When TCP Meets Reconfigurations: A Comprehensive Measurement Study

Kaan Aykurt*, Johannes Zerwas*, Andreas Blenk*†, Wolfgang Kellerer*
*Chair of Communication Networks, Technical University of Munich, Germany
†Siemens AG, Munich, Germany

***Abstract*—The diversity of deployed applications in data centers leads to a complex traffic mix in the network. Reconfigurable Data Center Networks (RDCNs) have been designed to fulfill the demanding requirements of ever-changing data center traffic. However, they pose new challenges for network traffic engineering, e.g., interference between reconfigurations, transport layer protocols, and congestion control (CC) algorithms. This raises a fundamental research problem: can the current transport layer protocols handle frequent network updates?**

This paper focuses on TCP and presents a measurement study of TCP performance in RDCNs. In particular, it evaluates diverse traffic mixes combining TCP variants, UDP, and QUIC transport protocols. The quantitative analysis of the measurements shows that migrated TCP flows suffer from frequent reconfigurations. The effect of reconfigurations on the cost, e.g. increased Flow Completion Time (FCT), depending on the traffic mix is modeled with Machine Learning (ML) methods. The availability of such a model will provide insights into the relationship between the reconfiguration settings and the FCT. Our model explains 88% of the variance in the FCT increase under different reconfiguration settings.

***Index Terms*—reconfigurable data center networks, TCP measurements, QUIC measurements**

I. Introduction

Over the past decades, the introduction of cloud computing transformed a significant portion of the information technologies industry [1]. With this transformation, many new applications appeared in DCNs with different requirements. For instance, distributed machine learning applications demand high bandwidth to transmit large amounts of data while other applications need to fulfill strict latency and high availability constraints such as deployments of 6G core networks [2].

As a result, the bandwidth and latency requirements of modern data center networks are far more different than those of conventional infrastructures. Conventional fully provisioned static DCNs are either too costly or fail to meet this challenging demand since it is not feasible to create permanent high bandwidth links across the racks [3]. Therefore, the research community has introduced reconfigurable network elements, such as Optical Circuit Switches (OCS), to enable temporary high bandwidth connections between source and destination pairs on demand, and thereby, meet the challenging demands [3]–[7]. The adaptation of the topology, i.e., the creation and destruction of new links, causes the flows to be re-routed and to be migrated to different links without the knowledge of the receiver or sender. The frequencies of these migrations depend on the reconfiguration scheme of the OCS.

The RDCNs enable better performance for DCNs by establishing low latency and high throughput. However, they also present a research question: what is the effect of introducing temporary paths with different provisioning periods on the FCT performance?

The performance analysis of RDCNs exists in the literature to some extent [3], [5]–[8]. However, these analyses mainly take into account the flow-level analysis [3], [5]–[7] or take a more theoretical perspective [5], [8]. In reality, the consideration of packet-level characteristics may pose a challenge to utilize the full potential of RDCNs. Combinations of high-bandwidth circuit networks and traditional packet-switched networks introduce non-trivial problems such as flow interruptions due to reconfiguration downtimes and bandwidth fluctuations [9]. Consequently, the changes in the bottleneck link capacity that emerge from path reconfigurations, pose a problem for end-to-end network connections: can transport layer protocols utilize the congested link capacity efficiently with rapid fluctuations in the available bandwidth?

The heterogeneity of network traffic loads, the dynamic nature of modern data centers, and the existence of complex, diverse paths between two end-hosts make the role of network transport protocols more critical for ensuring seamless end-to-end connectivity. Transmission Control Protocol (TCP) is the current *de facto* standard transport protocol of modern DCN architectures. TCP connections adjust their sending rates according to the available bandwidth. The rate limitations, in general, are modeled with the consideration of the capacity of a bottleneck link and a particular Round Trip Time (RTT). The networking community has analyzed TCP behavior in static environments [10]–[13]. However, in public cloud environments, multiple tenants can co-exist and the performance requirements, e.g. low-latency or high throughput, for them may differ from one another. To meet the performance goals, tenants may prefer different CC algorithms that lead to a shared network of different applications and interference of various CC algorithms [14]–[16]. The changing nature of such DCNs means that a mix of transport layer protocols and their variants can coexist [17], [18]. Some examples of these are TCP variants (CUBIC, Reno, BBR, Westwood, etc.), User Datagram Protocol (UDP), and QUIC. While the available studies for measuring the interaction between TCP and QUIC include performance benchmarks and analyze the interaction of TCP and QUIC in a static link [19], [20], they do not include dynamic scenarios in their analyses. Additionally, available studies for TCP performance in dynamic environments are also limited and lack the interaction of co-existing variants and congestion [9]. Specifically, the existing work does not include an in-depth analysis of various coexisting CC algorithms in a testbed environment. To the best of our knowledge, performance analysis of TCP behavior on dynamic architectures with coexisting flows of different natures is not yet available.

This paper analyzes the FCT performance of various TCP variants and QUIC under different reconfiguration schemes, measured in a testbed environment. In particular, loss-based, capacity-based, hybrid, and ECN-based CC is evaluated. The measurement results are used to model the effect of frequent path migrations on the migrated flow. In particular, this model predicts the increase of the FCT given the reconfiguration settings and involved TCP variants. The outcomes of the model provide insights into the interaction between the reconfiguration scenarios and the FCT. Additionally, a model to classify the TCP variants is introduced to enable network operators to gain insights into the traffic mix in their RDCNs.

This work contributes to the networking community by shedding light on the interactions of various TCP variants and transport layer protocols in a dynamic environment. The findings of this paper indicate that both the traffic mix and the reconfiguration period in a dynamic environment affect TCP performance.

Initial results of this paper have been presented at IEEE CNSM 2022 [21]. In this journal version, we extend our measurement framework with the inclusion of DCTCP, UDP, and QUIC in the traffic mix. Additionally, we present an analysis with more than two flows and extend our modeling to cover TCP variant classification. Our extended findings show that in addition to the CC algorithm, its implementation also plays an important role in TCP performance.

The remainder is structured as follows: Sec. II gives a brief overview of RDCN and widely used transport layer protocols. Sec. III lists related measurement studies. Sec. IV describes the testbed and measurement procedure. Sec. V presents the analysis of flow migrations under different reconfiguration scenarios. Finally, Sec. VI introduces an ML model to predict the FCT prolongation and TCP variant classification. We conclude and discuss future work in Sec VII.

II. Background

This section gives an introduction to RDCNs, briefly provides an overview of widely utilized TCP variants, and introduces UDP and QUIC.

A. Reconfigurable Data Center Networks

Fig. 1 shows a hybrid RDCN with N racks. The traditional, static packet-switched network is augmented by a circuit-switching element to provide rack-level direct connectivity. The packet network is a static environment, where the topology cannot be configured. The circuit switch, typically realized by a reconfigurable device such as an OCS, introduces dynamicity to an RDCN. In a typical setup, all racks are connected via the packet network permanently to provide basic connectivity for

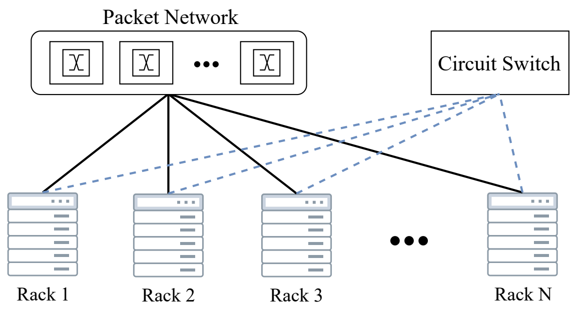


Fig. 1. Overview of an RDCN. A circuit switch is present to introduce dynamicity in addition to the conventional packet network.

latency-sensitive flows [4]. Additionally, the circuit switch, e,g. OCS, creates temporary links between rack pairs on demand. Limited availability on ports of the circuit switch and the high financial costs of full provisioning means that the high bandwidth network cannot be utilized all the time by the racks. Therefore, a scheduling algorithm for the circuit switch network is required to connect rack pairs on demand [3], [22].

The authors of [23], [24] showed that during circuit reconfiguration, no circuit links can be used, which causes a reconfiguration downtime. In RDCNs, a typical circuit reconfiguration schedule consists of 90% circuit up-times and a 10% circuit downtime [22]. We refer to circuit up-time as the *day time* and downtime as the *night time*.

Flow-level simulation studies of RDCNs have shown that their performance is superior in comparison to the static topologies, e.g., [3], [4]. However, these analyses consider 100% link utilization immediately after reconfigurations and do not consider the mix of TCP variants in the traffic pattern. When packet-level traffic characteristics are taken into account, the assumption of efficient and fair link usage may not hold. Therefore, the real cost of frequent reconfigurations on RDCN performance requires the analysis of transport layer protocols and their behavior under frequent reconfigurations.

B. TCP

The TCP/IP stack, which is still a part of today's internet, was introduced back in the 1980s [25]. The transport layer presented in the modern TCP/IP stack is implemented in the operating system kernel and has an end-to-end view of the connection, which considers only a single logical link between the two endpoints [26]. However, particular characteristics of individual links in each hop between the endpoints may influence the transport protocol's behavior.

TCP is one of the most popular transport layer protocols. It enables as fast as possible message exchange across networking entities with guaranteed delivery and reliability through the acknowledgment mechanism [25]. Although the mechanism appears to be straightforward, there are essential parameters in a TCP connection. One of these parameters is the ordering of the packets received. The burst of packets may change the order of sending during transmission in the link and cause overhead to the receiver. These packets have to be reassembled in the receiver. The other important aspect is the size of the group of packets transmitted in each period before waiting for an ACK from the receiver. This is referred to as the bytes-in-flight or the congestion window size in the literature. The determination of this parameter is essential for ensuring fast delivery. A smaller congestion window size leads to slower transmission, whereas a larger congestion window size may cause packet losses. The packet losses incurred by the TCP connection cause retransmissions. Missing packets that are not successfully received by the receiver have to be retransmitted by the TCP protocol to successfully deliver a message stream.

The availability of a set of parameters enables flexibility in design for various TCP implementations. The trade-off between latency and reliability allows for different TCP CC designs. Additionally, these algorithms also determine the distribution of the available bandwidth between flows. Therefore, in a broader sense, the goal of TCP CC is to maximize the sending rate while ensuring reliable and fair communication. Based on their nature, TCP variants can be categorized into five groups: loss-based, delay-based, capacity-based, hybrid, and explicit feedback-based CC.

Loss-based CC: Packet losses are one of the most commonly used congestion signals. The detection of a packet loss triggers the loss-based algorithms to adjust their congestion window size. The investigated loss-based TCP variants in this paper are CUBIC [27] and Reno [28]. These algorithms grow their congestion window sizes until a packet loss occurs. If a loss occurs after the growth period, this indicates congestion in the link. This causes the congestion window size to shrink. Different implementations use different approaches for increasing and decreasing the window size. CUBIC estimates the packet loss with a cubic function of the last time since a packet loss occurred. In contrast, Reno employs an Additive Increase, Multiplicative Decrease (AIMD) scheme.

Delay-based CC: Another technique for detecting congestion is continuously monitoring packets' RTT. An increase in RTT is used to indicate queue buildup in the network. Pure RTT-based congestion detection, such as TCP Vegas [29] introduces a fairness problem. RTT is the first parameter that increases if more than two flows compete on the same link since the queues build up before a packet loss happens. This introduces a problem for competition for delay-based CC algorithms and causes them to suffer in terms of achieved throughput. This paper omits the analysis of delay-based algorithms, as they are known to compete poorly with more aggressive, loss-based schemes [30].

Capacity-based CC: The links have limited capacity, and they need to be shared between multiple flows in general. One of the methods of adjusting congestion window size is to estimate the available capacity. The capacity-based algorithms track the estimated bandwidth to adapt the congestion window size after a loss [26]. The estimation phase is determined in the start phase and updated throughout the serving process. An example of this implementation investigated in this paper is TCP Westwood [31].

Hybrid CC: One other alternative to detect congestion is to combine more than one metric. TCP BBR [32] is an example of the category of hybrid CC mechanism which continuously measures RTT and link capacity to determine the congestion window size. The capacity estimation relies on predicting the Bandwidth-Delay Product (BDP). BDP is the product of a link's capacity and the RTT. The consideration of these two parameters enables a hybrid mechanism to detect congestion. Our analysis focuses on TCP BBR v1 in the remainder of this paper.

Explicit Feedback-based CC: Finally, a subset of TCP variants relies explicitly on feedback from the network to detect congestion. While TCP is implemented in the operating system kernel, the availability of an external notification field that indicates congestion helps detection much easier and more efficiently. DCTCP [33], PowerTCP [34], and XCP [35] are manifestations of the explicit feedback-based CC category. Analysis has shown that explicit feedback can benefit congestion window size adjustment significantly in controlled DCN environments [36]. This paper focuses on the investigation of DCTCP [33].

C. UDP

UDP is one of the core transport layer implementations as an alternative to TCP. In comparison to TCP, UDP speeds up transmission by transferring data without establishing a successful connection, i.e., it is connectionless. This means that UDP does not provide guarantees for packet delivery. Therefore it is mainly used for loss-tolerating connections that require low latency. This paper uses UDP as background traffic and measures the impact of UDP traffic on TCP performance in the presence of reconfigurations.

D. QUIC

QUIC [37] protocol was developed by Google in 2012. It was deployed in the Chromium source code and has gained popularity since then. It is estimated that over 40% of Google's traffic uses QUIC [38]. QUIC is implemented in user space on top of UDP. Although it is built on top of UDP, it also provides TCP's reliable transmission guarantees. However, the user space implementation means that QUIC can be customized and tailored to application needs. Among many existing implementations, we focus on LiteSpeed QUIC (LSQUIC) [39] due to its high rate transmission performance in the remainder of this paper.

III. Related Work

We are not aware of any thorough analysis of TCP flows subject to frequent migrations. However, several related works exist, which investigate TCP behavior under different scenarios. Most of the existing TCP analyses focus on static DCN topologies and employ simulation studies. The analysis of widely used TCP variants in static DCNs is presented in [36]. The performance analysis of coexisting DCTCP and other TCP variants are investigated in [14]. Improvements to DCTCP and an overview of existing CC implementations are discussed in [40]. Jain et al. discuss the TCP implementations focusing on wireless and, in general, low-bandwidth lossy links [41]. TCP performance in static DCNs under different CC schemes is analyzed in [42], [43]. The benchmark of TCP and QUIC in isolated settings are also analyzed in-depth in the literature [19]. Additionally, the literature has also focused on measuring TCP performance when there are also QUIC flows in the traffic mix [20]. However, none of these works considers flow migrations. Overall, these papers do not take into account the dynamic nature of RDCNs.

In RDCNs, the effect of packet re-ordering on TCP throughput has also been analyzed and known for a very long period of time [44]. Recently, Cârpa et al. have built on top of this work and analyzed the effect of switch reconfigurations on TCP performance [45]. This work considers multiple flows competing on the same bottleneck. However, it only focuses on TCP CUBIC and lacks a variation of CC algorithms.

Mukerjee et al. [9] use an open-source RDCN emulator to analyze the performance of TCP variants under reconfigurations. Specifically, they investigate the TCP variants' ability to ramp up their sending rate after frequent reconfigurations. They suggest an adaptation of TCP to account for short-term link capacity changes. However, this work lacks the interaction of different TCP variants, transport protocols, and does not consider heterogeneous traffic mixes. Moreover, they consider the migration of all the flows to an uncongested link, whereas in a real RDCN environment, congestion in the static packet network is also likely to occur. Existing works do not reflect real-world network environments' complex nature. To close this gap, we present a comprehensive testbed measurement study to investigate the behavior of CC algorithms subject to frequent reconfigurations with varying traffic mixes.

In addition to empirical measurement findings, the successful modeling of TCP throughput may benefit the CC algorithms. Prophet [46] framework provides a model to predict the throughput of TCP flows under static DCNs and strict conditions. Authors of [47] build an ML model to infer TCP characteristics from a set of passive measurements in static DCNs. CCAC [48] tool relies on theoretical analysis and presents a model to verify certain properties of CC algorithms before deployment and lacks modeling of multiple flows competing on the same bottleneck. The existing ML models predict the throughput to enable better congestion window size adjustment in static DCNs. However, these analyses have a limitation to their approach. The generalized prediction of these models to RDCNs does not hold. In order to close this gap, this paper proposes an ML model to relate the TCP variants and reconfiguration settings to the FCT.

IV. Measurement Setup

The goal of this paper is to compare the network utilization of flows under varying reconfiguration scenarios on a small representative setup. This section establishes a programmable data plane that emulates an OCS and describes the measurement settings.

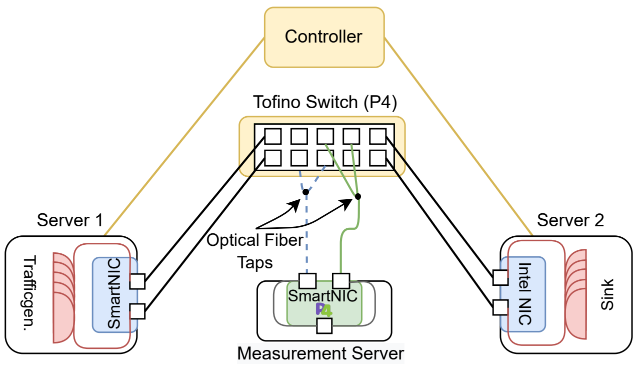


Fig. 2. Testbed overview.

A. Testbed

Fig. 2 shows the testbed. It consists of three servers, a `Switch` and a `Controller`. The servers are running Ubuntu 18.04 (4.15.0-173-generic kernel) and this study uses the official CC implementations available in the Linux Kernel. `Server 1` and `Server 2` are connected to the `Switch` via two 10 Gbps optical cables (black solid lines). The existence of two separate physical links allows packet generation of up to 20 Gbps in `Server 1` without facing a bottleneck in the uplink to the `Switch`. The server CPU and bus are capable of handling speeds up to 40 Gbps, and hence 20 Gbps traffic generation can be achieved. Four ports of the `Switch` are connected via loopback cables (green lines). The objective of these links is to introduce congestion of packets only in the loopback link. With this setup, a typical OCS behavior is emulated where the congestion vanishes when the flows are migrated to a newly set up dedicated link and it increases when the OCS link is torn down. The `Measurement Server` is connected to the optical taps in the loopback links. The `Controller` has 1 Gbps connection with all the entities to orchestrate the experiment.

The traffic is generated by using iPerf version 2.0.10 [49] for flows greater than 1 GB in `Server 1`. Smaller flows are generated via conventional server-client fashion HTTP file transfer. The packet sizes are set to 9000 Bytes, to reduce the overhead of packet processing and collection. As for the NIC, Netronome Agilio SmartNIC NFP-4000 [50] is used without any custom P4 programs. `Server 2` is designated as the traffic sink, and two separate server processes are being run on two different ports of the Intel X710 series NIC [51].

The incoming packets from `Server 1` to the `Switch` are initially configured to traverse the color-coded loopback links on the `Switch`. The forwarding rules are installed manually. The outgoing direction of the loopback links is tapped, and the data packets are collected in the `Measurement Server`. The `Switch` is then programmed to forward data packets coming from the loopback links to their final destination on `Server 2`. With this architecture, the traffic is tapped passively and the state of TCP is preserved on separate physical ports of `Server 2`.

The Measurement Server features a SmartNIC with two ports. A custom P4 program is loaded to SmartNIC to forward incoming packets from physical ports to a virtual interface. During the forwarding process, the IP ToS field of the packets is modified to distinguish the incoming packets from different physical interfaces, as it is not possible to differentiate from the virtual interface otherwise. *TCPDUMP* traffic collection is run on the virtual interface, and all the data packets are collected with software timestamping.

The Controller is present to orchestrate the measurement process. In addition to the remote experiment Controller shown in Fig 2, the Switch has its own local controller component that translates commands from the experiment controller. The local manager of the Switch is instantiated with the remote Controller at the beginning of the experiment and it allows for reconfiguring of the Switch with microsecond precision. We implement the local manager as a C++ program that receives high-level commands from the overall experiment controller and modifies the configuration of the P4 program or table entries accordingly. With the provided Software Development Emulator (SDE) by Intel, we utilize C++ bindings to control the programmable ASIC, e.g., to create new entries in match-action tables or to read/modify register values. Specifically, our local manager implements the loop that modifies the forwarding table entries to migrate the flows back and forth.

B. Measurement Procedure

Unless stated otherwise, the flows are created in the Server 1 at the same time. In the initial state, packets arriving at the Switch are forwarded to green and blue (dashed) loopback links such that they follow completely separated paths in the loopback link. In the second state (after one reconfiguration), blue packets traversing the blue loopback link are migrated to the green loopback link, causing congestion. The rules differentiating packets leaving the Switch to the destination server are kept intact during the rule update. With this approach, the migration takes place only in the loopback links independent of the flow source and destination. This reconfiguration scheme is then applied back and forth with a pre-defined reconfiguration period.

ARP entries to Server 1 and Server 2 are installed prior to the measurement. Therefore no ARP broadcasting and messages are being forwarded via the Switch. Moreover, the Switch is using a single buffer for all the incoming packets from all the ports and hence making the likelihood of ACK packet drops very high. Therefore, unlike the data packets in the outgoing direction, TCP ACK packets originating from Server 2 are routed around the Switch, meaning that they do not trace the loopback link. By bypassing the loopback link, ACK packets are transported instantly, hence reducing the likelihood of ACK packet drops. This behavior emulates the asymmetric routing in RDCNs, where large flows are mainly subject to migration and small flows are served via the static topology [4].

TABLE I
SUMMARY OF THE INVESTIGATED SCENARIOS.

	Values
Transport Protocols	TCP, UDP, QUIC
TCP Variants	CUBIC, BBR, Reno, Westwood, DCTCP
Reconfiguration Periods	None, $1\,\text{ms}$, $5\,\text{ms}$, $10\,\text{ms}$ $25\,\text{ms}$, $50\,\text{ms}$, $100\,\text{ms}$
Reconfiguration Downtimes	None, $1\,\text{ms}$, $2\,\text{ms}$, $5\,\text{ms}$, $10\,\text{ms}$
Number of Flows	1, 2, 8, 10, 20
Flow Sizes	2^n MB for n in [0, 8], $2\,\text{GB}$, $4\,\text{GB}$

The summary of analyzed scenarios, parametrized by the TCP variant, reconfiguration period, reconfiguration downtime, number of flows, and flow sizes are presented in Table I. The values column shows the available settings and the complete list of scenarios includes the combination of the specified settings. The most popular TCP variants from different CC categories are selected for investigation. Reconfiguration periods range from 0 to $100\,\text{ms}$ to account for accurate OCS reconfiguration scenarios in RDCNs [52]. Reconfiguration downtimes reflect the widely proposed values in the literature and reconfiguration periods are configured to result in a 90% duty cycle which is the common choice in the literature [3]–[8]. Flow sizes are selected to cover elephant ($2\,\text{GB}$ and $4\,\text{GB}$) and mice flows. The number of flows represents anticipated edge cases: a single flow, 2 competing flows as well as multiple flows.

V. EVALUATION

Frequent migrations of the flows are expected to affect the achieved throughput of the flows and hence the FCT. The intuition behind this expectation stems from the fact that the flows might take some time to ramp up their sending rates (due to TCP CC algorithm behavior) after the reconfigurations and lead to the under-utilization of the links. In this section, we analyze the packet traces collected during the measurement process to investigate the effect of the reconfiguration period on TCP performance. The results report averages and 95% confidence intervals from 30 measurement runs.

The evaluations are structured as follows: Sec. V-A establishes the benchmark of the testbed with a single flow. Sec. V-B analyzes two competing TCP flows using the same CC algorithm. Sec. V-C evaluates the competition of two different TCP variants. Sec. V-D investigates the performance of DCTCP. Sec. V-E measures the effect of reconfiguration downtime on TCP performance. Sec. V-F considers the scenarios consisting of many small flows. Sec. V-G analyzes TCP behavior when UDP traffic is present in the background with bursts. Sec. V-H sheds light on the results when there are more than two flows in the traffic mix. Finally, Sec. V-I presents TCP performance when interacting with QUIC traffic.

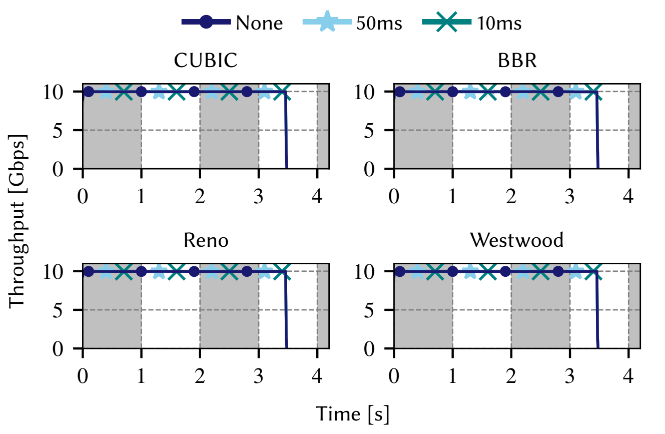


Fig. 3. Average throughput comparison of a single TCP flow under frequent reconfigurations. In the absence of congestion, there is no effect on the FCT.

A. Benchmark of the Testbed

In order to establish a benchmark of the testbed for comparing flow behavior in case of congestion, a single TCP flow with $4\,\mathrm{GB}$ volume is considered. This flow is migrated between two links with a fixed reconfiguration period for each measurement scenario. The measurement scenarios for benchmarking the testbed include reconfiguration periods of $10\,\mathrm{ms}$, $50\,\mathrm{ms}$, and no migration at all. The benchmark of the testbed indicates that the minimum average RTT $300\,\mathrm{us}$ is achieved with BBR, whereas the other variants' RTT is between $1\,\mathrm{ms}$ and $2.5\,\mathrm{ms}$ depending on the congestion window size.

Fig. 3 shows the achieved throughput for each TCP variant with respect to time and reconfiguration period. The 95% confidence intervals are also plotted with less opacity in the background with matching colors. However, it is not visible since the variance between the distinct measurement runs is very low. The results indicate that all the TCP variants achieve maximum theoretical throughput in less than $10\,\mathrm{ms}$. Regardless of the reconfiguration period, the flows can achieve $10\,\mathrm{Gbps}$ throughput and the flows are complete in $3.47\,\mathrm{s}$ as expected by theoretical calculations (also considering the headers in addition to the payload). Since the throughput is not affected by the reconfiguration period, the benchmarks indicate that the `Switch` does not drop packets during reconfiguration, and in the absence of congestion frequent path migrations do not affect the FCT.

B. Competition of Two Flows: Same TCP Variant

Congestion is likely to occur in a DCN environment, where multiple flows coexist. Accordingly, we investigate a scenario with two flows of $2\,\mathrm{GB}$ each. One of these flows is referred to as the *migrated* flow, which is periodically migrated in the loopback link shown in Fig. 2. The other flow (*main* flow) is served on the same link, which is shown by the green link in Fig. 2. Our migration approach, as introduced in Sec. IV-B, introduces periodic congestion in the loopback link, and after re-migration, the congestion vanishes.

Throughput: Fig. 4 shows the average throughput of the main and migrated flow with TCP CUBIC under different reconfiguration periods. With $50\,\mathrm{ms}$ reconfiguration period, the main and migrated flow's FCTs do not differ significantly. A jigsaw pattern of the throughput between $10\,\mathrm{Gbps}$ and $5\,\mathrm{Gbps}$ is evident. This indicates that during the congestion period, the rate is allocated fairly, and $50\,\mathrm{ms}$ gives enough time for TCP CUBIC to ramp up its sending rate on the uncongested link. However, at $25\,\mathrm{ms}$, a decrease in the average throughput of the migrated flow is observed. This directly translates into an increase in the migrated flow's FCT. Below $10\,\mathrm{ms}$, the magnitude of the FCT increase grows and the 95% confidence intervals of the throughput do not overlap anymore. At $1\,\mathrm{ms}$, the growth of the TCP congestion window after reconfiguration is not fast enough and, hence the migrated flow achieves less throughput consistently. The findings indicate that the FCT prolongation and the reconfiguration period are inversely related, which confirms the observations of Mukerjee et al. [9]. Although not shown in this figure, an increased FCT for the migrated flow is similarly observed for the other TCP variants as well. These extensive measurements serve as a baseline for network operators to fine-tune the reconfiguration period for fair bandwidth allocation.

Congestion Window Size: In order to elaborate on the analysis of varying FCTs with different reconfiguration scenarios, Fig. 5 shows the congestion window size of both flows. A baseline (from the measurements in Sec. V-A) of the congestion window size is also plotted to outline the benchmark values. The ideal CC behavior is to ramp up the maximum size and keep it constant throughout the transmission to achieve a small FCT. For our specific testbed environment, the maximum allowable congestion window size without encountering a packet loss is determined as around $3000\,\mathrm{KB}$ from baseline measurements. Since TCP CUBIC uses a cubic function to estimate the congestion window size, a concave component where the window size ramps up sharply and a convex component where CUBIC slowly probes for more bandwidth can be observed in the baseline measurements. At $50\,\mathrm{ms}$ reconfiguration period, the congestion window size of CUBIC indicates a pattern with peaks and valleys. The valleys correspond to the immediate decrease in the congestion window size after reconfiguration, hence encountering a packet loss due to congestion. The peaks align with the time instances just before reconfiguration, where the algorithm is trying to ramp up its maximum allowable bytes in flight. However, even at $50\,\mathrm{ms}$, the peaks are only at 60% of the baseline values , and the convex shape cannot be observed in the periodic congestion event. As the reconfiguration period is decreased further, the jigsaw pattern consisting of peaks and valleys becomes less evident and the migrated flow suffers significantly from small congestion window size. Overall, the analysis of congestion window size yields results parallel to the throughput observations and serves as an explanatory factor for increased FCT of the migrated flow with a decreasing reconfiguration period.

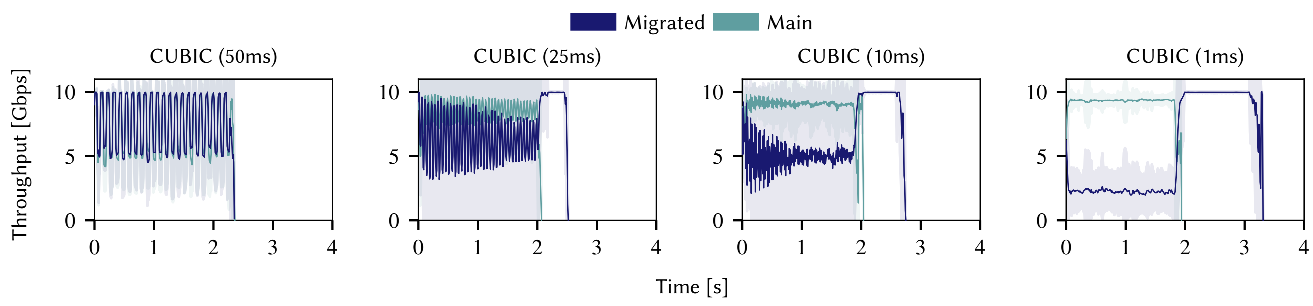


Fig. 4. Average throughput of TCP CUBIC flows with varying reconfiguration periods. 95% confidence intervals are presented in the background with matching colors. The migrated flows ramp up their sending rate after the main flows finish. Overall, decreasing the reconfiguration period increases the migrated flow's FCT.

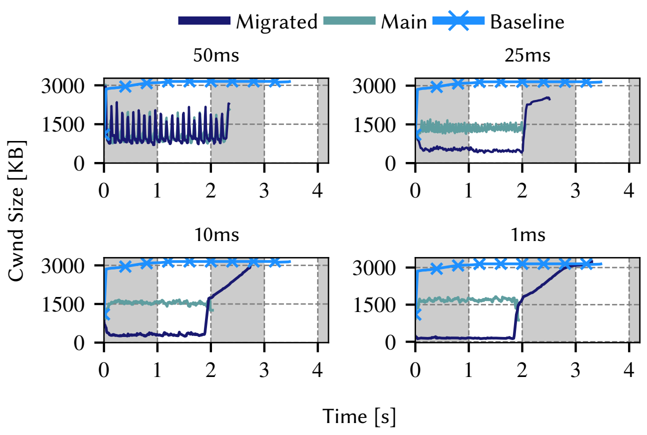


Fig. 5. Average congestion window size of TCP CUBIC under different reconfiguration periods. The congestion introduced by frequent migrations hinders TCP CUBIC's ability to ramp up its congestion window size. The migrated flow can only ramp up its congestion window size after the main flow finishes at around $t = 2\,\mathrm{s}$.

C. *Competition of Two Flows: Different TCP Variants*

Previous studies have shown that many TCP variants coexist alongside each other [9], [14]. The goal of comparing two different TCP variants is to shed light on the scenarios involving the interaction of different TCP variants. Fig. 6 shows the average throughput of migrated and main flows of different TCP variants. Similar to the previous sections, it presents the mean throughput and 95% confidence interval per time instance for 30 different measurement runs.

Fig. 6(a) presents the interaction of a migrated flow using CUBIC with main flows that use other variants and a reconfiguration period of $50\,\mathrm{ms}$. Unlike for two competing TCP CUBIC flows, the bandwidth is not shared fairly. The actual bandwidth distribution depends on the traffic mix.

The first sub-figure of Fig. 6(a) shows the main flow using BBR. It can be seen from the figure that although being migrated, CUBIC consistently receives higher throughput than BBR. The ideal case for all the scenarios would be that the flows share equal percentages of the available bandwidth. However, this plot shows that BBR becomes too passive by trying to be fair against CUBIC even when it is being migrated. Moreover, it can be seen that during the period where congestion is absent, BBR cannot ramp up its sending rate to the link's capacity ($10\,\mathrm{Gbps}$). The peaks are constant around $5\,\mathrm{Gbps}$ link utilization. Also, after the CUBIC flow finishes, BBR continues to utilize only 50% of the available bandwidth. This indicates that the BDP calculation is not updated after the initialization phase, leading to inefficiency even after the competing flow vanishes. The inefficient bandwidth utilization of BBR and its fairness property causes its FCT to be much higher.

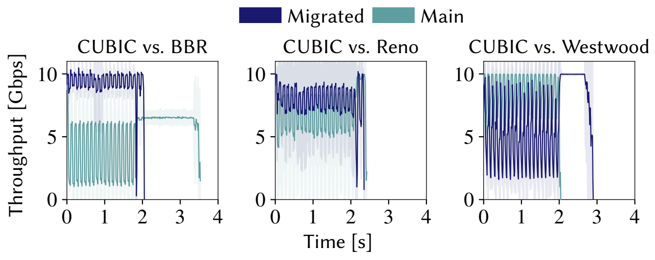
The competition between CUBIC and Reno, presented in the second subfigure of Fig. 6(a), has the fairest allocation among all scenarios. CUBIC and Reno, both loss-based CC schemes, behave similarly. Finally, Westwood (last sub-figure of Fig. 6(a)), a variant designed mainly for wireless, unreliable, and lossy links, dominates the other variants in terms of throughput competition.

Since the CC algorithms of the variants are different from one another, their interaction exhibits different characteristics than the competition of the same variants. An aggressive congestion window growth rate leads to a higher share of the available bandwidth than the other variant. In contrast, a conservative congestion window growth rate may lead the flow to under-utilize the link.

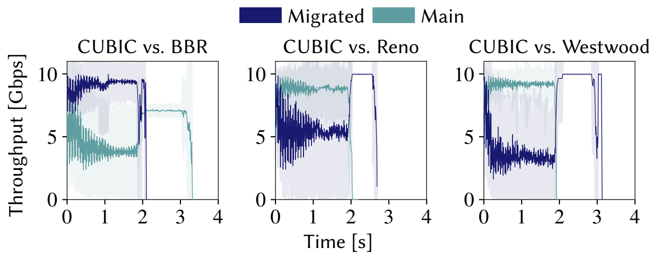
Furthermore, when compared to a smaller reconfiguration period (presented in Fig. 6(b)), the flow behavior is observed to be similar. While the magnitude of the FCT difference is not the same, the overall behavior is similar. This indicates that in addition to the reconfiguration period, the interacting TCP variants also play a role in determining the effect on the FCT.

D. *Analysis of DCTCP*

DCTCP is a CC algorithm that is specifically designed for DCNs. It relies on the ECN field in the IPv4 header. The switches mark a packet as congested, i.e., set the ECN field to *2b11*, if the queue depth exceeds a certain threshold. Depending on the ECN field, DCTCP adjusts its congestion window size. The queue depth threshold is a configurable parameter. For a rule of thumb, the switches mark a packet if the egress queue has more than 17% of BDP [33]. In our setup, the queue depth is read when enqueuing a packet in the egress queue. We vary the queue depth threshold with



(a) Reconfiguration Period: 50ms.



(b) Reconfiguration Period: 10ms.

Fig. 6. Average throughput of one migrated and one static (main) flow. The first TCP variant in each title refers to the migrated flow and the second one indicates the main flow. Moreover, 95% confidence intervals are presented in the background with matching colors. In addition to the reconfiguration period, the mix of interacting TCP variants affects the achieved throughput as well.

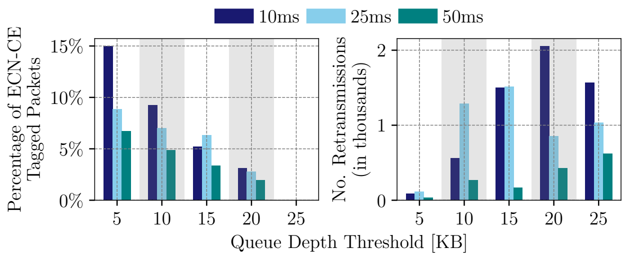


Fig. 7. Analysis of different queue depth thresholds with varying reconfiguration periods. The investigated metrics are the percentage of ECN-CE marked packets and the number of retransmissions. The results are obtained from the competition of two DCTCP flows and averages of 30 distinct measurement runs are reported. Increasing the queue depth threshold leads to less number of tagged packets, hence leading to more packet losses.

respect to reconfiguration periods and determine the best-suiting threshold.

Fig. 7 shows a bar plot comparing the percentage of ECN Congestion Experienced (CE) packets and the number of retransmitted packets for various queue depth thresholds and different reconfiguration periods. In order to fine-tune the threshold, we swipe the threshold from 5 KB to 25 KB with 5 KB increments.

The first subplot of Fig. 7 indicates the percentage of ECN-CE marked packets. When the threshold is set to 5 KB, around 15% of the packets are marked. Intuitively, the number of marked packets decreases when the threshold is increased. For a given threshold, decreasing the reconfiguration period leads to more marked packets, which is in parallel to our observations that decreasing the reconfiguration period leads to more congestion, and hence, more tagged packets.

The second subplot of Fig. 7 shows the number of retransmitted packets for various threshold values and reconfiguration periods. When analyzed together with the percentage of ECN marked packets, there seems to be a negative correlation between the number of marked packets and retransmissions. However, no clear consistent observation for retransmissions can be observed. Upon analyzing various thresholds, we select 5 KB as the ECN marking since it leads to the least number of retransmission and the ratio of marked packets are in line with previous analysis [36].

Fig. 8 visualizes the throughput of two competing flows. Fig. 8(a) analyzes the competition of two DCTCP flows, whereas Fig. 8(b) investigates the competing CUBIC and DCTCP flows with reconfiguration periods of 50 ms and 10 ms. Fig. 8(a) shows that two DCTCP flows can share the available bandwidth during congested periods and they can ramp up to the full link capacity in case of 50 ms reconfiguration period. Unlike the previous observations with TCP CUBIC, DCTCP can ramp up its sending rate when the congestion vanishes even in the case of 10 ms reconfiguration period. As a result, it fairly shares the bandwidth between the main and the migrated flow.

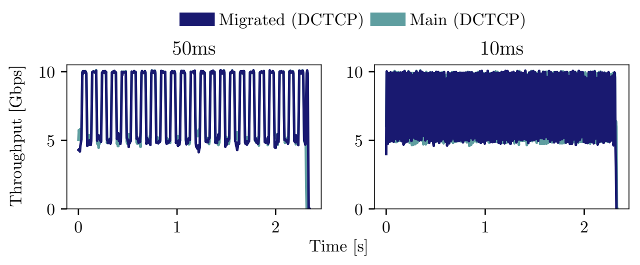
Fig. 8(b) illustrates that when a CUBIC flow is migrated to a static link with DCTCP flow, CUBIC aggressively gains over 90% of the bandwidth and causes the DCTCP flow to suffer in terms of achieved bandwidth. Since more packets are ECN marked, DCTCP fails to ramp up its sending rate upon the detection of congestion. Our observations confirm that in controlled RDCN environments, DCTCP CC might be the best choice in terms of fair allocation of bandwidth, however, in uncontrolled environments, DCTCP performance is subject to degradation.

E. Reconfiguration Downtime

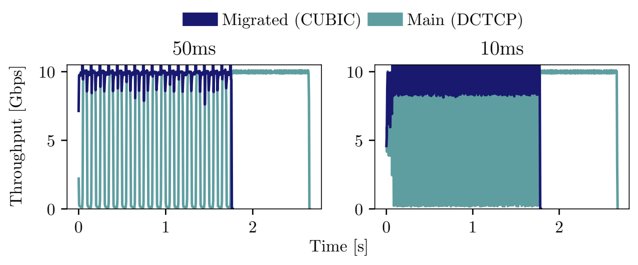
The scenarios so far considered an instantaneous reconfiguration of the circuits and re-routing of the flows. However, OCS generally incur a reconfiguration cost [22], [52], [53]. This means that the circuit is destroyed during a reconfiguration process and the new circuit and link require some time to be set up. Packets arriving during the reconfiguration process are, in the worst case, dropped. Since the `Switch` does not drop any packets during reconfiguration, the reconfiguration downtime is emulated by inserting a rule to drop packets during the downtime.

For reconfiguration downtime analysis, two flows with 2 GB sizes using TCP CUBIC are considered. One of them is continuously migrated while the other one is served statically. The reconfiguration period is varied across scenarios and a standardized 10% duty cycle is investigated, e.g. 50 ms *day time* and 5 ms *night time*.

Fig. 9 shows the average throughput of migrated and main CUBIC flows under different reconfiguration settings. Commercially available OCS have reconfiguration downtimes in the order of milliseconds. Accordingly, we evaluate values



(a) DCTCP vs. DCTCP.



(b) CUBIC vs. DCTCP.

Fig. 8. Average throughput of one migrated and one static (main) TCP flow. Competing flows are indicated in the plot legends, whereas the titles show the reconfiguration period. While two DCTCP flows can fair share the bandwidth even in the case of 10 ms reconfiguration period, CUBIC gains an aggressive share of the link rate when competing with DCTCP.

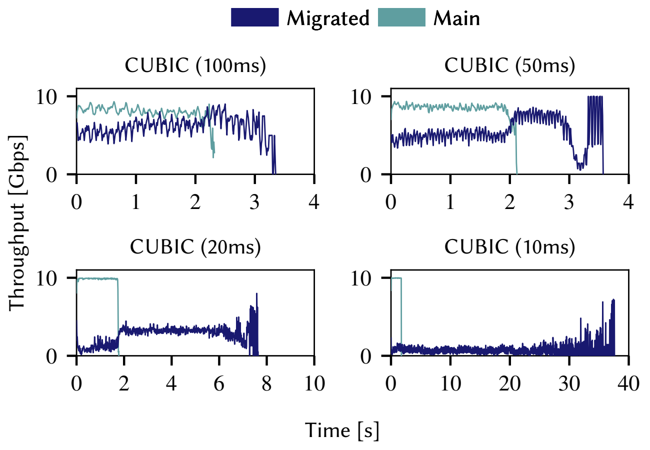


Fig. 9. Average throughput of TCP CUBIC flows with varying reconfiguration downtimes. Reconfiguration periods are reported in the title and the duty cycle is set to 90% in all scenarios. For instance, 10 ms reconfiguration downtime is created in case of a 100 ms reconfiguration period. Dropping packets during downtime increases the FCT of the migrated flow significantly.

of 1, 2, 5 and 10 ms. The upper left sub-figure illustrates the scenario where the circuit is provisioned for 90 ms and the reconfiguration incurs a downtime of 10 ms. Unlike the zero downtime case, it can be observed that even at 100 ms reconfiguration period migrated flow achieves consistently lower throughput, and hence it translates into higher FCT for the migrated flow. The following sub-figures further display the low link utilization of the migrated flow.

The last sub-figure presents the most outstanding behavior where the main flow achieves over 95% of the bandwidth during congestion. After dropping packets, 9 ms does not allow enough time for TCP CUBIC to ramp up its sending rate. Therefore, the migrated flow is unable to create congestion that will reduce the throughput of the main flow. Moreover, it can also be seen that even after the congestion vanishes, the migrated flow suffers from migrations due to the reconfiguration downtime. In this case, reconfiguration downtime not only affects the migrated flow's FCT negatively, but it also affects the main flow's FCT positively.

TABLE II
AVERAGE THROUGHPUT OF MIGRATED AND MAIN FLOWS.

Reconfiguration Period	Migrated Flow	Main Flow
50 ms	1.43 Gbps	1.24 Gbps
25 ms	1.39 Gbps	1.28 Gbps
10 ms	1.31 Gbps	1.27 Gbps

F. Small Flows and Random Arrivals

Empirical studies of DCNs show that over 80% of the flows are less than 10 KB and inter-arrival time (IAT) of flows are in the range of microseconds [54]. To improve scalability, RDCN designs such as [3], [4], [7] target rack-to-rack circuits. In such cases, the total rack-to-rack demand is considered and hence, groups of multiple flows of different sizes become subject to reconfigurations. In order to emulate this situation, we consider a scenario with flows in the range of 2^n MB for n in [0, 8]. The flows are separated into two groups: *migrated* and *main*. The flow volumes are randomly selected from the available volumes such that the total volume per group is 1 GB. The maximum allowed number of flows per group is set to 20. Flows arrive in pairs (one migrated and one main) with periods of 10 ms. All flows use TCP CUBIC.

Table II shows the average throughput of flows in each group under different reconfiguration periods. While the average throughput of the main flows varies less than 2% across scenarios, the migrated flows' throughput decreases as the reconfiguration period is decreased. Overall, this hints at the inefficient bandwidth utilization of flows when the reconfiguration period is decreased.

G. UDP Background Traffic

Although, TCP is considered as *de facto* standard transport protocol, there are also applications that use UDP, e.g., loss-tolerating or latency-sensitive applications such as industrial networks. The interaction of multiple TCP flows has been analyzed in the previous sections. The continuous feedback mechanism of TCP enables smoother interaction between the variants. The variants are designed to consider fairness between competing flows and guarantee minimal latency with maximum throughput. However, UDP is a much more conven-

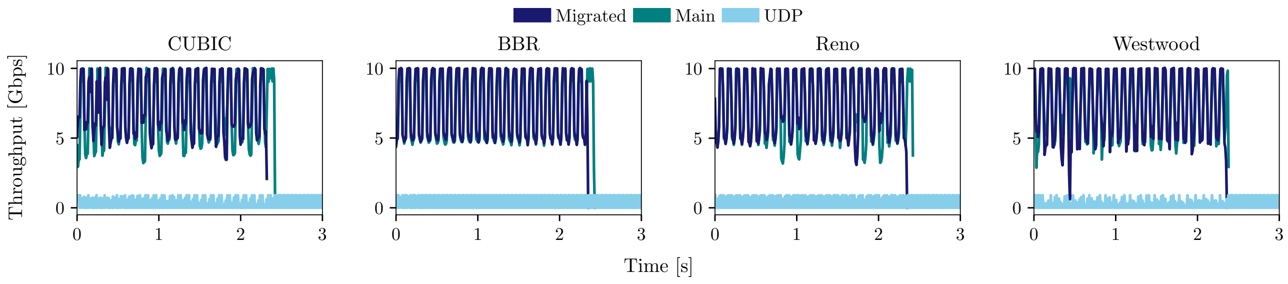


Fig. 10. Average throughput of one migrated and one static (main) TCP flow where the main link is exposed to bursty UDP traffic. Competing flows use the same TCP variants. Various TCP variants are plotted in different subplots. The reconfiguration period is set to 50 ms. UDP bursts are sent with an inter-arrival time of 25 ms. The inclusion of bursty UDP traffic hinders TCP's ability to adjust its sending rate. TCP BBR outperforms other variants in terms of adjusting the rate to the available capacity.

tional protocol that does not control successful transmission. UDP transfers packets without any control mechanism and it does not account for other traffic on the same link. Therefore, the interaction of TCP and UDP might differ from TCP to TCP interaction.

There are two major scenarios for the interaction of a UDP and a TCP stream: The first scenario considers a continuous stream of UDP packets. Packets are sent at a constant rate, i.e., there are no bursts of packets. The continuous stream occupies a constant amount of the link's capacity given by the target rate of the UDP stream, and TCP tries to adjust its sending rate up to the remaining capacity. Since UDP does not consider successful packet transmission, this is the expected behavior. Therefore, a more interesting analysis considers the second scenario: bursty UDP background traffic.

The second scenario is created by sending UDP traffic in bursts. The change in the available capacity in the link, which is caused by the bursts, means that TCP needs to adjust its sending rate to prevent packet loss. This scenario, including background UDP traffic is created via the addition of another server. The third server is configured to send 128 packets, each of which is 9000 bytes, and the burst IAT is set to 25 ms. UDP packet generation is achieved via MoonGen [55] packet generator. The background traffic is introduced only in the bottleneck link. TCP flow sizes are set to 2 GB, and the investigated reconfiguration period is 50 ms without any downtime. Therefore, the duration of traffic bursts equals half of the migration period. This means that both the migrated and the main TCP flow face a burst of UDP traffic for 25 ms during congestion. Additionally, the main flow faces a burst of UDP traffic for 25 ms in the absence of TCP congestion (when the migrated and the main flow are served on separate links).

Figure 10 shows the mean throughput of the TCP variants when background traffic interference is present. Averages of 30 distinct measurements are plotted. The figure shows that the link capacity occupied by UDP traffic forms around 10% of the link capacity during burst periods. The existence of background traffic in the congested links affects the TCP variants' ability to adjust their sending rate. Since the background traffic's IAT is half of the reconfiguration period, the main flow is expected to be affected more than the migrated flow from the bursts. Hence, the theoretical FCT of the main flow is expected to be higher than for the migrated flow. The throughput plots for all the variants are in parallel with the theoretical expectations. It can be seen that the main TCP flows finish later than the main flow for all the variants.

For CUBIC and Reno, the inconsistency between the valleys and the peaks indicates that packet losses trigger the loss-based CC algorithm. These flows demonstrate an irregular pattern as a result of the background traffic. The capacity limitations in the bottleneck link favor the migrated TCP flow by allocating it more bandwidth in general. For BBR, the jigsaw pattern is much more evident than for the other variants. This is possible, because of the available spaces in the buffers of the switch, as BBR operates at the sending rate where the queues in the path are empty. Hence, UDP packets do not create a packet loss, so the sending rate is not affected. Westwood achieves similar FCT for the migrated and the main flow, indicating that it allocates the link capacity most fairly among other variants in case of a bursty UDP background traffic.

In conclusion, the effect on CUBIC and Reno main flow is much greater, while Westwood can allocate bandwidth fairly across variants with the background traffic.

H. Competition of More than Two Flows

In a more realistic DCN environment, many flows coexist alongside each other. Therefore, the analysis of the number of flows in addition to the TCP variant analysis is beneficial to understand if the observed patterns with two flows can generalize to bigger scenarios.

Fig. 11 shows the throughput plots arising from the competition of more than two flows. Specifically, we investigate scenarios with 8 and 10 flows. The flow sizes are adjusted, such that the total transmitted volume equals the analysis in Sec. V-B, i.e. 4 GB. The average throughput of each migrated and main flow can be seen with transparent color-coded lines. At 50 ms reconfiguration period, CUBIC flows can adjust their sending rate up to the link capacity both in the presence and in the absence of congestion. However, at 10 ms reconfiguration period, the average achieved throughput of the migrated flow is less. Finally, similar to the previous observations, 50 ms

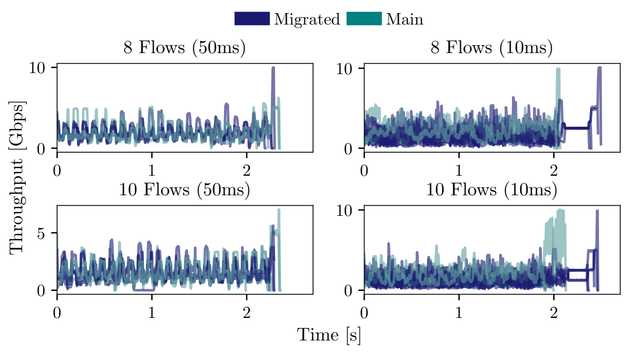


Fig. 11. Average throughput of migrated and main TCP CUBIC flows. The titles report the total number of flows and reconfiguration periods. We consider cases where migrated and main flows equally contribute to the total number of flows, e.g. there are 4 migrated and 4 main flows in case of 8 flows scenario. Each flow's throughput is plotted separately with transparent colors. Previously observed phenomena of migrated flow's increased FCT generalize to scenarios with more than 2 flows.

reconfiguration period does not affect the migrated CUBIC flow's FCT, whereas 10 ms hinders the migrated flow's ability to adjust its sending rate during congestion. Therefore, we conclude that our analysis of two flows can generalize to an increased number of flows as well.

I. Interaction with QUIC

We investigate TCP performance, when there is also QUIC traffic in the mix. For measuring a TCP flow's performance subject to frequent migrations, we migrate TCP traffic while keeping the QUIC flow fixed as the main flow. Among many existing QUIC implementations, we select LSQUIC, and all the reported results are generated with this implementation. In particular, we use LSQUIC v3.3.1 together with QUIC version h3-29.

Our initial measurements have shown that QUIC is not able to saturate a 10 Gbps link. Therefore, for analyzing the interaction of QUIC and TCP, the bottleneck link capacities are adjusted to 1 Gbps. Fig. 12 shows the measurement setup for these measurements. Similar to the first testbed (Fig. 2), two servers are present for creating and collecting the traffic. iPerf version 2.0.10 [49] is used for generating TCP flows, whereas QUIC traffic is generated via the example client and server Docker container provided by the LSQUIC developers [39].

In addition to the `Tofino Switch`, the setup includes a `Dell S3048-ON OpenFlow-capable Switch`, which features 4x10G ports and 48x1G ports. The figure illustrates 10G ports with red and 1G ports with orange color. The goal of the `Dell Switch` in this setup is to physically rate limit the ports to 1 Gbps, as the `Tofino Switch` does not have ports with 1 Gbps capacity. Although rate limiting can be implemented artificially via token bucket filtering (as done in downlink port to `Server 2`), in the case of flow migrations, our benchmarks have shown that the measurement results show a slight mismatch. Therefore, our setup includes a second switch for pure rate-limiting purposes.

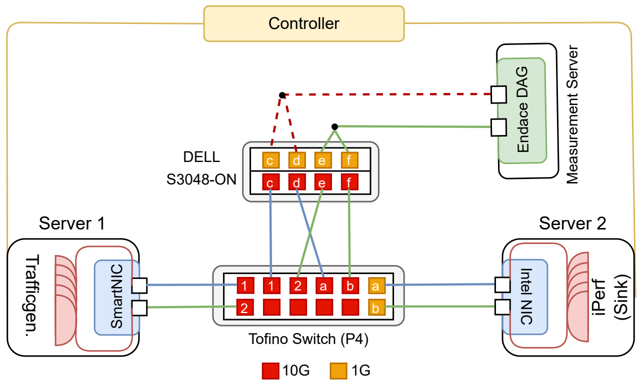


Fig. 12. Testbed overview for measuring TCP's interaction with QUIC.

Finally, all the traffic is tapped via a copper tap, collected, and timestamped in the `Measurement Server` via Endace DAG 4x1G measurement card for analysis, and the `Controller` is existent to orchestrate the experiment.

The procedure for the measurement includes the manual installation of forwarding rules to both of the switches. The manual forwarding rules are shown in the figure with letter and number matchings. For example, the `Dell Switch` is statically configured to forward packets incoming from the red interface 'c' to the orange interface 'c'. Similar to the previous migration approach, a rule update in the `Tofino Switch` is triggered with various reconfiguration periods. The rule update takes place in numbered ports, i.e., the update triggers to forward incoming packets from Port 2 to Port 1. This is repeated periodically with varying reconfiguration periods. This rule update causes congestion in the copper loopback links, which are shown in red, on the `Dell Switch` indirectly due to the manually and statically installed forwarding rules. Similar to the previous approach, the packets sent by `Server 2` are routed around the `Tofino Switch`, meaning that they do not trace the loopback links.

Fig. 13 presents the measurement results in case of 50 ms reconfiguration period. QUIC flow's CC algorithm is set to CUBIC and it is served constant on the static link as the main flow, whereas TCP flow is being migrated. Different CC algorithms of the TCP flow are plotted in different subplots. Each plot shows 30 distinct measurement runs, and the runs are plotted individually with transparency, i.e., there is one line per run.

On the contrary to the observations of the competition of two TCP flows, the upper left subplot shows that even at 50 ms reconfiguration period, the migrated flow's FCT is much higher than the main QUIC flow. During the congested periods, it is seen that the migrated flow achieves much less than the fair-share bandwidth of 500 Mbps, and this translates into higher FCT. This observation is similar between TCP CUBIC and TCP Reno.

TCP BBR shows a much more stable pattern, where the jigsaw pattern is more evident, and in this case, the migrated flow can achieve slightly higher bandwidth in comparison

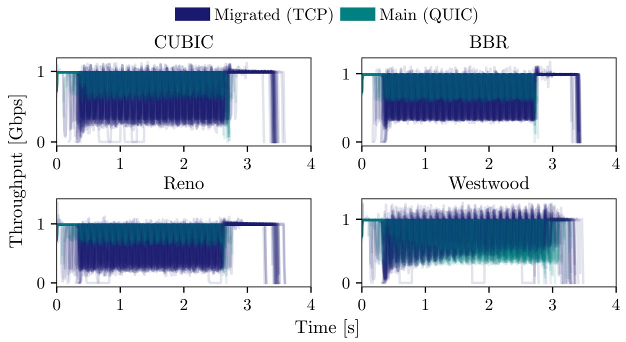


Fig. 13. Throughput of migrated TCP flows with varying CC algorithms against a main QUIC flow with CUBIC CC. 30 runs are plotted individually with transparent colors. The flow sizes are 128 MB. The reconfiguration period is set to 50 ms. TCP CUBIC, Reno, and BBR suffer from competition with QUIC significantly at 50 ms reconfiguration period, whereas TCP Westwood manages to compete with QUIC.

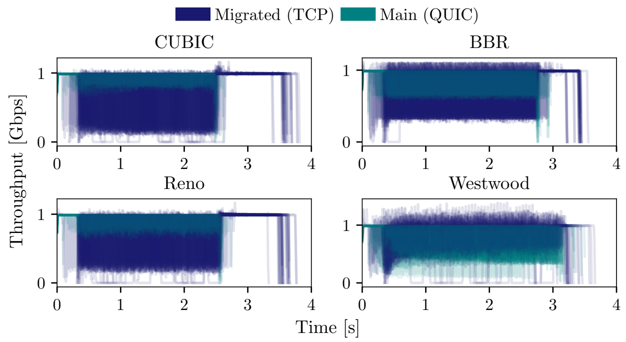


Fig. 14. Throughput of migrated TCP flows with varying CC algorithms against a main QUIC flow with CUBIC CC. 30 runs are plotted individually with transparent colors. The flow sizes are 128 MB. The reconfiguration period is set to 25 ms. The magnitude of the impact of reconfigurations is observed to be higher than 50 ms.

to TCP CUBIC and Reno. While the competition of a TCP CUBIC and TCP BBR flow resulted in TCP CUBIC gaining dominance over the link (Fig. 6(a)), it can be seen that a migrated TCP BBR flow achieves a smaller FCT when competing with a QUIC than TCP CUBIC or Reno. Finally, TCP Westwood, which was observed to be the most dominant CC algorithm among the investigated ones, manages to compete with QUIC and achieves much better FCT than the other TCP variants.

Fig. 14 presents the measurement results in case of 25 ms reconfiguration period. The results show that the observed patterns are similar to the case of 50 ms reconfiguration period (Fig. 13), however, the magnitude of the effect is higher. For TCP CUBIC and Reno, it can be seen that the migrated TCP flow is suffering significantly during congestion (even more than in the case of 50 ms). For BBR and Westwood the observed measurement results do not differ significantly.

Overall, the findings indicate that, in addition to the CC algorithm, its implementation, e.g. Linux Kernel implementation and LSQUIC implementation, also plays an important role. The previous findings reported from the competition of two TCP CUBIC flows do not hold when a TCP CUBIC flow is competing with a QUIC CUBIC flow.

VI. Modeling

The analysis of the measurements in Sec. V showed that, in general, the migrated flow's FCT increases when the reconfiguration period decreases. This section extends the evaluations to build up an ML model to predict the effect of the reconfiguration period on the FCT prolongation of the migrated flow. Additionally, it presents a model to predict the TCP variant given the traffic patterns. The availability of such models will enable the network operators to better understand the traffic mix in their infrastructures, and hence it will enable them to fine-tune their network management algorithms accordingly.

A. Prediction of FCT Prolongation

The proposed model determines the factors of influence on the FCT prolongation and formulates a prediction. The insights gained from the predictions of the model will serve as a tool to estimate the cost of reconfiguration scenarios on the FCT and enable the network operators to tune link-scheduling algorithms according to the traffic mix and reconfiguration scenario.

The FCT prolongation is defined as the time difference between the migrated flow's FCT and the main flow's FCT.[1] The target variable to predict in this model is the FCT prolongation. The input variables for the model are the migrated TCP variant, the main TCP variant, *day time*, and the *night time*. For the dataset, TCP CUBIC, Reno, and Westwood measurements with two flows with 2 GB volume are used.

For better interpretability, a random forest model is used for training [56]. The training dataset consists of the FCTs measured from all of the scenarios introduced in Sec. IV.

The random forest model is optimized via a grid search of over 288 combinations of model parameters and evaluated using 10-fold cross-validation. The grid search returns the best parameters as 500 estimators, bootstrapping enabled, maximum tree depth of 100 with a minimum of two samples per leaf and two samples to split at each internal node.

Overall, the model predicts the FCT prolongation with a Mean Squared Error (MSE) of $4.4\,\mathrm{s}^2$. Moreover, the R^2 of the model is 0.88, which indicates that 88% of the variance in the results can be explained via this model. The unexplained variance is mainly related to the scenario with 10 ms reconfiguration period and 1 ms *night time*. Since the actual FCTs vary significantly in these scenarios, many outliers exist and the model performs worse.

Fig. 15 presents the scatter plot of the predictions of the best-performing model and the actual FCT prolongations. The

[1] Since the difference between the migrated flow's and the main flow's FCT is always positive in our case, there is no difference to using the absolute difference.

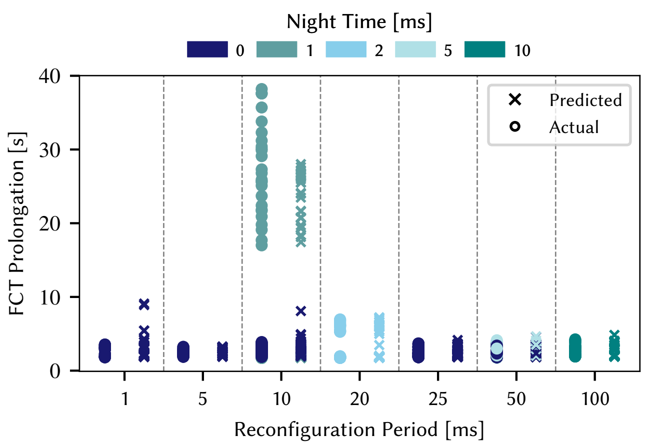


Fig. 15. Scatter plot of the predicted and actual FCT prolongations. The actual and predicted FCTs are plotted with different markers side-to-side with each other. The prediction is achieved with an MSE of $4.4\,\mathrm{s}^2$.

prolongation values range from no prolongation to as high as $40\,\mathrm{s}$.

The prolongation effect is significant when the *night time* is $1\,\mathrm{ms}$ and the *day time* is $9\,\mathrm{ms}$. Reconfiguration downtime has an important effect on FCT prolongation. However, the primary determinant is the *day time* in which a flow is served via the provisioned link. This effect is also shown clearly in the figure. At $2\,\mathrm{ms}$ *night time* with $20\,\mathrm{ms}$ *day time*, the FCT prolongation effect is much less. The model predicts the effect of the reconfiguration period on the FCT prolongation with considerable performance.

B. *TCP Variant Classification*

The model presented for FCT prolongation prediction relied on the knowledge of a TCP variant. To relax this assumption, we provide a model to classify a TCP variant from a traffic trace. Additionally, such a model will enable the network operators to gain insights into the traffic mix in their DCNs. The classification of TCP variants in wireless and static DCN architectures has been analyzed to some extent in the literature [57].

The input variables to our model are the congestion window size and RTT of the flow for $500\,\mathrm{ms}$ with respect to time. Hence, the model is able to make a decision within $500\,\mathrm{ms}$ in a live trace. The target variable is the TCP variant, an integer-encoded variable in the training dataset. The goal of the ML algorithm is to predict the TCP variant given the congestion window and RTT information which can be extracted from traffic traces.

The classification of the TCP variant relies on an LSTM model. The choice for LSTMs is based on the findings in [57] which establishes that LSTMs are a better choice than random forest for the classification of the TCP variant given the traffic traces. The congestion window size and RTT values are expressed in kilobytes and microseconds, respectively. Therefore, the value range they can take is very diverse. As suggested by [57], to overcome the effect of high variance in the input layer, an embedding layer before the LSTM layer is inserted into the neural network structure. The embedding layer's role is to reduce the dimension of the input space by sampling the input. The next layer consists of an LSTM, and the final layer is composed of a dense layer.

The traces from all measurements with two flows, excluding DCTCP are used for modeling. Since the target variable is a categorical variable with more than two classes, the neural network is trained by minimizing the categorical cross-entropy loss function. The hyperparameter optimization of the neural network model is done by varying the embedding layer and LSTM layer dimensions. Overall, the best-performing model is trained for 250 epochs, and TCP variant classification is achieved with 93.3% accuracy.

TABLE III
CONFUSION MATRIX OF TCP VARIANT CLASSIFICATION.

		Predicted			
		CUBIC	BBR	Reno	Westwood
Actual	CUBIC	100	0	17	4
	BBR	0	107	0	0
	Reno	3	0	101	0
	Westwood	1	0	4	95

Table III presents the confusion matrix of the model. Each row of the matrix represents the instances in an actual class, while each column represents the instances in a predicted class. The table shows that 29 variants out of 430 are classified incorrectly. Classification of CUBIC has the most errors. The model is prone to predict a CUBIC flow as Reno since the behavior of CUBIC and Reno are very similar. It can be seen that BBR has 100% classification accuracy. BBR is the only variant in this mix that does not use the buffers of the switch, and hence it demonstrates an entirely different congestion window size and RTT behavior. This translates into better classification performance. Finally, Reno and Westwood are easier to detect than CUBIC, with better classification accuracy.

In conclusion, the detection of TCP variants can shed light on the traffic mix in a DCN. Overall, our models to predict the FCT prolongation and classify TCP variants provide a proof-of-concept methodology that network managers can employ to better understand the TCP traffic mix in their DCNs and fine-tune their architectures accordingly.

VII. CONCLUSION

Achieving the full performance of RDCNs depends on TCP's ability to utilize the temporary high bandwidth links efficiently. Sub-millisecond reconfiguration periods, emerging transport protocols, such as QUIC, pose a threat to the performance of state-of-the-art TCP implementations. The short duration of the high-bandwidth link provisioning periods hinders TCP's ability to ramp up its sending rate and leads to lower link utilization with higher FCTs.

This paper analyzed the most popular TCP variants' behavior and interaction with QUIC and UDP under different reconfiguration scenarios. The findings indicate that reconfiguring the switch more frequently leads to a significant FCT increase in the migrated flow and the magnitude of this effect depends on the traffic mix. Inclusion of UDP and QUIC in the traffic mix hinders TCP's ability to adjust its sending rate in the presence of reconfigurations. Finally, the proposed ML model predicts the FCT prolongation and classifies TCP variants in the traffic mix. It will benefit the networking society to gain insights into the relationship between the reconfiguration scenario and FCT.

A generalized model that will serve as a foundation for predicting FCT prolongation with the incorporation of more input DCN parameters, e.g., switch buffer sizes and cable lengths is a direction for future work. The findings reported in this paper outline the benchmarks and pave the way for future work to design reconfiguration scenarios according to the analyzed traffic mix and the proposed ML model.

Acknowledgments

We acknowledge the financial support by the Federal Ministry of Education and Research of Germany (BMBF) in the program of "Souverän. Digital. Vernetzt." joint project 6G-life, project identification number 16KISK002 and Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 438892507.

References

[1] E. Jonas, J. Schleier-Smith, V. Sreekanti, C. Tsai, A. Khandelwal, Q. Pu, V. Shankar, J. Carreira, K. Krauth, N. J. Yadwadkar, J. E. Gonzalez, R. A. Popa, I. Stoica, and D. A. Patterson, "Cloud programming simplified: A berkeley view on serverless computing," *CoRR*, vol. abs/1902.03383, 2019. [Online]. Available: http://arxiv.org/abs/1902.03383

[2] V. Ziegler, H. Viswanathan, H. Flinck, M. Hoffmann, V. Räisänen, and K. Hätönen, "6g architecture to connect the worlds," *IEEE Access*, vol. 8, pp. 173 508–173 520, 2020.

[3] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A hybrid electrical/optical switch architecture for modular data centers," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, p. 339–350, Aug. 2010.

[4] C. Griner, J. Zerwas, A. Blenk, M. Ghobadi, S. Schmid, and C. Avin, "Cerberus: The power of choices in datacenter topology design-a throughput perspective," *POMACS*, vol. 5, no. 3, pp. 1–33, 2021.

[5] H. Liu, M. K. Mukerjee, C. Li, N. Feltman, G. Papen, S. Savage, S. Seshan, G. M. Voelker, D. G. Andersen, M. Kaminsky *et al.*, "Scheduling techniques for hybrid circuit/packet networks," in *Proc. 11th ACM CoNEXT*, 2015, pp. 1–13.

[6] M. Ghobadi, R. Mahajan, A. Phanishayee, N. Devanur, J. Kulkarni, G. Ranade, P.-A. Blanche, H. Rastegarfar, M. Glick, and D. Kilper, "Projector: Agile reconfigurable data center interconnect," in *Proc. ACM SIGCOMM 2016*. New York, NY, USA: Association for Computing Machinery, 2016, p. 216–229.

[7] W. M. Mellette, R. Das, Y. Guo, R. McGuinness, A. C. Snoeren, and G. Porter, "Expanding across time to deliver bandwidth efficiency and low latency," in *Proc. 17th USENIX NSDI*, 2020, pp. 1–18.

[8] K.-T. Foerster, M. Ghobadi, and S. Schmid, "Characterizing the algorithmic complexity of reconfigurable data center architectures," in *Proc. of the 2018 Symposium on Architectures for Networking and Communications Systems*, ser. ANCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 89–96.

[9] M. K. Mukerjee, C. Canel, W. Wang, D. Kim, S. Seshan, and A. C. Snoeren, "Adapting TCP for reconfigurable datacenter networks," in *USENIX NSDI*. Santa Clara, CA: USENIX Association, Feb. 2020, pp. 651–666. [Online]. Available: https://www.usenix.org/conference/nsdi20/presentation/mukerjee

[10] M. T. Naing, T. T. Khaing, and A. H. Maw, "Evaluation of tcp and udp traffic over software-defined networking," in *2019 International Conference on Advanced Information Technologies (ICAIT)*, 2019, pp. 7–12.

[11] K. Miyazawa, S. Yamaguchi, and A. Kobayashi, "Performance evaluation of tcp bbr and cubic tcp in smart devices downloading on wi-fi," in *2020 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-Taiwan)*, 2020, pp. 1–2.

[12] K. Ratna Pavani and N. Sreenath, "Performance evaluation of tcp-reno, tcp-newreno and tcp-westwood on burstification in an obs network," in *ADCOM*, 2012, pp. 19–24.

[13] K. Sasaki, M. Hanai, K. Miyazawa, A. Kobayashi, N. Oda, and S. Yamaguchi, "Tcp fairness among modern tcp congestion control algorithms including tcp bbr," in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*, 2018, pp. 1–4.

[14] S. M. Irteza, A. Ahmed, S. Farrukh, B. N. Memon, and I. A. Qazi, "On the coexistence of transport protocols in data centers," in *Proc. IEEE ICC*, 2014, pp. 3203–3208.

[15] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network," *ACM SIGCOMM computer communication review*, vol. 45, no. 4, pp. 183–197, 2015.

[16] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, 2011, pp. 242–253.

[17] B. Cronkite-Ratcliff, A. Bergman, S. Vargaftik, M. Ravi, N. McKeown, I. Abraham, and I. Keslassy, "Virtualized congestion control," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 230–243. [Online]. Available: https://doi.org/10.1145/2934872.2934889

[18] K. He, E. Rozner, K. Agarwal, Y. J. Gu, W. Felter, J. Carter, and A. Akella, "Ac/dc tcp: Virtual congestion control enforcement for datacenter networks," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 244–257. [Online]. Available: https://doi.org/10.1145/2934872.2934903

[19] K. Nepomuceno, I. N. d. Oliveira, R. R. Aschoff, D. Bezerra, M. S. Ito, W. Melo, D. Sadok, and G. Szabó, "Quic and tcp: A performance evaluation," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, 2018, pp. 00 045–00 051.

[20] Y. Yu, M. Xu, and Y. Yang, "When quic meets tcp: An experimental study," in *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*, 2017, pp. 1–8.

[21] K. Aykurt, J. Zerwas, A. Blenk, and W. Kellerer, "On the performance of tcp in reconfigurable data center networks," in *2022 18th International Conference on Network and Service Management (CNSM)*, 2022, pp. 127–135.

[22] W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren, and G. Porter, "Rotornet: A scalable, low-complexity, optical datacenter network," in *Proc. ACM SIGCOMM*, 2017, p. 267–280.

[23] H. Liu, F. Lu, A. Forencich, R. Kapoor, M. Tewari, G. M. Voelker, G. Papen, A. C. Snoeren, and G. Porter, "Circuit switching under the radar with reactor," in *Proc. 11th USENIX NSDI*, 2014, pp. 1–15.

[24] H. Liu, M. K. Mukerjee, C. Li, N. Feltman, G. Papen, S. Savage, S. Seshan, G. M. Voelker, D. G. Andersen, M. Kaminsky *et al.*, "Scheduling techniques for hybrid circuit/packet networks," in *Proc. 11th ACM ConNEXT*, 2015, pp. 1–13.

[25] J. Postel, "Transmission control protocol," Sep 1981. [Online]. Available: https://rfc-editor.org/rfc/rfc793.txt

[26] M. Polese, F. Chiariotti, E. Bonetto, F. Rigotto, A. Zanella, and M. Zorzi, "A survey on recent advances in transport layer protocols," *CoRR*, vol. abs/1810.03884, 2018. [Online]. Available: http://arxiv.org/abs/1810.03884

[27] 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.

[28] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM computer communication review*, vol. 18, no. 4, pp. 314–329, 1988.

[29] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "Tcp vegas: New techniques for congestion detection and avoidance," in *Proceedings of the conference on Communications architectures, protocols and applications*, 1994, pp. 24–35.

[30] R. Mittal, V. T. Lam, N. Dukkipati, E. Blem, H. Wassel, M. Ghobadi, A. Vahdat, Y. Wang, D. Wetherall, and D. Zats, "Timely: Rtt-based congestion control for the datacenter," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 537–550, 2015.
[31] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "Tcp westwood: Bandwidth estimation for enhanced transport over wireless links," in *Proceedings of the 7th annual international conference on Mobile computing and networking*, 2001, pp. 287–297.
[32] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: congestion-based congestion control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.
[33] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," in *Proc. ACM SIGCOMM*, 2010, pp. 63–74.
[34] V. Addanki, O. Michel, and S. Schmid, "{PowerTCP}: Pushing the performance limits of datacenter networks," in *Proc. USENIX NSDI*, 2022, pp. 51–70.
[35] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, 2002, pp. 89–102.
[36] M. Alizadeh, A. Javanmard, and B. Prabhakar, "Analysis of dctcp: stability, convergence, and fairness," *ACM SIGMETRICS Performance Evaluation Review*, vol. 39, no. 1, pp. 73–84, 2011.
[37] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9000
[38] J. Rüth, I. Poese, C. Dietzel, and O. Hohlfeld, "A first look at QUIC in the wild," in *Passive and Active Measurement*. Springer International Publishing, 2018, pp. 255–268.
[39] Litespeedtech, "Litespeedtech/lsquic: Litespeed quic and http/3 library." [Online]. Available: https://github.com/litespeedtech/lsquic
[40] T. Das and K. M. Sivalingam, "Tcp improvements for data center networks," in *2013 Fifth International Conference on Communication Systems and Networks (COMSNETS)*, 2013, pp. 1–10.
[41] A. Jain, A. Pruthi, R. Thakur, and M. Bhatia, "Tcp analysis over wireless mobile ad hoc networks," in *2002 IEEE International Conference on Personal Wireless Communications*. IEEE, 2002, pp. 95–99.
[42] K. G. Tsiknas, P. I. Aidinidis, and K. E. Zoiros, "Performance evaluation of transport protocols in cloud data center networks," *Photonic Network Communications*, 2021.
[43] A. Ganji, A. Singh, and M. Shahzad, "Characterizing the impact of tcp coexistence in data center networks," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, 2020, pp. 388–398.
[44] J. Bennett, C. Partridge, and N. Shectman, "Packet reordering is not pathological network behavior," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, pp. 789–798, 1999.
[45] R. Cârpa, M. D. de AssunçÃo, O. Glück, L. LefÈvre, and J.-C. Mignot, "Evaluating the impact of sdn-induced frequent route changes on tcp flows," in *Proc. IEEE CNSM*, 2017, pp. 1–9.
[46] J. Zhang, K. Gao, Y. R. Yang, and J. Bi, "Prophet: Toward fast, error-tolerant model-based throughput prediction for reactive flows in dc networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 6, pp. 2475–2488, 2020.
[47] D. H. Hagos, P. E. Engelstad, A. Yazidi, and Ø. Kure, "General tcp state inference model from passive measurements using machine learning techniques," *IEEE Access*, vol. 6, pp. 28 372–28 387, 2018.
[48] V. Arun, M. T. Arashloo, A. Saeed, M. Alizadeh, and H. Balakrishnan, "Toward formally verifying congestion control behavior," in *Proc. ACM SIGCOMM*, ser. SIGCOMM '21, 2021, p. 1–16.
[49] V. GUEANT, "Iperf - the ultimate speed test tool for tcp, udp and sctptest the limits of your network + internet neutrality test." [Online]. Available: https://iperf.fr/
[50] "Netronome agilio cx isa-4000-10-2-2: 2x 10g sfp+ smartnic." [Online]. Available: https://stordirect.com/shop/adapter-cards/network-interface-cards/netronome-agilio-cx-isa-4000-10-2-2-2x-10g-sfp-smartnic/
[51] "Intel® ethernet network adapter x710 product specifications." [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/series/189530/intel-ethernet-network-adapter-x710.html
[52] J. Zerwas, W. Kellerer, and A. Blenk, "What you need to know about optical circuit reconfigurations in datacenter networks," in *2021 33th International Teletraffic Congress (ITC-33)*, 2021, pp. 1–9.
[53] K.-T. Foerster and S. Schmid, "Survey of reconfigurable data center networks: Enablers, algorithms, complexity," *ACM SIGACT News*, vol. 50, no. 2, pp. 62–79, 2019.
[54] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *ACM SIGCOMM IMC*, 2010, pp. 267–280.
[55] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, "MoonGen: A Scriptable High-Speed Packet Generator," in *Internet Measurement Conference 2015 (IMC'15)*, Tokyo, Japan, Oct. 2015.
[56] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, "Machine learning interpretability: A survey on methods and metrics," *Electronics*, vol. 8, no. 8, 2019.
[57] A. Mishra, X. Sun, A. Jain, S. Pande, R. Joshi, and B. Leong, "The great internet tcp congestion control census," *POMACS*, vol. 3, no. 3, pp. 1–24, 2019.

Kaan Aykurt received his M.Sc. degree in Communications Engineering from the Technical University of Munich (TUM), Germany, in 2022. He joined the Chair of Communication Networks at the TUM as a research and teaching associate in May 2022. His research is focused on data center networks, multi-domain autonomous network management, and applications of machine learning in communication networks.

Johannes Zerwas received his M.Sc. degree in Electrical Engineering and Information Technology from the Technical University of Munich (TUM), Germany, in 2018. He joined the Chair of Communication Networks at the TUM as a research and teaching associate in February 2018. His research is focused on reconfigurable network topologies for data center and wide area networks, and data-driven networking algorithms.

Andreas Blenk is a Research Scientist at Siemens AG, where he has been a part of the Industrial Networks & Wireless group (T CED INW-DE) since April 2022. In his current role, he is focused on the automation, measurement, and validation of industrial networks, and is involved in defining and writing blueprints for the Siemens AG networking portfolio. Prior to joining Siemens AG, Andreas was a member of the Chair of Communication Networks, led by Prof. Wolfgang Kellerer, at the Technische Universität München (TUM) beginning in June 2012. He went on to receive his Doktor-Ingenieur (Dr.-Ing.) degree from TUM in May 2018, achieving the highest distinction of summa cum laude. From 2018 to 2022, Andreas continued his work at TUM as a Postdoc, further honing his expertise in the field of communication networks. During this time, he also served as a Senior Research Fellow at the Communication Technologies Group within the Faculty of Computer Science at the University of Vienna from March 2019 to January 2022.

Wolfgang Kellerer (M'96, SM'11) is a Full Professor with the Technical University of Munich (TUM), heading the Chair of Communication Networks at the Department of Electrical and Computer Engineering. Before, he was for over ten years with NTT DOCOMO's European Research Laboratories. He currently serves as an associate editor for IEEE Transactions on Network and Service Management and as the area editor for Network Virtualization for IEEE Communications Surveys and Tutorials.