Annulus: A Dual Congestion Control Loop for Datacenter and WAN Traffic Aggregates

Ahmed Saeed[♢], Varun Gupta[†], Prateesh Goyal[♢], Milad Sharif[‡], Rong Pan[§], Mostafa Ammar[♡], Ellen Zegura[♡], Keon Jang[♠], Mohammad Alizadeh[♢], Abdul Kabbani[♣], Amin Vahdat[♣]
[♢]MIT CSAIL, [†]AT&T, [‡]SambaNova, [§]Intel, [♡]Georgia Tech, [♠]MPI-SWS, [♣]Google

ABSTRACT

Cloud services are deployed in datacenters connected though high-bandwidth Wide Area Networks (WANs). We find that WAN traffic negatively impacts the performance of datacenter traffic, increasing tail latency by 2.5×, despite its small bandwidth demand. This behavior is caused by the long round-trip time (RTT) for WAN traffic, combined with limited buffering in datacenter switches. The long WAN RTT forces datacenter traffic to take the full burden of reacting to congestion. Furthermore, datacenter traffic changes on a faster time-scale than the WAN RTT, making it difficult for WAN congestion control to estimate available bandwidth accurately.

We present Annulus, a congestion control scheme that relies on two control loops to address these challenges. One control loop leverages existing congestion control algorithms for bottlenecks where there is only one type of traffic (i.e., WAN or datacenter). The other loop handles bottlenecks shared between WAN and datacenter traffic near the traffic source, using direct feedback from the bottleneck. We implement Annulus on a testbed and in simulation. Compared to baselines using BBR for WAN congestion control and DCTCP or DCQCN for datacenter congestion control, Annulus increases bottleneck utilization by 10% and lowers datacenter flow completion time by 1.3-3.5×.

CCS CONCEPTS

• **Networks** → **Transport protocols**; **Data center networks**.

KEYWORDS

Congestion Control, Data Center Networks, Wide-Area Networks, Explicit Direct Congestion Notification

ACM Reference Format:
Ahmed Saeed, Varun Gupta, Prateesh Goyal, Milad Sharif, Rong Pan, Mostafa Ammar, Ellen Zegura, Keon Jang, Mohammad Alizadeh, Abdul Kabbani, Amin Vahdat. 2020. Annulus: A Dual Congestion Control Loop for Datacenter and WAN Traffic Aggregates. In *Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication (SIGCOMM '20), August 10–14, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3387514.3405899

SIGCOMM '20, August 10–14, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7955-7/20/08.
https://doi.org/10.1145/3387514.3405899

1 INTRODUCTION

Large-scale cloud services are built on an infrastructure of datacenters connected though high bandwidth wide area networks (WANs) [24, 26, 27]. WAN traffic shares the datacenter network with intra-datacenter traffic, with the ratio of datacenter to WAN traffic typically around 5:1 [42]. Despite the small fraction of WAN traffic, we find that its impact on datacenter traffic is significant when both types of traffic are bottlenecked at the same switch. Storage racks are an example of such scenarios. A single rack serves applications both within its datacenter and in other datacenters. High volumes of small requests sent to a single rack can generate a large number of large responses, creating a bottleneck within the datacenter that impacts both WAN and datacenter flows. In production, we find that surges in WAN traffic originating from a datacenter increase tail latency of datacenter traffic by 2.5× (§2.1).

To better understand this behavior, consider that congestion control algorithms take a round-trip time (RTT) to react to changes in available bandwidth. When WAN and datacenter traffic are bottlenecked together, a datacenter flow will react to congestion hundreds of times before a WAN flow receives even its first feedback signal. Therefore, datacenter traffic takes the full burden of slowing down in response to congestion. WAN flows, on the other hand, build up long queues before their congestion control algorithms react, leading to packet drops and increasing latency for datacenter traffic. The behavior is exacerbated by datacenter congestion control algorithms that attempt to keep queues short [31, 34, 50].

WAN flows also suffer throughput loss due to the large variations in bandwidth caused by datacenter traffic, and the very small buffers in datacenter switches. Congestion control algorithms typically need buffering proportional to the bandwidth-delay product (BDP) of a flow to achieve high throughput [25], but datacenter switches have 1-2 orders of magnitude less buffer than the typical BDP of a WAN flow. The industry trend in recent years has been increasing link speeds (e.g., from 10 Gbps to 400 Gbps), while buffer sizes have stagnated (12 MB to 72 MB) [3, 10, 12, 35]. Buffer size requirements can be reduced for large numbers of flows [9] and by using better congestion control algorithms (e.g., DCTCP requires only 17% of BDP for high utilization [7]). However, competition between WAN and datacenter traffic creates significant challenges for WAN flows bottlenecked at shallow-buffered switches. Datacenter traffic is bursty and can create large fluctuations in bandwidth over sub-millisecond timescales. Since these fluctuations occur on a timescale much smaller than the WAN RTT, WAN flows cannot accurately track the available bandwidth without excessive buffering.

With shallow-buffered switches, even sophisticated buffer sharing or traffic isolation mechanisms cannot address the previous challenges. Buffer sharing techniques [36] allow a congested port to

grab a bigger share of a switch's total buffer, but the huge discrepancy between the amount of buffering available and what is needed by WAN flows makes these schemes less effective in our setting. Isolating datacenter and WAN traffic in separate queues can improve the performance of datacenter traffic. However, traffic isolation does not reduce bandwidth variability caused by datacenter traffic which negatively impacts the performance of WAN traffic. It also does not resolve the root cause of poor performance for WAN flows: the lack of adequate buffering relative to their BDP. Furthermore, datacenter switches have a small number of queues (e.g., 4-12 priority queues per port [3]), which are typically allocated to different traffic classes based on application and business requirements. To isolate datacenter and WAN traffic, we need two queues instead of one for each traffic class, wasting an already scarce resource.

Our insight is that fast reaction to congestion is necessary to remedy performance impairments of WAN traffic that shares a bottleneck with datacenter traffic. Fast reaction to congestion allows WAN traffic to adapt to fast changes in available bandwidth caused by demand variations of datacenter traffic. Further, it effectively reduces the BDP of WAN flows by several orders of magnitude, dramatically lowering their buffer requirements. There are several approaches to speedup reaction to congestion including. For instance, a bottleneck can generate explicit congestion notification messages to the sender [1, 40]. Another example is performing congestion on a per-link basis [33] or terminating and restarting connections using proxies, where congestion control is performed independently for different parts of the path [13].

One approach to building a scheme for improving the interaction between WAN and datacenter traffic could be to propose a new "one-size-fits-all" algorithm that works for both WAN and datacenter flows. Such a protocol would have to replace deployed algorithms that have undergone years of fine-tuning (e.g., DCQCN, TIMELY, HPCC, CUBIC, and BBR), and would likely face significant resistance to adoption (especially in the WAN). We take a more pragmatic approach. We design a solution that resolves performance impairments of WAN and datacenter traffic by augmenting existing WAN and datacenter protocols, imposing no limitations on their design. Moreover, we avoid making any changes to the network infrastructure (e.g., deploying proxies), requiring only minimal changes to the software stack of traffic sources.

In this paper we explore using *direct feedback* from switches, where datacenter and WAN traffic compete, to tackle the discussed challenges. When a datacenter switch experiences congestion, it sends a direct feedback signal to the senders of both datacenter and WAN flows. A direct signal reduces feedback delay by several orders of magnitude for WAN flows, thereby enabling these flows to react quickly to congestion. Direct feedback is effective only when it reduces delay feedback compared to mirrored feedback. Hence, approaches relying on direct feedback can improve performance compared to end-to-end approaches when the congestion occurs near the traffic source, which we show to be a common case in production settings. Providing direct feedback to both datacenter and WAN flows helps ensure fairness in the way they react.

We present *Annulus,* a congestion control system designed to handle the mixture of WAN and datacenter traffic. Annulus sources rely on two control loops to deal with two different types of congestion events: (1) congestion at nearby datacenter switches (e.g., ToRs) configured to send direct feedback; (2) congestion at other WAN or datacenter switches that do not send direct feedback. In the first case, Annulus reacts to the direct feedback signal using a "near-source" control algorithm, reducing reaction delay to near-source congestion. In the second case, it relies on an existing WAN or datacenter congestion control algorithm. Our design addresses two challenging aspects of such a dual control-loop protocol: the design of the near-source control algorithm (§3.2) and how it should interact with existing congestion control algorithms (§3.3).

We implement Annulus in a userspace network processing stack (§4). For direct feedback, we rely on the existing Quantized Congestion Notification (QCN) [1] mechanism, supported on most commodity datacenter switches. We evaluate the Annulus implementation on a testbed of three racks, two in one cluster, and one in separate cluster, connected by a private WAN. In the testbed, we compare Annulus to a setup where DCTCP is used for datacenter congestion control and BBR is used for WAN congestion control. We find that Annulus improves datacenter traffic tail latency by 43.2% at medium loads, and by up to 56× in cases where the majority of traffic at the bottleneck is WAN traffic. Annulus improves fairness between WAN and datacenter flows and supports configurable weighted fairness. We also find that Annulus improves bottleneck utilization by 10%. In simulations, we compare Annulus to TCP CUBIC, DCTCP, and DCQCN under various workloads. We find that Annulus reduces datacenter flow completion time by up to 3.5× and 2× compared to DCTCP and DCQCN, respectively. It also improves WAN flow completion time by around 10% compared to both DCTCP and DCQCN.

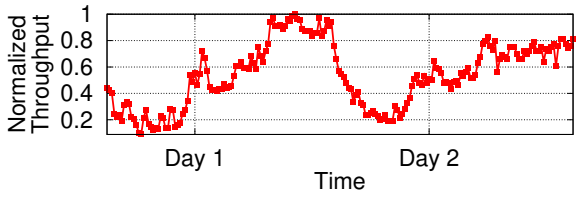
This work does not raise any ethical issues.

2 MOTIVATION

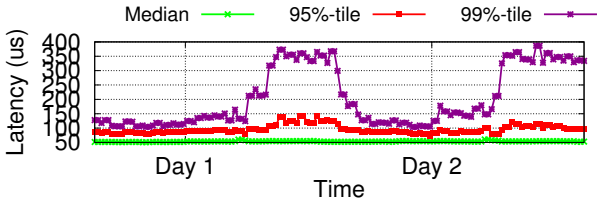
In this section, we explore the interaction of datacenter and WAN traffic at their shared bottlenecks. We start by observing their interaction in a production environment at a large-scale cloud operator, showing that surges in WAN demand lead to performance degradation in datacenter tail latency. We use simulations to validate that the performance degradation of datacenter and WAN persists regardless of the congestion control algorithm, congestion signal, and scheduling scheme at the switch. We find that performance degradation occurs due to the large delay of WAN congestion feedback, exacerbated by the limited buffer space in datacenter switches. Long delays in WAN feedback lead to two fundamental issues: 1) datacenter traffic reacts much faster, taking the full burden of slowing down to drain the queues while facing long queues caused by the slow reacting WAN traffic, and 2) feedback for the WAN traffic lags behind changes in capacity, making it difficult for the WAN congestion control to track available bandwidth accurately. This leads to either under utilization or more queuing. We show that a direct signal from the bottleneck switch to the traffic source can improve the performance of both WAN and datacenter compared to other approaches.

2.1 Interaction of WAN and Datacenter Traffic in the Wild

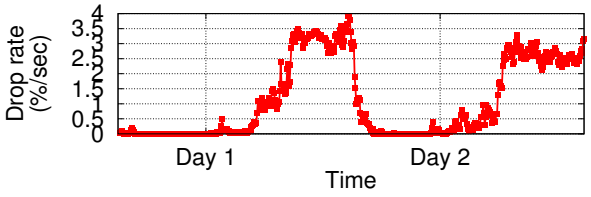
We collect measurements from two production clusters at a large cloud operator, over the period of a month. We record throughput,



(a) Normalized aggregate throughput of WAN traffic initiated from studied cluster



(b) Aggregate latency for intra-cluster traffic



(c) Aggregate drop rate at ToR uplinks

Figure 1: Analysis of the impact of WAN traffic on datacenter traffic from Cluster 1 over two days. All figures capture the same period.

averaged over periods of five minutes, and end-to-end Remote Procedure Call (RPC) latency, averaged over periods of twenty minutes. Both measurements are collected at end hosts and aggregated over all machines in the clusters. Thus, we collect data at a relatively low frequency in order to avoid interfering with server operations in terms of both processing and storage. We classify throughput measurements into WAN (i.e., traffic exiting the cluster)[1] and datacenter (i.e., traffic that remains in the cluster). Furthermore, we collect drop rates at switches, averaged over five minutes, to determine the bottleneck location and severity. We correlate end-host measurements with switch measurements. WAN and datacenter traffic both use the same priority group and compete for the same buffer space. For congestion control, WAN Traffic uses TCP BBR while datacenter traffic uses DCTCP. The total traffic load in both cluster is stable over the period of two days, with average load being 87% of the maximum load.

We focus on measurements collected over the period of two days in Cluster 1 (Figure 1). The WAN load varies significantly over the two days (Figure 1a), dropping to 20% of its maximum. As clear from Figures 1a and 1b, there is a strong correlation between changes in WAN demand and the 99th percentile of RPC latency of datacenter traffic. Surges in demand by WAN traffic lead to 2.5× increase in tail latency of datacenter RPCs, with a correlation coefficient of

[1]In this paper, we focus on WAN traffic existing the datacenter as we observe it to be the likely to compete with datacenter traffic.

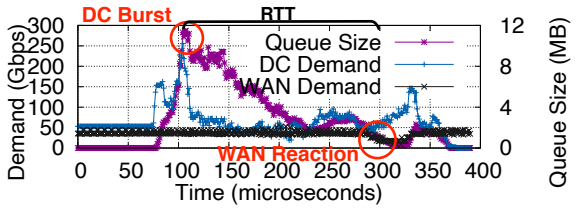


Figure 2: WAN flows start reacting to a burst in demand much later than the burst had occurred and datacenter flows have already reduced their rate.

0.8 between the two values. We find that drop rates are zero in all topology stages except at ToR uplinks (i.e., links connecting ToR switches to higher stages in the topology). We also observe no persistent bottlenecks in the rest of the path of WAN traffic. ToR uplinks are the first oversubscription point on the traffic originating from the cluster. Figure 1c shows a strong correlation between drop rate at ToR switches and WAN traffic behavior with a coefficient of 0.79. In Cluster 2, the correlation coefficients between WAN traffic and tail latency of datacenter RPC and drops at the ToR are 0.54 and 0.64, respectively. We also found that such behavior can persist for a month in one of the clusters. More figures are shown in Appendix B.

2.2 Causes of Performance Impairments

Long feedback delay of WAN flows is the crux of performance impairments observed in production clusters. First, consider the case where WAN and datacenter flows share the same buffer. Both WAN and datacenter congestion control aim at minimizing the occupancy of the shared buffer. However, WAN traffic reacts to congestion 10-10,000× later than datacenter traffic due to the difference in RTTs. Thus, datacenter flows will react and keep reacting to the buffer buildup caused by the inaction of the WAN flow, leading to long delays and potentially packet drops for datacenter traffic. Figure 2, based on simulations we present in the next section, illustrates this issue. The figure shows aggregate WAN demand, aggregate datacenter demand, and buffer occupancy. When datacenter demand surges, buffer occupancy increases. Datacenter traffic reacts quickly to buffer buildup and drains the queues. WAN reacts after an RTT (200 microseconds), after the queues have been drained.

A potential approach to improve the performance of datacenter and WAN traffic is to isolate them in separate queues at the switch. Isolation at the switch requires careful tuning of scheduling algorithms and allocation of buffer space through buffer carving; this in itself is limiting and wasteful (Appendix A). Still, even when used successfully, isolation leads to performance improvements only for datacenter traffic. The reason is that isolation does not address the fundamental root cause of performance problems for WAN traffic: the long feedback delay and the shallow buffers of datacenter switches. Congestion control algorithms typically need buffering proportional to the BDP to achieve high throughput [25], but datacenter switches have 1-2 orders of magnitude less buffer than the typical BDP of a WAN flow. For example, the classic buffer sizing rule of thumb [9] for New Reno suggests that a single TCP NewReno flow requires one BDP of buffer space to sustain 100% throughput. This amounts to 125MB for a 20ms RTT and bandwidth of 50 Gbps. Of course, the buffer requirement drops in the presence of more

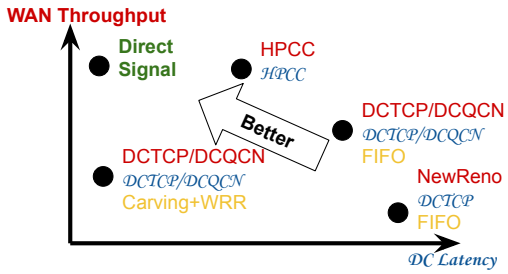


Figure 3: Summary of results for different configurations (WAN congestion control, DC congestion control, and buffer scheduling schemes).

flows and with less synchronization [9]. It can also be reduced by relying on better congestion control algorithms (e.g., DCTCP [7]). Nonetheless the mismatch between the amount of buffer required and that available in datacenter switches is large. For example, the Broadcom Trident II has only 12MB of buffer [10, 12, 35] which must be shared among all ports (and both datacenter and WAN traffic).

The relationship between buffer requirement and BDP is a known issue when designing WAN congestion control algorithms. However, the problem is worse when WAN traffic is competing for bandwidth with datacenter traffic, even when isolated at the switch. Buffer sizing rules relating BDP to buffer space requirements typically assume a fixed capacity bottleneck link [7, 9]. For WAN flows, the datacenter traffic breaks this assumption since it causes wide fluctuations in available bandwidth. These fluctuations occur at a timescale that is significantly smaller than the RTT of WAN flows. For instance, in a single WAN RTT of 20 milliseconds, thousands of datacenter flows can start and finish. This large variability makes it difficult for WAN flows to accurately track available bandwidth, leading to under utilization when they understand the bandwidth, or excessive buffering when they overestimate it. This would not be a major issue if WAN flows were allocated enough buffer space at the bottleneck, which is not feasible in datacenter switches.

2.3 A Closer Look at Interaction of WAN and Datacenter Traffic

In this section, we show that causes of performance degradation discussed earlier are fundamental to bottlenecks shared by WAN and datacenter traffic. We use simulations of various configurations (i.e., congestion control algorithm and buffer scheduling schemes). Our goal is to move from basic configurations to more sophisticated ones, understanding the behavior of each type of traffic with better reaction to congestion and added isolation. This methodology allows us to demonstrate and understand performance issues in easy to understand settings (e.g., using well understood algorithms like NewReno and DCTCP going through a tail drop buffer). Then, we show that similar issues persist as we use more complicated schemes (e.g., using buffer carving for DCQCN and DCTCP traffic).

We use TCP NewReno and DCTCP for WAN congestion control for long WAN RTT and DCQCN and HPCC for short WAN RTT. We use DCTCP, DCQCN, and HPCC for datacenter congestion control. *Note that we consider BBR in the previous section as well as our evaluation.* At the switch, we use buffer sharing where both types of traffic belong to the same priority group, compete for the same

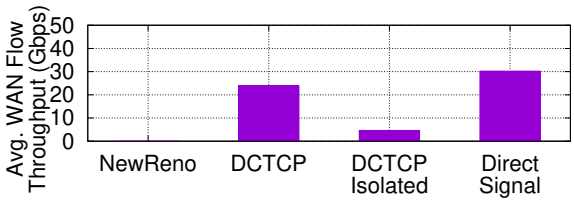


Figure 4: Average throughput of WAN flows.

buffer space, and are FIFO-scheduled. We also consider configurations where WAN and datacenter traffic are isolated through buffer carving combined with weighted round robin scheduling; this is referred to as as "Isolated" in figures. We use Flow Completion Time (FCT) to evaluate datacenter traffic performance and average throughput for WAN traffic. Figure 3 shows a summary sketch of the performance of all explored schemes.

We perform NS3 [41] simulations of a datacenter rack with 10 machines each connected with a 50 Gbps link to the ToR switch. The ToR switch is connected to the rest of the network through a single 100 Gbps link, creating a 5:1 oversubscription ratio. Datacenter traffic originates from 8 machines, with an overall average load of 40 Gbps, and flow sizes sampled from the distribution reported in [42]. The two remaining machines generate a single long WAN flow, each. Datacenter RTT is 8 microseconds while WAN RTT is 20 milliseconds. The only bottleneck in the path of WAN and datacenter traffic is the ToR uplink, where the ToR switch has 12 MB of buffers. The ECN marking threshold triggers with parameters K_{min} = 200KB and K_{max} = 800KB. We use the implementation of the authors of DCQCN and HPCC.

Ideally, the datacenter traffic should use 40% of the ToR uplink, maintaining low FCT. WAN flows should consume the rest of available bandwidth achieving an aggregate of 60 Gbps (average of 30 Gbps for two flows). We normalize the FCT of datacenter traffic for each scenario by its performance when it is using the network exclusively. We use the 30 Gbps average throughput as the ideal throughput for WAN traffic.

NewReno/DCTCP on FIFO queues: We start with the basic case where WAN traffic uses NewReno and datacenter traffic uses DCTCP. We configure NewReno to have an initial window than can achieve full link utilization so as to avoid its slow ramp up. Both types of traffic share buffer space. This combination represents the worst case scenario, with WAN flows building queues and only reacting to drops and datacenter flows waiting in long queues behind WAN traffic. Spikes in demand by datacenter traffic reduce bandwidth available to WAN. Due to large delays in WAN feedback, a long queue of WAN packets build up which easily exceeds the available buffer space in the switch, leading to drops and increase in datacenter FCT. Average WAN flow throughput is 1.2% of the ideal throughput, due severe window reduction by NewReno (Figure 4). Tail datacenter small flow completion time increases 7× compared to ideal throughput, due to the long queues (Figure 5).

Improving WAN congestion control: To overcome the severe reaction of NewReno, we employ a DCTCP for WAN congestion control. DCTCP has the advantage of modulating its reaction based on the severity of the congestion, similar to BBR v2 [17]. This allows it to require significantly less buffering than NewReno (17% of BDP compared to a full BDP). DCTCP significantly improves

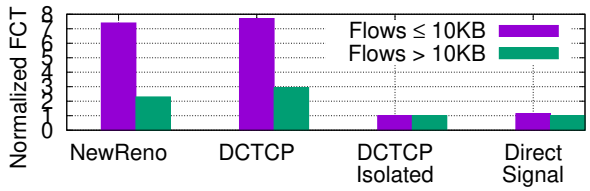


Figure 5: 99th Percentile FCT of datacenter flows.

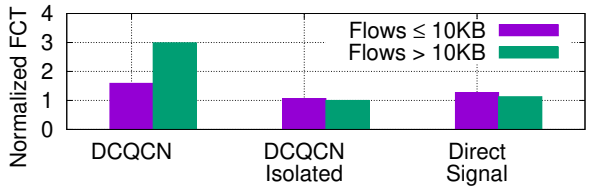


Figure 6: 99th Percentile FCT of datacenter flows when competing with short RTT WAN flows.

WAN throughput to only be 76% of the ideal throughput (losing only 24% of maximum throughput). However, the performance of datacenter traffic remains poor (similar to the previous scenario), due to long queues incurred by WAN flows and their slow reaction to changes in available bandwidth.

Adding buffer carving and scheduling at the switch: In the previous configuration, WAN traffic occupies most of the switch buffer space, leading to significant delay for datacenter traffic. Thus, in this configuration, we allocate buffer space at the switches such that datacenter traffic achieves its ideal performance and observe the performance of WAN traffic under such allocation. In particular, we configure the switches to allow WAN traffic to use a maximum of 25% of buffer space and use weighted round robin scheduling, providing datacenter traffic 4× the weight of WAN traffic. This scheme improves the performance of datacenter flows dramatically, leading to performance comparable to the baseline for both short and long flows (Figure 5). However, the performance of WAN traffic degrades, leading to average WAN throughput to be 13% of the ideal throughput (losing 87% of maximum possible throughput). This is caused by the large variability in available bandwidth for WAN traffic (esp. since the scheduling policy at the switch favors datacenter traffic), and the limited buffer space available for the WAN traffic.

WAN flows with short RTT: The problem is not limited to scenarios where WAN flows have long RTTs. We establish this fact by considering scenarios where WAN RTT is fairly small (e.g., inter-datacenter network within the same metro). We configure WAN RTT to be 200 microseconds. This short RTT allows us to use higher precision congestion control algorithms (i.e., HPCC and DCQCN) for WAN traffic, with the same configurations used in datacenter cases. When using DCQCN, the behavior of both WAN and datacenter traffic is similar in both small and large RTT settings. In particular, sharing buffer space leads to degradation in both WAN and datacenter performance where WAN throughput is 33% lower than ideal throughput and datacenter FCT is reduced by up to 3× (Figure 6). When isolated, DCQCN improves datacenter performance to baseline, while WAN performance degrades by 78%. HPCC reacts to overall buffer occupancy and rate at the bottleneck, complicating attempts for isolation at the switch (Appendix C).

In summary, improvements in congestion control algorithms and buffer management and isolation schemes can improve the interaction of WAN and datacenter traffic, compared to basic schemes. However, long tail latency for datacenter traffic and poor WAN throughput persist as long as both types of traffic share buffer space. The main culprits are the long feedback delay of WAN congestion control and the small buffer space in datacenter switches. The long delays mean that when available bandwidth decreases, a burst of WAN traffic has to be buffered at the bottleneck. This leads to long queues, increasing tail latency of datacenter traffic. Further, the limited buffer space at datacenter switches means that a WAN burst is mostly dropped because WAN BDP is much larger than available buffer space, leading to low WAN throughput.

2.4 Value of Direct Signals

A direct signal reduces feedback delay, leading to smaller BDP and consequently lowers buffer requirements. Furthermore, a direct signal provides a more recent view of the bottleneck, allowing for a more accurate tracking of available bandwidth (i.e., by avoiding bursts that exceed available capacity). Thus, direct feedback-based schemes can reduce tail latency of datacenter traffic and increase the throughput of WAN traffic. We develop a proof-of-concept implementation of a direct signal, presenting the full design of Annulus in the next section. The signal is generated based on the same rule that determines ECN marking of packets in DCTCP. A flow source reacts to the fast signal by halving its current transmission rate. It increases its rate again using similar rules as DCQCN. We implement this signal in the simulation setup presented above. The direct signal-based scheme provides the best compromise for WAN and datacenter performance, compared to schemes we presented earlier. In the case of long WAN RTT, it provides average WAN throughput that is 10% higher than ideal throughput, as due to random behavior of traffic generator utilization was slightly lower than 40%. Datacenter traffic achieves similar performance to the baseline for large flows, while achieving tail latency for small datacenter flows that is only 14% worse than the baseline. Another benefit of direct signals is that it can improve the performance of purely WAN traffic that is congested at a near-source datacenter bottleneck (§5.1.2).

There are several proposals for congestion control algorithms that rely on direct signals [1, 40, 48]. However, they typically require support from all switches, limiting their applicability (especially in the WAN). Furthermore, QCN as a standalone solution requires routing L2 packets through IP-routed datacenter networks which presents a significant overhead [1, 50]. Our approach requires only near-source switches to support direct QCN feedback (e.g., ToR which our measurements found to be the most bottlenecked). This allows for low-cost deployment relying on existing feedback signal. It also simplifies routing of QCN messages (§4). Another approach to solve the problem is to terminate flows when they exit and enter the datacenter, making all flows datacenter flows within the datacenter. This approach requires modification to the datacenter infrastructure to add such proxies at scale that support WAN demand. This approach is too costly compared to our proposal.

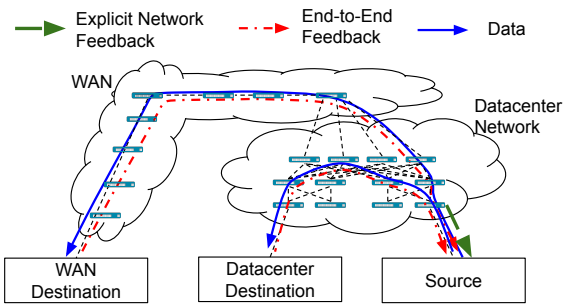


Figure 7: Feedback in Annulus.

3 ANNULUS DESIGN

Annulus is a dual control-loop congestion control system. It is a software only solution that requires no hardware modifications to existing NICs or switches. It employs direct feedback from only near-source switches, where datacenter and WAN traffic are likely to compete, leaving congestion control of other bottlenecks to be handled by their respective WAN or datacenter congestion control schemes. In particular, Annulus, as suggested by its name, the following control loops (Figure 7):

- End-to-end control loop: This control loop relies on existing congestion control algorithms to handle congestion that occur deep in the traffic path (i.e., not close to traffic sources where WAN and datacenter traffic mix).
- Near-source control loop: This control loop relies on congestion notification from switches near the source (e.g., ToR switches) to adjust its sending rate. This allows WAN and datacenter traffic to adjust their rate at the same time granularity, significantly improving WAN reaction time.

Annulus simply picks the minimum transmission rate of the two control loops. By relying on the minimum rate, Annulus makes a flow transmit at the rate accommodating the hop with least available capacity in the flow's path. When bottlenecks change for a specific flow, Annulus receives feedback signal from the new bottleneck reducing the rate of its respective loop, while the other control loop ramps up its rate due to lack of congestion signals.

3.1 Annulus Design Rationale

Annulus design has the following goals:

1) Performance at homogeneous bottlenecks (i.e., purely WAN or datacenter traffic) should remain at current standards. Datacenter and WAN traffic should retain the performance of the state-of-the-art algorithm they are currently using when the bottleneck is not shared. We aim to design an algorithm that resolves the issues presented in the previous section without impacting network performance for other, more common, types of bottlenecks.

2) Low switch overhead and preferably no switch modifications. Solutions requiring sophisticated behavior at switches are becoming more feasible [14]. However, a significant portion of current infrastructure does not support such programmability. Thus, it is preferable that the proposed approach does not introduce significant modifications to switches, if any.

Annulus design as a dual control-loop system does not constraint how homogeneous bottlenecks are controlled. We satisfy the first requirement by allowing existing algorithms to work in tandem with the near-source congestion control loop. The second requirement can be satisfied by relying on the QCN direct signal which part of IEEE 802.1Qau [1], implemented in many commodity datacenter switches. Generating more accurate signals might require switch modifications. However, the crux of our proposal is that a direct signal reduces feedback delay, leading to lower buffer requirements and more accurate estimation of available bandwidth.

3.2 Near-Source Congestion Control Loop

The near-source control loop has three components: 1) generation of the direct signal, and 2) rate control at the source.

Direct Signal Generation: We rely on QCN Congestion Notification (CN) messages as the direct signal from switches, indicating congestion. A switch maintains a congestion measure $F_b = (Q - Q_{eq}) + w \cdot (Q - Q_{old})$, where Q is the instantaneous queue size, Q_{eq} is the target queue size, Q_{old} is the queue size when F_b was last calculated, and w is a non-negative constant which is typically set to 2. The intuition is that F_b captures the queue-size excess $(Q - Q_{eq})$ and the derivative of the queue size $(Q - Q_{old})$. Hence, positive F_b means that either the buffer or the link are oversubscribed, indicating congestion.

The congestion measure F_b is calculated based on the state of the queue when a incoming sample packet arrives. Packets are sampled by selecting packets periodically every M arriving bytes, where M depends on F_b.[2] If the congestion measure is positive, a CN message is generated and sent to the source of the sampled packet. The CN message contains the F_b value. The CN can also include 64 bytes of the sampled packet, which is not part of the standard but implemented by some commodity switches. These 64 bytes are enough to identify the flow that sent the packet at the end host, as it includes the source and destination IPs and ports, mitigating one of the main drawbacks of QCN reported in [50].

Rate Control: The "near-source" control loop follows similar rules to that of Reaction Point in the QCN algorithm. We remove the details for brevity and refer interested readers to [37]. The main differences between Annulus and the QCN reaction point, is that Annulus is fully implemented in software, relying on ACK-clocking for rate increase. The standard QCN algorithm relies on a mix of timers and byte counters to increase the transmission rate. However, relying on counters can lead to oscillation of the transmission rate of WAN traffic as rate increases based only on the senders behavior and decreases based on feedback from the near-source bottleneck. Thus, we rely on ACK-clocking, where rate increases on ACKs, rather than based on byte counters or timers. This approach also helps simplify the implementation.

Window vs Rate: Window-based congestion control algorithms provide robustness to feedback delay by limiting the number of packets in flight to a maximum of the window size. This provides more robustness to congestion as the sender stops sending packets when it has a window worth of packets in flight. Rate-based algorithms keep sending at a specific rate and only slow down when they receive feedback. Despite the advantages of a window-based algorithm, the near-source control loop has to employ a rate-based algorithm. A window controls the number of packets in flight along

[2] The mapping between M and F_b is shown in [37].

Algorithm 1 Annulus Sender Algorithm. Each Flow F maintains two data structures for each control loop: End to End (E2E) and Near Source (NS). Each loop maintains a current rate and Annulus selects the minimum rate. EndToEndCCAlg is the existing end-to-end window-based congestion control algorithm. NearSourceCCAlg is the near-source congestion control algorithm that reacts to QCN messages.

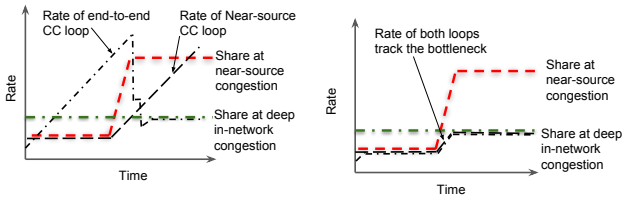
```
1:  procedure EndToEndRateCalc(Flow F)
2:      if F.E2E.currentRate < F.minRate × 2 then
3:          F.E2E.currentWindow = EndToEndCCAlg(F);
4:          F.E2E.currentRate = F.E2E.currentWindow × F.RTT;
5:      end if
6:      return ;
7:  end procedure
8:  procedure NearSourceRateCalc(Flow F)
9:      if F.NS.currentRate < F.minRate × 2 then
10:         F.NS.currentRate = NearSourceCCAlg(F);
11:     end if
12:     return ;
13: end procedure
14: procedure Annulus(Flow F)
15:     EndToEndRateCalc(F);
16:     NearSourceRateCalc(F);
17:     F.minRate = min(F.NS.currentRate, F.E2E.currentRate);
18:     return
19: end procedure
```

the whole path. The near-source control loop manages only a part of the path, requiring a window-based implementation to know the packets in flight within only that part of the path. This is not possible in modern networks as it requires per-packet acknowledgement from every hop in the path, leaving a rate-based implementation as the only viable solution. The drawback of rate-based algorithms is alleviated by relying on a direct signal because it allows for fast reaction to congestion.

3.3 Combining the Two Control Loops

Combining Windows and Rates: The end-to-end control loop can use either a window-based algorithm or a rate-based algorithm. The near-source control loop is rate-based. Thus, the overall algorithm is rate-based. Annulus chooses the minimum rate by converting the window set by the end-to-end control loop to a rate by dividing it over the *RTT* estimate, kept by all modern congestion control algorithms.

Utilization vs Cautious Probing: Each control loop manages congestion at a subset of a flow's path. The bottleneck in the path can change from the part of the path managed by one control loop to the other. This requires some coordination between the two control loops to improve the performance of Annulus. First, consider if the two control loops are uncoordinated. Each control loop sets its transmission rate independently of one another, R_1 of control loop C_1 and R_2 of control loop C_2. Annulus picks the smaller rate, say R_1, to accommodate the smaller bottleneck. In the absence of cross traffic at hops managed by C_2, it does not get any feedback and keeps increasing R_2. This increase has no effect on the actual transmission rate of the flow (Annulus is sending at R_1). Hence, C_2's view of the network is distorted by the lack of feedback. In the presence of cross traffic, C_2 gets feedback corresponding to a much higher rate than it should be operating at if it was to share the bottleneck fairly with cross traffic (§D). This inaccurate rate estimate by C_2 can be problematic when the bottleneck moves to the part of the path managed by C_2 (e.g., the current bottleneck is freed due to termination of flows). In that case, R_1 starts to increase, due to increased capacity for C_1 bottleneck, C_2 bursts at a very large R_2 which can take multiple RTTs to reduce to a reasonable level. Figure 8a shows the behavior of the rates produced by the two control loops when left uncoordinated.



(a) Behavior of the two loops if uncoordinated, which can result in loss

(b) Behavior when both loops are coordinated

Figure 8: Illustration of the impact of control loop coordination on Annulus rate behavior.

To avoid bursting, the control loop setting the minimum rate signals the other control loop to pause the growth of its window or rate. For example, in the previous scenario, C_1 signals C_2 to pause the growth of R_2 to $2\times R_1$ [20]. Furthermore, drops in the smaller rate are signalled to trigger proportional drops in the rate set by the other control loop. Figure 8b shows an illustration of the approach. The rate always tracks the minimum available bandwidth, avoiding exceeding the rate at the bottleneck. This avoids creating bursts when bottlenecks shift between near-source and deep path. We realize that this approach can lead to lower utilization in case the congestion does not change in the larger control loop. However, our goal is to avoid bursts or drops caused by the introduction of the additional control loop.

Algorithm 1 illustrates how the two control loops are combined.

4 IMPLEMENTATION

Figure 9 shows the architecture of Annulus at the end host. Feedback is obtained from both near-source switches and typical end-to-end signals. We leverage existing kernel implementation of the end-to-end control loop. This loop can be different depending on traffic type (i.e., datacenter or WAN). The implementation of end-to-end loops is modified so that each packet is tagged by a transmission time that is calculated based on a pacing rate inferred from the congestion window and RTT of its flow.[3] The near-source control loop is implemented in a userspace networking stack that captures packets after being processed by the host kernel [32]. The near-source loop keeps track of all active TCP connections and calculates their transmission rate based on received messages, translating it into a per-packet transmission time. The furthest timestamp in the future is picked as it represents the lowest rate. A timestamp-based rate limiter is used to schedule packet transmission [43]. The two control loops coordinate their rate growth, as discussed earlier. Note that Annulus can still be implemented in a virtualized

[3]The implementation of BBR in Linux already supports this functionality.

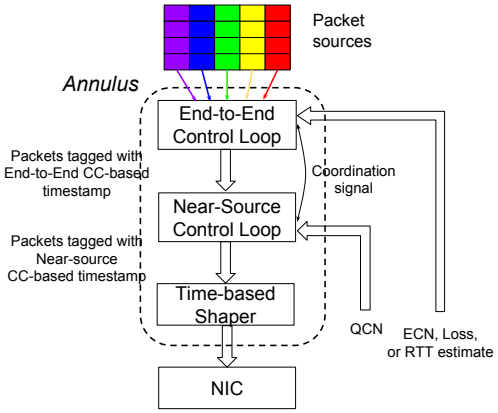


Figure 9: Architecture of Annulus end host.

setting, requiring the guest network stack to at least tag packets with their transmission time. This requires minimal changes to the driver in the guest OS so it can pass the timestamp information. Full functionality of Annulus requires the host OS to be able to pause the rate growth of flows in the guest OS. Such coordination between guest and host network stacks was shown to be beneficial in other settings as well [29, 49].

Routing QCN messages: The main challenge of deploying QCN is doing so at scale in an L3 routed network. To overcome this issue, we leverage some standard, but limited switch features, which makes deploying QCN feasible by focusing on congestion that happen only close to the source (i.e., ToRs where WAN and datacenter interaction is observed in practice). In particular, we leverage the L2 learning feature available on any L3 commodity switch today. L2 learning allows a switch to cache the source MAC address of a packet along with the corresponding input switch port number. The switch can then properly forward an L2 packet (e.g. QCN notification frame) to a MAC address that has been already cached through the corresponding cached port number. Today's data center switches typically have dedicated L2 tables, that cannot be used for L3 entries. These tables can accommodate at the order of 100,000 MAC entries. This is more than sufficient for maintaining the cached MAC address of the sampled packet long enough before its corresponding congestion control frame traverses back in the reverse direction using the cached information.[4]

Hence, all that is needed for QCN's notification packets to get routed back to their L2 sources is preserving their L2 source MAC address throughout the fabric (i.e. don't over-write that value end-to-end) and turning on L2 learning. Furthermore, we focus on deploying QCN close to traffic sources (i.e., ToR). This reduces the pressure on the memory. Thus, switches can continue to simultaneously routed IP packets based on the IP table information while forwarding non-IP L2 frames based on the L2 table information that gets populated based on the L2 header of IP packets. Note that the available memory for L2 learning allows for deployment in switches a level higher than ToRs in the topology, however, we didn't perform any experiments at that scale.

Note that routing QCN messages is a limitation of our implementation of Annulus. However, it is not a limitation of our design. In fact, Annulus provides performance improvements even if congestion occurs at switches connecting the datacenter to the WAN, as we show in simulation. Thus, we hope that this work motivates further development of direct signalling schemes whose messages can be routed in current datacenter networks.

5 EVALUATION

We evaluate the performance of Annulus on a testbed as well as using packet-level simulations in ns2 [2]. We compare the performance of Annulus to DCTCP and BBR in the testbed and to TCP CUBIC, DCTCP, and DCQCN in simulations. We evaluate Annulus in scenarios of traffic mixes and pure WAN and datacenter traffic. The testbed evaluation demonstrates the feasibility and value of Annulus, while the simulations allow us to evaluate Annulus under conditions that are hard to replicate in the testbed. Our main evaluation metrics are RPC tail latency for datacenter traffic and transfer latency and throughput for WAN traffic. This focus is because of the typical drive to operate WAN traffic at near capacity [44, 47] while datacenter traffic typically is more concerned with latency [6, 31, 34, 50].

5.1 Testbed Evaluation

Setup: We conduct experiments on three racks, two co-located in the same cluster with 20 machines each while the third is in a different cluster with 10 machines, where RTT between the two clusters is 8 ms. All machines are equipped with dual port 20 Gbps NICs. We maintain control over path congestion by forcing congestion to be at the ToR by setting a high oversubcription ratio of 5:1 to ToR uplink, and operating in non-peak hours. We configured QCN only at ToR switches. We set Q_{eq} to 100KB per port and a W of 8, where packets are samples every 15KB. **Note that while some figures label baselines as only DCTCP or BBR, all testbed evaluation scenarios use DCTCP exclusively for datacenter and TCP BBR exclusively for WAN.**

Workload: We generate synthetic RPC loads. RPC sizes are set to 127KB unless otherwise stated. We use a cross-rack all to all communication pattern, where all machines in one rack send traffic to all machines in another rack. This communication pattern generates a large number of flows and allows for stress testing the behavior of the congestion control algorithms. We generate three types of workloads: 1) Datacenter/WAN mixture with various mixture ratios, 2) edge caching traffic where the traffic is only WAN traffic, and 3) purely datacenter traffic that reflects the more common case in cluster traffic. WAN traffic is generated by sending traffic from 10 machines in one rack in one cluster to 10 machines in the rack in the other cluster, while datacenter traffic is generated by sending traffic from 20 machines in one rack to 20 machines in another rack.

5.1.1 WAN/Datacenter Mix. We start by looking at overall performance when WAN and datacenter traffic are mixed. In this scenario, we care about improving datacenter traffic isolation from WAN traffic, while retaining WAN traffic performance.

[4]The worst case here is when as many packets as the capacity of the L2 table arrive with a different MAC address each at the highest speed (i.e. from all ports). Conservatively assuming 100,000 L2 entries and a 16-port 40Gbps switch, with a 0.5KB average packet size, it is impossible that an L2 cached entry could be evicted in less than 600μs from the time it was added, which is much more than the round-trip time in any reasonably designed data center today.

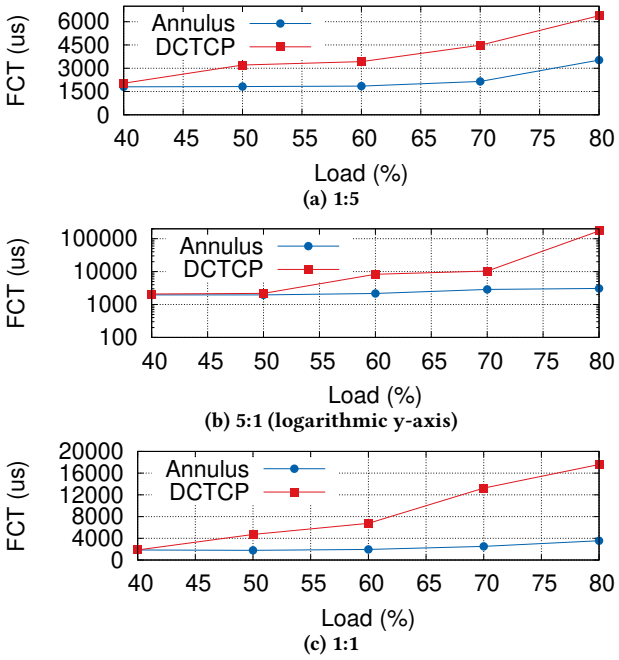


Figure 10: RPC latency of datacenter traffic for different WAN:Datacenter traffic ratios.

	Annulus	DCTCP/BBR	Max/Min Fair
Datacenter:WAN	5.8:1	1:7.6	4:1

Table 1: Ratio of achieved throughput under different schemes.

Datacenter Latency: Figure 10 shows the measured performance of 99th percentile RPC latency over datacenter. We find that at common mixes of 5:1 datacenter to WAN, even at small loads (i.e., 50%), latency is reduced by 43.2%. This is achieved without observing any changes in WAN traffic performance. On the other hand, in scenarios where traffic is more dominated by WAN traffic, where the ratio is reversed (i.e., 1:5 in Figure 10b), Annulus results in datacenter RPC latency that is 56x better at high loads. This is also combined by 40% reduction in average WAN traffic latency. Even in cases where traffic mix is even, we find that Annulus produces 2x improvements at 60% of bottleneck capacity. This shows that, as intended, Annulus reduces the impact of WAN traffic on datacenter traffic even at large shares of WAN traffic.

Fairness: We also perform experiments without forcing the traffic ratio between datacenter and WAN to observe the result sharing scheme when aggressive WAN congestion control interacts with datacenter traffic. Table 1 shows the results of such experiments. Annulus allows for a sharing policy that is proportional to the number of flows (400 flows for datacenter and 100 flows for WAN). Annulus can be used to provide different sharing ratios by changing how aggressively different flows react to congestion at the near-source bottleneck. This can be achieved by tuning the G_d value in the QCN algorithm, reducing or increasing the aggressiveness of the QCN control loop.

Bottleneck utilization: Experiments without forcing the traffic ratio allows us to also observe the maximum achievable goodput, summarized in Table 2. Annulus improves maximum achievable goodput by 5-10%. To better understand this impact, we look at the tcpdump for a single WAN flow over the period of a second. Figure 11 shows RTT and bytes in flights produced by both protocols. We find that Annulus produces more stable behavior, maintaining two segments in flight over the whole period while sustaining stable RTT between 10-13 ms by keeping the queues stable. On the other hand, BBR produces a fluctuating RTT ranging from 9-48ms which is caused by fluctuations in queue length. These fluctuations cause drops which leads BBR to frequently back off, reducing its goodput.

Setting	Annulus	DCTCP/BBR
Datacenter/WAN mix	91.51%	86.63%
Datacenter	90.42%	82.13%
WAN	92.1%	83.28%

Table 2: Bottleneck utilization under different traffic mixes.

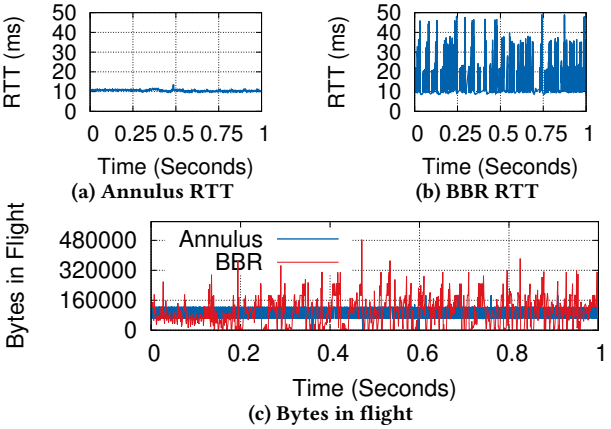


Figure 11: Behavior of a single flow over the period of one second showing bytes in flight and RTT for Annulus and BBR.

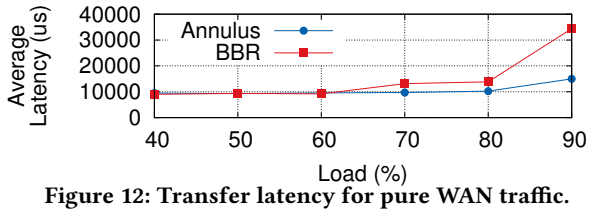
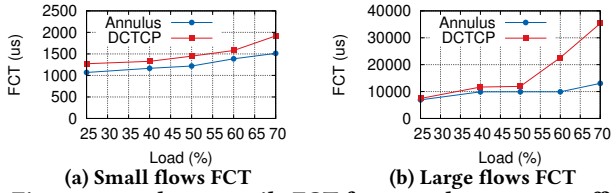


Figure 12: Transfer latency for pure WAN traffic.

5.1.2 Edge Caching (purely WAN traffic). In this case, we only look at the average latency of message transfer between ends as it better reflects the requirements of edge caching applications. Figure 12 shows the results. Annulus reduces average latency by 26% at 70% load and 80% at 90% load. It also improves latency by 2.2x at near bottleneck capacity while improving goodput by 10% (Table 2).

5.1.3 Purely datacenter traffic. We validate that Annulus does not cause regression in performance of datacenter traffic by comparing it to DCTCP under the scenario where traffic is 100% datacenter. In this experiment, we mix RPC load of small sized messages of 10KB and large sized messages of 1MB. The goal of this experiment is to show that Annulus does not introduce regression in the more common case of purely datacenter traffic. Figure 13 shows the



(a) Small flows FCT **(b) Large flows FCT**

Figure 13: 99th percentile FCT for pure datacenter traffic.

tail FCT for both jobs where Annulus improves performance of both jobs by up to 2.7x for the large RPC job at 70% utilization. This improvement in performance can be explained by the faster congestion signal.

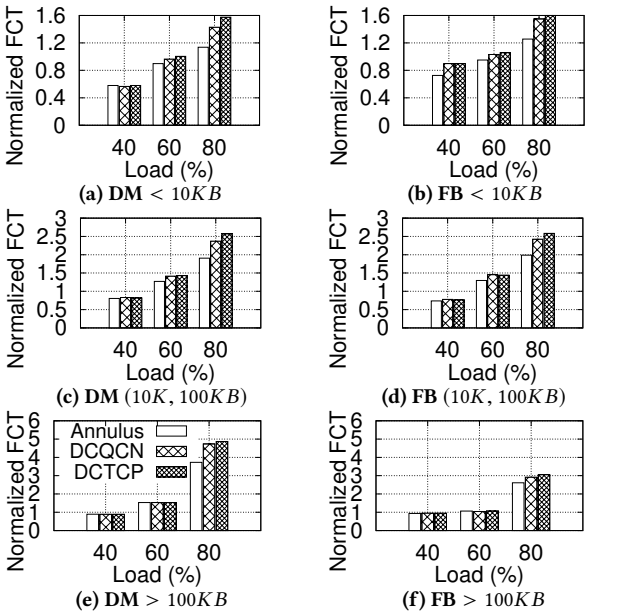
5.2 Simulations Evaluation

Setup: We use the fat-tree network [30]. The fabric interconnects 128 servers organized into four pods. Each pod consists of four aggregation switches and four top-of-rack (ToR) switches. Aggregation switches are connected to eight core switches resulting in a fabric with overall 4:1 oversubscription. Similar to Google's Jupiter architecture [45], the fabric is directly connected to the inter-cluster networking layer with an external cluster border router. Each pod is provided with a pool of 25% of aggregate intra-cluster bandwidth [45] for external connectivity. We use 10Gbps point-to-point Ethernet links across our entire network. All the switches in the topology have a per-port buffer capacity of 1MB. We also configure the host delay and intra-datacenter switching delay to be $1\mu s$ and $2\mu s$, respectively. Thus, the minimum RTT between two servers on different pods of a datacenter is $14\mu s$. In order to incorporate the effect of long RTT of inter-datacenter traffic, we set the propagation delay of the links connecting hosts to the external switch to $10ms$, resulting in RTT of roughly $\sim 20ms$ for datacenter-edge traffic. We use ECMP as our multipath routing scheme unless mentioned otherwise.

Implementation: We compare Annulus to TCP CUBIC, DCTCP, and DCQCN. DCTCP leverages ECN to convey congestion information to the end hosts and adjusts the congestion window size based on the fraction of marked bytes. As WAN networks generally do not support end-to-end ECN marking, we only enable ECN marking on intra-datacenter switches[5]. DCQCN [50] is another rate-based protocol that also relies on ECN marks. Note that DCTCP and DCQCN are used for datacenter traffic as well as WAN traffic (i.e., WAN traffic gets marked in the datacenter and its receivers mirror the ECN marking). This allows for comparing Annulus to systems where WAN traffic also receives feedback from near-source congestion points.

We use ns-2 FullTCP Sack implementation as our standard TCP protocol and build other schemes on top of it. For DCTCP, we set the parameters as described in [6]: (1) g, the factor for exponential weighted averaging, is set to $\frac{1}{16}$, and (2) K, the buffer occupancy threshold for setting the CE-bit, is set to $90KB$ (typical for 10 Gbps links). For a fair comparison, for QCN-based schemes, we set Q_{off} equal $90KB$. All other TCP functionalities are the same as in FullTCP Sack implementation. An important factor in flow completion times is the Retransmission Timeout (RTO) of TCP as dropped packets are retransmitted after expiration of an RTO. We use commonly used RTO values for inter and intra datacenter traffic [46, 48]. For intra-datacenter traffic, we set the RTO value as $1ms$. For inter-datacenter traffic, we use an RTO value of $100ms$.

[5]We did evaluate with ECN enabled on all of the switches, end-to-end, and have found marking on the inter-datacenter fabric to have almost no impact on our results, as most of the congestion happens locally in the intra-datacenter fabric.



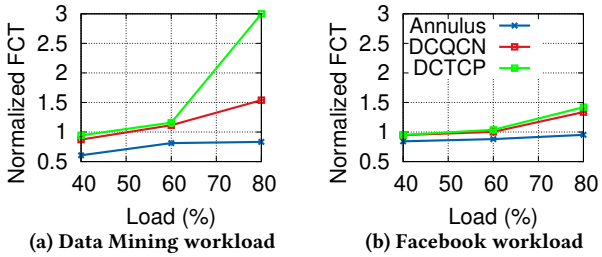
(a) DM $< 10KB$ **(b) FB** $< 10KB$ **(c) DM** $(10K, 100KB)$ **(d) FB** $(10K, 100KB)$ **(e) DM** $> 100KB$ **(f) FB** $> 100KB$

Figure 14: Average flow completion times for Data Mining (DM) and Facebook (FB) workloads. Numbers are normalized to FCT achieved by TCP CUBIC at 40% load. Note that the range of the y-axis is different.

Workload: We simulate empirical workloads based on observed distributions in production data centers. In particular, we consider two flow size distributions from a cluster running Data Mining workload [21] and a Facebook's Hadoop cluster workload [42]. We generate mix of inter- and intra-datacenter traffic with roughly 1:5 ratio similar to Facebook's production network [42]. In order to simulate the high utilization of the inter-datacenter fabric, we generate competing traffic originating from the external hosts. We keep the link utilization of the external links at about 80% [26]. For all our simulations, we select the source-destination pairs uniformly across all of the host.

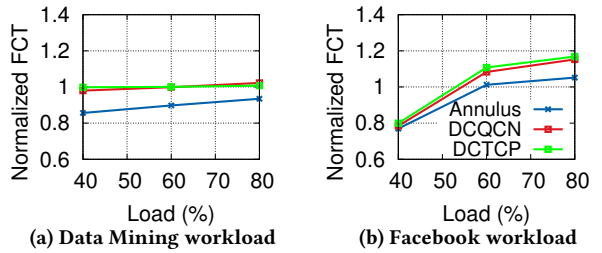
5.2.1 WAN/Datacenter Mix. Figure 14 shows the average completion times for intra-datacenter traffic achieved by each scheme as the fabric load varies from 40% to 80% for the two workloads. The results are obtained for simulations with more than 500,000 flows and normalized to the FCT achieved with TCP CUBIC at 40% load. We break down the results for small $[0, 10KB]$, medium $[10KB, 100KB]$, and large (> 100 KB) flows.

For the Data Mining workload, Annulus achieves about $4-5\times$ lower average FCT compared to TCP CUBIC for small and medium flows. Note that TCP CUBIC does not appear in Figure 14 as its performance is outside the plotted range. As expected, DCTCP and DCQCN have comparable performance as they both use ECN as the congestion feedback. However, Annulus outperforms DCQCN, achieving 30% lower average FCT across all flow sizes at high load. For the Facebook workload, the average flow completion times for



(a) Data Mining workload (b) Facebook workload

Figure 15: 99.9th percentile FCT for small intra-datacenter flows.



(a) Data Mining workload (b) Facebook workload

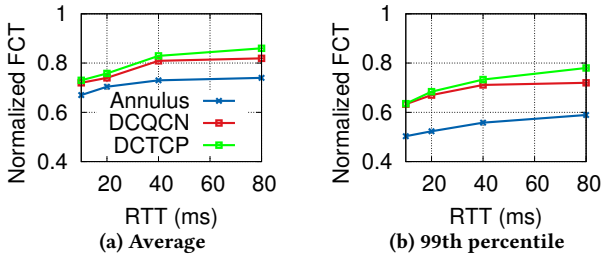
Figure 16: Overall average FCT for inter-datacenter flows.

Annulus are 4 – 5× smaller than TCP CUBIC for small flows. Similar to the other workload, Annulus achieves 16 – 31% lower average FCT compared to DCTCP and 12 – 24% lower than DCQCN.

The benefits of Annulus are more apparent in the tail latencies of small flows. Figure 15 shows the 99.9 percentile of FCTs for small flows for the two workloads. For the Data Mining workload, Annulus leads to 5×, 3.5×, and 2× lower tail latency at 80% fabric load compared to TCP CUBIC, DCTCP, and DCQCN, respectively. For Facebook workload, Annulus outperforms DCQCN and achieves 30% lower tail latency at high load. This is consistent with our testbed results.

We also found that Annulus improves the performance of the inter-datacenter flows as well in the case of datacenter and WAN traffic mixture. Figures 16a and 16b show the overall average FCT for all the inter-datacenter traffic for data-mining and Facebook workload, respectively. As it can be seen, Annulus improves the overall average FCT by ∼ 10% compared to other schemes. Note that inter-datacenter FCTs do not vary much as we always maintain the link utilization of the external links at 80% regardless of the intra-datacenter fabric load by generating competing flows. This is inconsistent with our testbed evaluation, where WAN traffic does not show improvements except at high WAN loads, is due to the difference in implementation of the networking stack. In particular, in our testbed evaluation, all traffic is paced. In the simulations, we show that Annulus retains its advantages even when pacing is not enabled, emphasizing the effect of the direct signal compared to burst avoidance.

5.2.2 Edge Caching (purely WAN traffic). We evaluated the value of Annulus for edge caching on the testbed. In simulation, we evaluate the effect of WAN RTT on the performance of Annulus. Intuitively, the longer the RTT, the higher the value of Annulus as end-to-end techniques react slower. To model such a scenario, we simulate a 20:1 incast for WAN flows with varying RTT between 20ms to 80ms.



(a) Average (b) 99th percentile

Figure 17: Normalized FCT of large transfers for various RTTs.

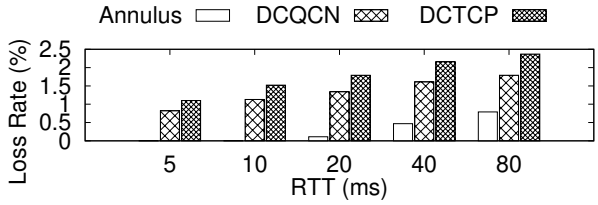


Figure 18: Packet loss rate at the congestion point.

We set the delay from the senders to the congestion point to 50μs. Each sender, transmits large transfers of 50MB to a single host. Once each transfer is complete, senders immediately initiate the next transfer. We run the simulation for 10,000 transfers and compute the average and 99th percentile FCT. Figure 17 shows the results for Annulus, DCTCP, and DCQCN. The numbers are normalized by the FCT achieved by TCP CUBIC. Annulus leads to almost 10% smaller average FCT and 20% smaller tail FCT when compared to DCTCP and DCQCN. This is mostly due to roundtrip-timescale reaction of DCTCP and DCQCN to the congestion, which is at order of tens of milliseconds. Annulus exploits the direct congestion feedback of QCN and reacts much faster that other scheme. This also lowers the drop rate (< 1%) at the congestion points (Figure 18).

6 DISCUSSION

Improving the performance of CDNs: We show that Annulus can be beneficial for cases where the traffic is mostly WAN. This shows that Annulus can be of significant use for edge caching applications, where congestion might not be only at the ToR, but also at the border switches connecting a cluster to the WAN. The delay of a direct feedback signal from border switches provides much faster feedback compared to end-to-end signals, allowing the benefits of Annulus to remain in effect. However, such deployment carries the challenges of fully routing L2 packets in IP-routed networks. We hope that the results we present here rekindle this discussion and encourage innovation that will allow for fast reaction to near-source congestion.

Leveraging flexible and open data planes: More switches are supporting programmable and open data planes including in-network telemetry (INT) [4, 5]. This has enabled the development of congestion control algorithms that leverages the INT information to accurately react to congestion (e.g., HPCC [31]). INT information is tagged on packets and mirrored by receivers by piggybacking the information on acknowledgments, increasing the delay of receiving such feedback. Our work with Annulus presents a motivation for developing schemes that can report INT signals directly to sender

from switches. This will enable the development of better near-source control loops.

Limitations: Our validation of Annulus is based on experiments. Constructing models for dual loop congestion control protocols remains an open and challenging problem. In particular, we believe our work should motivate further analysis of compatibility of different control loops. Such work should analyze the stability of a switched systems, like Annulus, where the overall behavior is defined by two control loops which are triggered based on the state of the system. Furthermore, our solution focuses on near-source congestion for bottlenecks shared by WAN and datacenter traffic. Theoretically, WAN and datacenter flows can also share bottlenecks near the receiver. However, addressing such bottlenecks requires a different approach (e.g., flow termination at the edge of the datacenter).

7 RELATED WORK

Congestion Control using delayed signals: The most common approach to congestion control relies on signals that are inferred from information piggybacked on acknowledgment packets. The most basic signals are loss [19, 22, 25] and delay [11, 15, 16] which are inferred by the sender based on the sequence numbers of the acknowledgment. Explicit signals, generated by switches, help senders infer congestion more accurately. Such signals include Explicit Congestion Notification (ECN) [6, 8, 17, 50] and In-band Network Telemetry (INT) [31] which are piggybacked on packets are mirrored by the receiver in acknowledgments. Some algorithms rely on switches to infer the sending rate of senders (e.g., RCP [18] and XCP [28]). However, all such algorithms suffer from issues caused by delayed feedback discussed earlier. Annulus leverages the advantages of using such algorithms by keeping existing end-to-end congestion control algorithms intact, while improving avoiding their main drawback of long delays by augmenting their decisions using the direct feedback signal.

Congestion Control using direct signals: Switches can use ICMP Source Quench to notify sources of congestion [40]. However, the deployment of ICMP Source Quench never took off and has been recently deprecated due to challenges associated with practical deployments. Further, ICMP source quench messages do not specify the state of the switch (e.g., queue length), providing a single-bit congestion signal. FastLane [48] provides direct feedback from switches to traffic sources in cases of packet drops. This allows Fastlane to improve the performance of short flows significantly. However, generating messages only in cases of drops can allow long queues build up. Further, FastLane is not supported by any hardware the authors are aware of. Both ICMP source quench and Fastlane operate above L2, allowing them avoid issues faced by QCN (e.g., routing messages in an L3 network and requiring NIC support). Annulus addresses QCN issues at the end host. We hope that our results will motivate better direct signals that are implemented above L2. Such signals and the algorithms using them can be implemented within the framework defined by Annulus.

Centralized scheduling: WAN and datacenter performance issues discussed earlier can be resolved if buffer occupancy within the datacenter is orchestrated by a central entity. Fastpass [39] and FlowTune [38] are examples of such systems. However, their messaging complexity proved prohibitive for practical consideration. Congestion control for datacenter-edge links has not been widely addressed. Most of the existing schemes rely on minimizing congestion at inter-datacenter or WAN links through traffic engineering (e.g., SWAN [23] and B4 [26]). Annulus allows for fast reaction to near-source congestion which improves WAN performance even in cases when WAN traffic is the majority of network traffic.

8 CONCLUSION

WAN and datacenter traffic have significantly different requirements and path characteristics. We show that when the two types of traffic compete for bandwidth and buffer space, both suffer performance degradation. WAN feedback delay is too large and buffer space at datacenter switches is too small. Datacenter traffic reacts fast, taking most of the burden of draining queues. Furthermore, feedback for the WAN traffic lags behind changes in capacity, making it difficult for the WAN congestion control to track available bandwidth accurately, leading to queue build up. We propose using direct congestion signaling to reduce feedback time of WAN traffic. We introduce Annulus, a dual congestion control loop that relies on QCN to implement the near-source congestion control loop. This approach significantly improves datacenter latency, utilization of bottlenecks, and fairness between WAN and datacenter traffic. We envision that this work will motivate further investigation of multi-control loop congestion control as well as better models of scenarios where WAN and datacenter traffic compete for bandwidth.

9 ACKNOWLEDGMENTS

We thank Yuchung Cheng, Neal Cardwell, Soheil Hassas Yeganeh, Aamer Mahmood, and David Wetherall for providing useful feedback. We thank our shepherd, Costin Raiciu, and anonymous SIGCOMM reviewers for their detailed and excellent feedback. This work was funded in part by an award from the Cisco Research Center and the National Science Foundation grants CNS 1816331, CNS 1910676, CNS 1751009, and CNS 1563826.

REFERENCES

[1] 2010. IEEE Standard for Local and metropolitan area networks– Virtual Bridged Local Area Networks Amendment 13: Congestion Notification. *IEEE Std 802.1Qau-2010 (Amendment to IEEE Std 802.1Q-2005)* (April 2010), c1–119. https://doi.org/10.1109/IEEESTD.2010.5454063

[2] 2011. NS-2 network simulator. http://nsnam.sourceforge.net/wiki/index.php/Mainpage.

[3] 2019. BCM56980 12.8 Tb/s Multilayer Switch Data Sheet. https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56980-series.

[4] 2019. In-Band Network Telemetry - A Powerful Analytics Framework for your Data Center. https://www.opencompute.org/files/INT-In-Band-Network-Telemetry-A-Powerful-Analytics-Framework-for-your-Data-Center-OCP-Final3.pdf.

[5] 2019. New Trident 3 switch delivers smarter programmability for enterprise and service provider datacenters. https://www.broadcom.com/blog/new-trident-3-switch-delivers-smarter-programmability-for-enterprise-and-service-provider-datacenters.

[6] Mohammad Alizadeh, Albert Greenberg, David A Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. 2011. Data center tcp (dctcp). In *Proc. of ACM SIGCOMM'11.*

[7] Mohammad Alizadeh, Adel Javanmard, and Balaji Prabhakar. 2011. Analysis of DCTCP: Stability, Convergence, and Fairness. In *Proc. of ACM SIGMETRICS'11.*

[8] Mohammad Alizadeh, Abdul Kabbani, Tom Edsall, Balaji Prabhakar, Amin Vahdat, and Masato Yasuda. 2012. Less is more: trading a little bandwidth for ultra-low latency in the data center. In *Proc. of USENIX NSDI'12.*

[9] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. 2004. Sizing Router Buffers. *SIGCOMM Comput. Commun. Rev.* (2004).

[10] Alex Arcilla and Tony Palmer. 2019. Broadcom Trident 3 Platform Performance Analysis: Achieving Predictably High Performance for Real-world Data Center Workloads. *ESG Technical Validation* (2019).

[11] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical Delay-Based Congestion Control for the Internet. In *Proc. of USENIX NSDI'18.*

[12] Wei Bai, Kai Chen, Shuihai Hu, Kun Tan, and Yongqiang Xiong. 2017. Congestion Control for High-Speed Extremely Shallow-Buffered Datacenter Networks. In *Proc. of the First Asia-Pacific Workshop on Networking (APNet'17).*

[13] John Border, Markku Kojo, Jim Griner, Gabriel Montenegro, and Zach Shelby. 2001. *Performance enhancing proxies intended to mitigate link-related degradations.* RFC 3135. Network Working Group.

[14] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. 2014. P4: Programming Protocol-Independent Packet Processors. *SIGCOMM Comput. Commun. Rev.* (2014).

[15] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. 1994. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proc. of ACM SIGCOMM'94.*

[16] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2016. BBR: Congestion-based congestion control. *Queue* (2016).

[17] Neal Cardwell, Yuchung Cheng, Soheil Hassas Yeganeh, Priyaranjan Jha, Yousuk Seung, Ian Swett, Victor Vasiliev, Bin Wu, Matt Mathis, and Van Jacobson. 2019. BBR v2: A Model-based Congestion Control - IETF 105 Update. https://datatracker.ietf.org/meeting/105/materials/slides-105-iccrg-bbr-v2-a-model-based-congestion-control-00.

[18] Nandita Dukkipati. 2007. *Rate Control Protocol (RCP): Congestion control to make flows complete quickly.* Ph.D. Dissertation. Stanford University.

[19] Sally Floyd, Tom Henderson, and Andrei Gurtov. 2004. *The NewReno Modification to TCP's Fast Recovery Algorithm.* RFC 3782. Network Working Group.

[20] Prateesh Goyal, Anup Agarwal, Ravi Netravali, Mohammad Alizadeh, and Hari Balakrishnan. 2020. ABC: A Simple Explicit Congestion Controller for Wireless Networks. In *Proc. of USENIX NSDI '20.*

[21] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. 2009. VL2: A Scalable and Flexible Data Center Network. In *Proc. of ACM SIGCOMM'09.*

[22] Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: A New TCP-friendly High-speed TCP Variant. *SIGOPS Oper. Syst. Rev.* 42, 5 (July 2008).

[23] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. 2013. Achieving high utilization with software-driven WAN. In *Proc. of ACM SIGCOMM'13.*

[24] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Kondapa Naidu B., Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, and et al. 2018. B4 and after: Managing Hierarchy, Partitioning, and Asymmetry for Availability and Scale in Google's Software-Defined WAN. In *Proc. of ACM SIGCOMM'18.*

[25] V. Jacobson. 1988. Congestion Avoidance and Control. *SIGCOMM Comput. Commun. Rev.* (1988).

[26] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, et al. 2013. B4: Experience with a globally-deployed software defined WAN. In *Proc. of ACM SIGCOMM'13.*

[27] Mikel Jimenez and Henry Kwok. 2017. Building Express Backbone. https://engineering.fb.com/data-center-engineering/building-express-backbone-facebook-s-new-long-haul-network/.

[28] Dina Katabi, Mark Handley, and Charlie Rohrs. 2002. Congestion Control for High Bandwidth-Delay Product Networks. In *Proc. of ACM SIGCOMM'02.*

[29] Praveen Kumar, Nandita Dukkipati, Nathan Lewis, Yi Cui, Yaogong Wang, Chonggang Li, Valas Valancius, Jake Adriaens, Steve Gribble, Nate Foster, and Amin Vahdat. 2019. PicNIC: Predictable Virtualized NIC. In *Proc. of ACM SIGCOMM'19.*

[30] Charles E Leiserson. 1985. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput.* (1985).

[31] Yuliang Li, Rui Miao, Hongqiang Harry Liu, Yan Zhuang, Fei Feng, Lingbo Tang, Zheng Cao, Ming Zhang, Frank Kelly, Mohammad Alizadeh, and Minlan Yu. 2019. HPCC: High Precision Congestion Control. In *Proc. of ACM SIGCOMM '19.*

[32] Michael Marty, Marc de Kruijf, Jacob Adriaens, Christopher Alfeld, Sean Bauer, Carlo Contavalli, Michael Dalton, Nandita Dukkipati, William C. Evans, Steve Gribble, Nicholas Kidd, Roman Kononov, Gautam Kumar, Carl Mauer, Emily Musick, Lena Olson, Erik Rubow, Michael Ryan, Kevin Springborn, Paul Turner, Valas Valancius, Xi Wang, and Amin Vahdat. 2019. Snap: A Microkernel Approach to Host Networking. In *Proc. of ACM SOSP'19.*

[33] Partho P. Mishra and Hemant Kanakia. 1992. A Hop by Hop Rate-Based Congestion Control Scheme. In *Proc. of ACM SIGCOMM'92.*

[34] Radhika Mittal, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, David Zats, et al. 2015. TIMELY: RTT-based congestion control for the datacenter. In *Proc. of ACM SIGCOMM'15.*

[35] Timothy Prickett Morgan. 2018. A Deep Dive Into Cisco's Use Of Merchant Switch Chips. https://www.nextplatform.com/2018/06/20/a-deep-dive-into-ciscos-use-of-merchant-switch-chips/.

[36] Eugene Opsasnick. 2011. Buffer management and flow control mechanism including packet-based dynamic thresholding. US Patent 7,953,002.

[37] Rong Pan. [n.d.]. QCN Pseudo Code: Version 2.2. http://www.ieee802.org/1/files/public/docs2008/au-pan-QCN-pseudo-code-ver2-2.pdf.

[38] Jonathan Perry, Hari Balakrishnan, and Devavrat Shah. 2017. Flowtune: Flowlet Control for Datacenter Networks. In *Proc. of USENIX NSDI'17.*

[39] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Devavrat Shah, and Hans Fugal. 2014. Fastpass: A centralized zero-queue datacenter network. In *Proc. of ACM SIGCOMM'14.*

[40] Jon Postel. 1981. *Internet Control Message Protocol.* RFC 792. Network Working Group.

[41] George F Riley and Thomas R Henderson. 2010. The ns-3 network simulator. In *Modeling and tools for network simulation.*

[42] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. 2015. Inside the Social Network's (Datacenter) Network. In *Proc. of ACM SIGCOMM '15.*

[43] Ahmed Saeed, Nandita Dukkipati, Vytautas Valancius, Vinh The Lam, Carlo Contavalli, and Amin Vahdat. 2017. Carousel: Scalable Traffic Shaping at End Hosts. In *Proc. of ACM SIGCOMM'17.*

[44] Brandon Schlinker, Hyojeong Kim, Timothy Cui, Ethan Katz-Bassett, Harsha V. Madhyastha, Italo Cunha, James Quinn, Saif Hasan, Petr Lapukhov, and Hongyi Zeng. 2017. Engineering Egress with Edge Fabric: Steering Oceans of Content to the World. In *Proc. of ACM SIGCOMM'17.*

[45] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Holzle, Stephen Stuart, and Amin Vahdat. 2015. Jupiter Rising: A decade of Clos topologies and centralized control in Google's datacenter network. In *Proc. of ACM SIGCOMM'15.*

[46] Vijay Vasudevan, Amar Phanishayee, Hiral Shah, Elie Krevat, David G Andersen, Gregory R Ganger, Garth A Gibson, and Brian Mueller. 2009. Safe and effective fine-grained TCP retransmissions for datacenter communication. In *Proc. of ACM SIGCOMM'09.*

[47] Kok-Kiong Yap, Murtaza Motiwala, Jeremy Rahe, Steve Padgett, Matthew Holliman, Gary Baldus, Marcus Hines, Taeeun Kim, Ashok Narayanan, Ankur Jain, Victor Lin, Colin Rice, Brian Rogan, Arjun Singh, Bert Tanaka, Manish Verma, Puneet Sood, Mukarram Tariq, Matt Tierney, Dzevad Trumic, Vytautas Valancius, Calvin Ying, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. 2017. Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering. In *Proc. of ACM SIGCOMM'17.*

[48] David Zats, Anand Padmanabha Iyer, Ganesh Ananthanarayanan, Rachit Agarwal, Randy Katz, Ion Stoica, and Amin Vahdat. 2015. FastLane: making short flows shorter with agile drop notification. In *Proc. of ACM SoCC'15.*

[49] Yimeng Zhao, Ahmed Saeed, Ellen Zegura, and Mostafa Ammar. 2019. zD: A Scalable Zero-Drop Network Stack at End Hosts. In *Proc. of ACM CoNEXT'19.*

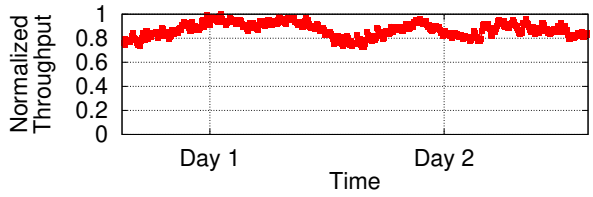
[50] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang. 2015. Congestion control for large-scale RDMA deployments. In *Proc. of ACM SIGCOMM'15.*

Appendix

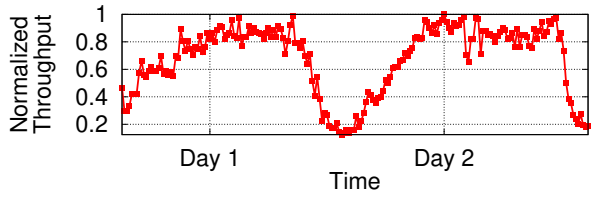
Appendices are supporting material that has not been peer-reviewed.

A THE CASE AGAINST BUFFER CARVING AND SCHEDULING AT THE BOTTLENECK

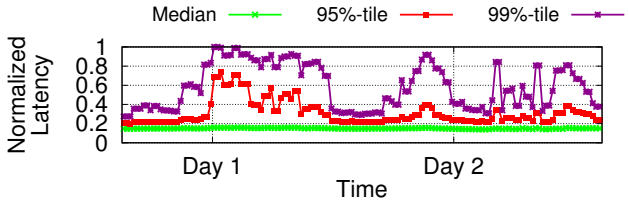
First, this approach only improves the performance of the type of traffic allocated more buffer space and prioritized by the scheduling algorithm at the switch, at the expense of the other type. As shown earlier, WAN traffic still suffer significantly leading to overall low network utilization. Second, we find it to be less effective with protocols like HPCC, which take total buffer occupancy as well as total bottleneck rate as a signal rather than portions used by individual types of traffic. Third, buffer carving is wasteful. Switches have a limited number of queues which are typically allocated and scheduled according to business-defined priorities (e.g., user facing traffic is strictly more important than background copy jobs). The number of jobs and their priorities far exceeds the number of



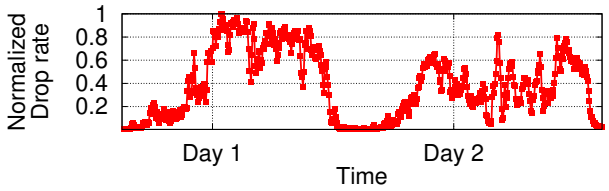
(a) Normalized aggregate throughput of all traffic initiated at studied cluster



(b) Normalized aggregate throughput of WAN traffic initiated from studied cluster

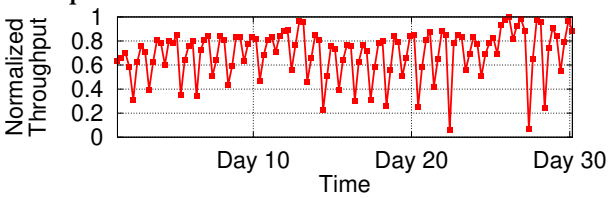


(c) Aggregate latency for intra-cluster traffic

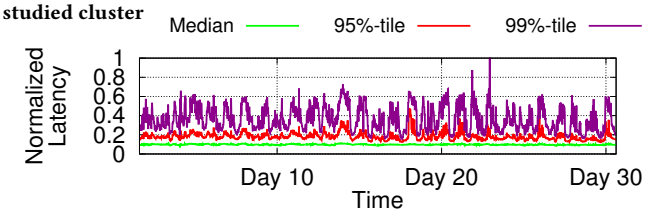


(d) Aggregate drop rate at ToR uplinks

Figure 19: Analysis of the impact of WAN traffic on datacenter traffic from Cluster 2 over two days. All figures capture the same period.



(a) Normalized aggregate throughput of WAN traffic initiated from studied cluster



(b) Aggregate latency for intra-cluster traffic

Figure 20: Analysis of the impact of WAN traffic on Local traffic from cluster 1 over 30 days.

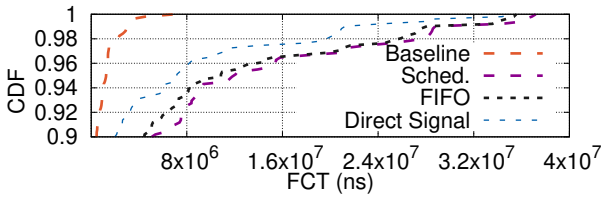


Figure 21: CDF of FCT for datacenter flows $> 10KB$ using HPCC for WAN and datacenter traffic.

available queues. Further dividing this limited resource based on whether the traffic within a single job or user is datacenter traffic or WAN traffic halves the number of available priority levels, which is already scarce. Another type of waste caused by carving is that it dedicates buffer resources to a certain type of traffic even when it is not needed. Finally, buffer carving and scheduling do not handle cases where congestion is caused purely by WAN traffic. In such cases, the delayed reaction of WAN traffic lowers flow completion time due to packet drops which take at least an RTT to recover from. Furthermore, it lowers utilization due to long ramp up time.

B MORE ON WAN AND DATACENTER TRAFFIC IN THE WILD

The behavior of Cluster 1, presented earlier, is also evident in the second cluster. Figure 19 shows the same data from second studied cluster. We note that, all clusters exhibit this behavior with various degrees depending on job placement, demand, and topology. These two clusters exhibit this behavior at cluster scale. This behavior leads to our first two observations.

We record the behavior of Cluster 2 over a period of a month. Figure 20a shows WAN traffic over the period of interest. Figure 20b shows surges of tail latency of local traffic tracking surges in WAN traffic. This behavior was also observed in the second cluster. Note that the tail latency of datacenter traffic, naturally, correlates with the aggregate throughput of all traffic in the cluster (both WAN and datacenter), with a correlation coefficient of 0.82 in case of cluster 2. It also exhibits consistent high correlation with the throughput of WAN traffic (e.g., a correlation coefficient 0.59 for the two-day period in Figure 19). In Section 2.3, we explain in details the reason behind this correlation.

C THE INTERACTION OF WAN AND DATACENTER TRAFFIC WITH HPCC

We find that HPCC behaves the same way whether traffic is isolated or not. This is because HPCC relies on overall buffer occupancy and overall bottleneck rate, ignoring how the buffer space and rate are divided between different traffic classes. HPCC achieves similar WAN throughput to ideal. Figure 21 shows tail FCT of flows with sizes larger than $10KB$. Behavior of small flows was similar to that of long flows, we do not include the figure for brevity. It is possible to change HPCC to handle buffer carving, requiring careful consideration of the impact of partitioning the signal on the algorithm. It will also require considering changes to be made to the ASIC collecting the INT information. Resolving these considerations is out of scope of this work.

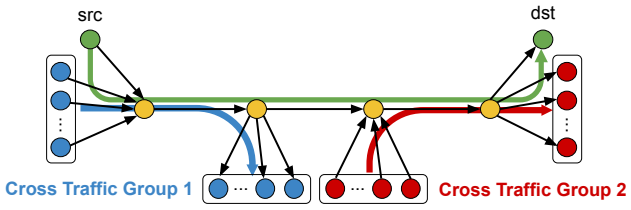


Figure 22: Topology of simulated network.

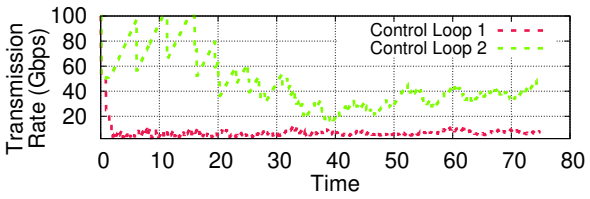


Figure 23: Rate deduced by each control loop when bottleneck 1 has 100 flows going through it while control loop 2 has 10 flows going through it.

D BEHAVIOR OF THE UNCONSTRAINED BOTTLENECK

We answer the following question: when one control loop is setting the minimum rate, what rate is the other control loop reaching? We answer the question through simulations. We create the topology in Figure 22. In this topology, one flow goes through two congested switches, where each switch is controlled by a separate control loop. The control loops rely on ECN marking to determine congestion and react to it using DCTCP. The flow sends at the minimum of the two. The two switches are congested by having Cross Traffic Group 1 and 2 go through them. By changing the number of flows is each group, we make the switch with more flows going through it more congested. We observe the rate deduced by each control loop. In this simulation, all links are 100Gbps with latency of 1 microseconds (i.e., RTT of main flow is 10 microseconds).

Figure 23 shows the rate set by each control loop, when bottleneck managed by control loop 1 has 100 flows going through while control loop two has 10 flows in addition to the main flow. Control loop 1 reaches a rate of around 6 Gbps, which the flow uses. The second loop reaches a rate of 45 Gbps on average. Both control loops reach a rate higher than their fair share which is an artifact of the underlying congestion control algorithm. However, the main take away here is that the second control loop still gets congestion feedback and reduces it rate to a cap that is reasonably within the transmission rate of cross traffic. We repeated the experiments with varying degrees of congestion for the second control loop. We found that the second control loop roughly tracks its fair share, as long as there is cross traffic at the bottleneck handled by the control loop. Cross traffic generates congestion signals, allowing the control loop to set its rate accurately.