Improving ECN Marking Scheme with Micro-burst Traffic in Data Center Networks

Danfeng Shan, Fengyuan Ren
Department of Computer Science and Technology, Tsinghua University, Beijing, China
Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing, China
E-mail: sdf13@mails.tsinghua.edu.cn, renfy@tsinghua.edu.cn

***Abstract*—In data centers, micro-burst is a common traffic pattern. The packet dropping caused by it usually leads to serious performance degradations. Therefore, much attention has been paid to avoiding *buffer overflow* caused by micro-burst traffic. In particular, ECN is widely used in data centers to keep persistent queue occupancy low, so that enough buffer space can be available as headroom to absorb micro-burst traffic. However, we find that instantaneous-queue-length-based ECN may cause problems in another direction — *buffer underflow*. Specifically, current ECN marking scheme in data centers is easy to trigger spurious congestion signals, which may result in overreaction of senders and queue length oscillations in switches. Since ECN threshold is low, the buffer may underflow and link capacity is not fully used.**

In this paper, we reveal this problem by experiments and simulations. Besides, we theoretically deduce the amplitude of queue length oscillations. The analysis result shows that overreaction of senders is caused by ECN mis-marking. Therefore, we propose Combined Enqueue and Dequeue Marking (CEDM), which can mark packets more accurately. Through simulations, we show that CEDM can greatly reduce throughput loss and improve flow completion time.

***Index Terms*—ECN marking, Micro-burst traffic, Large Segment Offload, Interrupt Coalescing.**

I. Introduction

Micro-burst is not uncommon in data centers. Generally, it refers to highly intense traffic appearing at a relatively short timescale. Recently, micro-burst has attracted much attention [1]–[9], because data center switches are usually shallow-buffered, and thus are prone to be overwhelmed by micro-burst traffic; the consequent packet dropping may trigger timeouts, which causes throughput collapse [10]–[12], and extends flow completion time [13], [14].

Fortunately, in recent transport proposals [4], [13]–[16], persistent queue occupancy is kept low, and thus the majority of buffer space can be available as headroom to absorb micro-burst traffic. This is achieved by a powerful tool called ECN, which can explicitly notify senders of congestion. Specifically, when the instantaneous queue length is higher than ECN threshold, packets are marked with ECN so that senders will slow down. Due to the effectiveness of ECN, it has been widely adopted by industry communities [17], [18].

However, although much attention has been paid to preventing buffer *overflow* caused by micro-burst traffic, we find that buffer can *underflow* using current data center ECN marking schemes. Specifically, as link rates in data centers have increased from 1Gbps to 10Gbps, and will increase to 40Gbps/100Gbps [18]–[20] in the near future, many batching schemes, such as Interrupt Coalescing (IC), have been used to reduce the CPU overhead, introducing bursty traffic into network. On the other hand, in all data center transport proposals using ECN, *instantaneous queue length* is used to make marking decision to quickly respond to fan-in traffic [13], regardless of whether traffic is bursty or not. As a result, the micro-burst traffic stemming from batching schemes may trigger spurious congestion signals, which not only does not help mitigate these kinds of micro-burst traffic, but also leads to overreaction of senders and consequent queue length oscillations in the switch buffer. What's worse, since ECN threshold in data centers is usually very low to reduce queueing latency [13], [15], [21], the buffer could underflow, leading to loss of throughput.

In this paper, we study the queue length oscillations caused by instantaneous-queue-length-based ECN marking scheme and batching-scheme-induced micro-burst traffic through both experiments and simulations (§II). We find that amplitude of queue length oscillations with batching is three times larger than that without batching, and the queue length oscillations may result in 14.3% of throughput loss.

Along with experiments and simulations, we also theoretically deduce the amplitude of queue length oscillations with batching scheme (§III). In particular, with DCTCP protocol, the amplitude of queue length oscillations increases from $O(\sqrt{BDP})$ without batching schemes to $O(BDP)$ with batching schemes, where BDP denotes Bandwidth Delay Product. We also quantitatively estimate ECN threshold settings to avoid throughput loss. The results show that the ECN threshold with batching scheme should be 50% (ECN*) or 61.6% (DCTCP) larger than that without batching scheme to fully utilize the link capacity.

Through analysis, we find that the overreaction of senders is caused by ECN mis-marking. Specifically, transient queue occupancy caused by multiplexing of micro-burst traffic should be excluded when making marking decision. Therefore, we propose *Combined Enqueue and Dequeue Marking (CEDM)* scheme (§IV) to make ECN marking more accurately. Similar to traditional marking scheme, CEDM marks packets based on instantaneous queue length to preserve the responsiveness to fan-in traffic. Different from traditional marking scheme, CEDM tries to identify and exclude transient queue occupancy through characteristics of queue length evolutions. The charac-

teristics are extracted from the slope of queue length evolution, and the queue lengths of two time points — packet enqueue and packet dequeue.

We implement and evaluate CEDM on ns-2 platform [22] (§V). The results show that the throughput loss is reduced by $1.6\times$ with ECN* and over $6\times$ with DCTCP. We also evaluate CEDM through large-scale simulations in a 10/40Gbps network with 144 hosts. We show that the average and 99th flow completion time of small flows can be reduced by $\sim$9.2-14.5% and $\sim$17.5-20.8%, respectively.

II. Background and Motivation

A. Sources of Micro-bursts

Micro-burst traffic stems from various elements. In this part, we summary these sources as follows:

1) TCP: Traffic burstiness can be introduced by TCP by a number of factors, including ACK compression [23], slow start [24], [25], ACK reordering [26], etc. Traffic burstiness induced by TCP can be effectively mitigated by TCP pacing [23], [24], [27], which has been studied a lot in the literature and thus is not our focus in this paper.

2) System Batching: To reduce CPU overhead in high-speed data center networks, lots of batching schemes have been introduced. We summarize these schemes as follows.

a) Interrupt Coalescing (IC): Using IC, the network interface is able to raise an interrupt for multiple packets. For example, in GNU/Linux, when a packet is received, raising of RX interrupt can be delayed for several microseconds. As a result, several packets can be received and acknowledged together. Consequently, the sender will introduce a bunch of packets into network. Using the default settings in GNU/Linux, the burst length generated by IC is about 3 packets in 10Gbps network.

b) Large Segment Offload (LSO): When enabling LSO, the transport layer can send out large virtual packets (up to 64KB in GNU/Linux), which are segmented into MTU-sized packets before entering into network. There are two techniques: Generic Segment Offload (GSO) and TCP Segment Offload (TSO). GSO is a software solution, while TSO is a hardware solution. Specifically, with GSO packets are segmented just before they are sent to NIC, while with TSO packets are segmented at NIC. Besides, TSO is only applied to TCP, while GSO is not limited to TCP.

Before segmentation, a number of packets need to be coalesced at transport layer. Since TCP uses sliding window protocols, without LSO only one (or two if send window increases) data packet is allowed to be transmitted when receiving an ACK. When LSO is enabled, transmission of the packet is deferred. In GNU/Linux, the transmission will always be deferred until some conditions are satisfied. These conditions are listed in TABLE I. Among these conditions,

TABLE I
Conditions of Stopping Deferring Transmission in GNU/Linux

#.	Condition	Rational
1	Data allowed to be sent is larger than a percentage (1/3 by default) of TCP send window.	A single LSO segment should not consume congestion window too much.
2	Data allowed to be sent is larger than maximum TSO size.	Reach max segment size.
3	`skb` is in the middle of write queue.	Delay sending does not help get more data.
4	The flow is not in `TCP_CA_open` state.	There may be congestion in the network.
5	There is FIN flag.	Delay sending may not help get more data.
6	SKB has delayed for over 2 clock ticks (2ms in general).	Packets should not be delayed too long.

TABLE II
Frequency of each condition's satisfaction

RTT	Cond. 1	Cond. 2	Others
$\sim 50\mu s$	96.8%	0%	3.2%
[300μs, 800μs]	2.5%	96.7%	0.7%

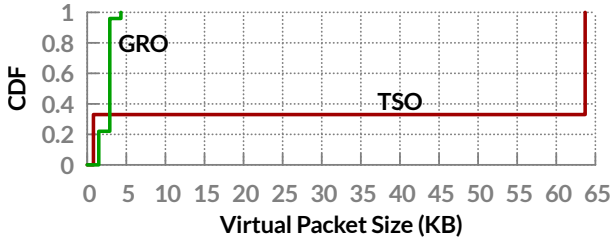


Fig. 1. An evaluation of virtual packet size. In the experiment, two hosts are connected to a commodity switch (Cisco Catalyst 2960 Plus Series). We start one-to-one flow using `iperf`. Virtual packets are captured through `tcpdump`. GRO (Generic Receive Offload) is a technique of Receive Side Offload in GNU/Linux.

Condition 1 and Condition 2 are easier to be satisfied, as is verified in our experiments[1](TABLE II).

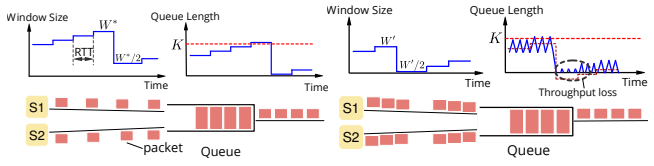
In summary, the burst length introduced by LSO is about $\min(L, \beta W)$, where L is maximum LSO size, W is the TCP window size and β denotes the fraction of congestion window that can be consumed by a single LSO segment. By default, $\beta = 1/3$ and $L = 64$KB in GNU/Linux.

c) Receive Side Offload: As opposed to LSO, there are also offload techniques at receive side, with which multiple received packets from the same flow can be merged into a large virtual packet. As a result, multiple packets are acknowledged together, which will result in bursty transmission from sender. The burst length introduced by Receive Side Offload is usually less than 5 packets in general, as shown in Fig. 1.

d) Jumbo Frame: Jumbo frames are Ethernet frames with more than 1500 bytes of payload. A single jumbo frame can carry up to 9000 bytes of payload [28], which result in traffic burstiness in the network.

Remarks: Although various batching schemes co-exist in the data centers, the traffic burstiness introduced by LSO is much more serious than other batching schemes. Therefore, in the following we mainly consider LSO.

[1]In the experiment, two hosts are connected to a NetFPGA-based switch (RTT $\approx 50\mu s$) or a commodity switch (RTT in [300μs, 800μs]), respectively. To get the frequency of a condition's satisfaction, we modify the kernel so that a message is printed whenever a condition is satisfied.



(a) Uniform Traffic (b) Bursty Traffic

Fig. 2. Motivation example

B. ECN Marking in Data Centers

ECN [29] is an effective tool, with which switches can explicitly notify senders of congestion. Historically, ECN is widely used by Active Queue Management (AQM) schemes [30], in which packets are marked according to average queue length.

In data centers, ECN marking [29] is also widely used by recent transport proposals [4], [13]–[16]. In all these protocols, packets are marked with ECN if the queue length is larger than an ECN threshold. Different from ECN in AQM, *instantaneous queue length* is used rather than average queue length in order to quickly notify senders of congestion [13].

What's more, ECN threshold is usually *very low* to reduce the queuing latency. Specifically, the ECN threshold (denoted by K) is set as follows:

$$K = \lambda \times C \times RTT \tag{1}$$

where C is link capacity and RTT is round-trip time. In particular, in DCTCP [21], $\lambda = 0.17$; in ECN* [15][2], $\lambda = 1$.

C. Problems of ECN Marking with Micro-burst Traffic

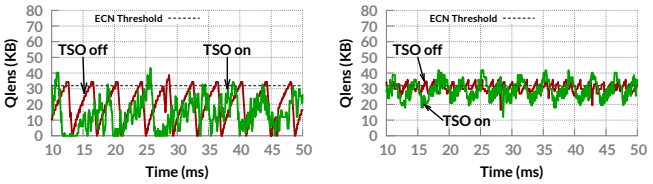
When there is micro-burst traffic induced by batching schemes in the network, current ECN marking scheme based on instantaneous queue length may lead to spurious congestion signals, which intensify queue length oscillations. As the ECN threshold is low, this may cause throughput loss. In the next, we use examples and experiments/simulations to show this.

1) An example: Consider two long-lived synchronized flows, who are with identical round-trip time (RTT) and sharing the same bottleneck. Both of them are in congestion avoidance phase. In the bottleneck queue, the ECN Threshold is K. In this scenario, the TCP window in each sender will increase by one for each RTT, as shown in Fig. 2.

First, if the packets are evenly spread across the entire RTT (Fig. 2(a)), then the queue length will increase by two (packets) for each RTT. We assume that the queue length reaches the ECN threshold K when the TCP window is W^*. Then after senders receive congestion signals, they will cut their windows in half. Accordingly, the queue length will decrease to Q_{min}. To achieve low latency while fully utilize the link capacity, the ECN threshold is usually set as low as possible on premise of $Q_{min} > 0$.

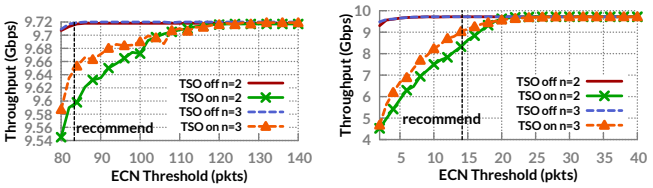
On the other hand, if there are batching schemes at senders, the arrival traffic to the bottleneck queue will be bursty (Fig.2(b)), and the queue length will be oscillating at sub-RTT level. As a result, the queue length may reach the ECN threshold when the TCP window is W' ($W' < W^*$). Then after senders cut their windows in half, the queue length will become Q'_{min}. Then we have $Q'_{min} < Q_{min}$. In other words, the amplitude of queue length oscillations becomes larger with batching schemes at sender. If the ECN threshold is so low that $Q_{min} \approx 0$, the queue could be empty and link is not fully utilized.

[2]In ECN*, standard ECN [29] is used at both TCP sender and receiver, and ECN marking in switch is based on instant queue length.



(a) ECN* (b) DCTCP

Fig. 3. Queue length evolution in our experiments. ECN threshold is set to 32KB. IC and Receive Side Offload are disabled.



(a) ECN* (b) DCTCP

Fig. 4. Throughput with different ECN thresholds in 10Gbps network.

2) Experiments: We conduct some experiments and simulations to show this. In our experiments, four hosts are connected to the same switch. Each host is a Dell Optiplex 780 desktop with an Intel® Core™ 2 Duo E7500 2930 MHz CPU, 4 GB memory, a 500GB hard disk, and an Intel® 82567LM Gigabit Ethernet NIC, running CentOS 5.11 with GNU/Linux kernel 2.6.38. The switch is a NetFPGA card with four Gigabit Ethernet networking ports. The base RTT is $\sim$50μs.

In our experiments, we let two flows from two hosts keep sending data to the same destination.

First, we try to show the queue length oscillations by examining the queue length evolution. To get fine-grained queue length in the switch, we let each packet carry the queue length information in its payload whenever it passes through the switch. Fig. 3 shows the queue length evolution in the bottleneck queue. With ECN* (Fig. 3(a)), the buffer can underflow (e.g., at 10.15ms) when TSO is turned on. With DCTCP (Fig. 3(b)), although queue length does not reach zero, the queue oscillations become much severer when TSO is turned on. Specifically, when TSO is turned off, the minimum queue length is only 3.65KB lower than ECN threshold. In comparison, when TSO is turned on, the minimum queue length is 11.11KB lower than ECN threshold.

3) Simulations: In the experiment, the throughput loss is not very serious. This is because in our testbed bandwidth delay product (BDP) is too small ($1\text{Gbps} \times 50\mu s = 4.2$ packets). The throughput is very high even when ECN threshold is 0. While in 10Gbps or 40Gbps networks, where BDP is much larger, the problem can be more serious. In order to show this, we implement TSO mechanism on ns-2 platform [22] following GNU/Linux's implementation. We use the same

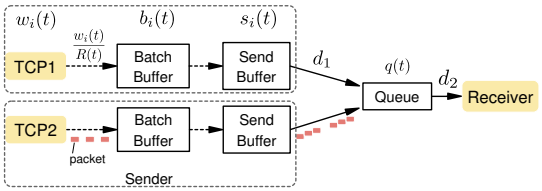
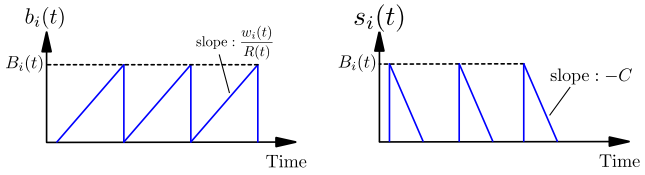


Fig. 5. Batching Structure



(a) batch buffer (b) send buffer

Fig. 6. Evolution of the number of packets in batch buffer and send buffer

settings as those in experiments, except that link capacity is 10Gbps and RTT is 100μs. Fig. 4 shows the throughput under different ECN threshold settings. When TSO is turned on, using the recommended ECN threshold settings, there are 127Mbps (Fig. 4(a)) and 1.391Gbps (Fig. 4(b)) throughput loss with ECN* and DCTCP, respectively.

III. Steady State Analysis

In this section, we theoretically analyze the effect of micro-burst traffic caused by system batching on queue length oscillations in data center networks. ECN* [15] and DCTCP [13] are considered because they have been officially supported by various OS kernels and thus are typical. We summarize main results as theorems and corollaries to improve readability.

During analysis, we consider N long-lived flows with identical round-trip times, who are sharing the same bottleneck. We assume that these flows are synchronized. This assumption is the same as that in [13]; as is mentioned in [13], it is realistic when N is small, which is true in data centers. We further assume that NIC speed is the same as the capacity of bottleneck link, which is denoted by C. The key notions are summarized in TABLE III for the sake of terseness.

To make the analysis easier to understand, we describe batching schemes using the structure shown in Fig. 5. In each sender, the transport layer is sending packets at a constant rate of $w_i(t)/R(t)$. These packets are accumulated at a batch buffer. The batch buffer won't deliver packets until the number of packets in it (denoted by $b_i(t)$) reaches the batch size $B_i(t)$. After that, the batch buffer will send all packets to the send buffer, where packets are sent out at a rate of NIC speed. In this way, packets are uniformly delivered by transport layer and injected into network in batches. The evolution of the number of packets in batch buffer and send buffer is shown in Fig. 6. Note that batch schemes at receive side can also be transformed to similar structure, where the batch buffer and send buffer should be at receive side. We do not separately consider receive side batching because results are similar as those at send side.

In this structure, the number of packets in the batch buffer or send buffer can be an arbitrary value in $[0, B_i(t)]$. As a result,

TABLE III
Key Notations in Our Analysis

Not.	Description
C	Link capacity
d	Base RTT (RTT without queueing)
d_1	Delay between sender and bottleneck queue
d_2	Delay between bottleneck queue and receiver
$w_i(t)$	Window size of sender i at time t
$R(t)$	RTT at time t
$b_i(t)$	The number of packets in the batch buffer in sender i at time t
$B_i(t)$	Batch size of sender i at time t
$s_i(t)$	The number of packets in the send buffer in sender i at time t
$q(t)$	The number of packets in the bottleneck queue at time t
K	ECN threshold
A	Amplitude of queue length oscillations
L	Maximum LSO size
β	Fraction of window that can be consumed by an LSO segment

the queue occupancy in the bottleneck queue also differs. The minimum and maximum queue length can be given by following lemmas:

Lemma 1 (Minimum Queue Length)**.** *At time t, the minimum bottleneck queue length can be given by*

$$q_{\min}(t) = \sum_{i=1}^{N} w_i(t) - \sum_{i=1}^{N} B_i(t) - Cd \tag{2}$$

Lemma 2 (Maximum Queue Length)**.** *At time t, the maximum bottleneck queue length can be given by*

$$q_{\max}(t) = \sum_{i=1}^{N} w_i(t) - \frac{\sum_{i=1}^{N} [B_i(t-\tau_i)w_i(t)]}{CR(t)} - Cd \tag{3}$$

where $\tau_i = \frac{B_i(t-\tau_i)}{C} + d_1$.

Proof of Lemma 1. Since the arrival traffic at bottleneck queue exhibits ON/OFF pattern, the queue length reaches its minimum value when arrival traffic from all sources is at the end of OFF period. Specifically, the batch buffer has delivered a batch of packets to the send buffer and the first packet in this batch is just arriving at the queue. In other words, it's d_1 after batch buffer delivers the batch to the send buffer. On the other hand, the number of packets sent out by transport layer is $\sum_i w_i(t)$. These packets are distributed in: ① batch buffer, ② send buffer, ③ flying between sender and bottleneck queue, ④ bottleneck queue, ⑤ flying between bottleneck queue and receiver, ⑥ arrived at receiver but the sender has not received their acknowledgements. Since $\sum_i \frac{w_i(t)}{R(t)} = C$, the number of packets in ① is $\sum_i b_i(t) = Cd_1$. The total number of packets in ② and ③ is $\sum_i B_i(t)$. The number of packets in ④, ⑤ and ⑥ is $q(t)$, Cd_2, and $C(d_1+d_2)$, respectively. Combining $d = 2(d_1+d_2)$, we have

$$\sum_{i=1}^{N} w_i(t) = q(t) + \sum_{i=1}^{N} B_i(t) + Cd \tag{4}$$

This is equivalent to equation (2). ■

Proof of Lemma 2. Similarly, since the arrival traffic at bottleneck queue exhibits ON/OFF pattern, the queue length reaches its maximum value when arrival traffic from all sources is at the end of ON period. This moment is d_1 after the send buffer

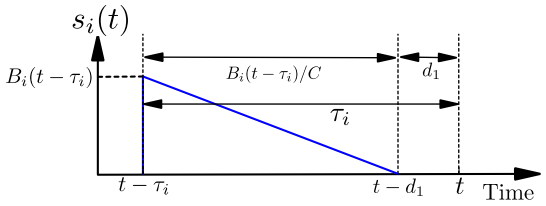


Fig. 7. The number of packets in send buffer before the bottleneck queue length reaches its maximum value

sent out all packets. During this time, evolution of $s_i(t)$ is shown in Fig. 7. Therefore, the number of accumulated packets in batch buffer i can be given by $\frac{w_i(t)}{R(t)} \cdot \left[\frac{B_i(t-\tau_i)}{C} + d_1\right]$. The total number of packets in batch buffer can be given by $\frac{\sum_{i=1}^{N}[B_i(t-\tau)w_i(t)]}{CR(t)} + Cd_1$. On the other hand, the total number of packets in other places can be given by $q(t) + 2Cd_2 + Cd_1$. Thus, we have

$$\sum_{i=1}^{N} w_i(t) = \frac{\sum_{i=1}^{N}[B_i(t-\tau)w_i(t)]}{CR(t)} + q(t) + Cd \quad (5)$$

This is equivalent to equation (3). ■

In our analysis, equation (2) and equation (3) can be further simplified, as we have assumed that flows are synchronized, namely, $w_1(t) = w_2(t) = \cdots = w_N(t) = w(t)$ and $B_1(t) = B_2(t) = \cdots = B_N(t) = B(t)$. We further assume that $B_i(t-\tau_i) = B_i(t)$, which is usually true as τ_i is in sub-RTT level and batch size usually does not change so frequently. Therefore, we can simplify equation (2) and equation (3) into

$$\begin{cases} q_{\min} = Nw(t) - NB(t) - Cd \\ q_{\max} = Nw(t) - B(t) - Cd \end{cases} \quad (6)$$

A. Analysis of ECN*

Theorem 1. *With ECN*, if the window size when packets are marked is $w(t)$, and it takes Δt for the senders to receive congestion signal after packets are marked with ECN. Then the amplitude of queue length oscillations can be given by*

$$A = \frac{N}{2}w(t) + NB(t+\Delta t) - B(t) \quad (7)$$

Proof. In ECN*, standard ECN [29] is used at end servers, i.e., the TCP sender will cut its window in half when an ACK with ECE bit is received. Therefore, if the window size when packets are marked is $w(t)$, and the sender receives congestion signal after Δt, then $w(t+\Delta t) = \frac{w(t)}{2}$. Therefore, the amplitude of queue length oscillations can be given by

$$\begin{aligned} A &= q_{\max}(t) - q_{\min}(t+\Delta t) \\ &= \frac{N}{2}w(t) + NB(t+\Delta t) - B(t) \end{aligned} \quad (8)$$

■

Corollary 1. *With ECN*, if $\beta w(t) < L$, the amplitude of queue length oscillations with LSO can be given by*

$$\frac{(\beta+1)N - 2\beta}{2(N-\beta)}(Cd+K) \leqslant A \leqslant \frac{(\beta+1)N-2\beta}{2N(1-\beta)}(Cd+K) \quad (9)$$

Proof. When LSO is enabled and $\beta w(t) < L$, $B(t) = \beta w(t)$ and $B(t+\Delta t) = \frac{\beta}{2}w(t)$. Substituting them into equation (7), we get

$$A = \frac{(\beta+1)N - 2\beta}{2}w(t) \quad (10)$$

On the other hand, $q_{\max} \leqslant K \leqslant q_{\max}$, from which we have

$$\frac{Cd+K}{N-\beta} \leqslant w(t) \leqslant \frac{Cd+K}{N(1-\beta)} \quad (11)$$

Substituting inequality (11) into equation (10), we can get inequality (9). ■

Remarks: Following the same way, we can get the amplitude of queue length oscillation without batching, which is $\frac{Cd+K}{2}$. Compared to it, *the amplitude of queue length oscillation can be increased by 25%* in the worst case when $N = 2$, $\beta = \frac{1}{3}$.

Corollary 2. *With ECN*, when LSO is enabled and $\beta w(t) < L$, to avoid throughput loss, the ECN threshold K should meet the following inequality*

$$K \geqslant \frac{(1+\beta)N - 2\beta}{(1-\beta)N}Cd \quad (12)$$

Proof. To avoid throughput loss, the queue should not underflow, namely,

$$q_{\min}(t+\Delta t) = \frac{N}{2}(1-\beta)w(t) - Cd \geqslant 0 \quad (13)$$

Combining equation (11), we must have

$$\frac{N}{2}(1-\beta)\frac{Cd+K}{N-\beta} \geqslant Cd \quad (14)$$

This inequality is equivalent to inequality (12). ■

Remarks: When $N = 2$, $\beta = \frac{1}{3}$, inequality (12) can be rewritten as $K \geqslant \frac{3}{2}Cd$. This result fits quite well with the simulation result in Fig. 4(a), where $Cd = 83.3$ packets and thus $\frac{3}{2}Cd = 125$ packets. In comparison, the ECN threshold without batching should be set as $K \geqslant Cd$ [15]. In other words, to achieve 100% throughput, *the ECN threshold when LSO is enabled should be increased by 50%.*

B. Analysis of DCTCP

Theorem 2. *With DCTCP, if the window size when packets are marked is $w(t)$, and it takes Δt for the senders to receive congestion signal after packets are marked with ECN. Then the amplitude of queue length oscillations can be given by*

$$A = \frac{\alpha}{2}Nw(t) + NB(t+\Delta t) - B(t) \quad (15)$$

Proof. Different than ECN*, with DCTCP, the sender will update its window as $w = w \times (1 - \frac{\alpha}{2})$ when receiving an ACK with ECE bit, where α denotes the fraction of marked packets. Therefore, if the window size when packets are marked is $w(t)$, and the sender receive congestion signal after Δt, then we have $w(t+\Delta t) = (1 - \frac{1}{\alpha})w(t)$. The amplitude of queue length oscillations can be given by

$$\begin{aligned} A &= q_{\max}(t) - q_{\min}(t+\Delta t) \\ &= \frac{\alpha}{2}Nw(t) + NB(t+\Delta t) - B(t) \end{aligned} \quad (16)$$

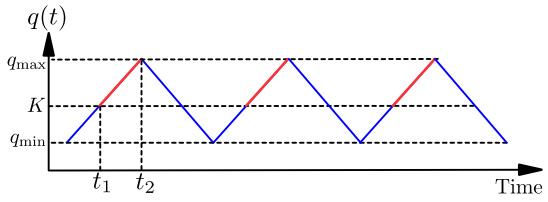


Fig. 8. Queue length evolution of DCTCP with batching

■

Corollary 3. *With DCTCP, if $\beta w(t) < L$, the amplitude of queue length oscillations with LSO can be given by*

$$\frac{\beta(N-1)}{N-\beta}(Cd+K) \leqslant A \leqslant \frac{\beta(N-1)}{N(1-\beta)}(Cd+K) + N\sqrt{\frac{(1-\beta)(Cd+K)}{2}} \quad (17)$$

Proof. If $\beta w(t) < L$, $B(t) = \beta w(t)$ and $B(t+\Delta t) = (1-\frac{\alpha}{2})\beta w(t)$. Substituting them into equation (15), we have

$$A = \left[\frac{\alpha}{2}N(1-\beta) + (N-1)\beta\right] w(t) \quad (18)$$

To further expand this equation, we need to estimate α. The queue length evolution when packets are marked is shown in Fig. 8. In a single batch, packets arriving between t_1 and t_2 are marked ECN. The number of packets arriving during this time is $\frac{q_{max}-K}{(N-1)C} \cdot NC$. The sender will receive congestion signal about an RTT later. During an RTT, there are $1/\beta$ batches. Therefore, the number of total marked packets can be given by $\frac{(q_{max}-K)NC}{(N-1)\beta C}$. Let $S(W_1, W_2)$ denote the number of sent packets, then (see [13] for more details), then we have $S(W_1, W_2) = \frac{W_2^2 - W_1^2}{2}$. Thus, we can estimate α by

$$\alpha = \frac{\frac{(q_{max}-K)NC}{(N-1)\beta C}}{S\left(w(t)(1-\frac{\alpha}{2}), w(t)\right)} \quad (19)$$

Substituting equation (6) into it, we have

$$\alpha^2 - \frac{1}{4}\alpha^3 = \frac{2(Nw(t) - \beta w(t) - Cd - K) \cdot \frac{N}{N-1} \cdot \frac{1}{\beta}}{w^2(t)} \quad (20)$$

Because α is very small [13], [21], this equation can be simplified as

$$\alpha \approx \frac{1}{w(t)}\sqrt{\frac{2N(Nw(t) - \beta w(t) - Cd - K)}{(N-1)\beta}} \quad (21)$$

Plugging it into equation (18), we get

$$A = N(1-\beta)\sqrt{\frac{N}{2(N-1)\beta}}\sqrt{(N-\beta)w(t) - (Cd+K)} + (N-1)\beta w(t) \quad (22)$$

Finally, plugging (11) into it gives the desired result. ■

Remarks: Corollary 3 reveals an important result of LSO: *with LSO, the amplitude of queue length oscillations of DCTCP is $O(Cd+K)$, which is much larger than $O(\sqrt{Cd+K})$ oscillations of that without batching* (The amplitude is $\frac{1}{2}\sqrt{2N(Cd+K)}$ without batching [13]). Thus, to avoid loss of throughput, ECN threshold should be increased a lot. Particularly, we have the following corollary for setting ECN threshold:

Corollary 4. *With DCTCP, when LSO is enabled and $\beta w(t) < L$, to avoid throughput loss, the ECN threshold K should meet the following inequalities*

$$\begin{cases} K \geqslant \frac{\beta(N-1)}{N(1-\beta)}Cd + \frac{N(N-\beta)^2}{8(N-1)\beta} \\ \quad \text{if } \left[\frac{N(N-\beta)}{\beta(N-1)}\right]^2 \frac{1-\beta}{8} \leqslant Cd+K \\ K \geqslant \frac{N^2(1-\beta)+N\sqrt{N^2(1-\beta)^2+8Cd(1-\beta)}}{4} \\ \quad \text{if } \left[\frac{N(N-\beta)}{\beta(N-1)}\right]^2 \frac{1-\beta}{8} > Cd+K \end{cases} \quad (23)$$

Proof. To avoid throughput loss, the queue should not underflow, namely,

$$q_{\min}(t+\Delta t) = N(1-\frac{\alpha}{2})w(t) - N\beta(1-\frac{\alpha}{2})w(t) - Cd \geqslant 0 \quad (24)$$

Substituting equation (21) into it, we have

$$w(t) - \sqrt{\frac{N}{2(N-1)\beta}} \cdot \sqrt{(N-\beta)w(t) - (Cd+K)} \geqslant \frac{Cd}{N(1-\beta)} \quad (25)$$

Let $f(w) = w - \sqrt{\frac{N}{2(N-1)\beta}} \cdot \sqrt{(N-\beta)w + (Cd+K)}$. Then $f(w)$ reaches its global minimum value when

$$w^* = \frac{Cd+K}{N-\beta} + \frac{N(N-\beta)}{8(N-1)\beta} \quad (26)$$

However, since $\frac{Cd+K}{N-\beta} \leqslant w(t) \leqslant \frac{Cd+K}{N(1-\beta)}$, there are two cases:

(1). $\left[\frac{N(N-\beta)}{\beta(N-1)}\right]^2 \frac{1-\beta}{8} \leqslant Cd+K$

In this case, $w^* \leqslant \frac{Cd+K}{N(1-\beta)}$, and $f(w)$ reaches its minimum value when $w = w^*$. Therefore,

$$f(w)_{\min} = f(w^*) = \frac{Cd+K}{N-\beta} - \frac{N(N-\beta)}{8(N-1)\beta} \quad (27)$$

To avoid throughput loss, we must have $f(w)_{\min} \geqslant \frac{Cd}{N(1-\beta)}$. Substituting (27) into it, we get

$$K \geqslant \frac{\beta(N-1)}{N(1-\beta)}Cd + \frac{N(N-\beta)^2}{8(N-1)\beta} \quad (28)$$

(2). $\left[\frac{N(N-\beta)}{\beta(N-1)}\right]^2 \frac{1-\beta}{8} > Cd+K$

In this case, $w^* > \frac{Cd+K}{N(1-\beta)}$. Therefore,

$$f(w)_{\min} = f\left(\frac{Cd+K}{N(1-\beta)}\right) = \frac{Cd+K}{N(1-\beta)} - \sqrt{\frac{Cd+K}{2(1-\beta)}} \quad (29)$$

To avoid throughput loss, we must have $f(w)_{\min} \geqslant \frac{Cd}{N(1-\beta)}$. Substituting (27) into it, we get

$$K \geqslant \frac{N^2(1-\beta)+N\sqrt{N^2(1-\beta)^2+8Cd(1-\beta)}}{4} \quad (30)$$

■

Remarks: When $N = 2$, $\beta = \frac{1}{3}$, we have $K \geqslant \frac{1}{4}Cd + \frac{25}{12}$ if $Cd + K \geqslant 8.3$ packets. This result fits quite well with the simulation result in Fig. 4(b), where $Cd = 83.3$ packets and thus $\frac{1}{4}Cd + \frac{25}{12} = 22.9$ packets. In comparison, the ECN threshold without batching could be set as $K \geqslant 0.17Cd = 16.7$ packets [21]. Therefore, *with LSO, the ECN threshold should be 61.6% larger than that without batching.*

IV. CEDM Scheme

Our analysis shows that micro-burst traffic greatly leads to queue length oscillations when instantaneous-queue-length-based ECN marking scheme is used. In this section, we propose *Combined Enqueue and Dequeue ECN Marking (CEDM)* scheme, which can mark packets more accurately.

A. Existing Solutions

Before proposing our approaches, we discuss several possible solutions:

(1) Eliminating micro-burst by packet pacing: A fundamental method is eliminating micro-burst by packet pacing so that queue length will not oscillate. However, as is summarized in §II-A, bursty transmission could stem from offload features inside NIC. Therefore, pacing must take place in NIC just before packets are injected into network. Besides, different flows should be separately paced and pacing rate should be dynamically determined, and thus current hardware-based pacing solutions [4], [31] do not apply. It is also difficult to implement such hardware pacer modules in high speed NICs.

(2) Increasing ECN threshold: If traffic burstiness in network is unavoidable, another solution is to prevent buffer underflow. The simplest method might be increasing ECN threshold, by which enough room is left for queue length fluctuation. However, this solution has two disadvantages: First, this increases the queueing latency of short flows. For example, in [13], authors suggest that the ECN threshold should be increased from 20 packets to 65 packets in 10Gbps network; as a result, the queueing latency might be increased by over three times. Second, as our analysis results (Theorem 1 and Theorem 2) show, the amplitude of queue length oscillations depends on the batch size, which varies under different network environments. Therefore, it's difficult to properly determine a generally applicable ECN threshold setting.

(3) Average queue length: When making ECN marking decision, instead of instantaneous queue length, we can use average queue length to accommodate bursty traffic, as is done in the literature [30]. However, ECN marking scheme based on average queue length may not apply to data center network, because senders may not be able to quickly react to congestion caused by fan-in traffic [13], which may lead to packet dropping and thus enlarging flow completion time.

B. Basic Idea

Due to the limitations of existing solutions, we propose our approach in this part. Rethinking the problem, we find that it occurs for two reasons: one is that traffic in the network is *bursty*; the other is that the switch only uses *instantaneous queue length* and *a single threshold* to make ECN marking decision. Since it's hard to drastically alleviate traffic burstiness, we can resort to improving ECN marking scheme by making marking decision more cautiously.

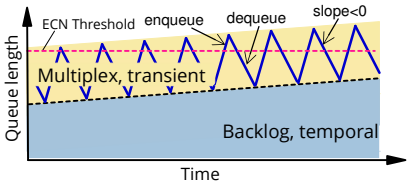


Fig. 9. Composition of queue occupancy

More specifically, as shown in Fig.9, the queue occupancies have two parts: one is caused by multiplexing of bursty traffic; the other is backlog queue representing the mismatching between congestion window size and pipeline capacity. Without bursty traffic, the queue occupancy is only caused by the second part, which is used for congestion detection. With bursty traffic, the queue occupancies caused by multiplexing of bursty traffic should be excluded when making ECN marking decision.

The exclusion could be achieved by utilizing the characteristics of queue length evolutions. Our key observation is that *the queue increasing caused by multiplexing of bursty traffic is transient.* Thus, if the queue length is larger than ECN threshold and becomes smaller than ECN threshold after a while, packets should not be marked. Furthermore, we can utilize another important information when making marking decision — *slope of queue length evolution*. Specifically, packets should not be marked if the queue length is decreasing.

C. Combined Enqueue and Dequeue ECN Marking

We now describe our approach. When a packet (with payload) is enqueued and dequeued, Algorithm 1 and 2 are performed, respectively. The algorithms are explained as follows:

Algorithm 1: Packet Enqueue

if *queue_length* $\geqslant K_2$ **then** Mark the packet;
else if $avg_s \geqslant 0$ **and** *queue_length* $\geqslant K_1$ **then**
  Mark the packet;
end

Algorithm 2: Packet Dequeue

/* update average slope */
$s \leftarrow \frac{q(t)-q(prev\_t)}{t-prev\_t}$; $prev\_t = t$;
$avg_s \leftarrow (1 - w_s)avg_s + w_s s$;
if *the packet is marked* **and** *queue_length* $< K_2$ **then**
  if $q(t) < K_1$ **or** $avg_s < 0$ **then** Unmark the packet;
end

Main mechanism: In our approach, ECN marking decision takes place at two time points: packet enqueue and packet

dequeue. When a packet is enqueued, it is marked if the instantaneous queue length is larger than ECN threshold (denoted by K_1) and the queue length is not decreasing (i.e., slope $\geqslant 0$). When the packet is dequeued, the marking is withdrawn if the queue length becomes smaller than ECN threshold or the queue length is decreasing (i.e., slope < 0).

Double Threshold: We use another threshold K_2 to prevent large queue occupancy. Specifically, when the queue length is larger than threshold K_2, packets are be marked anyway to prevent packet dropping. When N packet bursts with burst size B are arriving at the queue, the second threshold K_2 should be set as $K_2 = K_1 + (N-1)B$. For example, with LSO, given $N = 2$ and $\beta = \frac{1}{3}$, we have $K_2 = \frac{5}{4}K_1 + \frac{1}{4}Cd$.

Calculating slope: We have two considerations when calculating slope. First, slope is updated whenever a packet is dequeued rather than enqueued, because slope might be not accurate if it is updated when a packet is enqueued. Consider a scenario when the queue length is increasing at the beginning, and after that no packets are entering into the queue for a while. During this time, the queue length is decreasing but slope is still larger than 0 because it is not updated.

Second, average slope is used to accommodate transient decreases. In our approach, we use exponential weighted moving average (EWMA) to calculate average slope:

$$avg_s \leftarrow (1 - w_s)avg_s + w_s s \tag{31}$$

To determine the weight w_s, we assume that the transient decrease for L packets should be accommodated. Assume that the average slope is s_0 ($s_0 > 0$) before queue length begins to decrease, and the slope is s_1 ($s_1 < 0$) when queue length is decreasing. Then after L packets are dequeued, the average queue length can be given by

$$avg_s = s_1 + (s_0 - s_1)(1 - w_s)^L \tag{32}$$

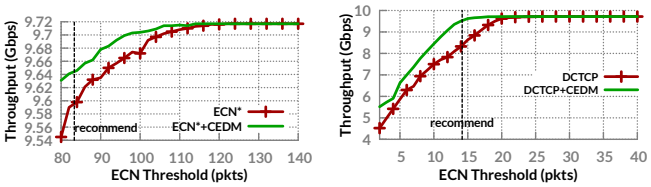
Assume that packets from N input ports are heading for the output port at line rate when queue length is increasing, and no packets are arriving when queue length is decreasing. Then $s_0 = (N-1)C$, $s_1 = -C$. To ensure the average slope is larger than 0, we should have $w_s \leqslant 1 - N^{-\frac{1}{L}}$. Given $N = 2$ and $L = 5$, for example, it is necessary to choose $w_s \leqslant 0.129$.

V. Evaluation

We implement LSO and CEDM scheme on ns-2 platform [22]. The reason that we choose LSO during evaluation rather than other batching schemes is that the traffic burstiness introduced by LSO is much severer than other batching schemes, as is analyzed in §II-A.

Four protocols are evaluated:

1) ECN*: This is the protocol proposed in [15].
2) ECN*+CEDM: Standard ECN [29] is used at end servers, and CEDM scheme is used to mark packets in switch.
3) DCTCP: This is the protocol proposed in [13].
4) DCTCP+CEDM: DCTCP protocol is used at end servers, and CEDM scheme is used to mark packets in switch.



(a) ECN* (b) DCTCP

Fig. 10. Throughput with different ECN thresholds

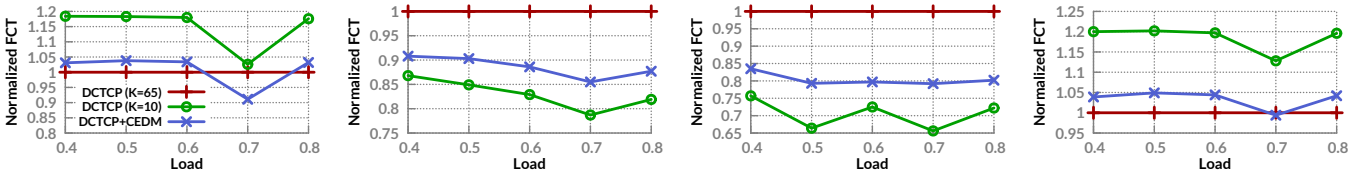
Throughput: To evaluate the throughput of CEDM scheme, we build a topology where 3 servers are connected to the same switch. The link capacity is 10Gbps and base RTT is 100μs. Two flows in two servers are keeping sending data to the third server. With CEDM, we set $w_s = 0.129$; and K_2 is set according to the guideline $K_2 = \frac{5}{4}K_1 + \frac{1}{4}Cd$. Fig.10 shows throughput with different ECN thresholds. With CEDM, the throughput is greatly improved. For example, when standard ECN is used at end servers (Fig. 10(a)), the throughput loss is reduced by 1.6 times (from 0.127Gbps to 0.076Gbps) using the recommended ECN threshold settings [15]. With DCTCP (Fig. 10(b)), when CEDM is used, the throughput loss is reduced by over 6 times (from 1.391Gbps to 0.210Gbps) using the recommended ECN threshold settings [21].

Large-scale simulations: We use the same simulation setup as prior work [32]–[34]. Specifically, we use a leaf-spine topology, where 9 leaf (ToR) switches are connected to 4 spine (Core) switches. In each leaf switch, 16 hosts are connected to it, thus, there are 144 hosts in total. Each leaf switch has 16 10Gbps downlinks connecting to hosts and 4 40Gbps uplinks connecting to spine switches. Each switch has 300KB buffer in each port. The base RTT across the spine is ~85.2μs. ECMP is used for load balancing.

We use realistic workloads in our evaluation. In the workload, the flow size distribution is derived from a data center running data mining jobs [35]. Flow arrivals follow a Poisson process.

In this simulation, DCTCP and DCTCP+CEDM are evaluated. In DCTCP, we set $K = 65$ packets, as is suggested by [13]. In DCTCP+CEDM, we set $w_s = 0.129$, $K_1 = 10$ packets ($0.17 \times C \times RTT$), and $K_2 = 30$ packets. In comparison, we also evaluate the DCTCP protocol when $K = 10$ packets to show the throughput loss when ECN threshold is low. Results with ECN* and ECN*+CEDM are similar and thus are omitted due to space limitation.

Fig. 11 shows the flow completion time (FCT) across different flow sizes at various loads. With CEDM scheme, lower threshold can be used, which decreases the queueing latency of small flows. Specifically, compared with DCTCP, the average and 99th percentile FCT of small flows (Fig. 11(b) and Fig. 11(c)) can be decreased by ~9.2-14.5% and ~17.5-20.8%, respectively. Meanwhile, the FCT of long flows (Fig. 11(d)) is almost the same as that of DCTCP protocol with $K = 65$. In comparison, if we simply decrease the ECN threshold without using CEDM scheme, the FCT of long flows is extended by ~19.6-20.2%, which denotes the loss of throughput.



(a) Overall: Average (b) $(0, 100\text{KB}]$: Average (c) $(0, 100\text{KB}]$: 99th percentile (d) $(10\text{MB}, \infty)$: Average

Fig. 11. Flow completion time across different flow sizes. FCT is normalized to the value achieved by DCTCP protocol for clear comparison.

VI. Conclusion

In this paper, we reveal that current instantaneous-queue-length-based ECN marking scheme and batching-scheme-induced micro-burst traffic in data centers can cause queue length oscillations, which may result in buffer underflow and throughput loss. We explore the problem through experiments and simulations. In addition, we theoretically deduce the amplitude of queue length oscillations. Through analysis, we find that the current ECN marking schemes based on instantaneous queue length and single threshold should be improved, because it may mis-mark packets when queue length oscillates. Therefore, we propose CEDM, which can mark packets more accurately. Through simulations, we show that CEDM can reduce throughput loss and improve flow completion time.

Acknowledgment

The authors gratefully acknowledge the anonymous reviewers for their constructive comments. This work is supported in part by National High-Tech Research and Development Plan of China (863 Plan) under Grant No. 2015AA020101, National Natural Science Foundation of China (NSFC) under Grant No. 61225011, and Suzhou-Tsinghua Special Project for Leading Innovation

References

[1] Arista Networks, "Myths about "Microbursts"," White Paper, 2009. [Online]. Available: http://www.arista.com/assets/data/pdf/7148sx-ixnetwork-microburst.pdf

[2] ——, "Microbursts, Jitter and Buffers," White Paper, 2014. [Online]. Available: https://www.arista.com/assets/data/pdf/TechBulletins/AristaMicrobursts.pdf

[3] F. Uyeda, L. Foschini, F. Baker, S. Suri, and G. Varghese, "Efficiently Measuring Bandwidth at All Time Scales," in *NSDI*, 2011.

[4] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is More: Trading a Little Bandwidth for Ultra-low Latency in the Data Center," in *NSDI*, 2012.

[5] R. Kapoor, A. C. Snoeren, G. M. Voelker, and G. Porter, "Bullet Trains: A Study of NIC Burst Behavior at Microsecond Timescales," in *CoNEXT*, 2013.

[6] V. Jeyakumar, M. Alizadeh, Y. Geng, C. Kim, and D. Mazières, "Millions of Little Minions: Using Packets for Low Latency Network Programming and Visibility," in *SIGCOMM*, 2014.

[7] D. Shan, W. Jiang, and F. Ren, "Absorbing micro-burst traffic by enhancing dynamic threshold policy of data center switches," in *INFOCOM*, 2015.

[8] S. Ghorbani, B. Godfrey, Y. Ganjali, and A. Firoozshahian, "Micro Load Balancing in Data Centers with DRILL," in *Proc. HotNets*, 2015.

[9] D. Shan, F. Ren, P. Cheng, and R. Shu, "Micro-burst in data centers: Observations, implications, and applications," *CoRR*, vol. abs/1604.07621, 2016. [Online]. Available: http://arxiv.org/abs/1604.07621

[10] A. Phanishayee, E. Krevat, V. Vasudevan, D. G. Andersen, G. R. Ganger, G. A. Gibson, and S. Seshan, "Measurement and Analysis of TCP Throughput Collapse in Cluster-based Storage Systems." in *FAST*, 2008.

[11] V. Vasudevan, A. Phanishayee, H. Shah, E. Krevat, D. G. Andersen, G. R. Ganger, G. A. Gibson, and B. Mueller, "Safe and Effective Fine-grained TCP Retransmissions for Datacenter Communication," in *SIGCOMM*, 2009.

[12] Y. Chen, R. Griffith, J. Liu, R. H. Katz, and A. D. Joseph, "Understanding TCP Incast Throughput Collapse in Datacenter Networks," in *WREN*, 2009.

[13] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," in *SIGCOMM*, 2010.

[14] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware Datacenter TCP (D2TCP)," in *SIGCOMM*, 2012.

[15] H. Wu, J. Ju, G. Lu, C. Guo, Y. Xiong, and Y. Zhang, "Tuning ECN for Data Center Networks," in *CoNEXT*, 2012.

[16] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion Control for Large-Scale RDMA Deployments," in *SIGCOMM*, 2015.

[17] G. Judd, "Attaining the Promise and Avoiding the Pitfalls of TCP in the Datacenter," in *NSDI*, 2015.

[18] A. Singh, J. Ong *et al.*, "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network," in *SIGCOMM*, 2015.

[19] C. Guo, L. Yuan *et al.*, "Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis," in *SIGCOMM*, 2015.

[20] C. Guo, H. Wu, Z. Deng, G. Soni, Y. Jianxi, J. Padhye, and M. Lipshteyn, "RDMA over Commodity Ethernet at Scale," in *SIGCOMM*, 2016.

[21] M. Alizadeh, A. Javanmard, and B. Prabhakar, "Analysis of DCTCP: Stability, Convergence, and Fairness," in *SIGMETRICS*, 2011.

[22] "ns-2," http://www.isi.edu/nsnam/ns/.

[23] L. Zhang, S. Shenker, and D. D. Clark, "Observations on the Dynamics of a Congestion Control Algorithm: The Effects of Two-way Traffic," in *SIGCOMM*, 1991.

[24] A. Aggarwal, S. Savage, and T. Anderson, "Understanding the performance of TCP pacing," in *INFOCOM*, 2000.

[25] H. Jiang and C. Dovrolis, "Source-level IP Packet Bursts: Causes and Effects," in *IMC*, 2003.

[26] J. Bennett, C. Partridge, and N. Shectman, "Packet reordering is not pathological network behavior," *Networking, IEEE/ACM Transactions on*, vol. 7, no. 6, 1999.

[27] D. Wei, P. Cao, S. Low, and C. EAS, "TCP pacing revisited," unpublished.

[28] Alteon Websystems, "Extended Frame Sized for Next Generation Ethernets," White Paper, 2000.

[29] K. Ramakrishnan, S. Floyd, D. Black *et al.*, "The Addition of Explicit Congestion Notification (ECN) to IP," RFC 3168, 2001.

[30] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *Networking, IEEE/ACM Transactions on*, vol. 1, no. 4, 1993.

[31] C. Lee, C. Park, K. Jang, S. Moon, and D. Han, "Accurate Latency-based Congestion Feedback for Datacenters," in *ATC*, 2015.

[32] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pFabric: Minimal Near-optimal Datacenter Transport," in *Proc. ACM SIGCOMM*, 2013.

[33] W. Bai, L. Chen, K. Chen, and H. Wu, "Enabling ECN in Multi-Service Multi-Queue Data Centers," in *NSDI*, 2016.

[34] L. Chen, K. Chen, W. Bai, and M. Alizadeh, "Scheduling Mix-flows in Commodity Datacenters with Karuna," in *SIGCOMM*, 2016.

[35] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A Scalable and Flexible Data Center Network," in *SIGCOMM*, 2009.