Reducing the Impact of RTT Unfairness using P4-Programmable Data Planes

Jose Gomez[a], Elie F. Kfoury[a], Jorge Crichigno[a], Gautam Srivastava[b,c,d]
[a]Integrated Information Technology Department, University of South Carolina, Columbia, SC, USA
[b]Department of Mathematics and Computer Science, Brandon University, Canada
[c]Research Centre for Interneural Computing, China Medical University, Taichung, Taiwan
[d]Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon
{gomezgaj, ekfoury}@email.sc.edu, jcrichigno@cec.sc.edu, srivastavag@brandonu.ca

***Abstract*—This paper presents a system that mitigates the Round-trip Time (RTT) unfairness issue in non-programmable networks using P4-programmable data planes. In traditional loss-based congestion control algorithms (CCAs), RTT unfairness occurs when the flows with shorter RTTs obtain higher bandwidth shares with respect to the flows with longer RTTs. This behavior occurs due to the faster recovery period that flows with shorter RTTs experience after a loss event. On the other hand, more recent CCAs, such as the Bottleneck Bandwidth and Round-trip Time (BBR), present the opposite behavior, where the flows with longer RTTs achieve higher throughput than the ones with shorter RTTs.**

In this paper, the proposed system employs a P4-programmable data plane to monitor the RTT of flows traversing a non-programmable router at line rate using passive taps. The P4-programmable data plane analyzes the RTT of each flow, subsequently segregating them into different queues. This separation is aimed at minimizing the interaction between flows with varying RTTs. Results show that implementing flow separation improves the fairness of long flows, reduces the RTT of individual flows allocated in different queues, and improves the Flow Completion Times (FCTs) of short flows.

***Index Terms*—P4, RTT unfairness, Transmission Control Protocol (TCP), Congestion Control Algorithm (CCA), Bottleneck Bandwidth and Round-trip Time (BBR).**

I. Introduction

The Transmission Control Protocol (TCP) plays a fundamental role in establishing reliable end-to-end communication. One of the fundamental functions of TCP is congestion control, which is a mechanism that constantly aims for the highest data transfer rate that can be used without causing network congestion. Congestion control algorithms (CCAs) typically operate as a function of the Round-trip Time (RTT), where they probe for a target transfer rate and use the RTT as a metric to determine the amount of data to inject into the network. CCAs adjust the sending rate based on feedback received during data transfer cycles, either increasing or decreasing it for the next cycle. This design that relies on RTT often results in a problem known as RTT unfairness [1]. RTT unfairness significantly impacts network performance in transport protocols. It typically occurs when two senders located at different distances from the receiver share the same bottleneck link. In such a scenario, these competing data flows go through varying recovery times after a congestion event, leading to an unequal bandwidth distribution. Traditional loss-based CCAs such as CUBIC [2] experience RTT unfairness, where flows with longer RTTs face significant disadvantages compared to those with shorter RTTs. CUBIC flows with shorter RTTs can probe the network more frequently, increasing their sending rate faster than flows with longer RTTs. This situation results in suboptimal network utilization and decreased overall Quality of Service (QoS). Conversely, newer CCAs like Bottleneck Bandwidth and Round-trip Time (BBR) [3] exhibit the opposite behavior, with flows having longer RTTs achieving higher throughput [4–6]. BBR aims to identify Kleinrock's optimal operating point [7] and restrict the amount of data in transit to one Bandwidth-delay Product (BDP), calculated as the product of the RTT and the bottleneck bandwidth (Btlbw) (i.e., BDP=RTT$\times$Btlbw). Consequently, BBR flows with longer RTTs obtain a larger bandwidth share.

This paper presents a system that utilizes P4-programmable data planes to identify TCP flows, calculate their RTTs, and classify them into distinct queues, aiming to reduce the impact of RTT unfairness. P4 is a programming language that enables the definition of how data planes process network packets [8]. The system employs passive optical taps and a P4-programmable switch as a measurement instrument. The P4-programmable switch receives a mirrored copy of traffic from a bottleneck link between two non-programmable routers. Then, the P4-programmable switch generates a set of rules to configure a non-programmable router to segregate TCP flows in different queues. This reallocation results in reducing the impact of the RTT unfairness. The P4-programmable switch implements the Jenks optimization method [9] to segregate flows with similar RTTs into separate queues. Additionally, the system adjusts the buffer size to minimize delays caused by bufferbloat [10]. The results demonstrate the system's capability to identify and classify TCP flows while improving fairness, reducing the RTT of individual flows resulting from queueing delay, and improving the Flow Completion Times (FCTs) of short flows.

A. Contributions

This paper proposes a system that separates TCP flows according to their RTT. The traffic is passively measured by

tapping on the egress interface of a non-programmable router. The P4-programmable data plane separates the flows using a classification algorithm. The contributions of this paper can be listed as follows:

- Reducing the impact of RTT unfairness observed when long TCP flows share a bottleneck link.
- Implementing a system that categorizes and segregates flows into distinct queues based on their RTT. This system leverages the programmability and granularity offered by P4-programmable data planes, enabling it to execute actions in non-programmable routers.
- Enhancing multiple performance objectives [11] such as fairness, FCTs, and RTTs of individual flows. Isolating the dynamics of competing TCP flows results in a more efficient utilization of resources.

The rest of the paper is organized as follows: Section II presents background on P4-programmable data planes and related works. Section III describes the proposed system. Section IV describes the experiments and the results. Section V concludes the paper.

II. Background and Related Work

A. P4-programmable Data Planes

P4-programmable data planes offer programmers the flexibility to define how packets are processed by a pipeline [8]. A P4 processing pipeline consists of three key components: a programmable parser, a programmable match-action pipeline, and a programmable deparser. The programmable parser interprets the stream of bits the switch receives and organizes them in standard and custom header fields previously defined by the programmer. Then, the match-action pipeline executes operations on packet headers and intermediate results. Lastly, the deparser reassembles the packet headers and serializes them for transmission.

P4-programmable data planes provide high-precision timers with nanosecond granularity and stateful memories such as registers, counters, and meters, all of which can be accessed at a line rate. These capabilities enable the execution of per-packet operations, which have found extensive use in adding visibility to network events and enhancing network performance [12–14]. In this paper, the data plane of a P4-programmable switch is programmed to identify and calculate the RTTs of individual flows. Then, the control plane employs a classification algorithm to determine the allocation of these flows into different queues.

B. RTT Unfairness Characterization

Gavaletz and Kaur [1] proposed a novel method for analyzing and decomposing the sources of RTT unfairness in transport protocols. They demonstrated that the sources of RTT unfairness vary between protocols. The authors also proposed a modification to TCP called FairTCP that addresses the feedback delay component of RTT unfairness by allowing connections to obtain more accurate congestion feedback. Tao *et al.* [15] proposed a mathematical model to study the problem of RTT unfairness in the BBR CCA. The authors evaluated a queueing model that captures the interactions between BBR flows and the underlying network. The authors also proposed a modification to BBR that addresses this issue by adjusting the pacing rate during the bandwidth probing phase based on the RTT of the connection.

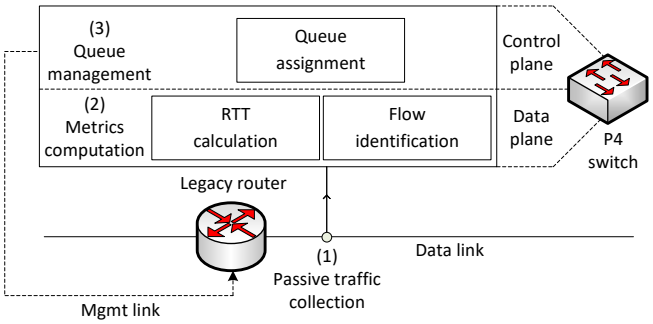


Fig. 1. High-level system overview. Step (1): a copy of the traffic is forwarded by the tap to the data plane of the P4 switch. Step (2): the RTT of each TCP flow is calculated and identified at the P4 switch's data plane. Step (3): A classification algorithm running in the P4 switch's control plane assigns a queue to each flow.

C. Flow Separation

Kfoury *et al.* [16] addressed the impact of buffer size at bottleneck routers on network application performance. The authors highlight the challenges of static buffer configurations, which can lead to increased packet losses, reduced link utilization, and higher latency. The paper proposes P4BS, a dynamic buffer sizing system that leverages programmable switches to measure key metrics such as long-lived flow counts, RTTs, packet loss rates, and queueing delays. P4BS optimizes buffer sizes using these metrics to minimize queueing delays and packet loss rates. The system was implemented and tested on a Tofino hardware switch, demonstrating improved quality of service across various applications like web browsing, video streaming, and voice over IP. In another work, Kfoury *et al.* [17] identified CCAs using P4-programmable data planes. The goal of this work is to reduce the impact of CCAs that present more aggressive behavior in terms of fairness and link utilization. P4CCI achieves this by computing and extracting the "bytes-in-flight" metric for each flow and then passing this data to a deep learning model for classification. Once classified, flows are sorted into dedicated queues based on their respective CCA types. The system was implemented and tested on real hardware. The results show that P4CCI's ability to detect and separate CCAs accurately leads to an enhancement in network performance.

III. Proposed System

A. Overview

Fig. 1 presents a high-level overview of the proposed system. This system employs passive taps to collect traffic from the data link of a non-programmable router by creating a copy of the traffic without causing any performance disruptions. Then, this traffic is directed to the data plane of a P4-programmable switch. Within this switch, RTT calculations

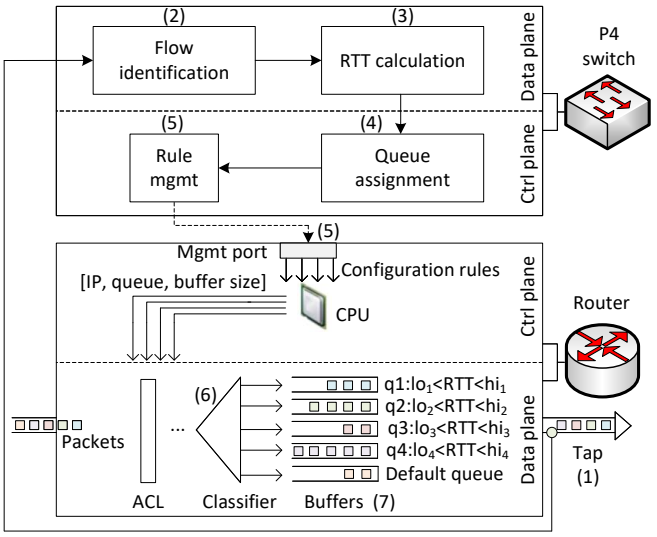


Fig. 2. Proposed system architecture. The P4 switch serves as a passive measurement tool for calculating the RTT on a per-flow basis.

are made for each individual flow, with a focus on TCP flows specifically. Once these flow metrics are computed, they are transmitted to the control plane. At this point, a classification algorithm is applied to determine which queues of the non-programmable router should receive the respective flows. This allocation action is executed via the management port of the non-programmable router.

Consider the architecture of the proposed system depicted in Fig. 2. The steps to identify flows, calculate the RTTs, and separate the flows in different queues are as follows:

1) The system maintains continuous passive monitoring of the traffic flowing through the non-programmable router. To achieve this, a passive tap device is deployed in the router's egress interface. The tap redirects the traffic to the data plane of a P4 switch that operates at line rate.
2) The packets are processed by the data plane of a P4 switch. In this stage, packets are parsed and the flows are identified.
3) The information resulting from the flow identification is transmitted to the RTT calculation module, which is responsible for pairing each flow with its respective RTT.
4) The data plane of the P4 switch sends to the control plane the RTTs of each flow to the classification module that implements the Jenks natural breaks algorithm [9] to separate the flows in different queues.
5) After the P4 switch's control plane identifies the flows assigned to each queue, it creates the control rules that implement the isolation of these flows within the non-programmable router. These rules consist of an Access Control List (ACL) that specifies the IP addresses associated with each queue and setting the buffer size of individual queues.
6) The classifier of the non-programmable router assigns the flows to their corresponding queues.

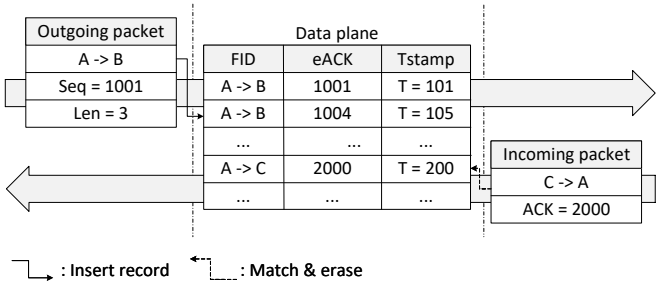


Fig. 3. RTT calculation in the data plane [19]. The RTT is calculated by subtracting the timestamp of an incoming TCP packet from the timestamp of the corresponding outgoing packet, using a flow identifier and acknowledgment number stored in a table for correlation.

7) The buffer size of each queue is adjusted following the Stanford rule [18] to prevent bufferbloat.

B. Metrics Computation

The proposed system employs a method detailed in [19] to calculate the RTT of individual flows. This method correlates the TCP sequence (SEQ) and acknowledgment (ACK) numbers found in incoming and outgoing packets. By calculating the time difference between these two packets, the system can derive the RTT. The method is described in Fig. 3, and it comprises the following steps:

1) For each outgoing packet, the system computes the flow identifier (FID) of the packet by applying a hash function to the 5-tuple, which consists of source/destination IP addresses, source/destination port numbers, and the communication protocol. Additionally, it calculates the expected acknowledgment number (eACK) by adding the SEQ number to the payload's length.
2) The current packet's timestamp is stored in a table, indexed by the combination of FID and eACK.
3) Upon receiving an incoming TCP packet, the system uses the FID and ACK number to search the table for an existing record. If a match is found, it calculates the difference between the timestamps, generating an RTT sample.

In real-world scenarios, devices may not acknowledge every packet promptly (e.g., due to delayed ACK, where a device sends a single ACK for multiple packets). Given the limited memory available on programmable switches (typically in the tens of megabytes), it is not feasible to store records indefinitely. Therefore, the system employs a timeout threshold. When this threshold is exceeded, the corresponding record is evicted from memory. Furthermore, the method utilizes a multi-stage hash approach due to constraints on accessing data plane memory.

The system also employs the Count-Min Sketch (CMS) data structure [16] to estimate the packet count for a specific flow. These packet counts are subsequently compared to a predefined threshold, helping determine whether a flow should be categorized as a long flow.

C. Classification Algorithm

The Jenks optimization method [9], also known as Jenks natural breaks algorithm, is a statistical technique employed for data classification and clustering. This method serves the purpose of grouping numerical data into meaningful and distinct categories. It proves particularly valuable when handling datasets characterized by substantial variability and identifying thresholds within the dataset.

The control plane of the P4 switch utilizes the Jenks natural breaks algorithm to establish the lower and upper limits for a set of K queues, which are determined according to the RTT values of individual flows. These limits are used to allocate flows with similar RTT in each queue. Algorithm 1 outlines the process of the Jenks optimization method. This method is employed to derive queue boundaries that maximize similarity within an input dataset comprising the RTTs of individual flows. Initially, the queue boundaries are defined by intervals with the same size. Then, the algorithm refines the boundaries to minimize the sum of squared deviations from the queue limits. The Goodness of Variance Fit (GVF) serves as a quality index in the Jenks algorithm, acting as a stopping criterion. The ideal data classification is achieved when the GVF=1. As a result, the Jenks algorithm will produce $[lo_j, hi_j]$ as the boundaries for j queues.

Algorithm 1: Jenks Natural Breaks Algorithm

*1: **Input:** $\mathbf{RTT}$ as the dataset containing the RTTs of individual flows and $\mathbf{K}$ as the number of queues.*

2: Define the boundaries for each queue: $[lo_j, hi_j]$ for $j = 1, 2, ..., \mathbf{K}$.

3: Calculate the sum of the squared deviation $SD_{\mathbf{RTT}}$ of the RTTs as follows:

$$SD_{\mathbf{RTT}} = \sum (rtt_i - \overline{rtt})^2,\ rtt_i \in \mathbf{RTT}$$

*4: **While** the GVF is lower than maximum value **do***

5: Calculate the sum of squared deviation for each queue SD_j as follows:

$$SD_j = \sum (rtt_{i,j} - \overline{rtt_j})^2,\ rtt_{i,j} \in [lo_j, hi_j]$$

6: Increase one standard deviation $\sigma = \sqrt{SD_j/\mathbf{K}_j}$ into the interval $[lo_j, hi_j]$ from queues with lowest SD_j by decreasing one σ_j into the interval from queues with largest SD_j.

7: Calculate the GVF as follows:

$$GVF = 1 - \sum_{j=1}^{K} SD_j / SD_{\mathbf{RTT}}$$

*8: **End while***

*9: **Output:** Return the queue limits, $[lo_j, hi_j]$ for $j = 1, 2, ..., \mathbf{K}$.*

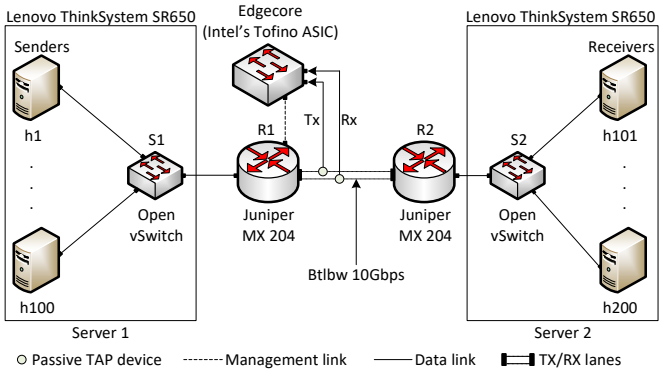


Fig. 4. Topology used to run the experiments.

IV. Results and Evaluation

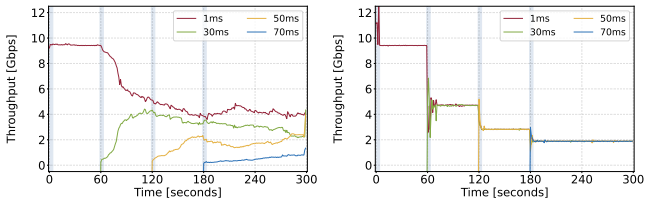
Fig. 4 illustrates the experimental setup, which comprises 100 senders (labeled h1 to h100), each establishing a TCP connection with corresponding receivers (h101 to h200). These hosts, created as network namespaces in Linux via Mininet [20] on a physical server, were allocated sufficient resources to ensure the results accurately reflect those obtained in a real-world scenario. The TCP send and receive buffers on the end hosts are set to 200MB. The senders run iPerf3 [21] to resemble large data transfers. The senders are connected to an Open Virtual Switch (OVS) [22] denoted as S1. S1 bridges to the server's (Server 1) network interface (i.e., Mellanox ConnectX-5 [23]), which, in turn, connects to a Juniper MX-204 router [24] (i.e., router R1). A similar configuration is observed on the side of Server 2. The link between the OVS switches and the routers has a bandwidth of 40Gbps, whereas the link connecting the routers has a bandwidth of 10Gbps. The connection between these routers is tapped in both directions (Tx/Rx) using an optical tap. These taps duplicate the traffic that is redirected to the P4 switch. The P4 switch used is the Edgecore Wedge100BF-32X [25], equipped with Intel's Tofino ASIC chip operating at 3.2 Tbps.

A. Flow Allocation and Bandwidth Distribution in Multi-Flow Scenarios

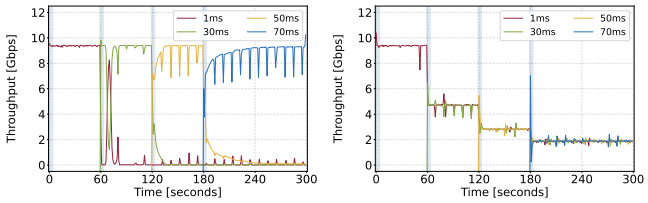
This test evaluates the system with four subsequent flows in scenarios with and without flow separation. Flows with 1ms, 30ms, 50ms, and 70ms are considered. Initially, the flow with 1ms RTT starts, and then, at intervals of 60 seconds, additional flows join. In this experiment, the P4 separates the flows and divides the available bandwidth equally as a function of the number of flows.

Fig. 5(a) shows a scenario with four CUBIC flows. The results indicate that flows with RTTs of 1ms and 30ms obtain a larger portion of the available bandwidth, while flows with RTTs of 50ms and 70ms experience performance degradation. Fig. 5(b) shows when flow separation is implemented. The system can identify and separate the flows in different queues and achieve a fair bandwidth share.

Fig. 5(c) shows what occurs when multiple long BBR flows interact in the same queue. The results indicate that when a



(a) CUBIC flows wo/separation (b) CUBIC flows w/separation



(c) BBR flows wo/separation (d) BBR flows w/separation

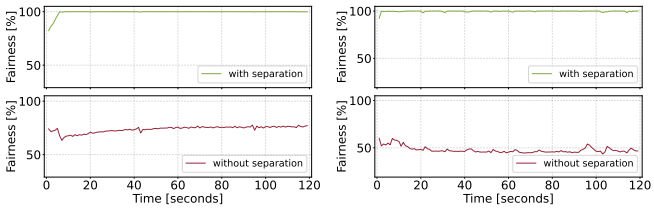
Fig. 5. The figure shows the throughput of TCP flows. (a) Four CUBIC flows wo/separation. (b) Four CUBIC flows w/separation. (c) Four BBR flows wo/ separation. (d) Four BBR flows w/separation.

flow with a longer RTT joins, it immediately deteriorates the performance of the remaining flows, with the flow having the longer RTT eventually achieving the higher throughput. Implementing flow separation, as observed in Fig. 5(d), ensures that each BBR flow will evenly share the available bandwidth and consequently converge to better fairness.

B. Assessing the Effectiveness of the Classification Algorithm

This experiment considers 100 simultaneous flows to be sharing the bottleneck. Considering that the router has up to four queues, the P4 switch implements the Jenks optimization method to group the closely related RTTs in a specific queue. In this way, the impact of RTT unfairness is minimized. The RTT distribution assigned to each flow follows a random distribution with values varying from 1ms to 100ms. When the system is tested without separation, the buffer size of the single queue is set to BDP considering the flow with longer RTT (i.e., 100ms). When the system implements flow separation, the buffer size is adjusted following the Stanford rule [18], which takes into account the BDP divided by the square root of the number of flows (BDP/$\sqrt{N}$). With this approach, the system also minimizes the queueing delay resulting from bufferbloat on a per-queue basis.

Fig. 6(a) shows the fairness index observed for CUBIC flows. In the lower graph are presented the results without separation. The fairness value settles around 80%, whereas the upper graph shows that with flow separation, the 100 CUBIC flows are identified in less than 2 seconds and allocated in four different queues. The reported fairness considers the aggregated throughput of the flows present in each queue. In Fig. 6(b), it is observed the interaction of 100 BBR flows. The lower graph shows a scenario without flow separation. The fairness index settles around 50%, whereas, in the upper graph, it is observed that the aggregated throughput of the four queues converges to fairness.



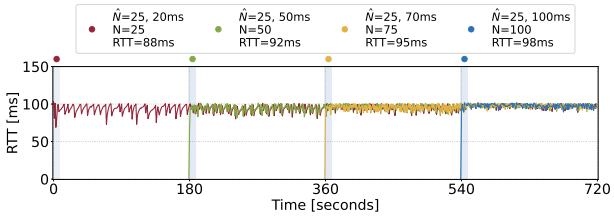
(a) CUBIC (b) BBR

Fig. 6. Fairness index with and without separation for (a) 100 CUBIC flows and (b) 100 BBR flows.

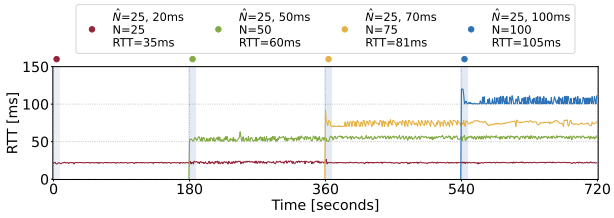
Fig. 7 shows the RTT observed in each queue with and without separation. The experiment consists of 25 CUBIC flows joining every 180 seconds. Each group of 25 flows has 20ms, 50ms, 70ms, and 100ms respectively. The results observed in Fig. 7(a) show that without separation, the flows experience an RTT of around 100ms due to a large buffer. Fig. 7(b) shows when the system separates the flows into different queues. In this scenario, the RTT observed in each queue settles around the value corresponding to individual flows.

C. Separating Long Flows from Short Flows

This experiment assesses the system's capability to enhance the FCT of short flows by segregating them from long flows. The type of traffic used in this experiment resembles web browsing competing against large data transfers. The FCT represents the time from the first packet transmission to the arrival of the last packet at its destination. This test examines how segregating flows based on their RTT and duration affects the FCT of short flows. This test presents a scenario where short flows share the bottleneck link with long flows. The experiment generates 100 long flows over a 10Gbps bottleneck link. The buffer size for the scenario without separation is 100ms. These long flows consist of a mix of 50% CUBIC and 50% BBR. Simultaneously, a sender initiates 10,000 short

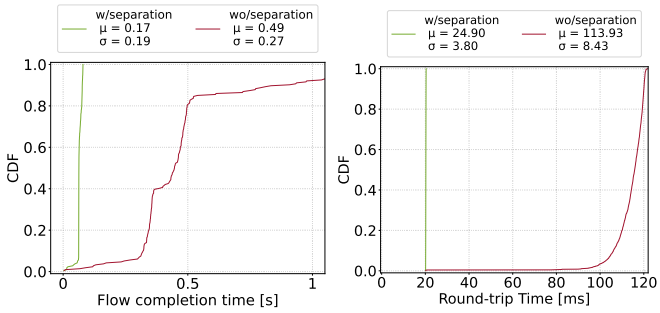


(a) wo/separation



(b) w/separation

Fig. 7. RTT of individual queues. (a) wo/separation. (b) w/separation.



(a) (b)

Fig. 8. Cumulative Distribution Functions (CDFs). (a) FCT of short flows. (b) RTT of short flows. The experiment encompasses scenarios both with and without separation.

flows with inter-connection times following an exponential distribution with an average of one second.

Fig. 8 depicts the Cumulative Distribution Functions (CDFs) of the FCT and RTT of short flows. It is observed that without traffic separation, the average FCT of at least 60% of short flows takes more than 0.49 seconds to complete. On the other hand, separating long flows from short flows reduces the average FCT to around 0.17 seconds for most flows. Fig. 8(b) shows that without separation, the average RTT of the short flows is similar to the long flows (i.e., $\sim$ 113.93ms), whereas, with separation, the RTT of short flows is reduced to an average of 24.9ms. By reducing the average FCT and RTT of short flows, the system can enhance the quality of service and improve network efficiency.

V. Conclusion and Future Works

This paper presented a system that reduces the impact of RTT unfairness in non-programmable networks using P4-programmable data planes. The system implemented a classification algorithm to separate relocate flows with similar RTTs in different queues. Results show that performance metrics including fairness, FCT, and RTT of individual flows can be improved by isolating the dynamics of competing TCP flows.

A limitation of the system is that when separated flows within a queue are not fully utilizing the available bandwidth, it results in the queue imbalance problem. To address this limitation, the authors intend to extend this work by implementing a mechanism to measure the throughput of individual flows. This mechanism will ensure that each queue efficiently utilizes its allocated bandwidth, and any remaining bandwidth can be reallocated to other queues as needed. Moreover, the authors aim to test the system's realism and scalability by using real traffic traces.

Acknowledgment

This work was supported by the U.S. National Science Foundation (NSF), under grant number 2118311, and by the Office of Naval Research (ONR), grant number N00014-23-1-2245.

References

[1] E. Gavaletz and J. Kaur, "Decomposing RTT-unfairness in transport protocols," in *2010 17th IEEE Workshop on Local & Metropolitan Area Networks (LANMAN)*, 2010.

[2] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS operating systems review*, 2008.

[3] N. Cardwell, Y. Cheng, C. Gunn, S. Yeganeh, and V. Jacobson, "BBR: congestion-based congestion control," *Communications of the ACM*, 2017.

[4] E. Kfoury, J. Gomez, J. Crichigno, and E. Bou-Harb, "An emulation-based evaluation of TCP BBRv2 alpha for wired broadband," *Computer Communications*, 2020.

[5] J. Gomez, E. Kfoury, J. Crichigno, E. Bou-Harb, and G. Srivastava, "A performance evaluation of TCP BBRv2 alpha," in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, 2020.

[6] J. Gomez, E. Kfoury, J. Crichigno, and G. Srivastava, "Understanding the performance of TCP BBRv2 using FABRIC," *2023 IEEE BlackSeaCom, Istanbul, Turkiye*, 2023.

[7] L. Kleinrock, "Internet congestion control using the power metric: Keep the pipe just full, but no fuller," *Ad hoc networks*, 2018.

[8] E. 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.

[9] G. Jenks, "The data model concept in statistical mapping," *International yearbook of cartography*, 1967.

[10] J. Gettys, "Bufferbloat: Dark buffers in the internet," *IEEE Internet Computing*, 2011.

[11] J. Crichigno, N. Ghani, J. Khoury, W. Shu, and M. Wu, "Dynamic routing optimization in WDM networks," in *2010 IEEE Global Communications Conference (GLOBECOM)*, 2010.

[12] E. 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)*, 2019.

[13] J. Gomez, E. Kfoury, J. Crichigno, and G. Srivastava, "A survey on TCP enhancements using P4-programmable devices," *Computer Networks*, vol. 212, p. 109030, 2022.

[14] A. AlSabeh, J. Khoury, E. Kfoury, J. Crichigno, and E. Bou-Harb, "A survey on security applications of P4 programmable switches and a STRIDE-based vulnerability assessment," *Computer Networks*, 2022.

[15] Y. Tao, J. Jiang, S. Ma, L. Wang, W. Wang, and B. Li, "Unraveling the RTT-fairness problem for BBR: A queueing model," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018.

[16] E. Kfoury, J. Crichigno, and E. Bou-Harb, "P4BS: Leveraging passive measurements from P4 switches to dynamically modify a router's buffer size," *IEEE Transactions on Network and Service Management*, 2023.

[17] E. Kfoury, J. Crichigno, and E. Bou-Harb, "P4CCI: P4-based online TCP congestion control algorithm identification for traffic separation," in *IEEE International Conference on Communications (ICC), Rome, Italy*, 2023.

[18] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," *ACM SIGCOMM Computer Communication Review*, 2004.

[19] X. Chen, H. Kim, J. 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*, 2020.

[20] K. Kaur, J. Singh, and N. Ghumman, "Mininet as software defined networking testing platform," in *International conference on communication, computing & systems (ICCCS)*, 2014.

[21] J. Dugan, S. Elliott, B. Mah, J. Poskanzer, and K. Prabhu, "iPerf3, tool for active measurements of the maximum achievable bandwidth on IP networks," *URL: https://github. com/esnet/iperf*, 2014.

[22] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, and P. Shelar, "The design and implementation of open vSwitch," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI15)*, 2015.

[23] Nvidia Corporation, "Mellanox ConnectX-5." [Online]. Available: https://tinyurl.com/5d7ct845, Accessed on 09-26-2023.

[24] Juniper Networks, "MX204 universal routing platform." [Online]. Available: https://tinyurl.com/yz86p3vx, Accessed on 08-14-2023.

[25] Edgecore Networks, "Wedge 100BF-32X." [Online]. Available: https://tinyurl.com/2xay8kky, Accessed on 09-26-2023.