Introduction

To address this problem, Data Center TCP (DCTCP) provides a more detailed window control scheme. Depending on the extent of the congestion informed by Explicit Congestion Notification (ECN) , DCTCP adjusts its send window properly to achieve high throughput while maintaining low buffer. It only needs a single parameter at the switch, and a few amends at the end hosts compared to TCP. Because of its simplicity both in algorithm and deployment, DCTCP has its dominant influence and receives the favor of many following protocols. In academic circles, D2 TCP is proposed as a novel protocol, which builds on the top of DCTCP to achieve deadline-aware goal. Data Center Congestion Control is another improvement of DCTCP, which uses a less compute intensive modification to ECN. In industry area, Microsoft introduces DCTCP in Windows Server 2012.

Although DCTCP provides significant performance improvements both in throughput and latency, there is still a small defect of DCTCP algorithm in maintaining the queue length. Through our observation from the simulation of DCTCP, with the growing number of flows, the bottleneck queue gradually oscillates with increasing amplitude, which strays away from its initial designing control objective. Through further analysis, we find that the single-threshold may result in the phenomenon of too late to inform increasing and decreasing congestion window, thus causing oscillation in DCTCP. In data center networks, because the burst of flows concurrently arrive the queue length will increase rapidly in a short time, and a lot of packets will be marked with ECN. Then the marked packets return back to the senders and all the notified senders will decrease their congestion windows. As a result, after one RTT, the queue length will face a rapid decline. The marking process will continue until the queue length drops back to the settled threshold(e.g. K packets in DCTCP), but the window size decline process will still last for a while because of the delay. We think this conventional single-threshold marking mechanism is not enough to start and end the marking process properly. Thus, we conclude that the nonlinear marking process of single-threshold is the essential reason for oscillation.

Therefore, we propose a new marking mechanism, the Double-Threshold DCTCP(DT-DCTCP). DT-DCTCP uses two parameters $K1$ and $K2$ to share the load of K, one is to start ECN marking in advance, and the other is to stop in advance. Then we analyze why oscillation happens in theory. Some papers have theoretically analyzed DCTCP system in framework of feedback control theory based on the continuous-time model or(and) discrete-time model(such as [4]). However, these linear control theoretic approaches are not appropriate to analyze the nonlinear component of marking mechanism. Thus we introduce the conception of Describing Function (DF) approach, which is well-developed in nonlinear control theory. Using DF approach and stability criterion, we analysis the stability of DCTCP and DT-DCTCP, and prove that the latter is more stable than the former.

Algorithm of DT-DCTCP

The design of dt-dctcp is motivated by the performance impairments described in x 2.3. The goal of dt-dctcp is to achieve high burst tolerance, low latency, and high throughput, with commodity shallow buffered switches. To this end, dt-dctcp is designed to operate with small queue occupancies, without loss of throughput. Dt-dctcp achieves these goals primarily by reacting to congestion in proportion to the extent of congestion. dt-dctcp uses a simple marking scheme at switches that sets the Congestion Experienced (CE) codepoint of packets as soon as the buffer occupancy exceeds a fixed small threshold. The dt-dctcp source reacts by reducing the window by a factor that depends on the fraction of marked packets: the larger the fraction, the bigger the decrease factor. It is important to note that the key contribution here is not the control law itself. It is the act of deriving multi-bit feedback from the information present in the single-bit sequence of marks. Other control laws that act upon this information can be derived as well. Since dt-dctcp requires the network to provide only single-bit feedback, we are able to re-use much of the ECN machinery that is already available in modern TCP stacks and switches. The idea of reacting in proportion to the extent of congestion is also used by delay-based congestion control algorithms [5, 31]. Indeed, one can view path delay information as implicit multi-bit feedback. However, at very high data rates and with low-latency network fabrics, sensing the queue buildup in shallow-buffered switches can be extremely noisy. For example, a 10 packet backlog constitutes 120s of queuing delay at 1Gbps, and only 12s at 10Gbps. The accurate measurement of such small increases in queueing delay is a daunting task for today's servers. The need for reacting in proportion to the extent of congestion is especially acute in the absence of large-scale statistical multiplexing. Standard TCP cuts its window size by a factor of 2 when it receives ECN notification. In effect, TCP reacts to presence of congestion, not to its extent 2. Dropping the window in half causes a large mismatch between the input rate to the link and the available capacity. In the high speed data center environment where only a small number of flows share the buffer (x 2.2), this leads to buffer underflows and loss of throughput.

(1) Marking at the Switch: DCTCP employs a very simple active queue management scheme. There is two parameters, the marking thresholds, K1 and K2. An arriving packet is marked with the CE codepoint if the queue occupancy is greater than K1 upon its arrival. Otherwise, it is not marked. This scheme ensures that sources are quickly notified of the queue overshoot. The RED marking scheme implemented by most modern switches can
be re-purposed for DCTCP.We simply need to set the low threshold to K1 and the high threshold to K2 and mark based on instantaneous, instead of average queue length.

(2) ECN-Echo at the Receiver: The only difference between a dt-dctcp receiver and a TCP receiver is the way information in the CE codepoints is conveyed back to the sender. RFC 3168 states that a receiver sets the ECN-Echo flag in a series of ACK packets
until it receives confirmation from the sender (through the CWR flag) that the congestion notification has been received. A dt-dctcp receiver, however, tries to accurately convey the exact sequence of marked packets back to the sender. The simplest way to do this is

to ACK every packet, setting the ECN-Echo flag if and only if the packet has a marked CE codepoint. However, using Delayed ACKs is important for a variety of reasons, including reducing the load on the data sender. To use delayed ACKs (one cumulative ACK for every m consecutively received packets 3), the dt-dctcp receiver uses the trivial two state state-machine shown in Figure 10 to determine whether to set ECN Echo bit. The states correspond to whether the last received packet was marked with the CE codepoint or not. Since the sender knows how many packets each ACK covers, it can exactly reconstruct the
runs of marks seen by the receiver.

(3) Controller at the Sender: The sender maintains an estimate of the fraction of packets that are marked, called , which is updated once for every window of data (roughly one RTT) as follows:

Alpha = (1-g)xalpha + gxF.

where F is the fraction of packets that were marked in the last window of data, and $0 < g < 1$ is the weight given to new samples against the past in the estimation of Given that the sender receives marks for every packet when the queue length is higher than
K1 and does not receive any marks when the queue length is below K2, Equation (1) implies that estimates the probability that the queue size is greater than K1 and less than K2. Essentially, close to 0 indicates low, and close to 1 indicates high levels of congestion.

Benefits

DCTCP alleviates the three impairments discussed in as follows.

Queue buildup: DCTCP senders start reacting as soon as queue length on an interface exceeds K. This reduces queueing delays on congested switch ports, which minimizes the impact of long flows on the completion time of small flows. Also, more buffer space is available as headroom to absorb transient micro-bursts, greatly mitigating costly packet losses that can lead to timeouts.

Buffer pressure: DCTCP also solves the buffer pressure problem because a congested port's queue length does not grow exceedingly large. Therefore, in shared memory switches, a few congested ports will not exhaust the buffer resources harming flows passing through other ports.

Incast: The incast scenario, where a large number of synchronized small flows hit the same queue, is the most difficult to handle. If the number of small flows is so high that even 1 packet from each flow is sufficient to overwhelm the buffer on a synchronized burst, then there isn't much DCTCP—or any congestion control scheme that does not attempt to schedule traffic—can do to avoid packet drops.

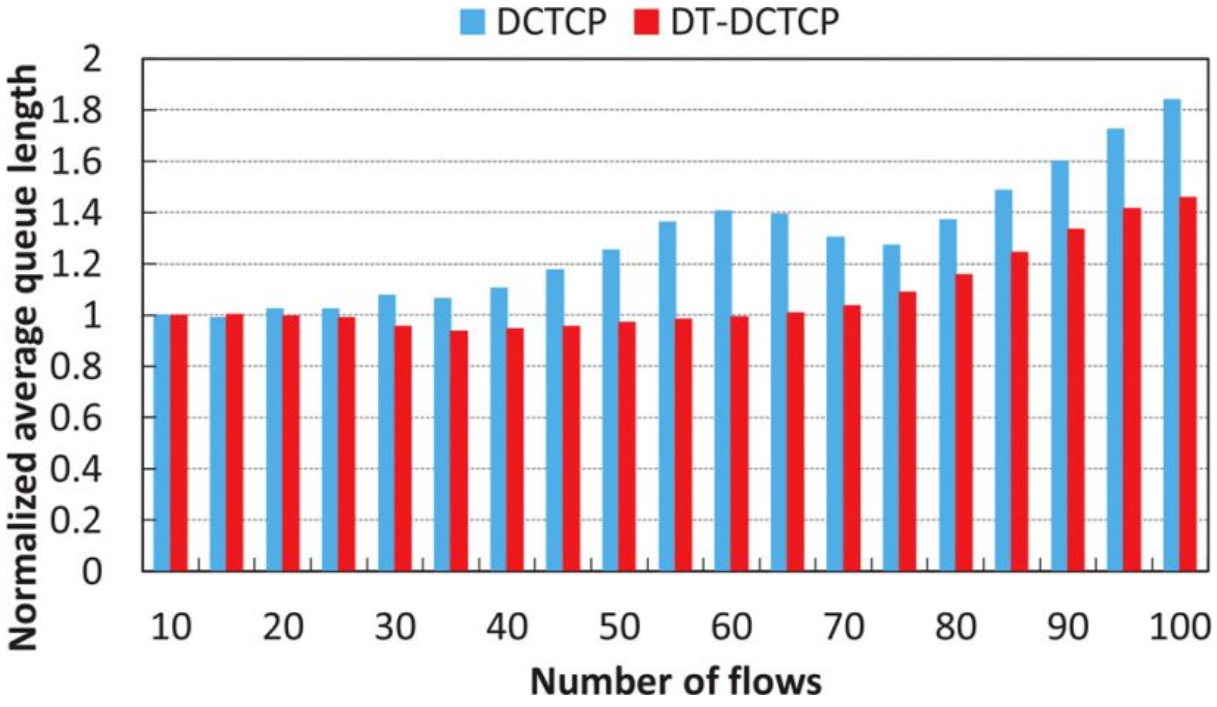
However, in practice, each flow has several packets to transmit, and their windows build up over multiple RTTs. It is often bursts in subsequent RTTs that lead to drops. Because DCTCP starts marking early (and aggressively – based on instantaneous queue length), DCTCP sources receive enough marks during the first one or two RTTs to tame the size of follow up bursts. This prevents buffer overflows and resulting timeouts.

DT-DCTCP is more stable than DCTCP.

We will substitute some certain parameters of DCTCP and DT-DCTCP into the Nyquist diagram(Eq. (19) of DCTCP and Eq. (24) of DT-DCTCP), to compare their stability deeply. For DCTCP, the configuration parameters are set as follows: R=0.001s,C=10Gbps,K=40 packets, g=1/16. As shown in Figure 9, $K_0G(jw)$ shifts to the left as the number of flows N increases. It

means with more flows, the oscillation will more easily happen. For DCTCP in Figure 9(a), the $-1/N_{0dc}$ locus is not surrounded by the $K_{0dc}G(jw)$ locus before N reaches 60. In other words, depend on Theorem 1, after N exceeds 50, $\max(-1_{N_{0dc}})$ is bigger than $K_{0dc}G(jw)$, which means DCTCP may occur oscillation.

In order to compare with DCTCP, we set configuration parameters in DT-DCTCP as K_1=30 packets, K_2=50 packets, which maintains an average of 40 packets, and keep other parameters unchangeable. The results are shown in Figure 9(b). The red dot matrix represents $-1/N_{0dt}(X)$, it intersects with N=70 where $K_{0dt}G(jw)$ locates. In fact, if we plot $-1/N_{0dc}(X)$ and $-1/N_{0dt}(X)$ in the same Nyquist diagram, we will find that the max values in real axis of them are almost the same. So we can conclude that, besides the maximum values of $-1/N_{0dc}(X)$ and $-1/N_{0dt}(X)$, the shapes and locations of $-1/N_{0dc}(X)-1/N_{0dt}(X)$ and $K_{0dt}G(jw)$ determine the intersections. Because $-1/N_{0dt}(X)$ has the positive imaginary part, which makes it far from $K_{0dt}G(jw)$ locus with smaller N , the intersection would occur with lower probability.



In conclusion, $-1/N_{dc}(X)$ is earlier to intersect with the $G(jw)$ locus, and $-1/N_{dt}(X)$ is less likely to have intersections. The earlier the intersections occur, the more unstable the system is. Therefore, DT-DCTCP algorithm is more stable.

Observation And Perceptual Analysis

We simulate DCTCP to observe its queue length at the switch with the variable number of flows using ns-2 simulator. Figure 1 shows the results of N=10 and N=100 long-lived DCTCP flows

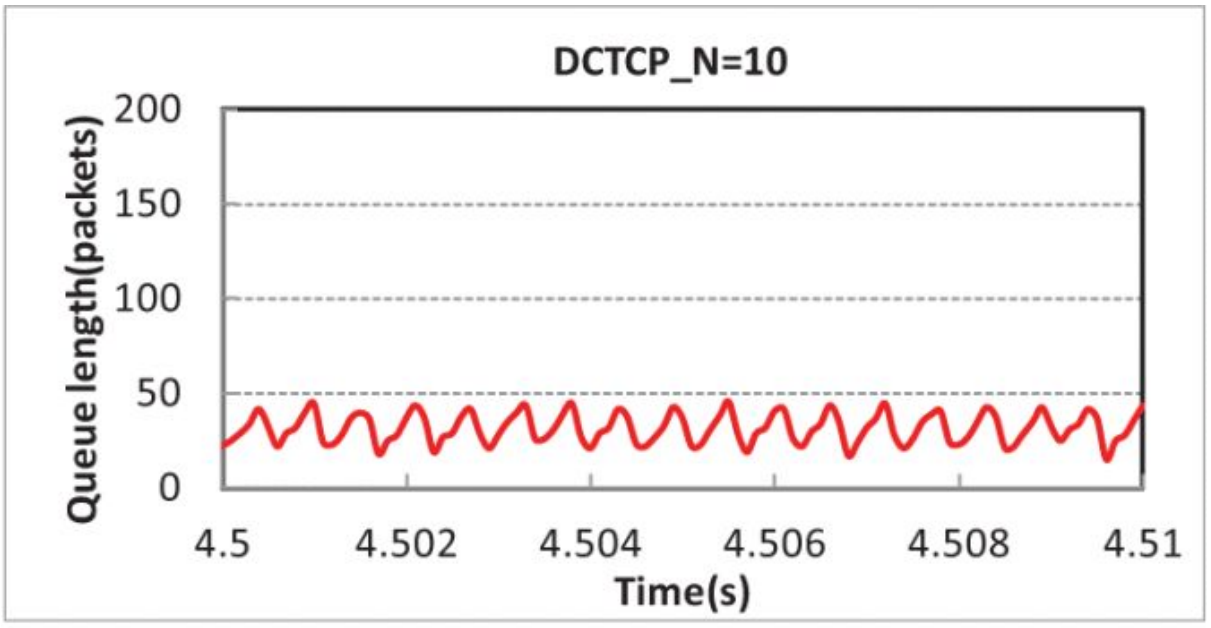
sharing a 10Gbps bottleneck, with $100\mu s$ RTT. When the number of flows reaches 100, the oscillation can't be ignored. The amplitude of queue in N=100 is likely 3 or 4 times of that in N=10. We can conclude from the simulation that DCTCP cannot maintain a stable queue length, and oscillates severely as the number of flows increases.

We may look deeper into the reason of oscillation. D-CTCP has one threshold K to determine both the start and the end of ECN marking. When the queue length climbs up to K, the switch marks the packets to inform the senders to decrease their congestion windows, and when the queue length falls back to K, the switch will release the congestion signal. However, one threshold may postpone the start as well as the end of marking. When a burst of flows arrive at the switch, the network becomes congested at once, but the senders will receive the congestion notice at least after one RTT. In this time period, the queue length will increase rapidly. Then the marked packets return back to the senders and all the notified senders will decrease their congestion windows. In the next few RTTs, the queue length will face a rapid decline. The marking process will continue until the queue length drops back to K, but the decline process will still last for a while because of the delay. Continuous increase and decrease in queue length cause the phenomenon of oscillation.

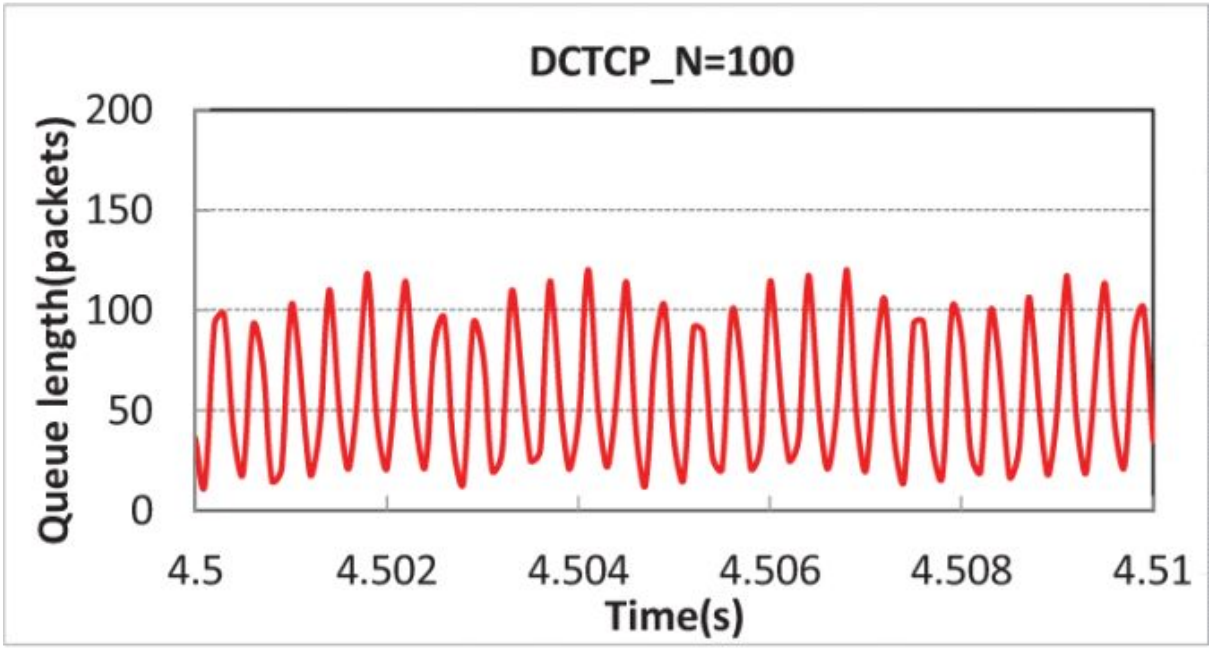
In order to reduce the average queue length, we expect K to be as low as possible, which will avoid Incast as well as queue buildup. In achieving this, we need ECN marking the earlier the better. However, if K is too small that, because of the oscillation, the queue length may drop to the bottom of the buffer, we may lose throughput. In other words, we also have to stop the marking process earlier when the queue length is at a decreasing state. Above all, we need such a mechanism that has one signal to inform congestion earlier to accelerate the marking process, and another signal to release the marking process earlier in order to prevent the queue length from dropping too much.

The original intention of Double-Threshold DCTCP is to improve the single-threshold mechanism of DCTCP. We are inspired from the discussion above that, one threshold K is not enough to control the total marking process. We can split K into two thresholds K_1 and K_2. One is to start ECN marking in advance, and the other is to stop in advance. In other words, the Double-Threshold model is designed to share the load of one threshold K. When the queue length increases beyond the lower threshold K_1, the network is having a potential congestion, and should set CE to inform the senders to decrease their window size. When the queue length decreases under the higher threshold K_2, the switch should release the message of congestion. This approach of “double-threshold” is more flexible than the “single-threshold” mechanism of DCTCP. The difference between these two marking mechanisms can be seen in Figure 2. The switch in DCTCP will mark the packet if the buffer occupancy is equal to or larger than K packets upon its arriving(Figure 2(a)). In DT-DCTCP, the packet will be marked when the queue length increases to K_1, and the marking process continues until the queue length falls back to K_2 (Figure 2(b)).

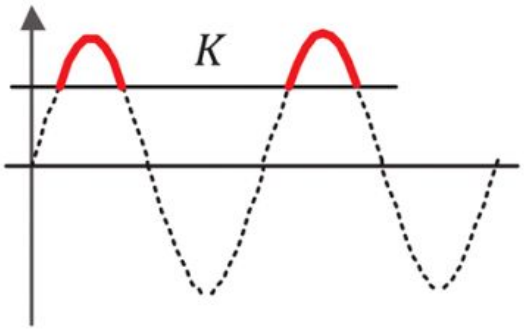
In fact, we regard the single-threshold component as a relay. In control theory, this structure always corresponds to oscillation, which means the controlled objective always fluctuates up and down the settled threshold, and never converges to a certain value. What we do is to replace the relay with a hysteresis loop, which restricts the queue length between the two thresholds, thus the oscillation can be better controlled.



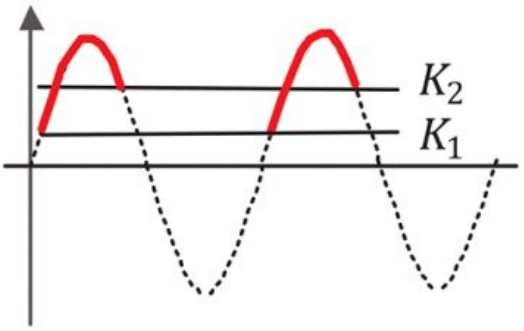
$(a) N = 10$



$(b) N = 100$



(a) DCTCP



(b) DT − DCTCP