Dynamic Router's Buffer Sizing using Passive Measurements and P4 Programmable Switches

Elie Kfoury*, Jorge Crichigno*, Elias Bou-Harb† and Gautam Srivastava‡
*Integrated Information Technology, University of South Carolina, Columbia, U.S.A
†The Cyber Center For Security and Analytics, University of Texas at San Antonio, San Antonio, U.S.A
‡Department of Mathematics and Computer Science, Brandon University, Canada
Email: ekfoury@email.sc.edu, jcrichigno@cec.sc.edu, elias.bouharb@utsa.edu, srivastavag@brandonu.ca

***Abstract*—The router's buffer size imposes significant implications on the performance of the network. Network operators nowadays configure the router's buffer size manually and statically. They typically configure large buffers that fill up and never go empty, increasing the Round-trip Time (RTT) of packets significantly, and decreasing the application performance. Few works in the literature dynamically adjust the buffer size, but are implemented only in simulators, and therefore cannot be tested and deployed in production networks with real traffic.**

Previous work suggested setting the buffer size to the Bandwidth-delay Product (BDP) divided by the square root of the number of long flows. Such formula is adequate when the RTT and the number of long flows are known in advance. This paper proposes a system that leverages programmable switches as passive instruments to measure the RTT and count the number of flows traversing a legacy router. Based on the measurements, the programmable switch dynamically adjusts the buffer size of the legacy router in order to mitigate the unnecessary large queuing delays. Results show that when the buffer is adjusted dynamically, the RTT, the loss rate, and the fairness among long flows are enhanced. Additionally, the Flow Completion Time (FCT) of short flows sharing the queue is greatly improved. The system can be adopted in campus, enterprise, and service provider networks, without the need to replace legacy routers.

***Index Terms*—Buffer size, packet switching, programmable data planes, P4 language, congestion control, bandwidth-delay product, passive measurement.**

I. Introduction

Due to the bursty nature of Internet traffic, routers and switches are designed to include packet buffers. A buffer is essential to accommodate transient bursts and to absorb traffic fluctuations. By buffering traffic, bursts are smoothed and packet drop rates are significantly reduced [1].

The size of buffers imposes significant implications on the performance of the network (e.g., [2–5]). Based on the aforementioned observations and facts, one might think that large buffers are generally favorable to improve the network performance. This intuition is however incorrect; large buffers incur unnecessary delay since incoming packets must wait in the output buffer. The size of the buffer has to be appropriately selected so that packets are not suffering from long delays, while transient bursts are absorbed and packet drops are prevented. For a long time, the general rule-of-thumb was to configure the buffer size equal to the $C \cdot RTT$, where C is the capacity of the port and RTT is the average round-trip time (RTT) [6]. The quantity $C \cdot RTT$ is referred to as the bandwidth-delay product (BDP) hereon. Appenzeller et al. [7] then demonstrated that the buffer size can be reduced to BDP/$\sqrt{N}$, where N is the number of long (persistent over time) flows traversing the port.

The Internet is largely dominated by the Transmission Control Protocol (TCP) as the transport protocol to establish reliable end-to-end sessions. TCP has several variations of Congestion Control Algorithms (CCA) that employ different rate adaptation techniques. A class of CCA follows the Additive Increase Multiplicative Decrease (AIMD) model where TCP increases the congestion window by approximately one maximum segment size (MSS) every RTT until a packet loss event is detected. When packet loss is detected, the flow decreases its congestion window by half, and the cycle is repeated again. This class of CCA is referred to as loss-based CCA, and CUBIC, which is the default algorithm used in multiple Linux distributions and in recent versions of Windows and MacOS, falls under this category [8]. While the recommended buffer sizes apply particularly to loss-based CCA, experiments in this paper show that CCAs from various categories (e.g., delay-based, rate-based, etc.) still achieve the the full link utilization when the buffer size is set to BDP/$\sqrt{N}$.

Network operators today configure the router's buffer size manually and statically. The buffer size is often configured to large values, without considering the characteristics of flows (e.g., RTT) and dynamic traffic patterns. Large values cause the buffer to fill up and to never drain completely. Essentially, *ping* measurements running on the Internet today produce RTTs much larger than the propagation delay, suggesting that the networks are using large buffers [5].

Setting the router's buffer size to BDP/$\sqrt{N}$ would require determining the current average RTT and the number of flows, N. This could be achieved by passively capturing traffic crossing the router and forwarding it to a general-purpose CPU. However, such devices cannot cope with high traffic rates, especially in high-speed networks. Sampling techniques (e.g., NetFlow [9]) cannot be applied either since they are

This work was supported by the U.S. National Science Foundation, Office of Advanced Cyberinfrastructure, Awards #1925484 and #2118311. The authors recognize J. Gomez for his help with the network topology.

not accurate enough and often lose measuring information (§III-B). Programmable switches have lately emerged as a promising approach to customize data plane behavior [10]. Due to their high precision, low cost, and compute power, recent work [11], [12] has investigated using these switches as instruments to process traffic measurements at terabits per second rates.

A. Contributions

This paper proposes a cost-efficient scheme that dynamically modifies the router's buffer size based on current network conditions. The conditions are passively measured by tapping on the router's ports and forwarding the traffic to a programmable switch. The programmable switch tracks the number of long flows and computes their RTTs, which are used to modify the router's buffer to the newly determined size. The contributions are:

- Devising a scheme that relies on passive measurements, which can process traffic at terabits per second rates.
- Identifying long flows, tracking their counts, and measuring their RTTs entirely in the data plane.
- Using network measurements to modify the buffer size of the router on-the-fly, to improve performance.
- Reducing queuing delays and improving the fairness of flows, regardless of the CCA.
- Improving the flow completion time of short flows sharing the bottleneck link with long flows.

The rest of this paper is organized as follows. Section II describes the related work. Section III describes the proposed system. Section IV presents the experimental setup and compares the performance of the proposed system against a network where the buffer size is fixed. Section V concludes the paper and describes future work.

II. Related Work

A. Sizing router buffers

Static buffer tuning. Villamizar et al. [6] established the initial rule-of-thumb, which states that the buffer size is equal to the BDP. Appenzeller et al. [7] demonstrated that the buffer size can be reduced to BDP/$\sqrt{N}$, which has been extensively evaluated using static buffer sizes [4, 5, 13, 14]. Dhamdhere et al. [15] considered packet loss rates in their formula, and argued that to limit the maximum loss rate, the buffer should be proportional to the number of long flows. Spang et al. [3] reported on measurement of Netflix video traffic to understand how the buffer size affects the quality of the videos. The authors statically configured different buffer sizes and observed that buffers that are too small and too large worsen quality. Beheshti et al. [16] presented buffer sizing experiments at Facebook. The authors manually modified buffer sizes and noted that small buffers produce tolerable degradation in some metrics (e.g., packet drop rates) and significant enhancements in others (e.g., latency). No conclusive adequate buffer size is provided.

Dynamic buffer tuning. The idea of dynamically modifying the buffer started with Flow Proportional Queuing (FPQ) [17] which adjusts the amount of buffering according to the number of TCP flows. Further schemes [18–20] also considered modifying the buffer size based on network traffic. However, such schemes have a main limitation; they assume that their methods will be implemented on contemporary routers. Such process is length and costly and most likely, router manufacturers will not modify their existing devices [21]. In fact, such schemes and other Active Queue Management (AQM) algorithms have been proposed more than ten years ago, and are still not implemented on contemporary routers in the market today, though the problem of buffer sizing is still being thoroughly researched [5].

B. Measurements using programmable data planes

Ghasemi et al. [11] proposed Dapper, a system that uses programmable switches to diagnose the cause of congestion. Metrics estimated in Dapper include the Round-trip Time (RTT), in-flight bytes, and loss rate. Chen et al. [22] proposed a system that leverages programmable switches to passively compute the RTT of TCP traffic. Kagami et al. [12] proposed CAPEST, a method that collects timing information to estimate the network capacity and the available bandwidth. Kfoury et al. [23] proposed a P4-based method to automate end-hosts' TCP pacing.

III. Proposed System

Fig. 1 illustrates an overview of the proposed system. The scheme can be used in both access and core networks. The steps to dynamically modify the router's buffer size are: 1) a copy of the traffic is forwarded to a programmable switch by passively tapping on routers' ports; 2) the programmable switch identifies, tracks, and computes the RTT of long flows. Afterwards, the computed statistics are pushed to the control plane where the buffer size is calculated; and 3) the programmable switch modifies the legacy router's buffer size.

It is worth noting that programmable switches are significantly cheaper than contemporary switching/routing devices. This is because they are whiteboxes, and are equipped with

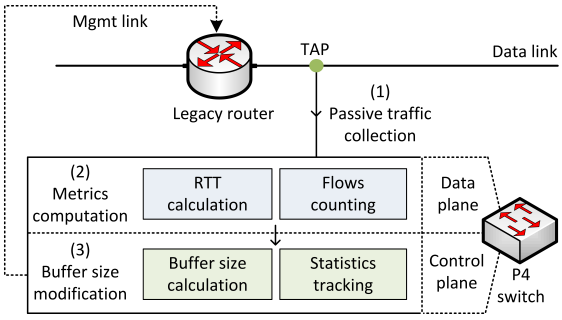

Fig. 1. High-level system overview. Step (1): a copy of the traffic is forwarded by the TAP to the P4 switch. Step (2): the RTT of individual flows is computed at the P4 switch's data plane. Step (3): The P4 switch's control plane modifies the router's buffer size according to the equation BDP/$\sqrt{N}$.

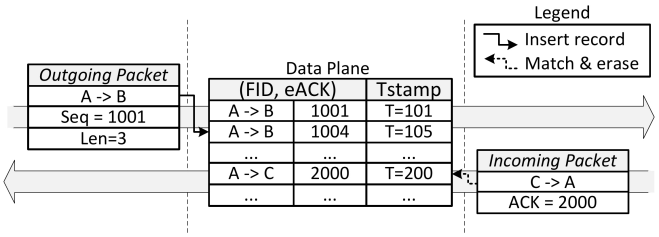

Fig. 2. RTT calculation in the data plane [22].

a small packet buffer. This makes the proposed system cost-efficient, especially since a single programmable switch can be used as a passive instrument for several routers.

A. Passive traffic collection

In Step (1), Fig. 1, a copy of the traffic is forwarded to the switch. The network TAP is a hardware device that operates at the physical layer and provides full visibility of data streams. The TAP captures the exact traffic even when the network is saturated. Computing the RTT (see §III-B) requires associating packets' acknowledgment numbers with sequence numbers. Such process is only achievable by using bidirectional TAPs. Note that a network TAP does not alter timing information and packet orders, which may occur with other schemes such as port mirroring operating at layer 2 and layer 3 [24].

B. Metrics computation

The proposed system adopts the method presented in [22] to calculate the RTT[1]of individual flows. The idea is to relate the TCP sequence (SEQ) and acknowledgement (ACK) numbers of incoming and outgoing packets. The RTT can then be inferred by calculating the time difference between the two packets. Consider Fig. 2. For each outgoing packet, the flow identifier (FID) of the packet is computed by hashing the 5-tuple (source/destination IP, source/destination port, and protocol), and the expected ACK (eACK) is calculated by adding the SEQ number to the length of the payload. The timestamp of the current packet is stored in a table indexed by the FID and the eACK. Upon receiving an incoming TCP packet, the FID and the ACK number are used to fetch the table for an existing record. If there is a match, the timestamps are subtracted, and an RTT sample is produced. In reality, devices might not acknowledge every packet (e.g, with delayed ACK, a device might send a single ACK for multiple packets). Since the memory on programmable switches is limited (tens of megabytes), it is not possible to infinitely store records. The solution is to use a timeout threshold that, once exceeded, will force the eviction of the corresponding record. The method further uses multi-stage hash due to the constraints on accessing the data plane memory.

[1]The source code for measuring the RTT in Tofino is publicly available at https://github.com/Princeton-Cabernet/p4-projects/tree/master/RTT-tofino

Algorithm 1: Long flows counting algorithm

input : Packet headers hdr; flow identifier FID; switch timestamp S_{tstamp}; flows timestamps $tstamps$, indexed by FID; packet counts $counts$, indexed by FID; current flow count C, time threshold `T_THRESH`, packet count threshold `C_THRESH`

output: Updated long flow count

begin
- $prev\_tstamp \leftarrow tstamps[FID]$
- $tstamps[FID] \leftarrow S_{tstamp}$
- **if** $tstamps[FID] - prev\_tstamp <$ `T_THRESH` **then**
  - **if** $counts[FID] =$ `C_THRESH` **then**
    - $C \leftarrow C + 1$
  - **else**
    - $counts[FID] \leftarrow counts[FID] + 1$
- **else**
  - $counts[FID] \leftarrow 0$
- **if** $hdr.tcp.flags =$ `FIN` **then**
  - **if** $counts[FID] =$ `C_THRESH` **then**
    - $C \leftarrow C - 1$

C. Buffer size modification

Each individual RTT sample is pushed to the control plane of the switch where a smoothed RTT (SRTT) is calculated. The smoothed RTT is computed as follows:

$$SRTT_{i+1} = \alpha SRTT_i + (1 - \alpha) RTT_i, \qquad (1)$$

where $SRTT_{i+1}$ is the new smoothed average, α is the smoothing factor ($0 \leq \alpha \leq 1$), and RTT_i is the RTT sample retrieved from the data plane. The value of $\alpha = 0.875$ is used in this work.

The number of flows in the buffer size formula (BDP / $\sqrt{N}$) is based on the count of long flows sharing the bottleneck link. Short flows on the other hand are not considered since they have very small effect on the buffer [7]. Consequently, there is a need to differentiate between long flows and short flows in order to determine N. Algorithm 1 outlines the steps implemented in the proposed system to identify the long flows in the data plane. The idea is to check whether the number of packets from the same flow exceeds a predefined threshold (`C_THRESH`) in a certain time window. Upon receiving a packet, the previous timestamp of the flow to which the

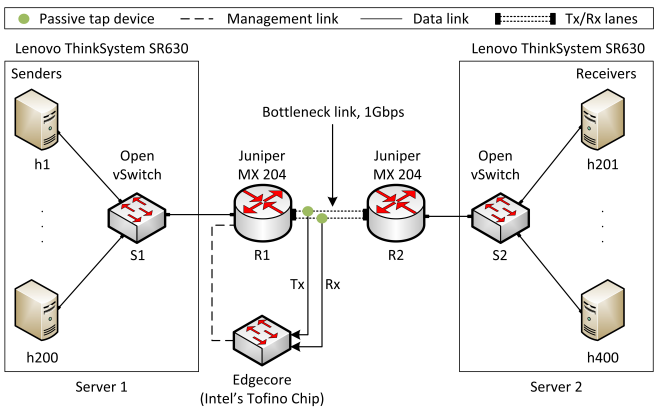

Fig. 3. Topology used for the experiments.

packet belongs is fetched and the flow's current timestamp is updated. If the difference between the current and the previous timestamps is below a predefined threshold (`T_THRESH`), and if the packet count reaches `C_THRESH`, then the flow is identified as a long flow. The counts are incremented if packets are received within the time window; otherwise, they are reset to zero. Finally, if the packet has the FIN TCP flag present (flow is leaving) and if it was a long flow, the counter C is decremented.

Since the $SRTT$ and N are now identified, the buffer size is set to BDP/$\sqrt{N}$, where BDP=$C \cdot SRTT$. Note that the capacity of the port C is known in advance and can be hard-coded and configured by the operator.

$SRTT$ can change over time, even when N remains the same. For instance, a flow might use an alternative path that has an average RTT significantly different from that of the initial path. In such scenarios, the buffer size must be recomputed and reconfigured on the legacy router. Since $SRTT$ is being computed continuously, the proposed system computes the difference between the current $SRTT$ and the $SRTT$ recorded when the buffer size was last modified. If there is a significant difference between the two (i.e., $|SRTT - R\_SRTT| > \gamma \times R\_SRTT$, where R_SRTT is the recorded $SRTT$, and γ is a configurable fraction of R_SRTT that triggers a buffer size recalculation), the buffer size is recalculated.

IV. Experimental Setup and Results

Fig. 3 shows the topology used to conduct the experiments. The topology consists of 200 senders (h1, h2, ..., h200), each opening a TCP connection to a corresponding receiver (h201, h202, ..., h400). The hosts in the experiments are network namespaces in Linux started through Mininet [25] on a physical server. The emulation was carefully designed, and sufficient resources were allocated (usage of CPUs was at all times below prudent levels), avoiding misleading results. The 200 senders are long flows running iPerf3 to conduct the measurements. The senders are connected to an Open Virtual Switch (OVS) (S1), which is bridged to the server's (Server 1) network interface. The server's interface is connected to a Juniper router MX-204 (R1). The bandwidth of the link connecting switch S1 and router R1 is 40Gbps, while the bandwidth of the link connecting router R1 and router R2 is 1Gbps. Since the former bandwidth exceeds the latter, the queue will buildup at router R1. The size of the TCP send and receive buffers on the end-hosts was set to a large value (200MB). For all the experiments, the assumed default buffer size of the router is 200ms, which reflects large buffers configured in production networks [5]. Edgecore Wedge100BF-32X [26] is the programmable P4 switch used. It is equipped with a programmable ASIC chip (Intel's Tofino) that operates at 3.2 Tbps.

Two scenarios are considered. The first scenario uses the default buffer size on the router, without any dynamic modification. We refer to this scenario as "wo/ buffer modification". The second scenario uses the programmable P4 switch to measure and modify the buffer size of the router. We refer to this scenario as "w/ buffer modification".

A. Test 1: Multiple long flows, CCAs, and propagation delays

This experiment evaluates the average Jain's fairness index ($\overline{\mathcal{F}}$), the average link utilization ($\overline{\rho}$), and the average RTT ($\overline{RTT}$) of various number of long flows (N) belonging to the same CCA, considering various propagation delays (RTT_{prop}). Moreover, the experiment includes results for overlapped flows belonging to different CCAs.

The results of this test are depicted in Fig. 4. The first row shows $\overline{\rho}$; the second row shows $\overline{\mathcal{F}}$; and the third row shows $\overline{RTT}$. Each row is further divided into three sub-rows that represent the configured RTT_{prop} (20ms, 50ms, and 100ms). The columns represent N for each CCA. The last column in each scenario refers to the total number of overlapped flows belonging to *different* CCAs (N_{mixed}). The distribution of those flows is extracted from [27] which characterized the CCAs deployed in the wild for the Alexa Top 20,000 websites. BBR is excluded from the *mixed* case due to the effect it has on the performance of other CCAs, regardless of the buffer size.

As expected, without buffer modification, $\overline{\rho}$ was 100% regardless of the experiment settings. This is because the

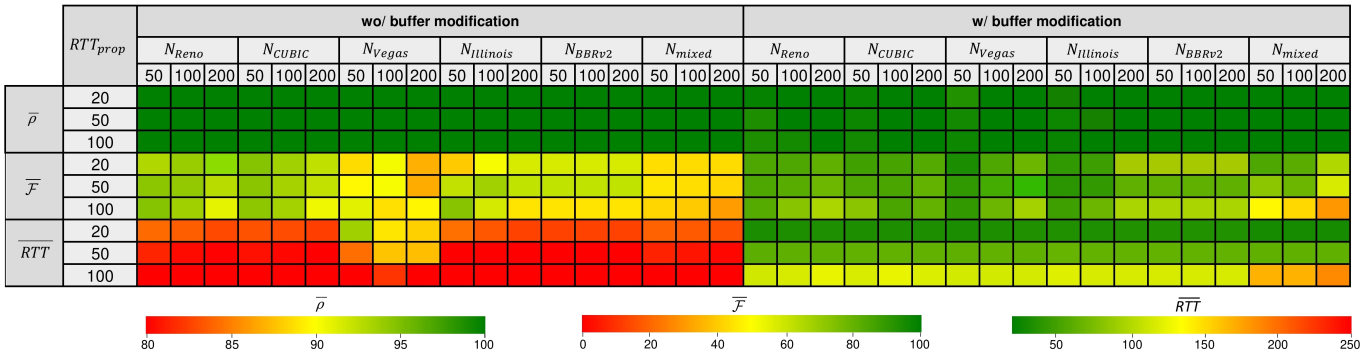

Fig. 4. Average link utilization ($\overline{\rho}$), average fairness index ($\overline{\mathcal{F}}$), and average RTT ($\overline{RTT}$) with various number long flows belonging to different CCAs, and considering various propagation delays (RTT_{prop}). The results for both scenarios (wo/ buffer modification and w/ buffer modification) are demonstrated.

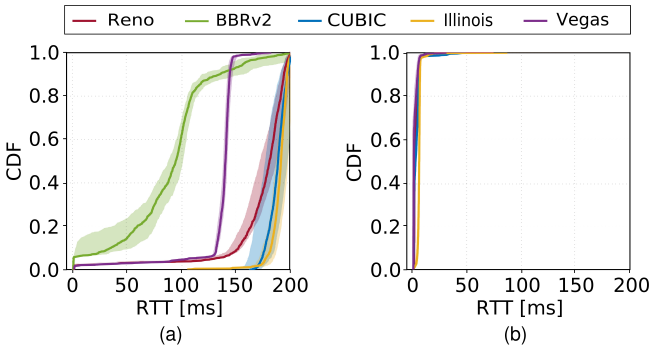

Fig. 5. CDF of $\overline{RTT}$ of short flows. (a) w/out buffer modification. (b) w/ buffer modification. The shaded area demonstrates the CDF of the minimum RTT and the maximum RTT of each connection.

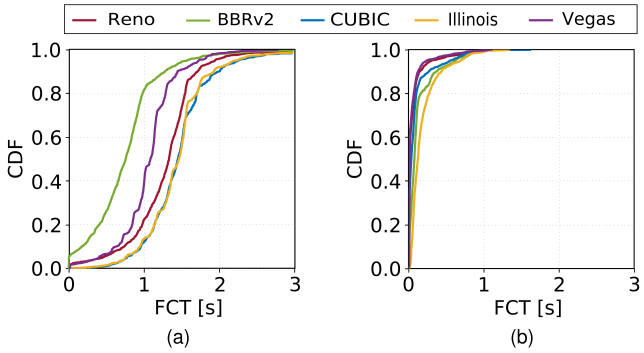

Fig. 6. CDF of the average FCT of short flows. (a) w/out buffer modification. (b) w/ buffer modification.

buffer is significantly large (200ms). With buffer modification, $\overline{\rho}$ was also $\approx 100\%$ in all settings. This is interesting as the buffer is significantly smaller than the default setting, and yet, $\overline{\rho}$ is $\approx 100\%$, even with non loss-based CCA. Recall that the BDP/$\sqrt{N}$ was originally designed based on the dynamics of loss-based CCA. However, it can be seen that this formula also works well with more recent CCAs (e.g., BBR). Note that this experiment was also executed for smaller N, and the results are consistent with the reported ones. The proposed system also provides a major improvement in $\overline{\mathcal{F}}$ when compared to the results of a fixed large buffer. Such improvements are valid regardless of N, or the CCA to which the N flows belong.

Finally, perhaps the most clear improvement is in $\overline{RTT}$. With the exception of TCP Vegas and BBR, all other CCAs have $\overline{RTT} \approx$ 250ms, regardless of the current RTT_{prop}. TCP Vegas and BBR also suffer from an increased $\overline{RTT}$ when RTT_{prop} is large (100ms). With buffer modification, $\overline{RTT}$ is only slightly larger than RTT_{prop}. This means that minimal queueing delay is contributing to $\overline{RTT}$. This improvement is valid regardless of N, or the CCA to which the N flows belong.

B. Test 2: Performance of short flows sharing the bottleneck with long flows

This experiment evaluates $\overline{RTT}$ and the Flow Completion Time (FCT) of short flows when sharing the bottleneck link with long flows, in both scenarios. FCT is defined as the amount of time that elapses from when the first packet is sent until the last packet reaches the destination. In this test, 1000 short flows are arriving according to a Poisson process, and the flow size distribution resembles a web search workload. The flow sizes ranged from 10KB to 1MB. In addition to the short flows, there are 200 competing long flows with RTT_{prop} =50ms.

Fig. 5 shows the Cumulative Distribution Functions (CDFs) of the minimum RTT, maximum RTT, and $\overline{RTT}$, for both scenarios. Without buffer modification (Fig. 5 (a)), $\overline{RTT}$ of all CCAs is significantly high. BBRv2 has the lowest $\overline{RTT}$ among the CCAs, and the results are aligned with [8]. Nevertheless, approximately 50% of the flows have an $\overline{RTT}$ $>$ 100ms. On the other hand, when the buffer is modified (Fig. 5 (b)), the $\overline{RTT}$ of all flows is $<$ 10ms, indicating a significant improvement for all CCAs.

Fig. 6 shows the CDF of the FCTs of short flows in both scenarios. There is an improvement when the buffer size is automatically adjusted. For example, when the buffer is not modified, most flows had FCTs between 1 and 3 seconds. On the other hand, when the buffer is modified, 90% of the flows had FCTs smaller than $\approx$ 0.5 seconds.

C. Test 3: Long flows with different emulated propagation delays

This experiment evaluates the link utilization, RTT, and packet loss rate of long flows in both scenarios. The experiment consists of 100 long flows, divided into four groups of each 25 flows each. Each group starts three minutes after the other. The propagation delays of the flows are as follows: group1, 20ms; group2, 50ms; group3, 30ms; and group4, 100ms. The CCA used in this test is CUBIC.

Fig. 7(a) shows the results when the buffer size is not modified. Each vertical gray stripe denotes a new joining group of flows. The figure shows that the link was fully utilized and there are some packet losses throughout the test with high variations. The RTT of all flows is significantly high, exceeding 200ms, irrespective of per-group RTT_{prop}. Fig. 7(b) shows the results when the buffer is modified. The improvements here can be specifically seen in the RTT of each group of flows which approximates the propagation delay. Note the sudden increase in packet loss when new flows join. This is happening because the buffer size is being adjusted while the buffer is already holding packets. Moreover, since the buffer is already small, the burstiness of the joining flows is not absorbed, causing packets to be dropped. Note however the steadiness in the packet loss variations. While the scheme (as expected) slightly increases the losses compared to deep buffers, their effects can be tolerated, especially with the significant improvements in RTT.

V. Conclusion and Future Work

This paper presented a scheme that uses passive measurements collected by programmable P4 switches to determine

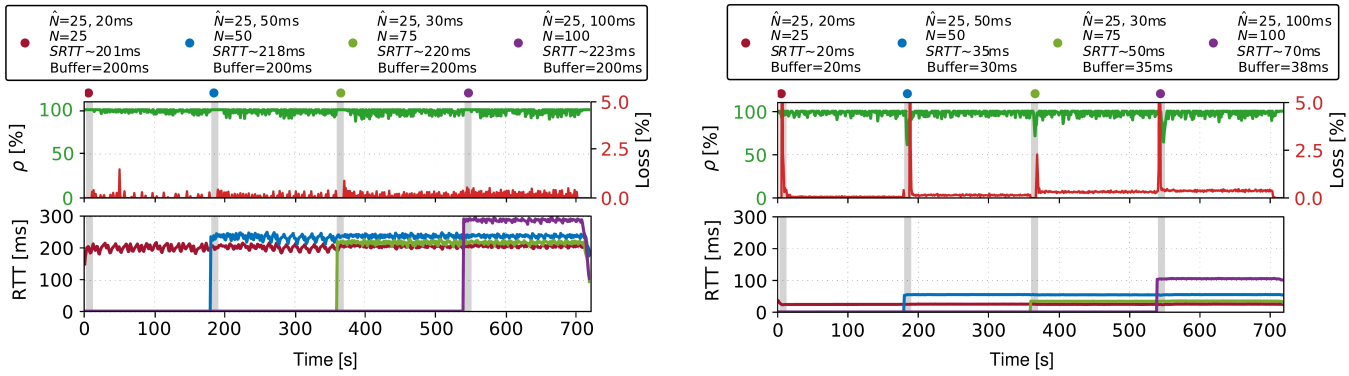

Fig. 7. Link utilization (ρ), packet loss, and RTT, considering various RTT_{prop}. The reported RTT is computed for each RTT_{prop} group. $\hat{N}$ is the current number of joining flows. The CCA used in this experiment is CUBIC. (a) w/out buffer modification. (b) w/ buffer modification.

the RTT and the number of flows crossing a legacy router. These variables are used to modify the buffer size of the router based on the well-known formula (BDP/$\sqrt{N}$). Experiments conducted on real hardware demonstrate the improvements in the RTT, packet loss rate, fairness, and FCT of flows over a network where the buffer size is statically configured. The formula BDP/$\sqrt{N}$ was criticized for inducing large packet losses in some scenarios. For future work, the authors plan to measure the packet loss rate among other metrics, and use them in the buffer size selection algorithm. Furthermore, the authors plan to test the scheme on a production network with traffic that includes a variety of CCAs, non-TCP streams, and media traffic.

References

[1] G. Huston, "Sizing the buffer, APNIC," [Online], Available: https://blog.apnic.net/2019/12/12/sizing-the-buffer/.

[2] N. Beheshti, P. Lapukhov, and Y. Ganjali, "Buffer sizing experiments at Facebook," in *Proceedings of the 2019 Workshop on Buffer Sizing*, December, 2019.

[3] B. Spang, B. Walsh, T.-Y. Huang, T. Rusnock, J. Lawrence, and N. McKeown, "Buffer sizing and video QoE measurements at Netflix," in *Proceedings of the 2019 Workshop on Buffer Sizing*, Palo Alto, CA, USA, December, 2019.

[4] N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, and G. Salmon, "Experimental study of router buffer sizing," in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, Vouliagmeni, Greece, October, 2008.

[5] N. McKeown, G. Appenzeller, and I. Keslassy, "Sizing router buffers (redux)," *ACM SIGCOMM Computer Communication Review*, vol. 49, no. 5, pp. 69–74, 2019.

[6] C. Villamizar and C. Song, "High performance TCP in ANSNET," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 5, 1994.

[7] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, August, 2004.

[8] E. F. Kfoury, J. Gomez, J. Crichigno, and E. Bou-Harb, "An emulation-based evaluation of TCP BBRv2 alpha for wired broadband," *Computer Communications*, vol. 161, pp. 212–224, 2020.

[9] B. Claise, G. Sadasivan, V. Valluri, and M. Djernaes, "RFC 3954: Cisco systems netflow services export version 9," October, 2004.

[10] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on P4 programmable data plane switches: taxonomy, applications, challenges, and future trends," *IEEE Access*, 2021.

[11] M. Ghasemi, T. Benson, and J. Rexford, "Dapper: data plane performance diagnosis of TCP," in *Proceedings of the Symposium on SDN Research*, April, 2017.

[12] N. Kagami, R. da Costa, and L. Gaspary, "Capest: Offloading network capacity and available bandwidth estimation to programmable data planes," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, March, 2020.

[13] A. Dhamdhere and C. Dovrolis, "Open issues in router buffer sizing," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, January, 2006.

[14] Y. Ganjali and N. McKeown, "Update on buffer sizing in internet routers," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 5, October, 2006.

[15] A. Dhamdhere, H. Jiang, and C. Dovrolis, "Buffer sizing for congested Internet links," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 2. IEEE, 2005, pp. 1072–1083.

[16] N. Beheshti, O. Baldonado, and P. Lapukhov, "Buffer sizing experiments at Facebook," in *2019 Workshop on Buffer Sizing*, 2019.

[17] R. Morris, "Scalable TCP congestion control," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*, vol. 3. IEEE, 2000, pp. 1176–1183.

[18] C. M. Kellett, R. N. Shorten, and D. J. Leith, "Sizing Internet router buffers, active queue management, and the Lur'e problem," in *Proceedings of the 45th IEEE Conference on Decision and Control*. IEEE, 2006, pp. 650–654.

[19] Y. Zhang and D. Loguinov, "ABS: adaptive buffer sizing for heterogeneous networks," in *2008 16th Interntional Workshop on Quality of Service*. IEEE, 2008, pp. 90–99.

[20] J. Lee, T. Tan, D. Kakadia, E. M. D. Reyes, and M. G. Lam, "Dynamic setting of optimal buffer sizes in IP networks," US Patent 8,223,641, July, 2012.

[21] X. Chen, S. L. Feibish, Y. Koral, J. Rexford, O. Rottenstreich, S. A. Monetti, and T.-Y. Wang, "Fine-grained queue measurement in the data plane," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, 2019, pp. 15–29.

[22] X. Chen, H. Kim, J. M. Aman, W. Chang, M. Lee, and J. Rexford, "Measuring TCP round-trip time in the data plane," in *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, August, 2020.

[23] E. F. Kfoury, J. Crichigno, E. Bou-Harb, D. Khoury, and G. Srivastava, "Enabling TCP pacing using programmable data plane switches," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, 2019, pp. 273–277.

[24] J. Zhang and A. Moore, "Traffic trace artifacts due to monitoring via port mirroring," in *2007 Workshop on End-to-End Monitoring Techniques and Services*, Munich, Germany, May, 2007.

[25] R. De Oliveira, C. Schweitzer, A. Shinoda, and R. Prete, "Using mininet for emulation and prototyping software-defined networks," *2014 IEEE Colombian Conference on Communications and Computing (COLCOM), Bogota, Colombia, June, 2014.*

[26] "Wedge 100BF-32X, 100GbE data center switch, Barefoot Networks, an Intel® company," [Online], Available: https://tinyurl.com/sy2jkqe.

[27] A. Mishra, X. Sun, A. Jain, S. Pande, R. Joshi, and B. Leong, "The great Internet TCP congestion control census," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2019.