Tuning ECN for Data Center Networks*

Haitao Wu†, Jiabo Ju†⋆, Guohan Lu†, Chuanxiong Guo†, Yongqiang Xiong†, Yongguang Zhang†
{hwu, v-jiju, lguohan, chguo, yqx, ygz}@microsoft.com
†Microsoft Research Asia, China
⋆Peking University, China

ABSTRACT

There have been some serious concerns about the TCP performance in data center networks, including the long completion time of short TCP flows in competition with long TCP flows, and the congestion due to TCP incast. In this paper, we show that a properly tuned instant queue length based Explicit Congestion Notification (ECN) at the intermediate switches can alleviate both problems. Compared with previous work, our approach is appealing as it can be supported on current commodity switches with a simple parameter setting and it does not need any modification on ECN protocol at the end servers. Furthermore, we have observed a dilemma in which a higher ECN threshold leads to higher throughput for long flows whereas a lower threshold leads to more senders on incast under buffer pressure. We address this problem with a switch modification only scheme - dequeue marking, for further tuning the instant queue length based ECN to achieve optimal incast performance and long flow throughput with a single threshold value. Our experimental study demonstrates that dequeue marking is effective for increasing the maximum incast senders close to the performance limit of ECN, achieving a gain anywhere from 16% to 140%.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network topology, Packet-switching networks

General Terms

Algorithms, Design

Keywords

TCP, ECN, Incast congestion, RED, Data center networks

1. INTRODUCTION

Data center networks are designed to support various applications [11, 8] and diverse traffic patterns [14]. Advanced topologies and structures [1, 13, 12] have been proposed to achieve higher bandwidth in data center networks. However, there have been some serious concerns about the TCP performance in data centers. For example, TCP incast congestion has become a practical issue in data center networks [22]. TCP incast occurs when many-to-one short flows are barrier synchronized by the applications. A recent study [22] shows that extreme high bandwidth and low latency, true of most data center networks, are preconditions for incast congestion. Moreover, without well-recognized service differentiation support, short-duration TCP flows will further suffer in the interaction with long-duration TCP flows [2]. Short flows require low queue occupancy for smaller task completion time while a large queue has been built up by long flows. A recent study [2] proposes a solution called DCTCP that requires both changing the switch ECN settings and modifying end server ECN in TCP stacks.

In this paper, we propose a further look at ECN implementation at the switches, and see if there is a way to address the above two TCP performance issues with a simple ECN tuning at the switch-side only. First, we have observed that the network latencies across data center networks are extremely low (under a millisecond). This has inspired us to study the use of instant queue length instead of average queue length in ECN. We hypothesize that, if instant queue length is used to trigger ECN, and if the threshold is properly tuned, ECN may actually alleviate the TCP performance impact without requiring end-host modifications. We use ECN* in this paper to denote such a scheme in which instant queue length based ECN is used at the switches and standard ECN congestion control is used at the end hosts.

The notion of using instant queue length to trigger ECN is not new. It has been proposed in previous literature, most recently in DCTCP to obtain the ratio of marked packets in [2], and to some extent to decouple ECN from Active Queue Management (AQM) in [15]. However, DCTCP requires modifying the end-point ECN implementation to make use of the new information, and it remains an open question on how to tune the ECN threshold for instant queue length to achieve a general buffer management scheme on commodity switches for TCP performance gains.

Through a brief analysis of TCP throughput, we discovered that the lower bound of the ECN* threshold is determined by the Bandwidth Delay Product (BDP). Our measurement in a real data center with 40,000 servers revealed that the TCP Round Trip Time (RTT) is hundreds of mi-

*This work was performed when Jiabo was an intern at Microsoft Research Asia.

Co-NEXT'12, December 10-13, 2012, Nice, France.

croseconds, which means the BDP can be as low as tens of kilobytes for Gigabit Ethernet. Furthermore, an ECN threshold based on a small BDP is already effectively supported by existing commodity switches. The preconditions for using ECN* are thus satisfied in data center networks.

Our experimental results show that ECN* achieves a similar performance to DCTCP on low latency networks when the ECN threshold is set by BDP. We believe that this clearly demonstrates the effectiveness of instant queue length based ECN, as ECN* only uses standard ECN[20] at the end servers. However, the results also reveal that there is a dilemma with the ECN threshold setting for both DCTCP and ECN*: a higher threshold leads to a high throughput for TCP long flows while a lower threshold is better for controlling TCP short flows on incast. We therefore designed a switch modification only approach to achieve the optimal performance of both long flows and incast using one threshold: ECN with dequeue marking. By checking whether instant queue length is over the ECN threshold when packets are dequeued, the marking of dequeued packets at the head of queue accelerates the congestion information delivery to the end system.

This paper has two key technical contributions. First, we give the threshold lower/upper bounds for ECN* and show a valid setting on commodity switches. Through our analysis, comprising latency measurements in a real data center and experiments in our tesbed, we have shown that ECN* achieves a similar performance to DCTCP for both short flows competing with long flows as well as incast in low latency data center networks. Second, we have solved the dilemma of setting the ECN threshold for instant queue length schemes, including DCTCP and ECN*. We have proposed, developed, and evaluated a switch modification only approach, namely dequeue marking, that achieves close to optimal incast performance for ECN using the threshold that leads to the high throughput of long flows.

The rest of the paper is organized as follows. Section 2 discusses research background. Section 3 describes ECN*. Section 4 presents dequeue marking. Section 5 presents experimental results. Section 6 discusses issues related to ECN. Section 7 presents related work. Finally, Section 8 concludes the paper.

2. RESEARCH BACKGROUND

2.1 TCP incast

TCP incast has been identified and described by Nagle et al. [18] in distributed storage clusters. In distributed file systems, the files are deliberately stored in multiple servers. When multiple blocks of a file are fetched from multiple servers at the same time, TCP congestion can occur at or near the receiver. This is called TCP incast. Several application specific solutions have been proposed in the context of parallel file systems. In data center networks, there can be many applications and TCP incast problem has become a practical issue [22, 2].

Figure 1 shows the incast goodput achieved on multiple connections versus the number of concurrent senders. Note that this paper uses the term goodput and throughput interchangeably to denote effective throughput obtained at the applications. The results are measured in a testbed as detailed in Section 5. The multiple TCP connections are barrier synchronized in our experiments as follows. We first establish multiple TCP connections between all senders and the receiver respectively. Then the receiver sends out multiple small request packets (one per sender) to request each sender to transmit data. TCP connections are issued round by round, and one round ends when all connections in that round have finished their data transfer to the receiver. We present two setups for incast: 1) each connection has a fixed traffic amount with the number of senders increasing, as in [19]; 2) the total traffic amount of all connections is a fixed one, as in [22].

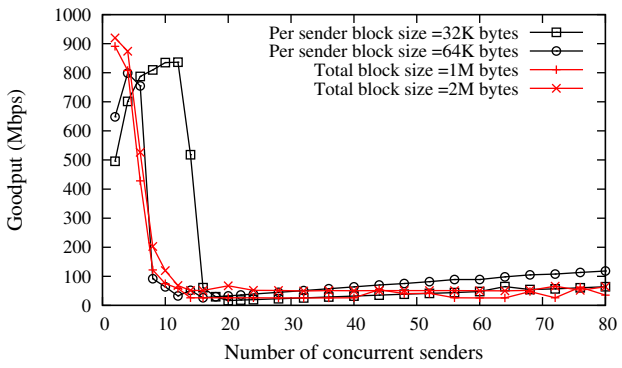

Figure 1: The incast goodput of multiple barrier synchronized TCP connections versus the number of senders.

2.2 ECN

As an explicit congestion notification protocol, DECbit [21] is proposed to detect possible congestion at routers. A bit at data packet header is set by intermediate nodes to notify the congestion. When the receiver gets the marked packets, it echoes the information to the sender by marking another bit at the ACK. The sender then decides whether the congestion window should be cut by the ratio of marked ACKs.

Recent switches/routers follow the standard set by RFC [20] in implementing ECN for TCP, which is closely related to Random Early Detection (RED) [10]. RED addresses the global synchronization issue of multiple TCP connections sharing the same bottleneck link, which is caused by a large number of packet losses occurring when the drop tail buffer overflows. For ECN/RED, an average queue length is maintained at the switch, and two (low and high) thresholds for the average queue length are used. In RED, when the average queue length is between the two thresholds, the incoming packet is dropped with a probability capped by a parameter, namely the max drop probability. When the average queue length is over the high threshold, all incoming packets are dropped.

In ECN, the incoming packets will be marked on Congestion Experienced (CE) bits instead of being dropped. In contrast to RED, which works without changing TCP end points, ECN requires a TCP receiver to continuously pass the CE bits as ECE (ECN echo) bits to the TCP sender. The TCP sender then cuts the congestion window in half and marks the first new data with a CWR (congestion window reduced) bit to suppress the ECE bit from the receiver. A TCP sender responds to the ECE bit only once for a win-

dow of data. Later studies have shown that RED/ECN may improve the performance of TCP if the parameters are properly set. However, the setting of the parameters are for the most part very subtle [9].

2.3 DCTCP

DCTCP[2] proposes changes to ECN to address the interaction of short and long TCP flows as well as TCP incast. DCTCP introduces two distinct features that differ from the ECN standard. First, DCTCP uses instant queue length at the switch instead of average queue length to trigger CE marking. Second, DCTCP cuts the congestion window at the sender in proportion to the ratio of CE marking. It does so by requiring the TCP receiver to echo the ECE for every packet (instead of per window of packets). If all ACKs are marked with ECE for a window of data, DCTCP cuts the congestion window size in half. To smooth the reduction of the congestion window, an exponential filter is introduced for the ratio of marked packets at the DCTCP sender.

The use of instant queue length combined with the use of only one threshold can greatly simplify the ECN setup on switches. The idea of using the ratio of marked packets (obtained by one-to-one mapping of ECE on ACKs) changes the binary feedback in ECN into multiple levels. In contrast to DECbit, which reduces the window when the ratio of marked packets is over a certain parameter, DCTCP cuts the congestion window using the value of half of the marked packets ratio. With these two changes, DCTCP can greatly improve the TCP performance with ECN enabled.

3. INSTANT QUEUE LENGTH BASED ECN

In this section, we present a generalized strategy for using instant queue length and a single threshold in ECN at intermediate switches. The scheme using instant queue length ECN with standard ECN [20] at the end servers is denoted as ECN*. We provide an analysis for the threshold setting and present the measurement results to show that existing data center networks are well within the operational region. As an example, we show that this strategy can be easily implemented in existing ECN-capable switches and it achieves similar results with DCTCP (but without end-system modifications).

3.1 ECN*

As already mentioned, instant queue length based ECN has been proposed in previous research to decouple ECN from AQM [15] and for frequent marking of the congestion state [2]. Here, we look at the instant queue length from a more generalized angle. Instant queue length represents the congestion window of all TCP connections sharing the same bottleneck. If the focus is to deal with temporal congestion caused by traffic burstiness, e.g,. incast congestion, a congestion control scheme like ECN can use instant queue length directly. Therefore, as long as the ECN threshold is set according to the window reduction taken by TCP, the TCP throughput performance will not be degraded.

Our generalized ECN strategy works as follows. At the intermediate switch/router, the instant queue length value is compared with a pre-configured threshold value whenever a packet is processed. If the instant queue length is greater than or equal to the threshold, the packet is marked with a CE bit at the IP header. The only configurable parameter that can tune the behavior of this scheme is the ECN threshold.

Our generalized ECN strategy does not make any assumption on the congestion control scheme used by the TCP sender and receiver to handle the marked packets, but we believe different schemes should achieve similar performance and have a similar tradeoff on parameter settings. The congestion control schemes include DCTCP [2], standard ECN as defined in RFC3168 [20], and DECbit.

We designate ECN* as the scheme that uses instant queue length and a single threshold at the switches and uses standard ECN[20] at the TCP sender/receiver.

ECN* is very well supported by today's commodity ECN-capable switches. First, ECN switches allow a weight parameter to adjust the exponential factor for updating the average queue length. By setting this weight to 1, the average queue length is effectively the same as the instant queue length because the values in the history are ignored. Second, ECN switches accept two threshold parameters, the low and high thresholds. By setting the two thresholds to the same value, they become one single threshold and the region in between the low and high thresholds is no longer in effect.

The support of ECN* at end-servers follows the ECN standard[20]. We assume that all packets are ECN-capable in data center networks and will discuss issues of non-ECN-capable packets suggested by the ECN standard [20] in Section 6.

3.2 Analysis of ECN*

We will now analyze the performance of ECN* when congestion occurs. We are interested in the parameter setting of the ECN* threshold, and seek to generalize it for future congestion control strategies deployed by the TCP sender/receiver. As standard ECN cuts the congestion window in half for a marked packet, we believe that the ECN* threshold serves as the upper bound for other schemes like DCTCP that decreases the congestion window more conservatively.

3.2.1 Lower bound of ECN threshold*

We analyze the lower bound for the ECN* threshold at which the TCP throughput performance won't be affected after the congestion window reduction by ECN. To simplify the analysis, we assume that the switch buffer size is large enough so that no packet is dropped due to buffer overflow. Additionally, we start with a case where there is only one TCP connection at the bottleneck link.

We assume that the connection is in a steady state, i.e., in congestion avoidance mode when an ACK with an ECE is received. The TCP sender then cuts its congestion window in half, which leads to buffer draining at the bottleneck link. To ensure that the throughput of this TCP connection does not degrade, the queue length at the bottleneck link should never drain to zero. Therefore, the TCP window saw-tooth phenomena is similar to the case described and analyzed carefully in [4] for switch buffer sizing. The difference is that in our case packets are not dropped after ECN threshold is reached. Instead, the packets are marked and delivered to the receiver. The marked ECE on return ACKs results in congestion window reduction, which is actually the same assumption made in [4]. Note that [4] does not consider a timeout caused by lost packets to simplify the analysis. For ECN*, there is no timeout and no packet loss, which

means that our case closely follows the previous simplified analysis for a drop tail queue with a buffer size the same as the ECN* threshold. Therefore, the ECN threshold h that won't affect TCP throughput is similarly obtained as that by the well-known rule-of-thumb for drop tail buffer size, i.e., the BDP.

$$h \geq T \times C \tag{1}$$

where T is the averaged Round Trip Time (RTT) for TCP connections in the network, and C is the bottleneck link capacity. For an advanced congestion control scheme like DCTCP, the lower bound of the threshold has been shown to be $O(\sqrt{T \times C})$ in [2, 3]. However, as we will show later in this section, the absolute value of the difference between the lower bounds of DCTCP and ECN* is actually very small for data center networks, which allows ECN* to perform similarly to DCTCP.

The case of multiple TCP connections on the same bottleneck link can be similarly obtained as $h \geq T \times C/\sqrt{N}$, where N is the number of long TCP flows on the bottleneck link. However, as shown in a measurement study of DCTCP by [2], the number of concurrent long TCP connections to a server is generally 2 or 3. For shallow-buffered Top of Rack (ToR) switches that connect servers, synchronization of a small number of TCP connections still takes effect so that the lower bound stays close to BDP in practice.

3.2.2 Upper bound of ECN* threshold

We analyzed the upper bound for the ECN* threshold at which the congestion windows of the TCP connections sharing the bottleneck link are effectively controlled to avoid buffer overflow. In other words, when the ECN* threshold is lower than the upper bound, there is no TCP packet loss.

To obtain the upper bound, we begin with a simplified case where there is only one TCP connection and it has a slow start. We denote its congestion window size as w_e when the ECN threshold is reached at the intermediate switch. To simplify our analysis, we assume that the TCP sender only generates full-sized data packets, and the transmission time is δ for a Maximum Transmission Unit (MTU) on the bottleneck link. TCP aggressively increases the congestion window by the same amount of data acknowledged by the receiver during a slow start, so the largest queue length at this phase is our focus.

To illustrate the evolution of the TCP congestion window and the resulting queue length at the bottleneck link, we present the details of a TCP connection on a slow start in Table 1. A similar method was used in [17] to illustrate the advantage of TCP-Reno over TCP-Tahoe in a steady state. In contrast to previous scenarios, data center networks feature high-bandwidth and low-latency, which results in a small BDP compared to the switch buffer size. We discuss the measurement of latency in datacenter networks in Section 3.3.

In Table 1, the TCP initially starts with the congestion window as 2 packets, which directly results in the switch queue length also having 2 packets. An RTT (T) later the ACK for the first data packet arrives at the sender, which causes the window to increase from 2 to 3 and the packets with sequence 3 and 4 are sent. Since the buffer has drained to empty, the newly arrived packets 3 and 4 cause the queue length to reach 2 again. This phenomena continues, resulting in regular mini-cycled initial queue length at the switch with a mini-cycle length of T.

Time	Packets Acked	Window size	Packets sent	Queue length
0		2	1,2	2
T	1	3	3,4	2
$T+\delta$	2	4	5,6	3=2-1+2
$2T$	3	5	7,8	2
$2T+\delta$	4	6	9,10	3=2-1+2
$2T+2\delta$	5	7	11,12	4
$2T+3\delta$	6	8	13,14	5
$3T$	7	9	15,16	2
$3T+\delta$	8	10	17,18	3=2-1+2
$3T+2\delta$	9	11	19,20	4=3-1+2
$3T+3\delta$	10	12	21,22	5=4-1+2
$3T+4\delta$	11	13	23,24	6
$3T+5\delta$	12	14	25,26	7
$3T+6\delta$	13	15	27,28	8
$3T+7\delta$	14	16	29,30	9
		...		
		w_e		h
		w_e+1		$h+1$
		...		
		$2w_e = w_e + w_e$		$h+w_e$

Table 1: Evolution during a slow start phase.

There are two conditions for changing the mini-cycled switch queue length. First, the BDP in the data center must be small, e.g., around 30 packets from our measurement study in Section 3.3. The small BDP leads to insufficient time to completely drain the queue in a mini-cycle before the next mini-cycle starts. That is to say, the queue length increases continuously after the network pipe becomes full. Second, if the switch queue length reaches the ECN* threshold, then the switch starts to mark incoming packets with a CE bit. When the ACKs of those marked packets arrive at the sender, the sender decreases the congestion window. The half-reduced congestion window throttles the outgoing traffic and thus causes the draining of the switch queue.

When the switch queue length reaches threshold h, the congestion window size is w_e. This means that there are w_e packets on the flight (either data packets in the forward direction or the ACKs in the backward direction). Before the congestion information generated by the ECN reaches the sender, the congestion window continues to increase. As the TCP connection is still in the slow start phase, the congestion window will at most become $2w_e$. Considering equation 1 has to be satisfied to maintain TCP performance on throughput, the first condition will always be satisfied when the second condition is reached. Therefore, the queue does not drain and thus the queue length keeps increasing from h to $h + w_e$.

The value of window size w_e is bounded by queue threshold h and the BDP, as the network may hold the packets either in the switch queue or in the network pipe. Therefore, we have

$$w_e \leq h + T \times C. \tag{2}$$

Note that whether the bound in equation 2 is tight is determined by the value of h and the the BDP. For example, assume the BDP is 30 packets while h is set to 6 packets. From Table 1 we determine that w_e is actually 13 packets when ECN is triggered, and the value 13 is much less than

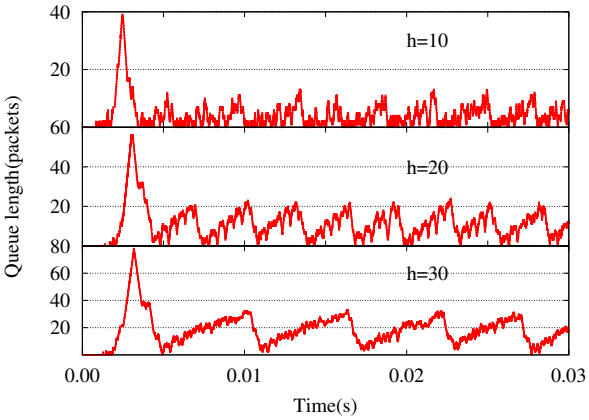

Figure 2: The instant queue length versus time with different ECN* threshold.

$6 + 30$. However, when h is set according to equation 1, the value of w_e is very close to the bound, as the network pipe is close to full. To ensure there is no packet loss, the switch buffer size B should be larger than the maximal value of queue length, i.e.,

$$h + w_e \leq B. \tag{3}$$

From equations 2 and 3, the upper bound for the ECN* threshold to avoid packet loss is

$$h \leq \frac{1}{2}(B - T \times C). \tag{4}$$

As the threshold h also has a lower bound in equation 1, equation 4 also gives the minimal switch buffer size to avoid buffer overflow as

$$B \geq 2h + T \times C \geq 3T \times C. \tag{5}$$

This condition holds for multiple synchronized TCP connections as long as the total congestion window of those connections can be successfully reduced[1].

3.2.3 *Validation in Experimental testbed*

We use experimental results to show that both the lower and upper bound obtained in equations 1 and 4 are valid. The details of our testbed are presented in Section 5. We used a Broadcom Pronto 3290 48-port GbE switch and developed software to log the instant queue length when an incoming packet arrives at the output port. We set up two TCP connections to the same receiver, and we changed the threshold h on the output port to the receiver and logged the queue length of the output port.

We ran the experiments 5 times and the queue length versus time with maximal the peak can be seen in Figure 2. The results show that the queue always jumps to the peak because of the slow start of the TCP connections, and then goes into cycled saw-tooth caused by congestion avoidance of the TCP connections. We observed that when threshold h is set to 20, the queue does not drain to empty and thus maintains the throughput. This phenomena echoes the condition of equation 1 as the BDP in our testbed is close to 20 packets. If we count the BDP, i.e., $T \times C$ as 20 packets, then the peak buffer occupation during the slow start phase clearly echoes the conditions of buffer size requirement as $2h + 20$ in equation 5.

[1]For incast, the total minimal congestion windows used in multiple TCP connections may cause switch buffer overflow, which is unavoidable even if DCTCP/ECN* is used in the switch.

3.3 RTT in data center networks

The threshold for ECN* should be $T \times C$, and the buffer size should be three times larger than the threshold. This requirement translates into a very large buffer requirement for Internet cases [4]. However, we will show that this requirement makes sense for data center networks and most commodity switches have the capacity to support it.

Our interest is to understand the value of $T \times C$. Most data center networks use Gigabit Ethernet Interface, so for the Top of Rack (ToR) switch, the link capacity to servers is one Gigabit. We start with the case of the ToR switch to check whether such a buffer and the ECN threshold requirements can be supported. Upper layers may have 10G or even 40G links, but certainly the buffer size on high profile switches/routers are much larger. We believe our evaluation of ECN* like protocols may provide more information on the buffer size ECN needs for switches in data center networks.

Another key element for BDP is the RTT in data center networks. Paper [2] showed that in the absence of queueing, the RTT is under 250us for inter-rack, and approximately 100us for inner-rack. What we are interested in is the RTT, including queueing latency, in a real data center, so that we can set and tune the ECN threshold and also set the buffer size.

To achieve this goal, we used software to setup TCP connections between different server pairs in a production data center with over 40,000 servers. The connections pattern was like a mesh for the whole data center at the rack level. We purposefully turned off our inner-rack connections to reduce traffic. All servers logged the RTTs for connection establishment then tore down the connections, so that no additional traffic was generated into the data center. We collected the RTT for those TCP connections as samples for RTT in the whole data center. Note that this data center was busy as it serves many customers using diverse products. Actually, we use the same data center to store and process our RTT logs.

Figure 3 shows the inter-rack latency distribution we obtained for one day in December 2011. We actually collected results for several months and found the patterns to be similar. The results show that over 23% of connections have an RTT of less than 200us, and over 74% of connections have an RTT of less than 300us. The connections have an RTT of less than 600us with a probability of 95%. As most of the RTT we observed was less than 400us (90% percentile), we estimate the BDP in data centers for ECN* threshold setting as follows:

$$BDP = T \times C = 1G * 400us = 50KB$$

The default MTU on the Ethernet is 1.5KB, so 50KB means 33 full sized packets.

We suggest using 20 to 30 packets as the ECN* threshold (h) at the switches, as determined by the RTT values in network. In addition, a buffer size in the same amount as the threshold should be reserved for each port on the switches. For a 48 port Gigabit Ethernet switch, the total reserved

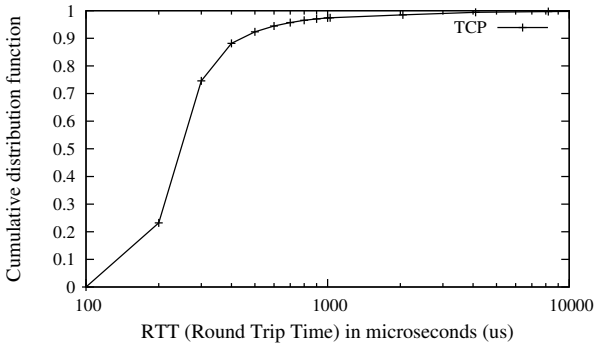

Figure 3: The inter-rack TCP RTT in a production data center with over 40,000 servers.

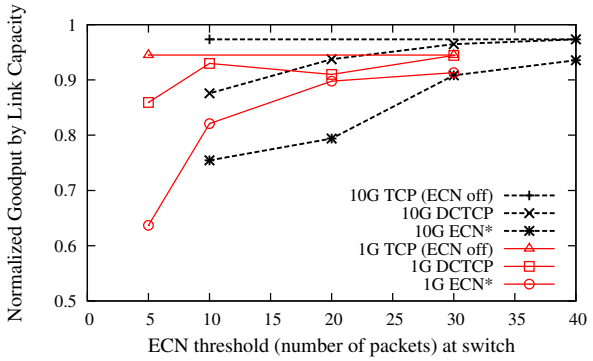

Figure 4: The goodput of two long flows to the same destination server on 1GbE and 10GbE networks.

buffer size is about 2MB(30*1.5K*48). Taking our shallow buffered broadcom switch as an example, it has 4MB shared buffer in total. DCTCP [2] also shows two other shallow buffered commodity ToR switches with a 4MB buffer, a Triumph 48-port, and a Scorpion 24-port. Thus, the ECN* threshold setting is valid for existing shallow buffered ToR switches. Note that ECN* also requires that the actual buffer size should be as large as $3TC$ to handle traffic peak. We consider that the buffer size over the reserved size (TC to $3TC$) can be allocated from the rest of the buffer in a shared way, instead of allocating to ports in a dedicated way. Such configurations for partial dedicated buffers per port and the rest buffers for limited sharing among all ports are actually supported in commodity switches [7].

3.4 Performance of DCTCP and ECN*

Next, we will briefly evaluate the performance of DCTCP and ECN* in our testbed. The setup of our testbed is described in Section 2.1. Both DCTCP and ECN* are designed to handle short flows, especially for incast. Ideally, we expect them to achieve similar performance on long flows and much better performance on incast.

We started by testing their performance on long flows. Figure 4 shows the throughput performance of two long TCP flows to the same destination server. The two flows contend the bandwidth at the switch output port to the server, and the total goodput is normalized by the link capacity. We chose two flows because paper [2] showed that the number of concurrent long flows is around 2 to 3, and 2 is the lowest number to cause contention. Additionally, as we discussed in Section 3.2, a larger number of TCP flows

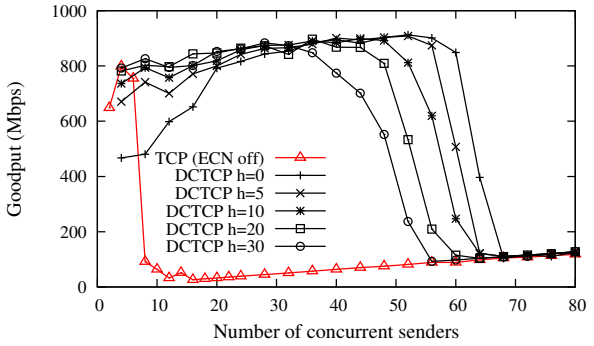

Figure 5: Incast Goodput of TCP (ECN off) and DCTCP with varied ECN thresholds(h).

requires a smaller threshold to maintain throughput. We also present the throughput achieved by TCP (ECN off) as the target throughput for DCTCP/ECN*. For both 1G and 10G link cases, the buffer size is set at 100 MTUs. We used 1500bytes MTU for 1GbE and 9000bytes jumbo frame MTU for 10GbE.

For both DCTCP and ECN*, the achieved throughput increases with the threshold and converges at TCP (ECN off). DCTCP [2] recommends using 20 packets as the threshold for a Gigabit Ethernet. ECN* can follow the same setup and achieve a similar performance. For 10GbE, we found the threshold should be set to 30+ packets, which is reasonable considering the ratio of link capacity to packet sizes. The figure also shows the slight throughput advantage of DCTCP on the smaller threshold requirements of 10GbE. The threshold requirements for DCTCP and ECN* to achieve high throughput on long flows are actually both well supported by existing commodity switches.

Next we evaluate the incast performance of TCP (ECN off), DCTCP, and ECN* respectively. One problem with incast is the question of how to set the threshold. As we discussed in Section 3.3, the ECN threshold used for ECN* should be around 30 packets considering the typical RTT in data center networks. To understand the impact of the ECN threshold on incast performance, we present the incast throughput of DCTCP and ECN* with different thresholds in Figure 5 and 6, respectively. The transmission block size for each sender is fixed at 64KB. Similar performance gaps are also observed for other incast setups, e.g., the total fixed block size as 1MB or 2MB.

Figures 5 and 6 show that the incast performance of DCTCP and ECN* is significantly affected by the ECN threshold. Larger thresholds lead to a smaller number of concurrent incast senders being supported. In the two figures, the curves of the threshold at 0 are just shown for the incast performance limit, i.e., the maximal number of parallel senders that can be supported on incast. When threshold is set to 0, all the incoming packets will be marked by CE unconditionally, so that the congestion window at the senders will never be increased. Note that such an unconditional marking strategy or a small threshold actually degrades the performance when the number of senders is small. The idea case for incast performance is non-degraded throughput until the largest number of incast senders is reached.

We have observed the tradeoff with the ECN threshold

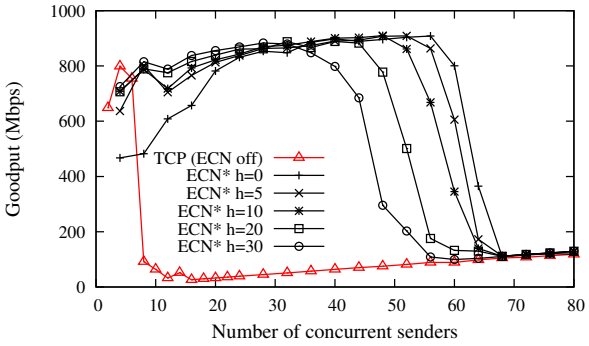

Figure 6: Incast Goodput of TCP (ECN off) and ECN* with varied ECN thresholds(h).

setting: a larger threshold leads to a higher long flow throughput but a smaller number of concurrent senders on incast, and a smaller threshold greatly improves incast performance but degrades the throughput of long flows. As we observe this same tradeoff for both DCTCP and ECN*, we believe this is a fundamental tradeoff for all congestion control schemes using instant queue based ECN.

Since the total number of supported incast senders is close to the number of the switch port at 48, why should we care about the performance difference. As we will show in Section 5, the maximum number of concurrent senders is actually determined by the switch buffer size. And as we have discussed in Section 3.3, for shallow buffered ToR switches, the buffer is actually shared among ports [7]. Therefore, with a smaller effective buffer size, e.g., due to high buffer pressures caused by buffer occupation on other ports, or with multiple incast applications occurring at the same time, the worst situation can be amplified.

4. DEQUEUE MARKING

4.1 Problem statement

To understand the impact of the ECN threshold setting on performance, we categorize the traffic in data center networks into three categories as follows.

First, all competing TCP flows are long flows. In this case, the performance is mainly determined by the throughput of the long flows. A recent study in [2] showed that there are usually 2 or 3 competing long flows. Second, a few short flows are trying to deliver a small volume of data as fast as possible, while there are other background long flows competing for bandwidth. In this case, the completion time of those short flows is the focus of performance. Third, during TCP incast, a large number of short flows are competing for a specific network port in a very short time. TCP incast happens whether or not there are ongoing background long flows. The performance of TCP incast is determined by the completion time of the last finished short flow.

As shown in Section 3.4, TCP (ECN off) actually works well for the first category. However, the large buffer occupation of TCP (ECN off) makes it work poorly for the second and third cases. ECN* and DCTCP use congestion control based on instant queue length. Thus, compared with TCP, both methods achieve similar performance for the first category and much better performance for the second and third categories. Considering the throughput of TCP connections is still fundamental to performance, we need a scheme that improves the performance of both DCTCP and ECN* during the last two categories by assuming the threshold is set for the first category.

In Section 3.2, we present the relationship of throughput and the threshold setting of ECN*. For DCTCP, the relationship has also been analyzed previously in [3]. Given that the ECN threshold has been set, we must next address how to make the instant queue length represent the traffic and thus trigger the threshold faster. Note that we don't wish to change the ECN threshold dynamically according to the traffic categories as we believe that such a solution is hard to implement due to the traffic dynamics in a data center.

4.2 Dequeue marking scheme

We have proposed, implemented, and evaluated a pure switch based solution - dequeue marking. The purpose of dequeue marking is not only to provide a complete switch only solution for ECN*, but also to understand the performance limit of instant queue length based ECN (both ECN* and DCTCP). Customers that have concerns with TCP stack modifications at the end server, e.g., DCTCP, can use dequeue marking and ECN*.

At the concept level, dequeue marking seems similar to the well-known drop-from-front method [16], which was proposed for TCP over ATM. A previous study in [16] shows that drop-from-front greatly improves performance over tail-drop, but for RED, front-drop and tail-drop are similar. In existing commodity switches, ECN follows the implementation of RED. As the switch buffer uses a First Come First Service (FIFO) rule, RED checks whether a packet should be dropped (or marked if ECN is enabled) when an incoming packet is queued at the switch output port. Such a dropping/marking policy for original RED/ECN works well as the rule is performed by an exponential filter based on the average queue length.

In this paper, we argue that for instant queue length based ECN, the marking policy performed when packets are queued is not longer efficient. For instant queue length based ECN, e.g., ECN* as analyzed in Section 3.2, the performance has been analyzed by assuming that congestion information is generated when the instant queue length is over the threshold. However, in existing ECN implementation, such congestion information (marked CE bit on packets just queued) must wait until the marked packet moves to the head of the queue. If the ECN threshold is set to a large value to accommodate TCP throughput of long flows, we believe that setting ECN mark when packets are queued severely delays the delivery of congestion information.

In this paper, we propose the use of dequeue marking for instant queue length based ECN at switches. When an ECN-capable packet is about to be dequeued, we check the instant queue length and the ECN threshold. If the instant queue length is larger or equal to the ECN threshold, then the packet is marked with a CE bit.

There are two benefits to dequeue marking. First, the latency to deliver the congestion information is reduced, so a better incast performance is expected. Second, our analysis and experimental results (skipped due to space limitation) for dequeue marking show a minimal buffer size (threshold upper bound) of ECN* as $B \geq h+T\times C \geq 2T\times C$, compared with equation 5 for the original enqueue marking.

Dequeue marking decides whether a packet should be marked when a packet is about to transmit, which is different from a straightforward extension of drop-from-front strategy of RED [16] to a mark-from-front strategy of ECN. This is because for both RED and ECN, the dropping/marking decision is made when a packet is enqueued. Considering that the traffic is highly bursty during TCP incast and multiple packets are enqueued at the same time, the front-mark of ECN may only mark the packets waiting for transmission when other packets are enqueued, while dequeue marking continuously marks all outgoing packets until the queue length is less than the threshold. To this end, front-mark may leave some "holes" (unmarked packets) and thus we believe dequeue marking is more suitable for instant queue based ECN.

4.3 Implementation on commodity switches

Dequeue marking does not require any change of the ECN protocol, as it changes the start time of when the packets are marked.

Readers may have concerns of whether this method will introduce large process latency for the outgoing packets if we modify the CE bits when the packet is about to dequeue. From our experience, changing bits on the packet header during dequeue is well supported on commodity switches. However, the switch does not provide a hardware solution to check the current buffer length when a packet is about to dequeue. To solve this problem, we developed a software solution using the broadcom switch SDK and started a dedicated thread to check queue length continuously. Our measurement results show that queue length reading costs about 6 us (microseconds). That is to say, the first packet to get a CE mark is 6us behind the time when the queue length is over the ECN threshold. Note that the delay is not the process latency introduced per packet. Since the hardware on the broadcom switch also requires a 4us interval for the queue length update during normal ECN marking (when packets are queued), we think that our software solution for dequeue marking at 6us is acceptable.

Similar operations can be made for other types of switches and we believe there is no other barrier to implementing dequeue marking on chips.

5. EXPERIMENTAL RESULTS

We deployed a testbed with 50+ servers and one Broadcom Pronto 3290 48-port Gigabit Ethernet switch. This switch supports ECN, and has 48 1GbE ports and 4 10GbE ports. The topology of our testbed is such that 47 servers connect to the GbE ports, and the other 4 servers connect to the 10GbE ports. Each server has two 2.2G Intel Xeon CPUs E5520 (four cores in total), 32G RAM, and a 1T hard disk. We use 47 Intel PRO/1000 PT Dual Port GbE Adapters and four Mellanox ConnectX 10GbE Adapters. The OS of each server is Windows Server 2008 R2 Enterprise 64-bit version. The CPU, memory, or hard disk was never a bottleneck in any of our experiments. We modified the iperf to create an incast scenario in which multiple sending servers generate TCP traffic to a receiving server under the same switch.

The implementation of DCTCP followed the description in paper [2]. For ECN*, we used existing Windows 2008r2 TCP/IP stack with no modification, which was New-Reno like. We didn't use SACK (Selective Acknowledgment) in our experiments as a previous study [19] shows that it does not help much with incast congestion. We turned on ECN support at both the sender and receiver servers. All other settings followed those in the data center network we accessed. The default timeout of TCP on Windows server is 300 milliseconds (ms).

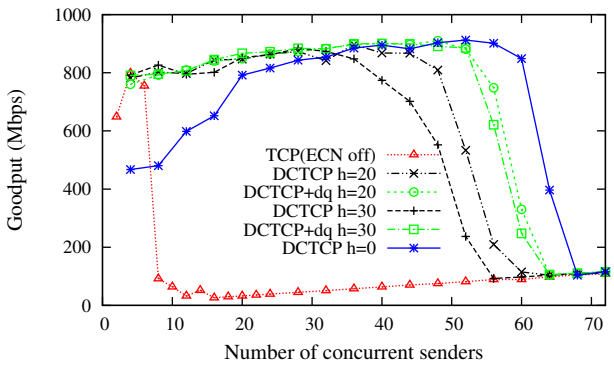

Figure 7: Incast performance of TCP and DCTCP with and without dequeue marking. ECN Threshold is set at 20 and 30 packets.

We use "**dq**" to denote the results which featured dequeue marking for DCTCP/ECN*. We evaluated instant queue based ECN and found improvements according to the traffic categories described in Section 4.1. Our experiments started with incast as our schemes are motivated by degraded incast performance on a large ECN threshold. Then we evaluated the interaction between short and long flows and finally the case of long flows only.

5.1 Incast

In our incast congestion experiments, we evaluated two cases as described in Section 2.1: fixed per sender block size and fixed total block size. The trends and improvement obtained were similar, so due to space limitation we only present the results for when the amount of data transmitted by each connection is fixed at 64kbytes. When the number of concurrent senders is less than 44, each server generates at most one connection. To evaluate a situation with more senders (over 44), each server may generate at most two connections. *The incast performance of DCTCP and ECN* with the same ECN threshold is fairly similar*, so we present two cases for DCTCP and ECN* with the threshold set at 20 and 30 packets respectively.

In Figures 7 and 8, we show the incast performance of DCTCP and ECN* versus TCP, respectively. TCP (ECN off), DCTCP, and ECN* with the ECN threshold set at 0 (h=0) are the same as those in Figures 5 and 6. The curves with h=0 show the maximum number of servers that can be supported by using instant queue length based ECN in practice.

Dequeue marking does not degrade performance when the number of senders is small, compare with the results achieved by setting h=0. In addition, we have observed that with dequeue marking enabled, both DCTCP and ECN* achieve incast performance close to the limit of ECN protocol by marking packets unconditionally (h=0), but without sacrificing throughput when the number of senders is small. Dequeue marking effectively mitigates the performance dif-

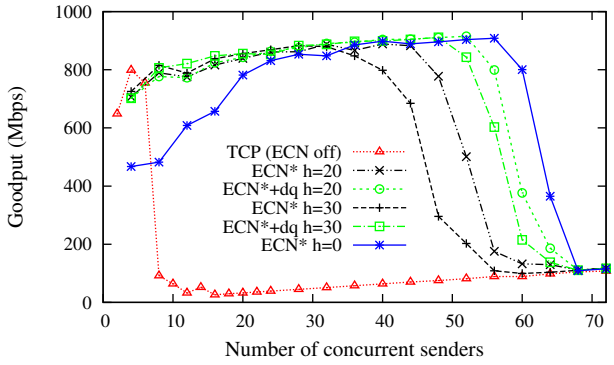

Figure 8: Incast performance of TCP and ECN* with and without dequeue marking. ECN Threshold is set at 20 and 30 packets.

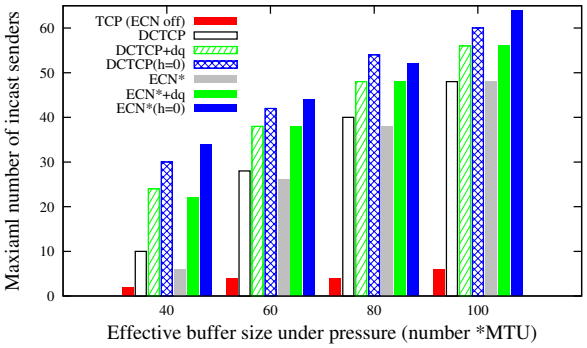

Figure 9: Incast performance of TCP, DCTCP, and ECN* with and without dequeue marking versus varied switch buffer size caused by buffer pressure. The ECN threshold at the switch is set at 20 packets.

ference of DCTCP and ECN* when using different thresholds.

Dequeue marking supports more senders during incast. To evaluate the performance when the buffer size on the switches is constrained, e.g., the shared buffer is occupied by flows on other ports, we designed an experiment for buffer pressure as follows. The ECN threshold at the switch was kept at 20 packets, and we varied the buffer size from 40 to 100 link MTU and compared the incast performance of TCP, DCTCP, and ECN*. For DCTCP/ECN*, we turned on dequeue marking and compared the performance with the ECN limit ($h = 0$), so that in total we have seven candidates in Figure 9.

The performance was judged by the maximum number of concurrent senders without obtaining goodput below 600Mbps. We observed that when the buffer pressure is high, caused by either a constrained buffer size per port or sharing the buffer occupied by other ports in practice, the relative gain obtained by dequeue marking was also higher. For example, when the buffer size was equal to 40(100) link MTU, dequeue marking increased the maximum number of DCTCP senders from 10 to 24 (48 to 56), achieving a gain of 140% (16%). The larger buffer size supports more concurrent incast senders and the performance of DCTCP and ECN* were almost the same for the buffer size satisfying equation 5.

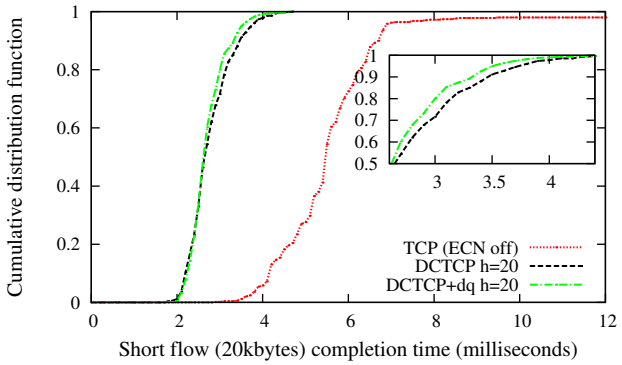

Figure 10: Short flow (20k bytes) completion time of TCP and DCTCP, with two background long TCP/DCTCP flows respectively. The ECN threshold (h) at switch was set at 20 packets.

We tried other incast setups and obtained larger improvement when the threshold was larger. We have observed that: *first, with dequeue marking enabled, the maximum number of incast senders of DCTCP and ECN* with a large threshold are fairly close to the performance limit of ECN (h=0).* Second, given a threshold and the varied buffer size caused by buffer sharing, the improved ratio of dequeue marking increases with smaller buffer size.

5.2 Interaction of long flows and short flows

Dequeue marking is proposed to address the ECN threshold setting problem for incast. Readers may have concerns over whether such modifications will make short flows unnecessarily conservative when competing with long flows that have a larger congestion window and queue occupation at the switches.

To evaluate the interaction between long and short flows, we first established two long connections to the same receiver that occupy the buffer on the bottleneck link. Then we continuously established and tore down a new TCP connection to transmit 20kbytes data to the same destination server of the two long flows. The duration of the long connections was just long enough to cover the transmission of all the 20kbytes short connections. Both long and short TCP connections were started using iperf. All the servers were under the same GbE switch. We selected 20kbytes as it has been used in related research [2].

We evaluated the performance according to two aspects of the interaction: the completion time observed by short flows and the throughput obtained by long flows.

In Figure 10 and Figure 11, we show the distribution of completion time of short flows. Note that the completion time of TCP (ECN off) was actually worse and had over a 2% probability of encountering a timeout (over 300ms and not shown in the two figures). As the completion time distributions of dequeue marking for both enabled and not enabled are fairly close, we provide a zoomed in look. We have observed that dequeue marking always slightly decreases the completion time for short flows. Moreover, the completion time of short flows was similar for DCTCP and ECN*, which

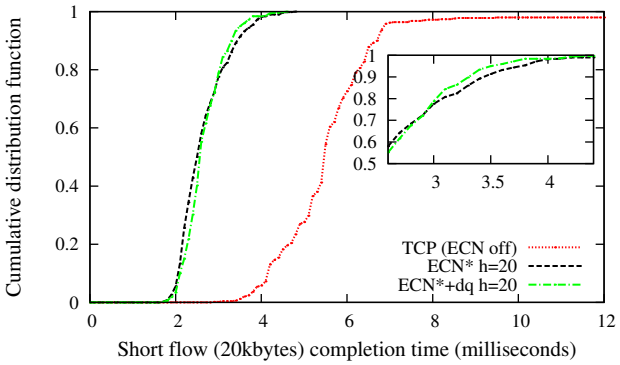

Figure 11: Short flow (20k bytes) completion time of TCP and ECN*, with two background long TCP/ECN* flows respectively. The ECN threshold (h) at switch was set at 20 packets.

Stacks	Original	Dequeue marking
TCP (ECN off)	944.85	-
DCTCP	918.7	903.87
ECN*	895.2	897.75

Table 2: Comparison of total achieved throughput (Mbps) of two long flows when there is a short flow continuously downloading 20kbytes traversing the same bottleneck link.

indicates that the queue occupancy caused by the two long TCP connections was similar for the two schemes.

In Table 2, we present the throughput of long flows when there are short flows continuously traversed. TCP (ECN off) achieves the largest throughput of long flows, with the worst completion time on short flows. The throughput of long flows under both DCTCP and ECN* are very close, especially when dequeue marking is enabled.

To this end, we found a slightly better large percentile completion time for short flows when dequeue marking was enabled. Meanwhile, the throughput of long flows on DCTCP and ECN* was close.

5.3 Long flows with large latency

We evaluated an extreme latency case for instant queue length based schemes, including both DCTCP and ECN*. In Figure 3 we have shown that the latency in data center network is less than 1ms with a probability around 98%. We are interested in the performance degradation in the remaining 2% of cases. The worst case occurs when two long flows are competing on the same bottleneck. We introduced extra latency by adding a fixed 1 millisecond delay using software for each outgoing packet at the two sender servers in our testbed. We measured the throughput achieved with these two long flows and the results are shown in Figure 12.

We observed that DCTCP achieves much higher throughput compared with ECN*, when the network latency was over 1ms but the threshold was set to a low value according to 90 percentile latency. For example, for ECN threshold h=30, DCTCP was 929.8Mbps while ECN* was 736.7Mbps (79%). Note that ECN* quickly gained more throughput if there were more connections, e.g., 3, and thus it's still as competitive as DCTCP in data center networks. Another observation is that dequeue marking slightly degraded throughput if the ECN threshold was set too low, e.g., lower than the recommended h=30 determined by the normal latency in data center networks.

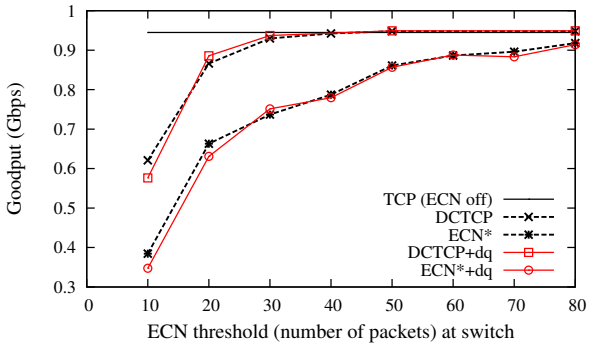

Figure 12: The goodput of two long flows to the same destination server on 1GbE. There is an additional 1ms latency at the sender servers.

6. DISCUSSIONS

The ECN standard defines ECN-capable (ECT) bits at the IP header to indicate switches that mark the packet in order to make the connection respond in the same manner as dropping. However, the behavior of switches handling non-ECT packets when ECN is triggered is not specified. By ECN triggering we mean the instant queue length is over the ECN threshold in our setting. We observe that the broadcom switch simply drops the packets that are not ECN-capable when ECN is triggered. This behavior gives the ECN-capable packets preference as the ECT packets just get marked.

The ECN standard states that TCP retransmitted packets and SYN packets must not be marked as ECT at the IP header. The concerns behind the statements are mainly rooted in security and the desires to make ECN more robust. The queue size in Figure 2 shows that the queue length is periodically over the ECN threshold. Therefore, if SYN packets for new connections or retransmitted packets meet the conditions during congestion, these packets may get dropped and the performance degraded. In this paper, we suggest that all such packets can be marked as ECT in data center networks.

7. RELATED WORK

Paper [15] describes instant queue length based ECN, and DCTCP uses the ratio of marked packets in its congestion control algorithm. In this paper, we analyze the ECN threshold setting. Our analysis and measurements of use in a data center show that ECN*, which uses instant queue at the switch and standard stack at the end system, can perform as well as DCTCP in three traffic categories.

Besides DCTCP, there are other schemes proposed for mitigating the impact of incast congestion. Paper [22] uses a smaller retransmission timeout for incast. We consider an instant queue length based ECN approach is complementary to smaller timeouts. Meanwhile, avoiding a timeout is more appealing than a fast recovery after packet loss. ICTCP [23] is proposed to mitigate incast by overloading the TCP

receive window for congestion control. ICTCP also targets the avoidance of packet loss caused by buffer overflow. We believe ECN* has several benefits compared to ICTCP: 1) ECN is more general and can be used for all bottlenecks in the network while ICTCP only works for the last hop congestion; 2) In addition to incast, ECN* also reduces the completion time of short flows by controlling queue occupation and mitigates the interaction of short and long TCP flows.

Previously, there have been some approaches for dynamic tuning RED thresholds, e.g., [9], but they are only designed for average queue length. Control theoretic analysis of RED and improved designs are proposed in [5, 6], namely the Proportional and the Proportional-Integral (PI) controller. The proportional controller also uses the instant queue length instead of the average queue length. However, the identified stability conditions for Proportional controllers can't be applied to ECN*. This is because ECN* sets the two (low and high) thresholds for ECN at the same value, which essentially is a bang-bang control (on-off control). Therefore, ECN* only uses the ECN implementations on commodity switches, but has different properties when compared with ECN.

Actually, both DCTCP and ECN* share the same difference. The stability of DCTCP is analyzed in [3]. This paper proposes a simple model to analyze the queue occupancy properties following the methods in [15] and [17]. Our analyzed bounds for ECN* thresholds are briefly validated using experiments, and we show that the setup for the threshold determines TCP performance, especially for incast.

Commodity switches use tail-drop when buffer overflows. In drop-from-front [16], when a cell arrives at a full buffer or meeting, the cell closet to being transmitted is dropped instead of the tail. With partial frame drop, drop-from-front achieves similar performance to RED. A previous study in [16] shows tail-drop and front-drop are equally good when either is applied to RED. Compared with the marking approach when the packet is queued, the idea of dequeue marking greatly speeds up congestion information delivery for instant queue length based ECN.

This paper uses instant queue length and demonstrates the feasibility of dequeue marking on commodity Broadcom switches. We believe that the final deployment of dequeue marking in switch chips has no fundamental technical barriers.

8. CONCLUSIONS

In this paper, we demonstrate that instant queue length based ECN* achieves a similar performance when compared with DCTCP in high-bandwidth low-latency networks. ECN* does not need to modify ECN protocols at end servers. We observed that both DCTCP and ECN* have a dilemma with the ECN threshold: a larger ECN threshold to achieve high throughput for long flows may have worse performance during incast. To achieve both high throughput and optimal incast performance with a single ECN threshold, we propose dequeue marking for DCTCP and ECN*. Dequeue marking checks the queue length when packets are dequeued and speeds up the delivery of the congestion signal by the packets at the queue head instead of the queue tail.

We have developed dequeue marking ECN on Broadcom switch Pronto 3290. We built a testbed with 50+ servers and a 48-port Ethernet Gigabit switch. Our experimental results demonstrate that dequeue marking is effective for enlarging the maximum incast senders of DCTCP and ECN* fairly close to the performance limit of ECN using unconditional marking. Depending on the buffer size and ECN threshold, the gain obtained by dequeue marking varies from 16% to 140%.

9. REFERENCES

[1] M. Al-Fares, A. Loukissas, and A. Vahdat. A Scalable, Commodity Data Center Network Architecture. In *Proc. SIGCOMM*, 2008.

[2] M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. DCTCP: Efficient Packet Transport for the Commoditized Data Center. In *Proc. SIGCOMM*, 2010.

[3] M. Alizadeh, A. Javanmard, and B. Prabhakar. Analysis of DCTCP: Stability, Convergence, and Fairness. In *Proc. SIGMETRICS*, 2011.

[4] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing Router Buffers. In *Proc. SIGCOMM*, 2004.

[5] C.V.Hollot, V.Misra, D.Towsley, and W.Gong. A Control Theoretic Analysis of RED. In *Proc. INFOCOM*, 2001.

[6] C.V.Hollot, V.Misra, D.Towsley, and W.Gong. On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In *Proc. INFOCOM*, 2001.

[7] S. Das and R. Sankar. Broadcom Smart-Buffer Technology in Data Center Switches for Cost-Effective Performance Scaling of Cloud Applications. http://www.broadcom.com/collateral/etp/SBT-ETP100.pdf.

[8] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *OSDI'04*, 2004.

[9] S. Floyd, R. Gummadi, and S. Shenker. Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management, 2001. http://www.icir.org/floyd/papers/adaptiveRed.pdf.

[10] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, Aug. 1993.

[11] S. Ghemawat, H. Gobioff, and S. Leung. The Google File System. In *ACM SOSP'03*, 2003.

[12] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu. BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers. In *Proc. SIGCOMM*, 2009.

[13] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu. DCell: A Scalable and Fault Tolerant Network Structure for Data Centers. In *Proc. SIGCOMM*, 2008.

[14] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken. The Nature of Datacenter Traffic: Measurements & Analysis. In *Proc. IMC*, 2009.

[15] A. Kuzmanovic. The Power of Explicit Congestion Notification. In *Proc. SIGCOMM*, 2005.

[16] T. V. Lakshman, A. Neidhardt, and T. J. Ott. The Drop from Front Strategy in TCP and in TCP over ATM. In *INFOCOM*, pages 1242–1250, 1996.

[17] T.V. Lakshman and U. Madhow. The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss. *IEEE/ACM Transactions on Networking*, Jun. 1997.

[18] D. Nagle, D. Serenyi, and A. Matthews. The Panasas ActiveScale Storage Cluster: Delivering scalable high bandwidth storage. In *Proc. SC*, 2004.

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

[20] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. *RFC3168*, Sept. 2001.

[21] K. Ramakrishnan and R. Jain. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks. *Digital Equipment*, 1990.

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

[23] H. Wu, Z. Feng, C. Guo, and Y. Zhang. ICTCP: Incast Congestion Control for TCP in Data Center Networks. In *Proc. Conext*, 2010.