Enabling Fairness in Flow Allocation using P4-programmable Data Planes

Jose Gomez*, Elie F. Kfoury†, Ali Mazloum†, Jorge Crichigno†
*Fort Lewis College, USA
†University of South Carolina, USA
jagomez@fortlewis.edu, {ekfoury, amazloum}@email.sc.edu, jcrichigno@cec.sc.edu

***Abstract*—This paper presents a system designed to enhance Transmission Control Protocol (TCP) fairness by rebalancing router queues and reducing the impact of Round-Trip Time (RTT) unfairness. The proposed system utilizes a P4-programmable Data Plane (PDP) to process a copy of the traffic from the link between two non-programmable routers. The PDP measures the throughput and calculates the RTT of competing flows in the data plane. Then, the control plane generates the rules to be implemented in a non-programmable router that will allocate flows in different queues to isolate their dynamics. The limits for each queue result from the Jenks optimization algorithm. This approach ensures that flows with similar characteristics share the same queue.**

The results demonstrate that the system efficiently identifies and segregates flows into multiple queues, thereby enforcing fairness among competing flows and enhancing the Flow Completion Time (FCT). The experiments were executed on traffic provided by Measurement and Analysis on the WIDE Internet (MAWI). The system effectively rebalances queues and dynamically redistributes underutilized bandwidth independently of the design principles of the transport protocol. Furthermore, the results show that the system effectively mitigates the effects of bufferbloat and successfully detects and reduces the impact of protocol abuses at the network layer.

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

I. Introduction

A key challenge in next-generation networks is to efficiently utilize computational and networking resources to deliver services that satisfy diverse Quality of Service (QoS) requirements. As the demand for high-speed data transfers increases, there is a need to distribute the available bandwidth fairly among competing flows. The Transmission Control Protocol (TCP) plays a key role in enabling reliable end-to-end communication over the Internet. TCP aims at optimizing data transfer rates while minimizing network congestion by regulating the sending rate through its congestion control mechanism. In the realm of congestion control mechanisms, there exists a diverse range of algorithms known as congestion control algorithms (CCAs). These algorithms regulate the sending rate dynamically, responding to network parameters such as packet loss and Round-Trip Time (RTT). However, since TCP is implemented in the end hosts, it lacks full visibility into network events, which can lead to unfair resource utilization.

For instance, RTT differences among competing TCP flows create an unfair bandwidth distribution, resulting in the RTT unfairness problem [1]. This issue arises when two senders, situated in different locations, encounter varying propagation delays in relation to the receiver. These delays can be attributed to factors such as physical distance, network capabilities, and QoS policies. Consequently, competing TCP flows experience different recovery times in the presence of packet losses. Traditional loss-based CCAs, such as CUBIC [2], exhibit RTT unfairness by favoring flows with shorter RTTs. Therefore, flows with shorter RTTs can increase their sending rates more quickly, leaving flows with longer RTTs at a disadvantage. Conversely, more recent CCAs such as the Bottleneck Bandwidth and Round-Trip Time (BBR) favor flows with longer RTTs [3]. Hence, flows with shorter RTTs achieve a lower throughput. This distinctive bias carries significant implications for the design of Internet protocols. Firstly, it introduces a trade-off between minimizing latency and maximizing delivery rate, undermining efforts to reduce end-to-end latency. For instance, prioritizing routes with minimal RTT using protocols such as Open Shortest Path First (OSPF) may become less effective, as flows along these routes can easily be overwhelmed when competing with others on less optimal routes with higher latency. Secondly, the advantage gained by flows with long RTTs creates a vulnerability, as malicious receivers can exploit this by artificially increasing their inbound traffic with higher latency, thereby obtaining more bandwidth from competing flows.

The emergence of P4-programmable Data Planes (PDPs) offers the functionalities for addressing these issues. P4 (Programming Protocol-Independent Packet Processors) [4] is a data plane programming language that allows network operators to define custom packet processing logic directly within the data plane, enabling fine-grained control over traffic management. Leveraging PDPs, this paper proposes a system to enhance throughput allocation and reduce the impact of RTT unfairness. The proposed system uses a PDP to passively process the traffic between two non-programable routers. This passive measurement mechanism is implemented with optical taps. The data plane measures the throughput, calculates the RTT of individual flows, and categorizes them into separate queues to reduce interaction among flows with significant differences. Queue boundaries are set using the Jenk's classification algorithm [5], which groups flows with similar RTTs together. Results indicate that the proposed system effectively identifies and separates flows into multiple queues, enhancing

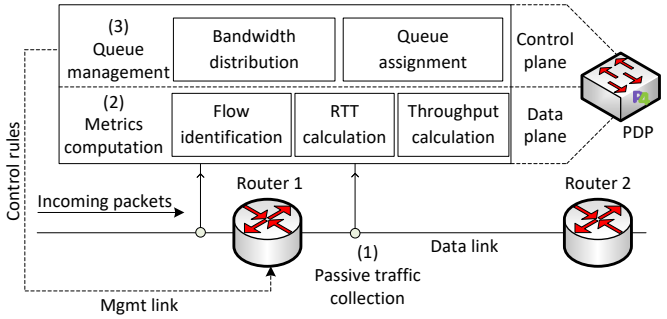


Fig. 1. High-level system overview. (1): a copy of the traffic is forwarded by the tap to the data plane of the PDP. (2): The data plane computes the throughput and RTTs of individual flows. (3): The control plane implements a classification algorithm and generates the rules to be implemented in a non-programmable router.

fairness among competing flows and improving the Flow Completion Time (FCT) for long flows reproduced from a MAWI trace. Moreover, the system is capable of rebalancing the queue, minimizing the impact of bufferbloat [6], and mitigating the effects of User Datagram Protocol (UDP) abuses.

This paper proposes a system that separates flows according to the throughput and RTT of individual flows. The traffic is passively measured by tapping on the egress interface of a non-programmable router. The PDP separates the flows using a classification algorithm. The contributions of this paper can be listed as follows:

- Improving the FCT of long flows by reallocating them in different queues.
- Enhancing the link utilization [7] by dynamically redistributing the bandwidth that is not fully utilized by competing flows.
- Detecting and reducing the impact of bufferbloat by separating flows that increase the average queue latency.
- Mitigating UDP abuses by enforcing a bandwidth policy on UDP flows that exceed a predefined threshold.

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 results and the experiments. Section V concludes the paper.

II. Background and Related Work

A. P4-programmable Data Planes (PDPs)

PDPs 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. PDPs 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 [9–11]. In this paper, the data plane of a PDP 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 Monitoring

Ma *et al.* [12] investigated the issue of RTT unfairness in BBR. The researchers identified BBR's RTT unfairness root cause as its tendency to send excessive data during bandwidth probing, favoring long RTT flows. To rectify this, they proposed BBQ, a variation of BBR that ensures flow fairness without altering design principles. Gavaletz and Kaur [1] introduced a novel method for analyzing RTT unfairness sources in transport protocols, revealing variations among CCAs. Additionally, they proposed FairTCP, a TCP modification mitigating feedback delay in RTT unfairness by providing more precise congestion feedback to connections. Tao *et al.* [3] developed a mathematical model to study RTT unfairness in BBR. They evaluated a queuing model capturing BBR flows' interactions with the network and proposed a modification to dynamically adjust BBR's pacing rate during bandwidth probing based on connection RTT.

C. Bandwidth Redistribution

Meng *et al.* [13] proposed a queue management scheme aimed at ensuring consistently low latency for real-time communication applications amidst competing flows. It identifies the root cause of disruptions as the mismatch between abrupt bandwidth allocation adjustments and gradual congestion window adjustments. The proposed system slows down bandwidth adjustment to align with congestion control reactions and gives time for the application to adapt. Kfoury *et al.* [14] proposed P4BS, a dynamic buffer sizing system utilizing PDPs to optimize buffer and minimize queueing delays and packet loss rates. Kfoury *et al.* [15] also identified CCAs using PDPs to mitigate the impact of the interaction among different CCAs.

III. Proposed System

A. Overview

Fig. 1 provides an overview of the proposed system. Passive taps are used to get a copy of the traffic from the data link of a non-programmable router without affecting performance. This copied traffic is then sent to the data plane of the PDP. In the data plane, the system identifies each flow, calculates the RTT for each flow, and measures the throughput of TCP flows. After computing these flow metrics, they are sent to the control plane. In this stage, a classification algorithm determines which queues of the non-programmable router should receive each flow. The control plane creates the allocation rules that are carried out through the management port of the non-programmable router.

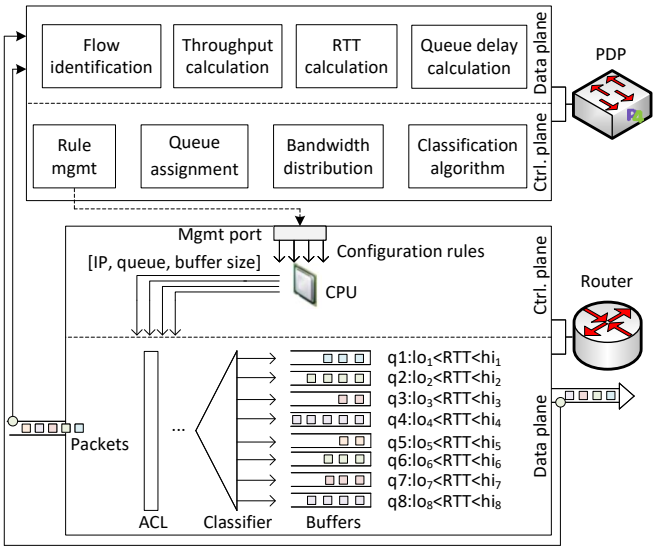


Fig. 2. Proposed system architecture. The PDP serves as a passive measurement tool for calculating the RTT on a per-flow basis.

The architecture of the proposed system is shown in Fig. 2. The process for identifying flows, calculating RTTs, measuring the throughput, and segregating the flows into distinct queues is as follows:

1) The system passively the traffic passing through the non-programmable router. To accomplish this, passive tap devices are deployed on the router's ingress and egress interface. This tap redirects the traffic to the data plane of a PDP operating at line rate.
2) The flow identification data is sent to the RTT calculation module, which pairs each flow with its RTT.
3) The throughput of each TCP flow is determined by extracting the packet length from the IPv4 header and summing up this value in a register every second.
4) The RTTs of each flow are relayed from the data plane of the PDP to the control plane. Here, the classification module employs the Jenks natural breaks algorithm [5] to segregate the flows into different queues.
5) Once the control plane of the PDP identifies the flows designated for each queue, it generates control rules to enforce the isolation of these flows within the non-programmable router. These rules include an Access Control List (ACL) detailing the IP addresses linked to each queue and configuring the buffer size for individual queues.
6) The buffer size of each queue is modified according to the Stanford rule [16] to mitigate bufferbloat.

B. Metrics Computation

The proposed system utilizes a technique outlined in [17] to compute the RTT of individual flows. This method correlates the TCP sequence (SEQ) and acknowledgment (ACK) numbers present in incoming and outgoing packets. By determining the time difference between these two packets, the system can calculate the RTT.

The system employs a hash function to calculate the Flow Identifier (FID) for each outgoing packet, based on the 5-tuple comprising source/destination IP addresses, source/destination port numbers, and the communication protocol. Additionally, it determines the expected acknowledgment number (eACK) by adding the sequence (SEQ) number to the payload's length. Subsequently, the system indexes the timestamp of each outgoing packet in a table using the combination of FID and eACK. When an incoming TCP packet is received, the system uses the FID and acknowledgment (ACK) number to search the table for a matching record. Upon finding a match, it computes the difference between timestamps to generate an RTT sample.

In real-world situations, devices might not promptly acknowledge every packet, such as when a delayed ACK occurs, where a device sends a single ACK for multiple packets. However, programmable switches typically have limited memory, usually in the tens of megabytes, making it impractical to store records indefinitely. Hence, the system implements a timeout threshold. Once this threshold is surpassed, the corresponding record is removed from memory. Additionally, the method employs a multi-stage hash approach because of constraints on accessing data plane memory.

The system also utilizes the Count-Min Sketch (CMS) data structure [14] to estimate the packet count for a particular flow. These packet counts are then compared to a predefined threshold, helping in determining whether a flow should be classified as a long flow. A flow is considered long if its size $>$ 10MB.

C. RTT Separation Method

The proposed system implements the Jenks optimization method [5] in the PDP's control plane to separate the RTTs into independent groups. The Jenks optimization method is a statistical technique used to organize data into meaningful categories by identifying distinct thresholds. In this approach, the algorithm sorts the RTT and groups them into classes defined by the number of queues. By calculating the sum of squared deviations within each potential breakpoint, the algorithm aims to minimize the total sum of squares across all classes. This effectively maximizes differences between classes while minimizing variation. Employing a dynamic programming approach, the algorithm explores various arrangements to find the optimal breakpoints. Once these optimal breakpoints are determined, the RTTs are then classified into their respective queues.

D. Bandwdith Allocation Method

The control plane of the PDP implements the Weighted Max-Min Share (WMMS) fair allocation algorithm [18], which is a method used to distribute the available bandwidth fairly among competing flows. In this application, the algorithm assigns weights to competing flows as a function of the RTT, where higher RTT implies lower priority. Initially, each flow receives a minimum share of bandwidth (maxmin share), proportional to the inverse of the number of flows. Iteratively, flows are given additional bandwidth until fairness is achieved.

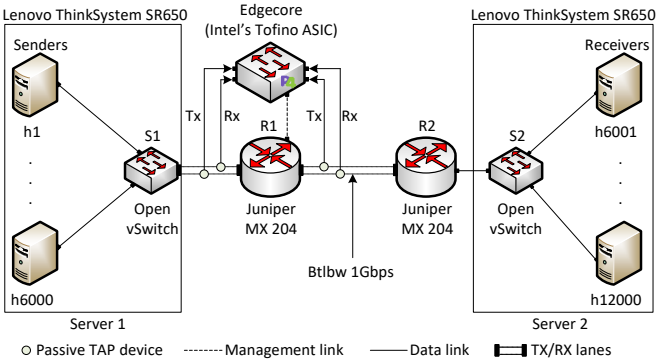


Fig. 3. Topology used to run the experiments.

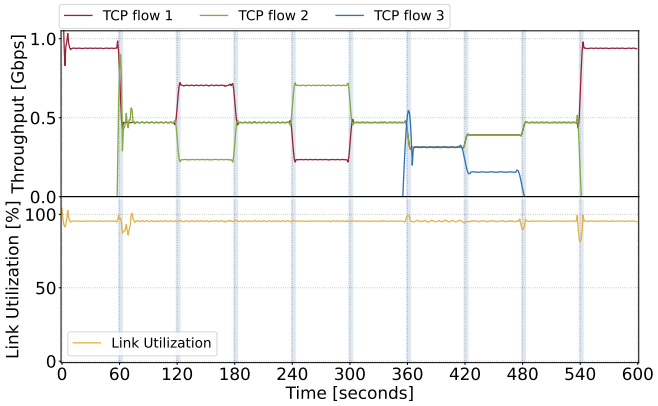


Fig. 4. Throughput and link utilization of three competing TCP flows over a 1Gbps link.

The goal is to maximize the minimum share while considering flow priorities, ensuring a balanced allocation.

Given the maximum available bandwidth as C and N as the number of flows, a TCP flow is guaranteed to obtain at least:

$$C_i = \frac{rttw_i}{\sum_{j=1}^{N} rttw_j} C \tag{1}$$

where $rttw = \{rttw_1, rttw_2, ..., rttw_N\}$ represents the normalized RTT weights of individual TCP flows.

IV. Results and Evaluation

Fig. 3 shows the experimental setup, consisting of 6000 senders (labeled h1 to h6000), each establishing a TCP connection with corresponding receivers (h6001 to h12000). These hosts, created as network namespaces in Linux using Mininet on a physical server, were allocated sufficient resources to ensure the results closely resemble those obtained in real-world scenarios. The TCP send and receive buffers on the end hosts are set to 200MB. The senders and receivers utilize iPerf3 [19] to exchange large data transfers.

The senders are connected to an Open Virtual Switch (OvS) denoted as S1. S1 bridges to the server's (Server 1) network interface, which connects to a Juniper MX-204 router [20] (router R1). A similar configuration is replicated for Server 2. The link between the OvS switches and the routers has a bandwidth of 40Gbps, while the link connecting the routers has a bandwidth of 1Gbps. An optical tap duplicates the traffic between these routers in both directions (Tx/Rx), redirecting it to the PDP. Similarly, there is a tap that duplicates the traffic between Server 1 and router R1. The PDP is the Edgecore Wedge100BF-32X [21] equipped with Intel's Tofino ASIC chip operating at 3.2 Tbps. The data plane of the PDP operates at line rate, enabling it to execute operations swiftly. This capability is leveraged to implement functions for measuring throughput and RTT. Meanwhile, the control plane of the switch receives and processes data plane metrics. These metrics serve as inputs for both the classification algorithm and the queue rebalancing process.

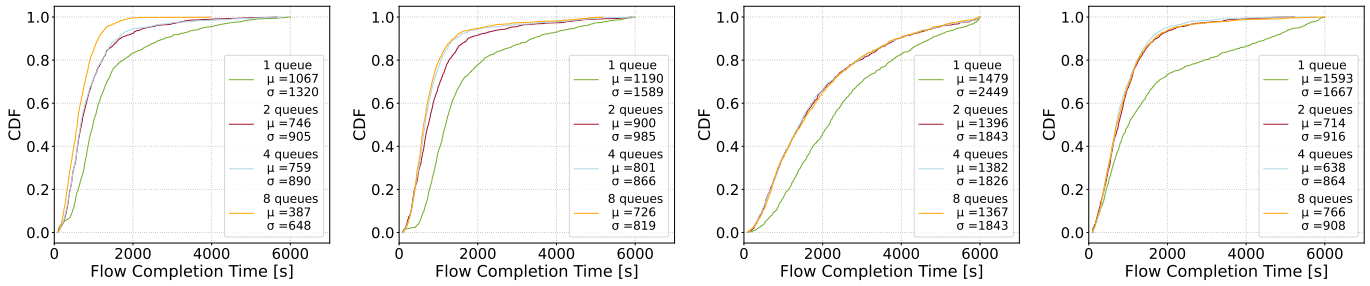
A. Test 1: Rebalancing the Queue

This test evaluates the system's adaptability to changing network conditions. This experiment aims to show how the system operates on a per-flow basis. In this scenario, some flows may not fully utilize the bottleneck link due to the bottleneck being located elsewhere. To address the underutilization of the link, the system is able to track the link utilization on a per-flow basis and reallocate the remaining bandwidth to flows capable of utilizing it fully.

Fig. 4 depicts a scenario involving three TCP flows. Initially, flow 1 fully utilizes the available bandwidth until flow 2 joins at t=60, where the bandwidth is evenly split between them. However, at t=120, flow 2 experiences degradation. Consequently, the system detects this fluctuation and reallocates the unused bandwidth to flow 1, resulting in an increased throughput of around 750Mbps. Conversely, at t=240, flow 1 experiences degradation, prompting the system to allocate the unused bandwidth to flow 2. At t=360, flow 3 joins, and the bandwidth is evenly distributed among the three flows. However, at t=420, flow 3 experiences degradation, and the remaining bandwidth is evenly distributed between flow 1 and flow 2. Similarly, when flow 3 leaves at t=480, flow 1 and flow 2 utilize the remaining bandwidth. As a result, the system optimizes link utilization by redistributing queue allocation.

B. Test 2: Reducing Flow Completion Time of Long Flows

This test assesses the FCT of large data transfers extracted from a MAWI trace dated February 15, 2024 [22]. The extracted values include the RTT of each individual transfer, the start times of the transfers, and the volume of data transferred from sender to receiver. Using this data, we construct 6000 sender-receiver pairs in Mininet to recreate the observed scenario in the MAWI trace. The senders are web servers hosting files ranging from 150 megabytes to 2 gigabytes, while the receivers request these files using the `wget` command. The RTT of each flow ranges from 1ms to 224ms. Metadata capturing the duration of the data transfers is stored for result analysis. Fig. 5 examines various scenarios involving different CCAs and multiple queues. It is worth



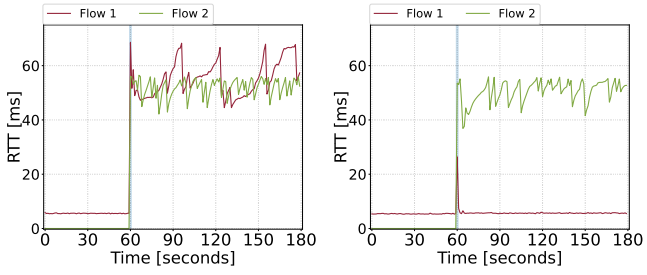
(a) CUBIC (b) HTCP (c) BBRv1 (d) BBRv2

Fig. 5. This experiment presents the Cumulative Distribution Function (CDF) of the FCT for 6000 long flows over a 1 Gbps link using the MAWI traces. 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.

noting that while the MAWI traces do not directly provide information on the CCA in use, most Internet applications commonly employ CUBIC and BBRv1 [23]. Additionally, the result also includes HTCP [24] which is a variant of TCP that is used for high-speed long-distance networks and BBRv2 [25] which is a newer version of TCP BBR. Fig. 5(a) illustrates that the system enhances the average FCT of long flows by segregating them into multiple queues and allocating bandwidth to each queue based on its average RTT. It's observed that employing multiple queues improves the average FCT compared to using just one queue, primarily by alleviating the bufferbloat issue. Bufferbloat occurs when routers have large buffer sizes, leading to additional delay as loss-based CCAs fill the buffer excessively. In Fig. 5(b), it's shown that HTCP exhibits a similar average FCT to CUBIC. Additionally, no significant difference is observed when flows are separated into 4 or 8 queues. Figs. 5(c-d) indicate that increasing the number of queues does not significantly impact the performance of BBRv1 and BBRv2 flows. This behavior can be attributed to BBR's design, which aims to estimate Kleinrock's operating point [26], thus reducing delays caused by bufferbloat. Consequently, segregating flows into more than two queues does not substantially alter BBR's operating point.

C. Test 3: Minimizing Bufferbloat

Bufferbloat arises when router buffers become oversized, holding more packets than the port's sending rate can handle



(a) wo/ bufferbloat prevention (b) w/ bufferbloat prevention

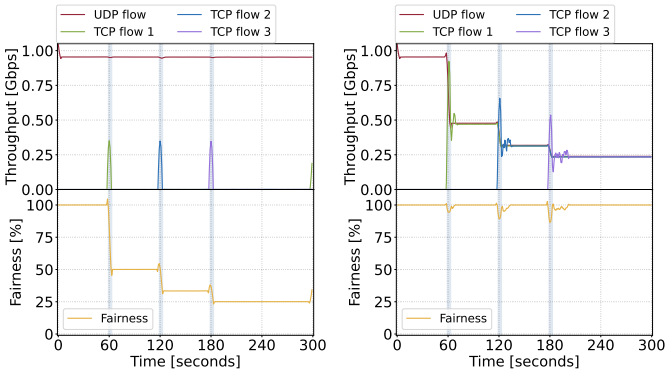
Fig. 6. RTT of two CUBIC flows. (a) wo/ bufferbloat prevention. (b) w/ bufferbloat prevention.

[6]. As a result, the end-to-end delay is increased. This issue is widespread across the Internet and is often incorrectly attributed to network congestion. Unlike end hosts, intermediary devices have a clearer view of network events beyond just packet losses and delays. They can discern the source of these events and complement the actions taken by CCAs on end hosts. The proposed system monitors flow RTT and identifies when a flow is introducing unnecessary latency. It then takes action by segregating such flows into different queues to isolate their effects. Fig. 6(a) illustrates how RTT changes over time when two flows share the same buffer. At first, flow 1 encounters a low RTT. However, when flow 2 joins and injects more packets than the egress port can manage, it leads to increased latency for both flows. In contrast, Fig. 6(b) demonstrates the bufferbloat prevention mechanism implemented by the system. Here, the operator can set a threshold for RTTs on a per-queue basis. If a new flow increases overall latency beyond this threshold, it is reallocated to a different queue alongside flows experiencing similar latency.

D. Test 4: Mitigating UDP Abuses

While widely utilized by many multimedia applications, UDP is a protocol that is susceptible to abuse [27]. Unlike TCP, UDP lacks any congestion control and connection management mechanisms such as the TCP three-way handshake. This limitation allows an application employing UDP to generate traffic at a high rate, potentially causing congestion and overflows at routers, switches, and other network devices. In contrast, TCP employs congestion control by significantly reducing the sending rate when congestion is detected. Consequently, UDP abuse can lead to detrimental effects such as one UDP flow monopolizing most of the network bandwidth, crowding out TCP flows, and resulting in unfair bandwidth allocations and starvation.

The proposed system can mitigate UDP abuses, which are common in Domain Name System (DNS) amplification attacks, which is type of Distributed Denial of Service (DDoS) attack [11, 28]. Fig. 7(a) shows a scenario without UDP abuse prevention. In this case, all flows share a single queue which is typically the case in enterprise routers. A UDP flow starts sending data at t=0 and and after 60 seconds, subsequent TCP



(a) wo/ UDP abuse prevention (b) w/ UDP abuse prevention

Fig. 7. UDP abuse scenario. (a) wo/ UDP abuse prevention. (b) w/ UDP abuse prevention.

flows join. It is observed that the performance of the TCP flows is degraded and the fairness index fluctuates from 50% to 33% finally settling around 25% when all the four flows are present. On the other hand, Fig. 7(b) shows a scenario with flow separation and bandwidth limitation. In this case, the UDP flow can fully utilize the available bandwidth if no other flow is present. When the TCP flows join, the system allocates them in different queues in order to isolate their dynamics and enforce a fair bandwidth utilization.

V. Conclusion and Future Work

This paper proposes a system designed to enhance the fairness among competing flows by computing their throughput and RTTs. The system employs a passive approach to take corrective actions to mitigate potential negative interactions among competing flows. The experiments illustrate the system's ability to reduce the FCT of long flows, considering a large number of flows and latencies observed in real-world traces. Additionally, the system can rebalance queues to minimize the impact of bufferbloat and mitigate the effects of well-known DDoS attacks at the network layer. Future work will focus on extending the system to perform flow separation based on the CCA employed by the flows.

Acknowledgment

The authors would like to acknowledge the National Science Foundation (NSF) for supporting this work, under award 2346726. The authors would also like to acknowledge the FABRIC [29] team for facilitating the platform used for some preliminary tests.

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] 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.

[4] 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*, 2014.

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

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

[7] J. Crichigno, W. Shu, M. Wu, "Throughput optimization and traffic engineering in wdm networks considering multiple metrics," in *Proceedings of the 2010 IEEE International Conference on Communications*, 2010.

[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] 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.

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

[11] 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.

[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] Z. Meng, N. Atre, M. Xu, J. Sherry, and M. Apostolaki, "Confucius queue management: Be fair but not too fast," *arXiv preprint arXiv:2310.18030*, 2023.

[14] 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.

[15] 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), Italy*, 2023.

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

[17] 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.

[18] T. Hill, "Partitioning general probability measures," *The Annals of Probability*, 1987.

[19] J. Dugan, S. Elliott, B. Mah, J. Poskanzer, and K. Prabhu, "iPerf3, tool for active measurements of the maximum achievable bandwidth on IP networks." [Online]. Available: https://github.com/esnet/iperf Accessed on 04-20-2024.

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

[21] Edgecore Networks, "Wedge 100BF-32X." [Online]. Available: https://tinyurl.com/2xay8kky, Accessed on 04-20-2024.

[22] MAWI Working Group Traffic Archive, "Packet traces from WIDE backbone." [Online]. Available: https://tinyurl.com/2e88vw2v, Accessed on 04-04-2024.

[23] 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.

[24] D. Leith and R. Shorten, "H-TCP: TCP for high-speed and long-distance networks," in *Proceedings of PFLDnet*, 2004.

[25] Google, "TCP BBRv2 Alpha/Preview Release." [Online]. Available: https://github.com/google/bbr/tree/v2alpha, Accessed on 04-20-2024.

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

[27] D. Thomas, R. Clayton, and A. Beresford, "1000 days of UDP amplification DDoS attacks," in *2017 APWG Symposium on Electronic Crime Research (eCrime)*, 2017.

[28] A. AlSabeh, E. Kfoury, J. Crichigno, and E. Bou-Harb, "P4DDPI: Securing P4-programmable data plane networks via DNS deep packet inspection," in *NDSS Symposium*, 2022.

[29] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "FABRIC: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, 2019.