Improving TCP Fairness in Non-programmable Networks using P4-programmable Data Planes

Jose Gomez[a], Elie F. Kfoury[a], Jorge Crichigno[a], Gautam Srivastava[b]
[a]University of South Carolina, USA
[b]Brandon University, Canada
{gomezgaj, ekfoury}@email.sc.edu, jcrichigno@cec.sc.edu, srivastavag@brandonu.ca

***Abstract*—Round-trip Time (RTT) unfairness arises in traditional TCP loss-based Congestion Control Algorithms (CCAs) such as CUBIC when flows with shorter RTTs are allocated more bandwidth than those with longer RTTs. Conversely, newer CCAs such as the Bottleneck Bandwidth and Round-trip Time (BBR) exhibit the opposite behavior, where flows with longer RTTs are allocated more bandwidth than those with shorter RTTs.**

This paper presents a system that reduces RTT unfairness by rebalancing the router's queue according to the RTT of TCP flows. The proposed system uses a P4-programmable Data Plane (PDP) as a measurement tool to process a copy of the traffic on the network, between non-programmable (traditional) routers. The PDP generates per-flow fine-grained measurements of the RTT and the throughput of TCP flows, and creates control rules which are applied to a non-programmable router. These rules distribute the TCP flows into separate queues, according to a classification algorithm that mitigates RTT unfairness. Results show that the system improves fairness among competing flows and reduces the Flow Completion Time (FCT) of long flows, regardless of the CCA. Moreover, the system is capable of measuring the throughput of individual flows and rebalancing the bandwidth allocated to each queue. This approach ensures that the available bandwidth is effectively used.

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

I. Introduction

The Transmission Control Protocol (TCP) enables reliable end-to-end communication over the Internet. TCP controls the sending rate through its congestion control mechanism, which aims at optimizing data transfer rates while reducing network congestion. Within TCP's congestion control mechanism lies the congestion control algorithms (CCAs), which are responsible for dictating the dynamics of the sending rate based on parameters such as packet loss and Round-Trip Time (RTT). Among the CCAs used in TCP, one of the fundamental principles is the Additive-Increase Multiplicative-Decrease (AIMD) scheme. Traditional CCAs such as Reno [1] implement the AIMD scheme, gradually increasing the sending rate until congestion occurs, upon which it promptly reduces the rate by half. This approach ensures network stability and enables bandwidth probing without overwhelming the network with excessive data traffic.

However, disparities in the RTT of individual flows induce an unfair bandwidth share causing the RTT unfairness problem. A typical scenario where RTT unfairness occurs is when two senders are located at different distances from a receiver. In such a scenario, competing data flows experience different recovery times after a congestion event, leading to an unequal distribution of bandwidth. Traditional loss-based CCAs, such as CUBIC [2] experience RTT unfairness, favoring flows with shorter RTTs over those with longer ones. The flows with shorter RTTs can increase their sending rate faster, leaving the flow with longer RTTs at a disadvantage.

In contrast, recent CCAs such as the Bottleneck Bandwidth and Round-Trip Time [3] (BBR) exhibit the opposite behavior, where flows with longer RTT achieve higher throughput. BBR is designed to estimate Kleinrock's optimal operating point [4] by limiting data in transit to one Bandwidth-delay Product (BDP), computed as the product of RTT and bottleneck bandwidth (Btlbw). This approach allocates larger bandwidth shares to BBR flows with longer RTTs, resulting in a bias against flows with shorter RTTs. This unique bias has significant implications on other protocols. Firstly, it introduces an unfavorable trade-off between low latency and high delivery rate, undermining the efforts invested in routing optimization [5]. For instance, prioritizing routes with minimal RTT using protocols like Enhanced Interior Gateway Routing Protocol (EIGRP) may no longer be convenient, as flows along such routes are easily overwhelmed when competing with others taking suboptimal routes with higher latency. Secondly, the advantage gained by long RTT flows creates a vulnerability, allowing malicious receivers to obtain more bandwidth from competing flows by artificially inflating their inbound traffic with higher latency.

This paper proposes a system to address RTT unfairness and enhance the performance of different TCP CCAs. The system leverages a P4-programmable Data Plane (PDP) as a measurement instrument to passively capture traffic between two non-programmable (traditional) routers. By computing the RTT and throughput of individual TCP flows, the system employs a classification algorithm to determine flow separation. P4, which stands for Programming Protocol-independent Packet Processors [6], is the programming language used to specify packet processing in the data plane. The PDP calculates the RTT for each flow and uses this information to create control rules. Rules are then applied to a non-programmable router, segregating traffic into different queues based on their respective RTTs.

A. Contributions

This paper introduces a system that segregates TCP flows based on their RTT. By passively measuring the traffic through the egress interface of a non-programmable router, the PDP efficiently separates the flows using a classification algorithm. The contributions of this paper can be summarized as follows:

- Enabling per-packet visibility on a traditional (non-programmable) network by using PDPs and passive optical taps.
- Leveraging PDPs to continuously generate per-flow fine-grained measurements at line rate, such as RTT and throughput.
- Comparing and analyzing the impact of RTT unfairness on CUBIC and BBR flows.
- Categorizing and separating flows into different queues according to their RTT regardless of the design principles of the CCA.
- Improving the fairness, Flow Completion Times (FCTs), and RTTs for each flow. By isolating the behaviors of competing TCP flows, resources are utilized more efficiently.
- Rebalancing router's queues by reallocating unused bandwidth from underutilized flows to those that can more efficiently utilize it.

The rest of the paper is organized as follows: Section II presents background on PDPs and related work. Section III describes the proposed system. Section IV describes the experiments and the results. Section V presents the conclusion and future work.

II. Background and Related Work

A. P4-programmable Data Planes (PDPs)

PDPs provide programmers with the flexibility to specify how packets are handled through a pipeline [7]. This pipeline comprises three main components: a programmable parser, a programmable match-action pipeline, and a programmable deparser. Initially, the programmable parser decodes the incoming bit stream received by the switch, organizing it into standard and custom header fields as predetermined by the programmer. Subsequently, the match-action pipeline carries out operations on packet headers and intermediate results. Finally, the deparser reconstructs the packet headers and prepares them for transmission by serializing them.

PDPs offer precise timers with nanosecond accuracy and stateful memories such as registers, counters, and meters, all accessible at line rate. These capabilities facilitate per-packet operations, which are extensively used to enhance network performance and provide visibility into network events [8–11]. This paper focuses on programming the data plane of a PDP to identify and compute the RTTs of individual flows. Subsequently, the control plane utilizes a classification algorithm to determine how these flows are allocated into different queues.

B. RTT Characterization

Ma *et al.* [12] investigated the RTT unfairness issue in BBR across geo-distributed cloud servers. Despite its aim to optimize delivery bandwidth while minimizing delay, BBR resulted in significant bandwidth disparities among competing flows, favoring longer RTT flows over shorter flows. Through empirical measurements and theoretical modeling, researchers identified the root cause of BBR's RTT unfairness: its tendency to send excessive data during bandwidth probing, disproportionately benefiting long RTT flows. To remedy this, they proposed BBQ, a variation of BBR that ensures flow fairness without altering the design principles. Gavaletz and Kaur [13] introduced a novel method for analyzing and breaking down the sources of RTT unfairness in transport protocols. Their study revealed that the origins of RTT unfairness differ among CCAs. Additionally, the authors proposed FairTCP, a modification to TCP aimed at mitigating the feedback delay component of RTT unfairness by enabling connections to obtain more precise congestion feedback.

Tao *et al*. [14] created a mathematical model to study RTT unfairness in BBR. They evaluated a queuing model that captures the interactions between BBR flows and the underlying network. Pan *et al.* [15] addressed BBR's RTT unfairness by introducing a gamma correction mechanism. This technique adapts the pacing rate during congestion avoidance based on the connection's RTT.

C. Flow Separation

Kfoury *et al.* [16] investigated the impact of buffer size on network application performance at bottleneck routers. They underscored the drawbacks of static buffer configurations, which often result in increased packet losses, decreased link utilization, and higher latency. The paper introduces P4BS, a dynamic buffer sizing system harnessing programmable switches to measure crucial metrics like long-lived flow counts, RTTs, packet loss rates, and queuing delays. P4BS optimizes buffer sizes based on these metrics to minimize queuing delays and packet loss rates. P4BS exhibited improved quality of service across various applications such as web browsing, video streaming, and voice-over IP.

In another study, Kfoury *et al.* [17] focused on identifying CCAs using PDPs. Their objective was to mitigate the impact of CCAs that exhibit aggressive behavior concerning fairness and link utilization. The proposed scheme computes and extracts the "bytes-in-flight" metric for each flow, feeding this data into a deep-learning model for classification. Once classified, flows are segregated into dedicated queues based on their respective CCA types. Results demonstrated a precise detection and separation of flows based on their CCAs.

III. Proposed System

A. Overview

The proposed system utilizes passive taps to collect traffic from the data link between two non-programmable routers as shown in Fig. 1. These taps create a copy of the traffic without causing any performance disruptions. Subsequently, this traffic is directed to the data plane of a P4 switch. Within this switch, RTT and throughput calculations are performed for each flow. Once these flow metrics are computed, they are transmitted

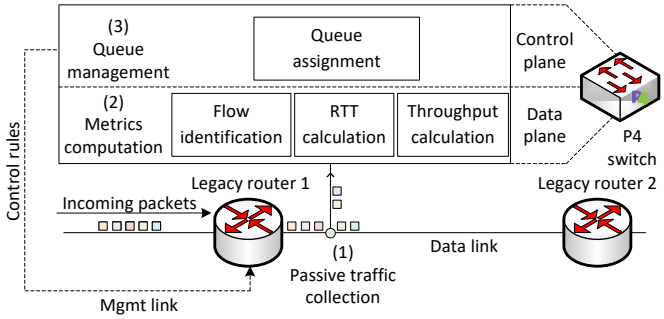


Fig. 1. High-level system overview. Stage (1): a copy of the incoming packets is forwarded by the tap to the data plane of the P4 switch. Stage (2): Each flow is identified and the RTT and throughput are calculated at the data plane of the P4 switch. Stage (3): The P4 switch's control plane employs a classification algorithm to allocate a queue for each flow.

to the control plane. At this stage, a classification algorithm determines which queues of the non-programmable router should receive the respective flows. The allocation action is executed via the management port of the non-programmable router.

B. Metrics Computation

Fig. 2 illustrates how the RTT is calculated on a per-flow basis. This method correlates the TCP sequence number (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. This is performed in the data plane by using the flow identification (FID) and the expected acknowledgment (eACK) of outgoing packets as the key of timestamp (Tstamp) values. The eACK is calculated by adding the packet length to the SEQ. Then, when the incoming packet matches the expected acknowledgment, an RTT sample is produced by calculating the differences between the timestamps.

C. Classification Algorithm

The Jenks optimization method [18] is a statistical tool used to classify data into meaningful categories by identifying distinct thresholds. The proposed system begins by sorting the RTTs and dividing them into classes based on the number of queues. By calculating the sum of squares within each potential breakpoint, the algorithm aims to minimize the total sum of squares across all classes, effectively maximizing differences between classes while minimizing variation. Utilizing a dynamic programming approach, the algorithm explores various arrangements to find the optimal breaks. Once these optimal breaks are determined, RTTs are classified into their respective queues.

IV. Results and Evaluation

Fig. 3 shows the experimental setup used for the evaluations. The topology consists of 1000 senders (labeled h1 to h1000), each initiating a data transfer with their corresponding receivers (h1001 to h2000). The end hosts are represented as network namespaces in Mininet [20]. The topology is hosted

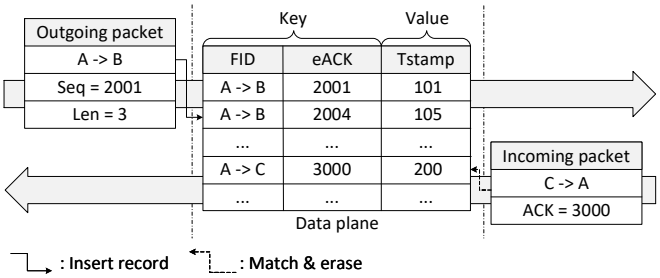


Fig. 2. RTT calculation. When an incoming TCP packet is received, its timestamp is subtracted from the timestamp of the corresponding outgoing packet. This calculation utilizes a flow identifier (FID) and the expected acknowledgment (eACK) as the key of a timestamp (Tstamp) value [19].

on a physical server provisioned with enough resources to ensure that the obtained results closely reproduce real-world scenarios.

The TCP send and receive buffers on the end hosts are configured to 200MB each. To perform large data transfers, the senders utilize iPerf3. These senders are linked to an Open Virtual Switch (OVS) denoted as S1, which bridges to the network interface of Server 1 (Mellanox ConnectX-5), further connecting to a Juniper MX-204 router [21] (Router R1). A similar setup is mirrored on the side of Server 2. The links between the OVS switches and the routers have a bandwidth of 40Gbps, whereas the connection between the routers operates at 10Gbps. Bidirectional optical taps are employed to monitor traffic flow between the routers (Tx/Rx), redirecting it to the P4 switch. The P4 switch is the Edgecore Wedge100BF-32X [22] that uses an Intel's Tofino ASIC chip operating up to 3.2 Tbps.

A. Experiment 1: TCP Flow Separation and Fairness Analysis

This experiment evaluates the performance of four TCP long flows with varying RTTs that share a common bottleneck link. Specifically, the RTTs for these flows are 2 milliseconds (ms), 20 ms, 40 ms, and 60 ms, respectively. The experiment examines both fairness and throughput in two scenarios: one without flow separation (denoted as wo/ separation) and another with

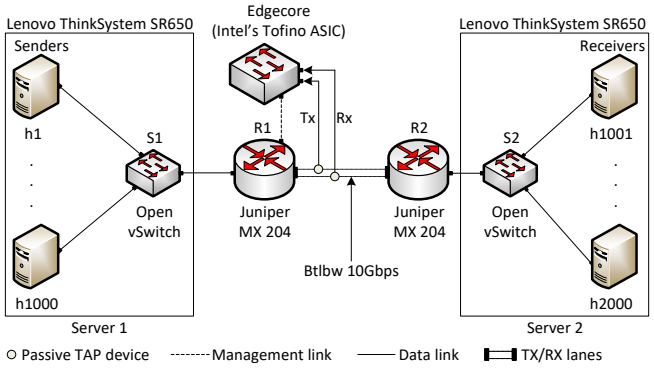
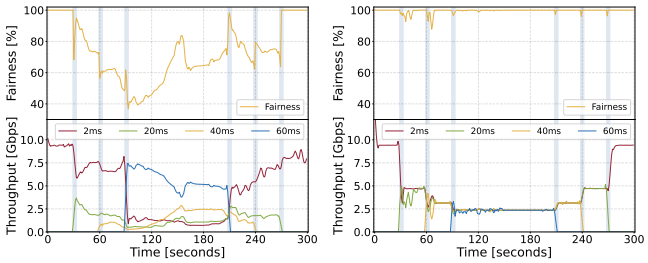
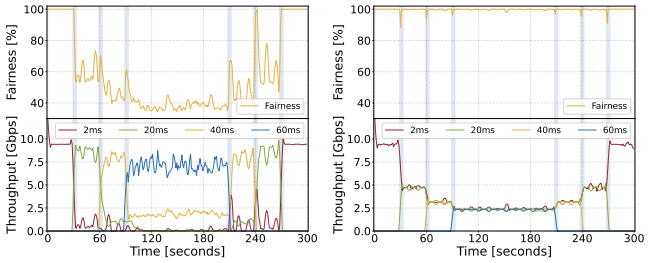


Fig. 3. The experimental topology employs bidirectional optical taps to extract a copy of the network traffic between two legacy routers and direct it to a PDP.



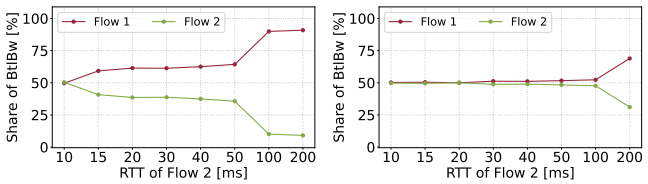
(a) CUBIC wo/ separation (b) CUBIC w/ separation



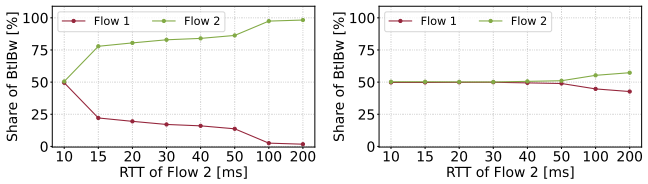
(c) BBR wo/ separation (d) BBR w/ separation

Fig. 4. The figure shows the fairness and 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. Shaded lines indicate when a flow enters or exits.

flow separation (denoted as w/ separation). In the scenario without flow separation, all four flows share the same queue, and the BDP is pre-adjusted based on the flow with the highest RTT. The objective is to assess the degree of coexistence among the flows in the absence of explicit separation. In the scenario with flow separation, the PDP allocates the flows into separate queues based on their RTTs. This approach isolates the interaction between TCP flows with different RTTs. Additionally, the system modifies the buffer size for each queue using the Stanford rule [23], which scales the buffer size to BDP/$\sqrt{N}$, where N represents the total number of flows. This adjustment aims to mitigate the impact of bufferbloat [24] in each queue. Fig. 4(a) illustrates the scenario without flow separation, where all flows share a single queue. In this configuration, the fairness index exhibits degradation after a new long flow joins, resulting in inconsistent fairness values throughout the experiment. In contrast, Fig. 4(b) depicts a scenario where a P4-switch identifies the RTT of individual long flows and allocates them to separate queues. As a result, the available bandwidth is equitably distributed, maintaining a fairness index of around 98% throughout the test duration. Similarly, Fig. 4(c) presents a scenario involving BBR flows. Unlike CUBIC's behavior, in scenarios without separation, the BBR flow with a higher RTT dominates the bandwidth share. However, Fig. 4(d) shows that by segregating the BBR flows into distinct queues, fairness is achieved, resulting in a balanced bandwidth distribution and a fairness index of approximately 98%. Flow separation enhances the stability of TCP flows, minimizing throughput fluctuations.



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



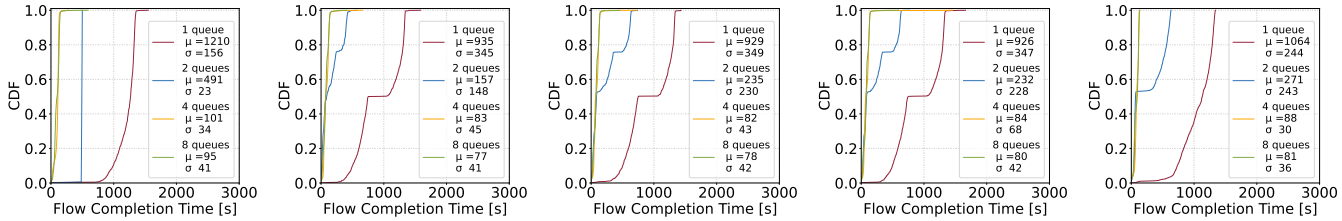
(c) BBR wo/ separation (d) BBR w/ separation

Fig. 5. When flow separation is absent, the disparity between two competing TCP flows worsens with increasing differences in RTT. However, segregating the flows into distinct queues leads to a convergence of bottleneck share towards fairness.

B. Experiment 2: Analyzing the Impact of RTT Disparities

This experiment aims to evaluate the impact of the RTT disparity in the share of the bottleneck bandwidth (BtlBw) and how the proposed system addresses this issue. In scenarios where two CUBIC flows compete over a bottleneck link, the flow with the shorter RTT gains an advantage, as it can increase its congestion window more rapidly due to the shorter intervals between packet losses. Conversely, BBR assigns inflight data in proportion to the RTT, leading to larger bandwidth allocation for flows with longer RTTs. This test systematically varies the RTT from 10 ms to 200 ms between two competing flows. The first flow maintains a constant RTT of 10 ms, while the second flow's RTT is increased. This range of RTTs is representative of typical Local Area Network (LAN) and Wide Area Network (WAN) scenarios. The experiment aims to provide insights into the bandwidth sharing dynamics under varying RTT conditions and the effectiveness of the proposed system in ensuring equitable bandwidth distribution.

Fig. 5(a) illustrates a scenario involving two competing CUBIC flows. It is noted that in the absence of flow separation, the CUBIC flow with the lower RTT does not completely dominate the bandwidth. Instead, it maintains approximately a 60% share of the BtlBw, leaving the other flow with a 40% share. Fig. 5(b) demonstrates that when the RTT disparity is less than 90 ms, the proposed system effectively enforces an equitable distribution of the BtlBw. However, the system begins to show signs of degradation when the disparity exceeds 200 ms. In contrast, Fig. 5(c) examines the behavior of BBR flows and shows that with a 5 ms difference in RTT, the flow with the longer RTT obtains at least 75% of the BtlBw. Finally, Fig. 5(d) shows that with flow separation, the flows achieve a fair distribution of bandwidth. This experiment demonstrates that the proposed system can enhance the fairness of competing flows, regardless of the design principles of the CCA being used.



(a) 100% CUBIC (b) 75% CUBIC, 25% BBR (c) 50% CUBIC, 50% BBR (d) 25% CUBIC, 75% BBR (e) 100% BBR

Fig. 6. This experiment presents the Cumulative Distribution Function (CDF) of the FCT for 1000 long flows over a 10 Gbps link. The experiment demonstrates how the number of queues impacts the FCT of long flows. Note that the results corresponding to 4 and 8 queues mostly overlap.

C. Experiment 3: FCT of Competing Flows

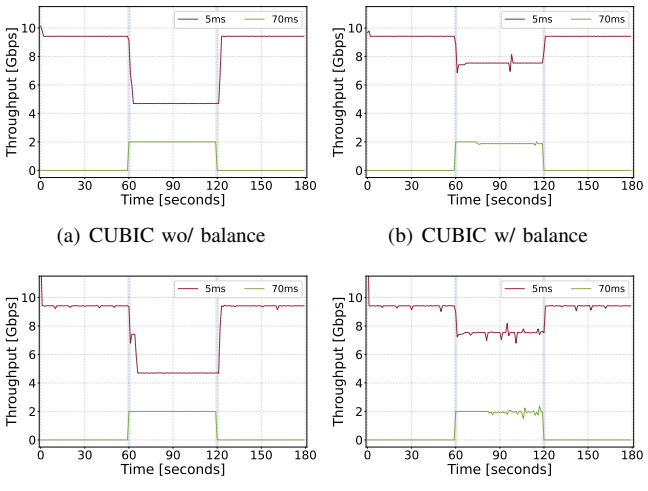
The experiment is designed to assess the FCT of 1000 long competing flows over a 10Gbps bottleneck link. The RTT for each sender-receiver pair is randomly selected from a predefined set of values that span from 1 ms to 100 ms. The experiment further investigates the system's efficiency by employing 1, 2, 4, and 8 queues to manage varying combinations of CUBIC and BBR flows. In the tests utilizing a single queue, the queue size is adjusted to match the BDP considering that the maximum RTT is 100 ms. In the experiments involving multiple queues, the queue sizes are adjusted based on the average RTT of the flows within each queue, following the Stanford sizing rule [23]. The configurations tested include 100% CUBIC flows, a 75:25 CUBIC to BBR ratio, an equal 50:50 split, a 25:75 CUBIC to BBR ratio, and 100% BBR flows. Each flow transfers 150 Megabytes of data, and the results display the transfer duration as a Cumulative Distribution Function (CDF).

Fig. 6(a) shows a scenario exclusively using CUBIC flows. The results indicate that a single queue configuration leads to an increased average FCT of 1210 seconds, attributed to bufferbloat effects inherent to CUBIC flows. Implementing two queues decreases the average FCT to 491 seconds, and employing four queues further reduces it to an average of 101 seconds. Expanding to eight queues does not result in a substantial decrease in the average FCT. In contrast, Fig. 6(b) depicts a 75:25 CUBIC to BBR flow ratio, where a single queue setup results in an average FCT of 935 seconds. Utilizing two, four, and eight queues progressively diminishes this value. Figs. 6(c) and 6(d) present similar outcomes for 50:50 and 25:75 CUBIC to BBR ratios, respectively. Fig. 6(d) demonstrates that in a 100% BBR flows environment, increasing the number of queues enhances the FCT for longer flows. The most significant improvements are evident when utilizing either four or eight queues.

D. Experiment 4: Rebalancing the Queue

This experiment extends the proposed system by incorporating measurements of individual TCP flow throughput. The primary objective of this modification is to optimize the distribution of allocated bandwidth within each queue. This queue imbalance occurs in scenarios where a TCP flow fails to fully utilize its designated queue capacity. The system continuously monitors the available bandwidth and tracks the bandwidth consumption of each flow. When an underutilized flow is detected, the system dynamically reallocates the unutilized bandwidth to other active flows. As a result, the system maximizes link utilization.

Fig. 7(a) illustrates a scenario without queue balance, where two CUBIC flows share a 10Gbps bottleneck link. Each flow is initially allocated 5Gbps bandwidth. However, the flow with a 70 ms RTT only utilizes 2Gbps, resulting in a link utilization of 70%. In Fig.7(b), the flow with a 70ms RTT does not utilize its full allocated bandwidth. Consequently, the system rebalances the queue, reallocating the unused 2Gbps to the flow with a 5ms RTT. As a result, the 5 ms flow fully consumes the available bandwidth, leading to a link utilization of approximately 100%. Similarly, competing BBR flows exhibit analogous behavior, as depicted in Fig. 7(c), where the flow with a 70 ms RTT fails to fully utilize its allocated bandwidth. On the other hand, in Fig. 7(d), the system effectively rebalances the queue for BBR flows. Consequently, the BBR flow with a 5 ms RTT achieves around 7Gbps, enhancing the link utilization. These findings emphasize the system's resilience and its capacity to optimize bandwidth distribution regardless of the underlying CCA.



(a) CUBIC wo/ balance (b) CUBIC w/ balance

(c) BBR wo/ balance (d) BBR w/ balance

Fig. 7. The figure shows the throughput of two competing TCP flows with 5 ms and 70 ms RTT respectively. (a) CUBIC flows wo/ balance. (b) CUBIC flows w/ balance. (c) BBR flows wo/ balance. (d) BBR flows w/ balance. Shaded lines indicate when a flow enters or exits.

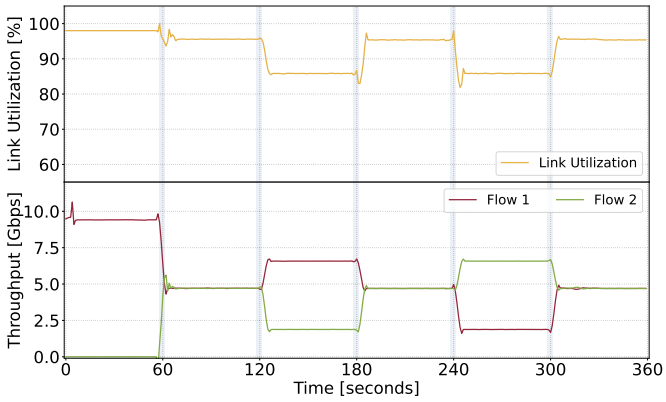


Fig. 8. Throughput comparison of two CUBIC flows as throughput fluctuates. Shaded lines highlight periods of throughput changes.

E. Experiment 5: System under Dynamic Conditions

This test evaluates the system's adaptability to varying throughput from flows sharing a bottleneck link. In Fig. 8, it is observed a scenario where a CUBIC flow (referred to as flow 1) initially fully utilizes the available link capacity. After 60 seconds, another CUBIC flow (referred to as flow 2) joins, and the bandwidth is evenly divided between them. At t=120, flow 2 reduces its throughput utilization, leading to the remaining bandwidth being allocated to flow 1. By t=180, both flows converge to a fair share. Subsequently, at t=240, flow 1 decreases its throughput, and the remaining throughput is allocated to flow 2. Finally, by t=300, both flows converge equally to a fair share. Throughout these transitions, the link utilization fluctuates between 85% and 95%. A similar performance is achieved with BBR flows.

V. Conclusion and Future Work

When flows with different RTTs share a single queue, it impacts the performance of competing flows due to uneven throughput distribution. This issue is an inherent challenge in TCP CCAs and is influenced by their design principles. The results demonstrated that the proposed system can enhance fairness and improve link utilization for long TCP flows by segregating flows with similar RTTs into separate queues. These improvements can be achieved regardless of the underlying CCA. Furthermore, the system extends its functionality to measure the throughput of individual TCP flows and dynamically rebalance the queue when a flow underutilizes its allocated bandwidth. As part of future work, the authors plan to validate the system's performance and scalability using real-world network traces.

Acknowledgment

This work was supported by the U.S. National Science Foundation (NSF) awards 2118311 and 2346726, and by the Office of Naval Research (ONR) award N00014-23-1-2245.

References

[1] M. Allman, "RFC 2581: TCP congestion control," April 1999.

[2] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[3] N. Cardwell, Y. Cheng, C. Gunn, S. Yeganeh, and V. Jacobson, "BBR: congestion-based congestion control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[4] L. Kleinrock, "Internet congestion control using the power metric: Keep the pipe just full, but no fuller," *Ad hoc networks*, vol. 80, pp. 142–157, 2018.

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

[6] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, and G. Varghese, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[7] 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*, vol. 9, pp. 87094–87155, 2021.

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

[9] 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.

[10] 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*, vol. 207, p. 108800, 2022.

[11] A. Mazloum, E. Kfoury, J. Gomez, and J. Crichigno, "A survey on rerouting techniques with P4 programmable data plane switches," *Computer Networks*, vol. 230, p. 109795, 2023.

[12] S. Ma, J. Jiang, W. Wang, and B. Li, "Fairness of congestion-based congestion control: Experimental evaluation and analysis," *arXiv preprint arXiv:1706.09115*, 2017.

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

[14] 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), Abu Dhabi, UAE*, 2018.

[15] W. Pan, X. Li, H. Tan, J. Xu, and X. Li, "Improvement of RTT fairness problem in BBR congestion control algorithm by gamma correction," *Sensors*, vol. 21, p. 4128, 2021.

[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. Jenks, "The data model concept in statistical mapping," *International yearbook of cartography*, vol. 7, pp. 186–190, 1967.

[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, Virtual Event, USA*, 2020.

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

[21] Juniper Networks, "MX204 universal routing platform." [Online]. Available: https://tinyurl.com/yz86p3vx, Accessed on 03-25-2024.

[22] Edgecore Networks, "Wedge 100BF-32X." [Online]. Available: https://tinyurl.com/2xay8kky, Accessed on 03-25-2024.

[23] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 281–292, 2004.

[24] J. Gettys, "Bufferbloat: Dark buffers in the internet," *IEEE Internet Computing*, vol. 55, no. 1, pp. 57–65, 2012.