size causes the short RTT flows to suffer from additional queuing delays which can be much larger than their round-trip propagational delays.

When two TCP flows with different RTTs compete at a bottleneck link, their throughput ratio according to Eq. 1 is

$$\frac{S_s}{S_l} = \frac{T_l}{T_s}\sqrt{\frac{p_l}{p_s}} \tag{2}$$

where S_s and S_l, T_s and T_l, and p_s and p_l, represent the throughput, round-trip time and loss rate of short and long RTT flows, respectively. Let T_q be the average queueing delay seen by both flows then:

$$\frac{S_s}{S_l} = \frac{T_l + T_q}{T_s + T_q} \tag{3}$$

when $p_l = p_s$. Since TCP is designed to fill buffers of any size [2], the difference in the throughput of short and long RTT flows can be reduced by increasing the buffer size. However, this comes at the cost of increasing the average queueing delay.

To validate this, we conducted ns-2 simulations on a dumbbell topology. We set the bottleneck capacity to 100 Mbps and the packet size to 1 KB. Fig. 1(a) shows the throughput ratio as a function of the bottleneck buffer size when a 10 ms RTT flow competes with a 990 ms RTT flow. Observe that when buffer sizes are small, the throughput ratio can be more than 100. This happens because short RTT flows fill up the small buffer very quickly, leaving little room for long RTT flows, thus causing the throughput of latter flows to degrade considerably (see Fig. 1(c)). For example, a 25 ms buffer results in a throughput ratio of 443. Note that this is similar to the one predicted by Eq. 2 as the product of the ratio of RTTs and the ratio of loss rates ($=\frac{990}{10} \times \sqrt{\frac{0.2}{0.012}} \approx 405$).

However, as the buffer size increases, the throughput ratio decreases. This happens because increasing the buffer size reduces the value of the RTT ratio. In addition, the loss rate decreases for both flows. While the short RTT flow still causes the buffer to fill up quickly but with increasing buffer size, the probability that the large RTT flow will find the buffer to be full decreases, which improves the throughput experienced by the long RTT flow. For example, when a 300 ms buffer is used, the average queueing delay is ∼230 ms, which results in a throughput ratio equal to 1220/240=5.1. The ns-2 simulation results yield a value of 5.6.

When buffer is sized according to the BDP rule, which results in a 500 ms buffer, the short RTT flow has to face large and unnecessary queueing delays due to the presence of the long RTT flow (since the buffer is sized according to the average RTT of both flows). For example, the short RTT flow experiences a queueing delay in excess of 300 ms (see Fig. 1(b)), however, if it were the only flow present then only 10 ms of buffering would have been needed to maintain full link utilization. In summary, varying the buffer size of a single link presents a tradeoff between delay and throughput ratio.

- Small buffer sizes lead to smaller queueing delays but considerably degrade the throughput of long RTT flows.
- Large buffer sizes improve throughput of long RTT flows, even though the throughput ratio remains significant, but also increase the average queueing delay. Consequently, when buffers are sized according to the BDP rule, short RTT flows experience large queueing delay due to the presence of long RTT flows. Large queue sizes can result in high jitter, which can hurt real-time applications such as VoIP and video conferencing.

III. Proposed Framework: SplitBuff

To allow long RTT flows to achieve high throughput while ensuring low delay for short RTT flows, we propose ***SplitBuff***, a framework for limiting the interaction between short and long RTT flows. With SplitBuff, the available buffer space is split into multiple sub-buffers and flows are mapped to sub-buffers based on their RTTs. Each sub-buffer carries flows with a smaller RTT range compared to the RTT range of all flows traversing the router. The size of each sub-buffer is chosen based on the RTT of flows traversing it.

In order to realize SplitBuff, three important questions need to be addressed: (a) *How do routers obtain flow RTT information?*, (b) *how should flows be mapped to sub-buffers?*, and (c) *how should the sub-buffers be sized?*.

A. Obtaining RTT Information

The RTT information can be obtained by a router using several methods. These methods can be divided into two categories: (a) *Passive RTT Estimation* and (b) *Active RTT Estimation*.

Passive RTT Estimation rely on observing the behavior of each flow at a router; either by observing the TCP SYN-ACK pairs and time stamping [5] or by using algorithms based on the frequency of TCP packets [6] in each flow. All of these techniques require per-flow state and an additional overhead of RTT estimation. In addition, they cannot be used if IP payloads are encrypted.

Active RTT Estimation rely on explicitly sending the RTT in flows' packets to the routers. This method has an overhead of conveying RTT information in the packet headers but have reduced RTT estimation time and complexity. The RTT

information can be quantized [7], [8] so that it can be easily incorporated in the IP header without requiring a new field or any changes in IP. Therefore, we use active RTT estimation in SplitBuff. We would like to point out existing buffer sizing rules [1], [2], [3] as well as several congestion control protocols [7], [8], [9] also require flow RTT information.

B. Mapping Flows to Sub-Buffers

An important decision in the design of SplitBuff is the choice of a criterion for mapping flows to sub-buffers. It is important for the criterion to be *scalable* (in the sense that it should not require per-flow state) and *efficient* (so that it does not introduce any significant overhead inside the routers). A simple criterion is to use the average RTT of flows traversing a router for mapping flows. In particular, when two sub-buffers are used, flows with RTT greater than the average are mapped to one buffer and the rest are mapped to the other. A downside of using the average RTT ($\overline{RTT}$) is that the RTT of each flow includes the queueing delay at the bottleneck link, which can vary widely. Therefore, we use the average round-trip propagation delay ($\overline{RTTP} = \overline{RTT} - T_q$) as a criterion for mapping flows to sub-buffers.

C. Sizing Sub-Buffers

Choosing the size of sub-buffers is a critical decision because it impacts the throughput and delay experienced by flows. As router buffers are sized according to the BDP rule [2], we therefore enforce that the cumulative size of all sub-buffers should not exceed the BDP. To achieve fair throughput for long RTT flows and low delay for short RTT flows, we size sub-buffers proportional to the average RTT of flows passing through them. Let B be the buffer size given by the BDP rule based on the average RTTP of *all* flows traversing the router. Let T_{B_X} and T_{B_Y} be the average RTTP of flows in the two sub-buffers X and Y, respectively. Then

$$B_X = \frac{T_{B_X} B}{T_{B_X} + T_{B_Y}}, B_Y = \frac{T_{B_Y} B}{T_{B_X} + T_{B_Y}} \quad (4)$$

where B_X and B_Y are the sizes of sub-buffers X and Y, respectively.

The above rule ensures that buffers sizes proportional to flow RTTs. For example, suppose two flows with a RTT of 10 ms and 990 ms, respectively, traverse a router using SplitBuff. Their average RTT is 500 ms, which yields a buffer size of 6250 pkts on a 100 Mbps link with 1 KB packet sizes. The sub-buffers would then have sizes of ∼62 pkts and ∼6187 pkts. While the 10 ms RTT flow would traverse the former sub-buffer, the 990 ms flow would traverse the latter. This allows the short RTT flow to achieve low delay and the long RTT flow to obtain high throughput. Note that packets from these queues are served using a fair round-robin scheduler. The working of SplitBuff with two sub-buffers is explained in Fig. 2.

Variables:
p = received packet.
$p \rightarrow rtt$ = RTT information in the received packet.
RTT_s = sum of the RTTs.
RTT_x, RTT_y = sum of the RTTs in sub-buffer x or y.
T_u = update time period.
T_q = average queuing delay.
T_{q_x}, T_{q_y} = average queuing delay in sub-buffer x or y.
$thresh$ = threshold for mapping flows.
B = total buffer size.
B_x, B_y = total size of sub-buffer x or y.
C = link capacity.
α = smoothing constant with value between 0 and 1.
On every packet *p* arrival in time period T_u:
  $RTT_s = RTT_s + (p \rightarrow rtt)$;
  if $((p \rightarrow rtt) - T_q) < thresh$ **then**
    $RTT_x = RTT_x + (p \rightarrow rtt)$;
    $T_{q_x} = update(T_{q_x}$, current number of packets in $B_x)$;
    append p to the tail of sub-buffer x;
  else
    $RTT_y = RTT_y + (p \rightarrow rtt)$;
    $T_{q_y} = update(T_{q_y}$, current number of packets in $B_y)$;
    append p to the tail of sub-buffer y;
  end
After every time period T_u:
  $(T_q, T_{q_x}, T_{q_y}) = update(T_q, T_{q_x}, T_{q_y}, \alpha)$;
  $B = (average(RTT_s) - T_q) * C$;
  $thresh = (average(RTT_s) - T_q)$;
  $T_{B_x} = average(RTT_x) - T_{q_x}$;
  $T_{B_y} = average(RTT_y) - T_{q_y}$;
  $B_x = \frac{T_{B_x} * B}{T_{B_x} + T_{B_y}}$; $B_y = \frac{T_{B_y} * B}{T_{B_x} + T_{B_y}}$;
  $RTT_s, RTT_x, RTT_x = 0$;

Fig. 2. Working of SplitBuff with two sub-buffers

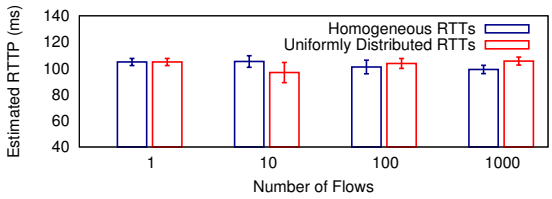


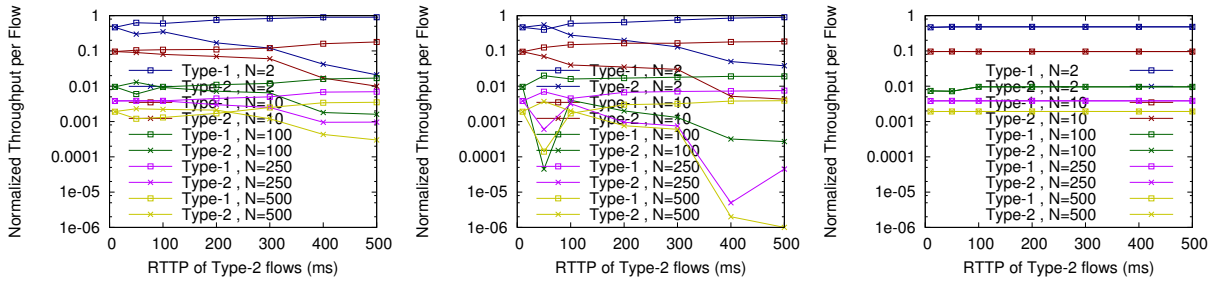
Fig. 3. RTTP estimation on a single buffer

IV. Evaluation

We now evaluate SplitBuff under diverse network scenarios using ns-2 simulations [10]. First, we evaluate the estimation accuracy of our RTTP measurements. Second, we compare the performance of SplitBuff with the single buffer case when different buffer sizing rules are used. Third, we compare SplitBuff with Blind-SplitBuff in which flows are mapped *randomly* to sub-buffers rather than based on their RTTs. Finally, we evaluate SplitBuff using real trace data collected from backbone Internet links. Unless stated otherwise, we use a dumbbell topology with a single bottleneck link. We use TCP SACK in all scenarios. All simulations are run for at least 500 s and results are averaged over 10 runs.

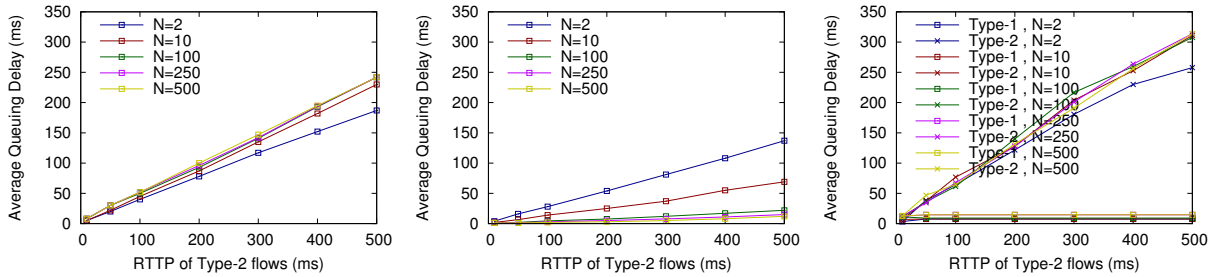
A. RTTP Estimation Accuracy

To evaluate the accuracy of estimated $\overline{RTTP}$ in SplitBuff, we conduct experiments under two scenarios with varying number of long-lived TCP flows: (a) Flows have homogeneous RTTP of 100 ms and (b) flows have heterogeneous RTTPs that are uniformly distributed in the interval [50 ms,150 ms] with a



(a) Throughput under BDP Rule (b) Throughput under SM Model (c) Throughput under SplitBuff

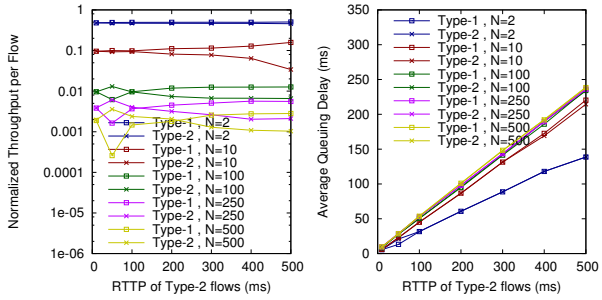
Fig. 4. Throughput comparison of (a) BDP Rule, (b) Stanford Model and (c) SplitBuff.



(a) Queueing delay under BDP Rule (b) Queueing delay under SM Model (c) Queueing delay under SplitBuff

Fig. 5. Queueing delay comparison of (a) BDP Rule, (b) Stanford Model and (c) SplitBuff.

mean of 100 ms. To avoid phase effects, flow starting times are chosen uniformly at random from [0 s,5 s]. Fig. 3 shows the estimated $\overline{RTTP}$ in both scenarios. Observe that the estimated RTTP closely approximates the actual $\overline{RTTP}$. In particular, the standard deviation remains within 5.2 ms and 7.7 ms of the actual RTTP under both scenarios, respectively.



(a) Throughput (b) Queuing Delay

Fig. 6. Performance under Blind-SplitBuff.

B. SplitBuff Evaluation

We now compare the performance of SplitBuff with the single buffer case under different buffer sizing rules as well as with Blind-SplitBuff. For ease of exposition, *short* and *long* RTT flows are termed as *Type-1* and *Type-2* flows, respectively.

1) Comparison with Single Buffer Models: For the single buffer case, we consider two buffer sizing rules: (a) BDP rule and (b) Stanford Model (SM) [2]. We fix the RTTP of Type-1 flows to 10 ms and vary the RTTP of Type-2 flows in the range [10 ms, 500 ms]. We also vary the number of TCP flows $N \in \{2, 10, 100, 250, 500\}$. In each simulation, there are $N/2$ Type-1 flows and $N/2$ Type-2 flows. We determine the normalized throughput (ratio of per-flow throughput and link capacity), average queueing delay and loss rate for both type of flows. Figures 4(a) and 4(b) show the throughput under the BDP and SM rules, respectively. Observe that across a range of flows, the throughput of long RTTP flows decrease substantially for both the models with increase in RTTP. This decline is much more under SM especially when N is large due to the fact that SM decreases buffer size with N. The queueing delay, as shown in Fig. 5(a) and 5(b), however, is much lower in SM compared to the BDP rule for the same reason.

With SplitBuff, both type of flows achieve the same throughput across a range of RTTP of Type-2 flows (see Fig. 4(c)). This happens because the aggressive Type-1 flows are sent to a separate sub-buffer, which limits their impact on Type-2 flows. Also observe that Type-2 flows experience slightly higher queueing delay compared to the BDP rule due to round-robin scheduling. However, the queuing delay of short flows reduces dramatically as shown in Fig. 5(c). Moreover, when both type of flows are considered, the average queuing delay remains less compared to the BDP rule.

2) Comparison with Blind-SplitBuff: We now compare SplitBuff with Blind-SplitBuff (that maps each flow randomly onto one of the sub-buffers) in order to isolate the benefits of using round-robin scheduling and RTT-awareness. We find that Blind-SplitBuff behaves similar to the single buffer case. We set the size of both sub-buffers in Blind-SplitBuff to $(BDP/2)$. Fig. 6(a) shows that for any number of flows, the throughput of long RTTP flows follow almost the same trend as shown in Fig. 4(a). However, the utilization of the bottleneck link improves because of splitting. The queuing delay is approximately the same in both sub-buffers as shown

in Fig. 6(b)) and is greater compared to SplitBuff when both types of flows are considered (see Fig. 5(c)). The queuing delays with Blind-SplitBuff are similar to the single buffer case (see Fig. 5(a)). These results show that random splitting of flows with round robin scheduling cannot improve fairness between short and long RTT flows because such flows can end up in the same sub-buffers. Therefore, it is the RTT-aware splitting which causes performance improvement.

C. Evaluation over Real RTT Trace Data

We now evaluate the performance of SplitBuff using trace data collected from some backbone Internet links [11], [12]. The RTT distribution was found to be heavy-tailed ranging from 10 ms to 1000 ms in [11]. In some traces captured from US, the distribution was found to be bimodal [12], with first mode at 180 ms (US coast-to-coast traffic with approximate $\overline{RTTP}$=130 ms) and second mode at 74 ms (US to Asian/European traffic with approximate $\overline{RTTP}$=60 ms). To mimic the RTT distribution in [12], we generate flows in ns-2 with RTTs that follow two different lognormal distributions corresponding to the two modes. For each distribution, we generate $N/2$ flows and categorize US coast-to-coast traffic as Type-1 flows and US Asian/European traffic as Type-2 flows. Observe that most of Type-1 flows will pass from sub-buffer-1 while most of the Type-2 flows will pass from sub-buffer-2, therefore the resulting RTTP estimate in each sub-buffer is close to the $\overline{RTTP}$ of the corresponding flows (see Table I).

To evaluate SplitBuff's performance with three sub-buffers, we define two thresholds for mapping flows to sub-buffers at $\overline{RTTP} - \frac{\sigma}{2}$ and $\overline{RTTP} + \frac{\sigma}{2}$, where σ is the standard deviation of RTTs. We observed a further improvement in the throughput of Type-2 flows. In particular, the normalized throughput was measured to be 0.0095 and 0.0096 for Type-1 and Type-2 flows, respectively. And the average of the queuing delays (T_q) for both type of flows was further reduced to 67.3 ms.

TABLE I
SPLITBUFF: FOR BIMODAL RTT DISTRIBUTION WITH N=100

Buffer Type	Flows	Estimated $\overline{RTTP}$ (ms)	Normalized Throughput	Queuing Delay (ms)
BDP Rule:				**Avg:** 108.7
	Type-1	-	0.012	-
	Type-2	-	0.0072	-
SplitBuff-2:				**Avg:** 98.25
Sub-buffer-1	Type-1	62.7	0.0106	52.2
Sub-buffer-2	Type-2	178.7	0.0086	144.4

V. RELATED WORK

Active Queue Management (AQM) schemes, such as RED [13] and ADT [14], *seek* to maintain low queues and high network throughput at the same time. However, choosing appropriate parameters for AQM schemes is a challenging task. Moreover, such schemes do not address issues that result due to the interaction of heterogeneous RTT flows.

Several buffer sizing rules have been proposed in the past which target reducing queue delays [1], [2], [3]. For instance, SM [2] proposes to reduce buffer sizes by leveraging the statistical multiplexing of flows. However, it can significantly increase the loss rate [3]. In addition, our results in Section IV show that it does not address throughput fairness issues. With SplitBuff, queuing delays can be reduced while maintaining high link utilization and low loss rates. In addition, it considerably improves throughput fairness.

The RD [15] services enable a user to choose between a higher transmission rate or low queuing delay at a congested network link through link scheduling and dynamic buffer sizing. However, RD does not differentiate between short and long RTT flows. Consequently, long flows can achieve very low throughput when competing with short flows. SplitBuff, on the other hand, improves delays as well as throughput fairness. Moreover, unlike RD services, SplitBuff does not maintain per-flow state.

VI. CONCLUSION

Flows on the Internet have heterogenous RTTs due to geographical distribution of clients. When short and long RTT flows compete at a bottleneck, they can adversely affect each other. We proposed SplitBuff, a framework which splits a buffer into multiple sub-buffers to improve performance in the presence of RTT heterogeneity. We showed the efficacy of SplitBuff using extensive ns-2 simulations. In the future, we plan to investigate the behavior of SplitBuff in the presence of rate-sensitive and delay-sensitive flows. We also plan to investigate whether it would be beneficial to increase the number of sub-buffers in SplitBuff.

REFERENCES

[1] C. Villamizar and C. Song, "High performance TCP in ANSNET," *ACM SIGCOMM CCR*, vol. 24(5), pp. 45–60, 1994.

[2] G. Appenzeller, I. Keslassy and N. McKeown, "Sizing router buffers," in *Proc. ACM SIGCOMM*, 2004.

[3] A. Dhamdhere, H. Jiang and C. Dovrolis, "Buffer sizing for congested internet links," in *Proc. IEEE INFOCOM*, 2005.

[4] Mathis, Matthew and Semke, Jeffrey and Mahdavi, Jamshid and Ott, Teunis, "The macroscopic behavior of the TCP congestion avoidance algorithm," *ACM SIGCOMM CCR*, vol. 27(3), July 1997.

[5] H. Jiang and C. Dovrolis, "Passive estimation of TCP round-trip times," in *Proc. ACM SIGCOMM*, 2002.

[6] D. Carra, K. Avrachenkov, S. Alouf, A. Blanc, P. Nain and G. Post, "Passive online RTT estimation for flow-aware routers using one-way Traffic," in *Proc. Networking*, 2010.

[7] N. Vasic, S. Kuntimaddi and D. Kostic, "One bit is enough: a framework for deploying explicit feedback congestion control protocols," in *Proc. COMSNETS*, 2009.

[8] I. A. Qazi and T. Znati, "On the design of load factor based congestion control protocols for next-generation networks," in *Proc. IEEE INFOCOM*, 2008.

[9] D. Katabi, M. Handley and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proc. ACM SIGCOMM*, 2002.

[10] Network Simulator. [Online]. Available: {http://www.isi.edu/nsnam/ns/}

[11] A. Acharya and J. Saltz, "A study of internet round-trip delay," University of Maryland, Tech. Rep. CS-TR-3736, 1996.

[12] S. Shakkottai, R. Srikant, N. Brownlee, A. Broido and K. Claffy, "The RTT distribution of TCP flows in the internet and its impact on TCP-based flow control," CAIDA, Tech. Rep., 2004. [Online]. Available: {http://www.ece.tamu.edu/~sshakkot/index_files/tr-2004-02.pdf}

[13] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Netw.*, Aug. 1993.

[14] R. Stanojevic, R. N. Shorten and C. M. Kellett, "Adaptive tuning of Drop-Tail buffers for reducing queueing delays," *IEEE Comm. Letters*, vol. 10(7), July 2006.

[15] M. Podlesny and S. Gorinsky, "Rd network services: differentiation through performance incentives," *ACM SIGCOMM*, 2008.