Congestion Control for High-speed Extremely Shallow-buffered Datacenter Networks

Wei Bai
HKUST

Kai Chen
HKUST

Shuihai Hu
HKUST

Kun Tan
Huawei

Yongqiang Xiong
Microsoft Research

ABSTRACT

The link speed in datacenters is growing fast, from 1Gbps to 100Gbps. However, the buffer size of commodity switches increases slowly, thus significantly outpaced by the link speed. In such extremely shallow-buffered datacenter networks, prior TCP/ECN solutions suffer from either excessive packet losses or significant throughput degradation. Motivated by this, we introduce BCC, a simple yet effective solution with only one more configuration (shared buffer ECN/RED) at commodity switches. BCC operates based on real-time shared buffer utilization. When the buffer is abundant, BCC delivers both high throughput and low packet loss rate. When it becomes scarce, BCC triggers shared buffer ECN/RED to prevent packet losses at the cost of sacrificing a small amount of throughput. Our preliminary results show that BCC maintains low packet loss rate persistently while only slightly degrading throughput when the buffer becomes insufficient. Compared to current practice, BCC achieves up to 94.4% lower 99th percentile completion time for small flows while only degrading large flows by up to 2.8%.

CCS CONCEPTS

• **Networks** → **Data center networks**;

KEYWORDS

Data center networks, Buffer, ECN, TCP

ACM Reference format:
Wei Bai, Kai Chen, Shuihai Hu, Kun Tan, and Yongqiang Xiong. 2017. Congestion Control for High-speed Extremely Shallow-buffered Datacenter Networks. In *Proceedings of APNet'17, Hong Kong, China, August 03-04, 2017,* 7 pages.

APNet'17, Hong Kong, China

DOI: 10.1145/3106989.3107003

DOI: 10.1145/3106989.3107003

1 INTRODUCTION

Datacenters applications generate a mix of workloads with both latency-sensitive small messages and throughput-sensitive bulk transfers. Hence, datacenter network (DCN) transport should provide low latency and high throughput simultaneously to meet the requirements of applications.

It is a challenge to achieve both goals that are essentially at odds, especially under the shared *shallow-buffered* commodity switches in production DCNs. This challenge has been identified 7 years ago by Microsoft researchers in their production DCNs. To address it, they leveraged ECN [19] to strike the tradeoff between high throughput and low latency, and showed that a properly configured per-port ECN/RED marking scheme could well utilize the shallow buffer to achieve both high throughput and low latency, while still reserving certain headroom to absorb micro-bursts [5]. Since then, ECN-based transports become flourishing [5, 16, 20, 21] and are widely adopted in industry.

However, in this paper, we show that this seemingly solved problem resurges and the solution is now being re-challenged, due to the recent industrial trend. The link speed of production DCNs is growing fast from 1Gbps to 100Gbps, whereas the buffer size of commodity switches increases slowly (*e.g.*, from 4MB at 1Gbps to 16MB at 100Gbps), significantly outpaced by the link speed. Consequently, the buffer size per port per Gbps drops from 85KB to 5.12KB , leading to an *extremely shallow-buffered* DCN environment (§2.3).

We show that it is hard for prior TCP/ECN solutions to remain effective when buffer is extremely shallow (§3). On the one hand, if we configure the ECN marking threshold as originally proposed [5, 21], it causes excessive packet losses even before ECN reacts when many ports are active simultaneously. On the other hand, if we configure a relatively lower ECN threshold than original one, it wastes bandwidth and degrades throughput unnecessarily when fewer ports are busy because ECN over-reacts.

This problem is severe, but receives little attention so far, and there is no readily deployable solution either. Thus, the key contribution of this paper is to expose this problem and its consequences experimentally, and introduce an *extremely*

simple, yet effective and readily deployable solution, named **BCC** (Buffer-aware Congestion Control), to it.

Our design of BCC is inspired by the understanding of modern switching chip functionalities. We are surprised to find that to solve our problem, *one more ECN configuration is enough*! At its core, BCC inherits the success of per-port ECN/RED by DCTCP [5], and further enables shared buffer ECN/RED to cope with the extremely shallow buffer problem. Shared buffer ECN/RED follows the RED [10] algorithm but tracks the occupancy of the shared buffer pool to mark packets. Packets get marked if the shared buffer occupancy exceeds pre-defined thresholds. While this function is there [4, 8], it was less understood and seldom used previously in literature. BCC perhaps exploits it for the first time.

In BCC, shared buffer ECN/RED and per-port ECN/RED work complementarily to each other. When fewer ports are active, the shared buffer is abundant. Hence, per-port ECN/RED will take effect first and strike the balance of high throughput and low latency as before [5]. When more and more ports become active, the shared buffer turns scarcer. Thus, shared buffer ECN/RED will automatically be triggered first to prevent packet losses—BCC trades throughput for latency when achieving both becomes impossible.

We evaluate the performance of BCC using ns-2 simulations (§5). At low loads, BCC fully utilizes the link capacity. It achieves up to 13.5% lower average completion time for large flows, compared to a conservative ECN configuration. At high loads, BCC keeps low packet loss rate while only sacrificing a small amount of throughput. It achieves up to 94.4% lower 99th percentile completion time for small flows while only degrading large flows by up to 2.8%, compared to a standard ECN configuration.

The rest of the paper is organized as follows. We introduce extremely shallow switch buffer and its impacts in §2 and §3, respectively. We present the design of BCC in §4. §5 presents evaluation results. We discuss related work in §6 and conclude the paper in §7.

2 BUFFER IS BECOMING EXTREMELY SHALLOW

In this section, we first understand the buffering logic of existing chips. Then, we quantify the buffer requirements of TCP[1] at high-speed. Finally, we show that the buffer space becomes increasingly insufficient as link speed increases.

2.1 Understanding the switch buffering

On the switching chip, the Memory Memory Management Unit (MMU) allocates the on chip buffer memory to incoming packets. The buffer memory is divided into several pools, which can be classified into following two categories:

- **Private Pools:** dedicated buffers reserved to egress queues.
- **Shared Pools:** shared buffers that can be used once the destination egress queue's private pool has been used up.

When a packet arrives, the MMU first tries to enqueue it into the private pool of the destination queue. If there is no enough buffer space, the MMU tries to enqueue it into the shared pool. The packet only gets dropped by MMU if neither the private pool nor the shared pool has enough space. Moreover, the MMU only drops new arriving packets. Packets in the pool cannot be pushed out and dropped.

2.2 Buffer requirement of TCP at high-speed

TCP is the dominant transport protocol in DCNs [5]. The switch buffer is crucial for TCP's performance. Moderate buffer occupancies are necessary for high throughput [7]. Futhermore, we also need some buffer headroom to absorb transient busrts [5]. Therefore, insufficient switch buffers cause (1) **low throughput**, thus slowing bulk transfers and (2) **excessive packet losses**, thus resulting in large tail completion times for small messages.

To achieve the desired performance, TCP requires *at least* $C \times RTT \times \lambda$ buffer space per port, where C is the link capacity, RTT is the average round-trip time and λ is a characteristic constant of the congestion control algorithm[2]. In recent years, the link speed in DCNs has increased greatly, from 1Gbps to 40Gbps and now to 100Gbps. However, the base latency does not change much as it is mainly determined by processing overhead introduced by various sources (*e.g.*, kernel network stack, driver, NIC and middlebox) along the path. Hence, the buffer demand of TCP almost increases in proportion to the link speed in DCNs.

Testbed measurement: In out testbed, three servers (Mellanox ConnectX-4 100Gbps NIC, Linux kernel 3.10.0) are connected to a Arista 7060CX-32S-F switch. The base latency in is $\sim 30\mu$s. We consider two TCP variants: DCTCP [5] and ECN* [21] (regular ECN-enabled TCP which simply cuts window by half in the presence of an ECN mark). We generate 16 long-lived flows using `iperf` from two senders to a receiver. We vary the RED marking threshold[3] and measure the aggregate throughput at the receiver side. For a TCP variant, its basic buffer requirement approximately equals to the minimum marking threshold delivering 100% link utilization.

Figure 1 shows aggregate throughput results with different thresholds. As expected, ECN* starts to achieve 100% throughput on 325KB which is close to the bandwidth-delay

[1] In this paper, by TCP we refer to various TCP-variants, such as DCTCP [5] and ECN* [21], *etc.*, that are designed for datacenters.

[2] Due to the small number of concurrent large flows in DCNs [5], we can assume large flows are synchronized here.

[3] We set the maximum and minimum queue length thresholds of RED [10] to the same value as previous work [5, 21] suggests.

ASIC	Broadcom 56538	Broadcom Trident+	Broadcom Trident II	Broadcom Tomahawk
Capacity (ports × BW)	48 p × 1 Gbps	48 p × 10 Gbps	32 p × 40 Gbps	32 p × 100 Gbps
Total buffer	4MB	9MB	12MB	16MB (4 MMUs)
Buffer per port	85KB	192KB	384KB	512KB
Buffer per port per Gbps	85KB	19.2KB	9.6KB	5.12KB

Table 1: Information of some commodity datacenter switching chips. Note that Tomahawk has 4 switch cores, each with its own MMU and 4MB buffer [1, 2]. Dynamic buffer sharing only happens within the single core.

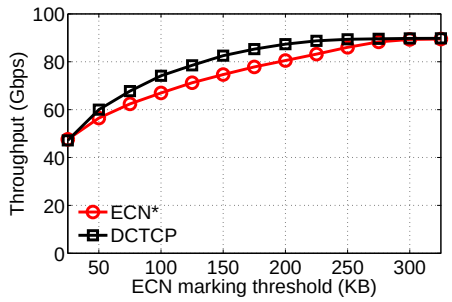


Figure 1: [Testbed] Aggregate TCP throughput with different ECN/RED marking thresholds

product (BDP) in our testbed. Our measurement also shows that DCTCP performs similar as ECN* in practice. The minimum ECN marking threshold that DCTCP requires for 100% throughput is 250KB. The reader may be curious that why our experiment observation of DCTCP seems inconsistent with theory results in [6] (0.17BDP buffering is enough for 100% throughput). We think this is mainly due to packet bursts that are caused by various interactions between the OS and the NIC (*e.g.*, TSO, GRO and interrupt moderation). Hence, a much larger ECN marking threshold is required to absorb bursts. Such complex burst behaviors are difficult to capture by ideal fluid model in [6], thus resulting in the theory-practice gap[4]. We also conduct the above experiment using Windows Server 2012 R2 and observe that DCTCP requires ∼60-70% BDP buffering for 100% throughput.

Production Datacenters: Compared to our simple small-scale testbed, production datacenters are more challenging and have larger base latency. At the end host, packets may experience high processing delay due to kernel scheduling. In the network, packets experience innegligible processing delay when going through various middleboxes (*e.g.*, firewall, IPSec gateway and load balancer). Long-distance cables and multiple switch hops also bring several-microsecond delay. Above factors greatly increase the actual latency in production environments. In [13], the authors show that even the 50th percentile inter-pod latency can exceed 200μs. Such latency eventually transfers to a large buffer demand. Consider a 100Gbps network with 80μs base RTT, the per-port buffer requirement of ECN* can easily reach 1MB.

[4]Such performance-theory gap has also been identified by previous work [21] and even DCTCP paper itself [5].

2.3 Buffer becomes increasingly insufficient

However, the buffer size of commodity switching chips does not increase as expected. We list buffer and capacity information of some commodity chips in Table 1. The capacity significantly outpaces the buffer size, resulting in decreasing buffer per port per Gbps (from 85KB to 5.12KB). The reasons of shallow switch buffers are at least two-fold.

- The memory used in switch buffers is high-speed SRAM. Compared to DRAM, SRAM is more expensive as it requires more transistors.
- The area increases with the memory size. When the area becomes large, the read/write latency will increase, making the memory access speed hard to match the link speed.

Therefore, most commodity switches in DCNs are shallow buffered. We envision that such trend will hold for future 200/400Gbps switching chips.

3 PROBLEMS CAUSED BY EXTREMELY SHALLOW BUFFER

In this section, we show that, in extremely shallow-buffered high-speed DCNs, existing TCP/ECN solutions use switch buffers either (1) too aggressively, thus causing excessive packet losses at high loads (§3.1) or (2) too conservatively, thus seriously degrading throughput at low loads (§3.2).

3.1 Standard ECN configuration causes excessive packet losses

To achieve 100% throughput, operators need to configure a moderate marking threshold (*e.g.*, $C \times RTT \times \lambda$). To the best of our knowledge, this is current operation practice in many production DCNs. However, the standard ECN configuration is likely to overfill extremely shallow buffers when many ports are congested simultaneously, thus causing excessive packet losses and poor performance for small flows.

We take Broadcom Tomahawk with 16MB buffer and 32 100Gbps ports as an example. If TCP desires 1MB ($100Gbps \times 80\mu s$) marking threshold per port, the buffer will be overfilled when more than half of the total ports are congested. What is

worse, Tomahawk has 4 switch cores to achieve desired performance at the high-speed. Each core has its own MMU and 4MB buffer [1, 2] and dynamic buffer sharing only happens within the single core. Therefore, the buffer of a Broadcom Tomahawk chip will be overfilled when more than 4 ports attached to a single core are congested simultaneously.

3.2 Conservative ECN configuration degrades throughput

Realizing the above limitation, a straight forward solution is to configure a lower marking threshold (*e.g.*, $\leq$ average per-port buffer), thus leaving headroom to reduce packet losses. However, this conservative ECN configuration causes much *unnecessary* bandwidth wastage when few ports are congested simultaneously. For example, when only a single switch port is congested, this method still throttles TCP throughput despite the sufficient switch buffer resource.

4 SOLUTION

4.1 Design Goals

We seek to achieve both high throughput and low packet loss rate simultaneously. However, as shown in §3, it is difficult to achieve both metrics when many ports are active simultaneously. When a conflict arises between the two metrics, we prefer to keep low packet loss rate at the cost of sacrificing a small amount of throughput. This is because the bandwidth is generally plentiful in datacenters, while a small increase in packet loss rate (*e.g.*, $\geq 0.1\%$) can seriously degrade the application performance and in turn, operator revenue [15]. Furthermore, our solution should work with existing commodity switches and be backward compatible with legacy network stacks. Modifying switch hardware is especially problematic as a new switch ASIC typically takes years to design and implement.

4.2 BCC Mechanism

We model the switch as a shared-buffer output-queued switch. Variables and parameters used in the model are listed in Table 2 and 3. We start from the simplest assumption that each switch port only contains a single egress queue[5] and no buffer is reserved for each queue. Hence, all buffers are dynamically allocated from a single shared buffer pool. The switch has B (shared) buffer space and N egress queues in total. An ECN-based transport [5, 16, 20, 21] is enabled at the end host. The standard ECN setting has been configured on each port/queue to achieve 100% throughput.

Today's commodity switching chip typically use Dynamic Threshold (DT) algorithm [9] for dynamic buffer allocation. The shared buffer allocated to a queue is controlled by a parameter α. At time t, the MMU will compute a threshold

[5]In §4.2 and §4.3, we use queue and port interchangeably.

Parameter	Description
B	Switch shared buffer size
N	Total number of switch egress queues
C	Capacity of the switch queue
RTT	Base round-trip time
α	Parameter for shared buffer allocation
B_R	Minimum per-queue required buffer for high throughput and low packet loss rate
K_{min}	Minimum marking threshold for shared buffer ECN/RED
K_{max}	Maximum marking threshold for shared buffer ECN/RED
P_{max}	Maximum marking probability for shared buffer ECN/RED
h	See Equation 3

Table 2: Shared buffer model parameters

Variable	Description
t	Time
$Q_i(t)$	Length of switch queue i at time t
$T(t)$	Queue length control threshold at time t

Table 3: Shared buffer model variables

$T(t)$ to limit the queue length. $T(t)$ is actually a function of the unused shared buffer size and α as follows:

$$T(t) = \alpha \times (B - \sum_{i=1}^{N} Q_i(t)) \quad (1)$$

A packet arriving in queue i at time t will get dropped if $Q_i(t) \geq T(t)$. As analyzed in [9], if there are M active queues, each queue can eventually get $\alpha \times B/(1 + M \times \alpha)$ buffer space. The more active queues we have, the smaller buffer space each queue can get from the shared pool. α values are typically powers of two for hardware implementation simplicity (*e.g.*, 1/128 to 8 in Tomhawk).

We assume that our ECN-based transport protocol requires at least B_R buffer space per queue to achieve both high throughput and low packet loss rate. We simply treat B_R as a known constant here and show how to determine B_R later in §4.3. When $T(t) > B_R$, it means that the switch has sufficient buffer space to achieve both goals simultaneously. Hence, BCC just marks packets like the standard ECN configuration without degrading throughput.

When $T(t) \leq B_R$, it indicates that the shared buffer pool is highly utilized by many concurrently active ports. In such scenarios, only relying on standard ECN configuration may cause excessive packet losses as analyzed in §3.1. Hence, BCC throttles the shared buffer occupancy to avoid excessive packet losses. By Equation 1 and $T(t) \leq B_R$, we derive that

$$\sum_{i=1}^{N} Q_i(t) \geq B - B_R/\alpha \quad (2)$$

Here $\sum_{i=1}^{N} Q_i(t)$ is the occupancy of the shared buffer pool at time t, and B, B_R and α are all known parameters. This implies that, to prevent excessive packet losses, BCC should throttle the shared buffer occupancy from exceeding a static threshold $B - B_R/\alpha$.

To realize this, we leverage the shared buffer ECN/RED functionality which has been widely supported in commodity switching chips [4, 8]. Shared buffer ECN/RED follows the original RED algorithm [10] but tracks the occupancy of a shared buffer pool to mark packets. It can effectively control shared buffer occupancies. Moreover, shared buffer ECN/RED can be used in combination with other switch ECN configurations. When several ECN configurations coexist, a packet gets marked if anyone decides to mark it first.

Summary: BCC is built on top of existing ECN-based transports and per-port standard ECN configuration (current practice). It further enables shared buffer ECN/RED at the switch to achieve buffer-aware congestion control.

- When few ports are active, the shared buffer resource is abundant and per-port standard ECN configuration will take effect first to strike the balance of high throughput and low latency as before [5]. Both high throughput and low packet loss rate can be achieved.
- When more and more ports become congested, the shared buffer resource turns scarcer. Shared buffer ECN/RED will be automatically triggered first to prevent packet losses at the cost of sacrificing a small amount of bandwidth.

4.3 Parameter Selection

We now derive several parameters for BCC. First, we determine B_R, the minimum per-queue (port) buffer size for both high throughput and low packet loss rate. With B_R fixed, we then decide marking thresholds and probability of shared ECN/RED. Note that in this section we give several useful rules-of-thumb to set parameters while leaving optimal parameter settings for future work.

Determine B_R: Statistics has shown that there is typically a small number of concurrent large flows to the same receiver in DCNs [5]. Hence, we consider a simple scenario where several synchronized long-lived flows share a bottleneck link. $C \times RTT \times \lambda$ per port buffering is required for 100% throughput. Furthermore, the lag in ECN control loop imposes extra buffer requirement to avoid packet losses. When a packet gets ECN marked at switch egress[6], the sender will reduce its window after one RTT. During this RTT interval, extra buffer space is required to absorb the queue increase. We consider the most challenging slow start phase. As an ACK packet can trigger two MTU-sized data packets, the aggregate sending rate reaches $2C$ and the switch queue gradient is C. Therefore we need $C \times RTT$ extra buffer space to avoid packet losses and $C \times RTT \times (1 + \lambda)$ buffer space in total to achieve both goals. Through ns-2 simulations, we confirm that $C \times RTT \times (1 + \lambda)$ also works well for a mix of small and large flows. As C and λ are both known and RTT can be measured [13, 21] in production DCNs, operators can easily compute the value of B_R.

Determine parameters for shared buffer ECN/RED: We leverage shared buffer ECN/RED to prevent the shared buffer occupancy from exceeding $B - B_R/\alpha$. To achieve fast reaction to bursty traffic, we mark packets based on the instantaneous buffer occupancy. Shared buffer ECN/RED has 3 parameters to configure: minimum threshold K_{min}, maximum threshold K_{max} and maximum probability P_{max}. When the buffer occupancy is: 1) below K_{min}, no packet is marked; 2) between K_{min} and K_{max}, packets are marked according to a probability; 3) exceeds K_{max}, all packets get marked.

Inspired by DCTCP [5], our first choice is to set $K_{min} = K_{max} \leq B - B_R/\alpha$, in which only a single threshold is required. However, with such cut-off setting, all flows sharing a buffer pool are likely to reduce their window at the same time, resulting in global synchronization problem and a further loss of throughput [10].

Therefore, we decided to perform a probabilistic marking by setting $K_{min} < K_{max} = B - B_R/\alpha$. The key here is to control the range between K_{min} and K_{max}. A too small $K_{max} - K_{min}$ will make buffer occupancy regularly ramp up beyond K_{max}, still causing global synchronization and even packet losses. As original RED work [10] suggests, $K_{max} - K_{min}$ should be made sufficiently large (*e.g.*, larger than typical increase in the shared buffer occupancy during a RTT) to avoid global synchronization. Hence, the choice of $K_{max} - K_{min}$ depends on both the number of ports N and link capacity C. In BCC, we set K_{min} as follows:

$$K_{min} = B - B_R/\alpha - C \times N \times h \quad (3)$$

where h is a parameter to control $K_{max} - K_{min}$. In our evaluation, we set h to 8μs. For the maximum marking probability P_{max}, we set it to 10% according to [10].

4.4 Discussion

Impact of multiple MMUs: Each MMU has its own shared buffer ECN/RED without interfering with each other. Hence, BCC supports multi-MMU chips (*e.g.*, Tomahawk).

Impact of different α values: Operators may configure different α values for different queues for differentiated network services. In such scenarios, we can choose the minimum value α_{min} among them and update shared buffer ECN/RED parameters as follows: $K_{max} = B - B_R/\alpha_{min}$, $K_{min} = B - B_R/\alpha_{min} - C \times N \times h$.

Impact of static reserved buffers: When both static reserved buffers and dynamic shared buffers exist, the MMU first tries to use static reserved buffers. Therefore, we should reduce

[6]Modern shared buffer switches mark packets at egress side [22].

B_R to incorporate the static reserved buffer into BCC. Let S_{min} denote the minimum static buffer size reserved for a single queue. Our recommended value for B_R should become $C \times RTT \times (1+\lambda) - S_{min}$

5 EVALUATION

In this section, we present ns-2 simulations results.

Topology: We simulate a 128-host 100Gbps leaf-spine topology with 8 leaf switches and 8 spine switches. We use ECMP for load balancing. The base fabric RTT across the spine is $\sim$80μs. The BDP is 1MB. The jumbo frame is enabled.

Workload: We generate traffic according to the web search workload (see [5] for more details about the distribution). We adjust the flow arrival intervals to achieve the desired load in the network core.

Buffer: To emulate Tomahawk chip, we attach every 8 switch ports to a 3MB shared buffer pool. We set α to 4 for all ports. In addition, each port has 128KB static reserved buffer. We allocate 10MB buffer for each NIC at the host.

Schemes compared: We use DCTCP [5] and set RTOmin to 5ms. We compare the following three schemes:

- **DCTCP K=720KB**: This is a standard ECN configuration (current practice). We configure the per-port (queue) ECN/RED marking threshold to 720KB (0.72BDP based on measurement in §2.2) to achieve 100% throughput.
- **DCTCP K=200KB**: This is a conservative ECN configuration. We configure the per-port (queue) ECN/RED marking threshold to 200KB, which is smaller than average per-port switch buffer size (512KB), to reduce packet losses.
- **BCC**: BCC requires two ECN configurations at the switch. We set per-port (queue) ECN/RED marking threshold to 720KB like the standard ECN configuration. Since λ is 0.72 for DCTCP and the per-port static reserved buffer size S_{min} is 128KB, $B_R = C \times RTT \times (1+\lambda) - S_{min}$ $\approx$1.6MB. Therefore, K_{max} $\approx$2.6MB, $K_{min} = K_{max} - C \times N \times h$ $\approx$1.8MB and P_{max}=10%.

Performance metrics: We use flow completion time (FCT) as the performance metric and breakdown FCT results across small (0,100KB], medium (100KB,10MB] and large (10MB,∞) flows. Since the performance of many real-time applications depends on the slowest flow, we consider the 99th percentile FCT for small flows.

Result analysis: According to Figure 2, we have the following two key observations.

- At low loads, BCC performs similar as DCTCP K=720KB while generally outperforming DCTCP K=200KB, especially for medium and large flows. With the sufficient buffer resource, BCC can fully utilize the link capacity without triggering shared buffer ECN/RED. By contrast, DCTCP K=200KB still conservatively marks packets, thus significantly degrading throughput. Compared to DCTCP K=200KB. BCC achieves up to $\sim$13.5% (6362μs to 5503μs) lower average FCT for large flows. DCTCP K=200KB only shows some performance advantage ($\sim$100μs) on small flows, due to its lower switch queueing.
- At high loads, BCC generally outperforms the other two schemes. For small flows, BCC achieves up to 94.4% (5174μs to 291μs) lower 99th FCT compared to DCTCP K=720KB. This is because DCTCP K=720KB causes excessive packet losses due to the exorbitant shared buffer utilization. The packet loss rate with DCTCP K=720KB exceeds 0.3% at 90% load. This results in frequent TCP timeouts, which seriously increases FCT by at least 5ms (RTOmin). By contrast, at 90% load, the packet loss rate with BCC is lower than 0.08%.

  For large flows, BCC's performance is within $\sim$0.4-2.8% of the DCTCP K=720KB. This suggests that BCC only slightly degrades large flows. We think that the lower packet loss rate with BCC can make up for throughput loss to some degree. By contrast, DCTCP K=200KB is still so conservative that it increases FCT by at least $\sim$9% compared to DCTCP K=720KB.

In summary, BCC can operate based on the real-time shared buffer utilization, thus keeping good performance in various scenarios.

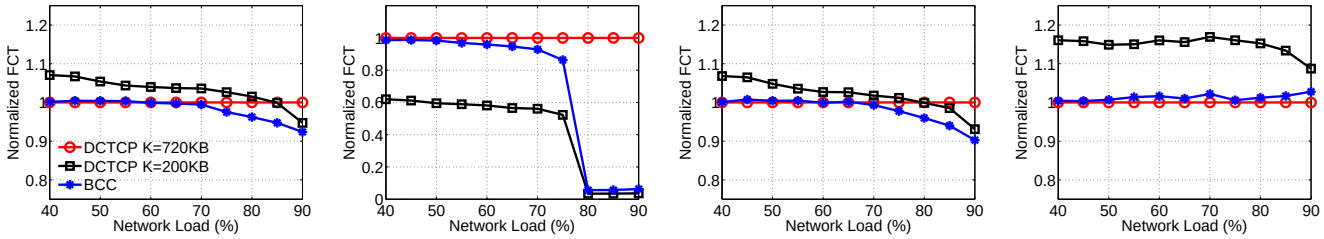
6 RELATED WORK

Bufferless Transports in DCNs: There are some bufferless transport designs in DCNs. But they may encounter various deployment challenges. PDQ [14] requires non-trivial modifications to switch hardware. Fastpass [18] and Flowtune [17] leverage a centralized scheduler, which is easy to suffer from failures and poor scalability. pHost [11] relies on the congestion free network core, which does not hold for many DCNs. By contrast, BCC is easy to deploy with only one more ECN configuration at commodity switches.

PFC: PFC (Priority-based Flow Control) [3] has been enabled in some DCNs to achieve lossless networks [12]. PFC needs to reserve enough buffer space as the headroom [12]. The size of the headroom is greatly affected the propagation delay. Deploying PFC in large-scale DCNs results in very large headroom size, which may not be affordable for commodity switches. Therefore, PFC is still limited in modest scale (*e.g.*. thousands of servers). Even so, commodity switches can still only support a small number (*e.g.*, 2) of lossless traffic classes [12]. Moreover, PFC may introduce deadlock problem, causing damage to the whole network [12].

7 CONCLUSION

In production DCNs, the increase of link speed significantly outpaces the increase of switch buffer size, resulting in an extremely shallow-buffered environment. Consequently, prior



(a) Overall: Avg (b) (0,100KB]: 99th percentile (c) (100KB,10MB]: Avg (d) (10MB,∞): Avg

Figure 2: [Simulation] FCT results for the web search workload. Results are normalized to values achieved by DCTCP K=720KB for clear comparison.

TCP/ECN solutions suffer from severe performance degradation. To address it, we introduced BCC, a simple yet effective solution with only one more shared buffer ECN/RED configuration at commodity switches. BCC maintains low packet loss rate persistently while only slightly degrading throughput when the buffer becomes insufficient. We demonstrated its superior performance using extensive simulations.

REFERENCES

[1] https://people.ucsc.edu/~warner/Bufs/7060CX.html. (????).

[2] https://people.ucsc.edu/~warner/Bufs/tomahawk. (????).

[3] IEEE DCB. 802.1Qbb - Priority-based Flow Control. http://www.ieee802.org/1/pages/802.1bb.html. (????).

[4] User Manual of Arista EOS version 4.15.0F. https://www.arista.com/assets/data/docs/Manuals/EOS-4.15.0F-Manual.pdf. (????).

[5] Mohammad Alizadeh, Albert Greenberg, David A. Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan. Data center TCP (DCTCP). In *SIGCOMM 2010*.

[6] Mohammad Alizadeh, Adel Javanmard, and Balaji Prabhakar. Analysis of DCTCP: stability, convergence, and fairness. In *SIGMETRICS 2011*.

[7] Guido Appenzeller, Isaac Keslassy, and Nick McKeown. Sizing Router Buffers. In *SIGCOMM 2004*.

[8] Wei Bai, Li Chen, Kai Chen, and Haitao Wu. Enabling ECN in Multi-Service Multi-Queue Data Centers. In *NSDI 2016*.

[9] Abhijit K. Choudhury and Ellen L. Hahne. 1998. Dynamic Queue Length Thresholds for Shared-memory Packet Switches. *IEEE/ACM Trans. Netw.* 6, 2 (April 1998), 130–140. https://doi.org/10.1109/90.664262

[10] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.* (????), 397–413.

[11] Peter X. Gao, Akshay Narayan, Gautam Kumar, Rachit Agarwal, Sylvia Ratnasamy, and Scott Shenker. pHost: Distributed Near-optimal Datacenter Transport over Commodity Network Fabric. In *CoNEXT 2015*.

[12] Chuanxiong Guo, Haitao Wu, Zhong Deng, Gaurav Soni, Jianxi Ye, Jitu Padhye, and Marina Lipshteyn. RDMA over Commodity Ethernet at Scale. In *SIGCOMM 2016*.

[13] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, Zhi-Wei Lin, and Varugis Kurien. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. In *SIGCOMM 2015*.

[14] Chi-Yao Hong, Matthew Caesar, and P Godfrey. Finishing flows quickly with preemptive scheduling. In *SIGCOMM 2012*.

[15] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, Emily Blem, Hassan Wassel, Monia Ghobadi, Amin Vahdat, Yaogong Wang, David Wetherall, and David Zats. TIMELY: RTT-based Congestion Control for the Datacenter. In *SIGCOMM 2015*.

[16] Ali Munir, Ihsan A Qazi, Zartash A Uzmi, Aisha Mushtaq, Saad N Ismail, M Safdar Iqbal, and Basma Khan. Minimizing flow completion times in data centers. In *INFOCOM 2013*.

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

[18] Jonathan Perry, Amy Ousterhout, Hari Balakrishnan, Deverat Shah, and Hans Fugal. Fastpass: A Centralized "Zero-queue" Datacenter Network. In *SIGCOMM 2014*.

[19] K Ramakrishnan, Sally Floyd, David Black, et al. 2001. RFC 3168: The addition of explicit congestion notification (ECN) to IP. (2001).

[20] Balajee Vamanan, Jahangir Hasan, and TN Vijaykumar. Deadline-aware datacenter tcp (d2tcp). In *SIGCOMM 2012*.

[21] Haitao Wu, Jiabo Ju, Guohan Lu, Chuanxiong Guo, Yongqiang Xiong, and Yongguang Zhang. Tuning ECN for data center networks. In *CoNEXT 2012*.

[22] Yibo Zhu, Monia Ghobadi, Vishal Misra, and Jitendra Padhye. ECN or Delay: Lessons Learnt from Analysis of DCQCN and TIMELY. In *CoNEXT 2016*.