Analyzing and Enhancing Dynamic Threshold Policy of Data Center Switches

Danfeng Shan, Wanchun Jiang, and Fengyuan Ren, *Member, IEEE*

Abstract—Today's data center switches usually employ on-chip shared memory; buffer management policy in them is essential to ensure fair sharing of memory among all ports. Among various polices, *Dynamic Threshold* (DT) policy is widely used by switch vendors. Meanwhile, in data centers, distributed applications such as MapReduce often introduce micro-burst traffic into network and the packet dropping caused by micro-burst usually leads to serious performance degradation. When micro-burst traffic arrives at switches, DT is unable to fully utilize the buffer to absorb it. Therefore, in this paper, we theoretically deduce the sufficient conditions for packet dropping caused by micro-burst traffic, and quantitatively estimate the free buffer size when packets are dropped. The results show that the free buffer size can be very large when the number of overloaded ports is small. What's worse, to ensure fair sharing of memory among output ports, packets from micro-burst traffic may be dropped even when the traffic size is much smaller than the buffer size. In light of these results, we propose the *Enhanced Dynamic Threshold* (EDT) policy, which can alleviate packet dropping caused by micro-burst traffic through fully utilizing the switch buffer and temporarily relaxing the fairness constraint. The simulation results show that EDT can absorb more micro-burst traffic than DT.

Index Terms—shared memory, dynamic threshold, buffer management, micro-burst, data center network

1 Introduction

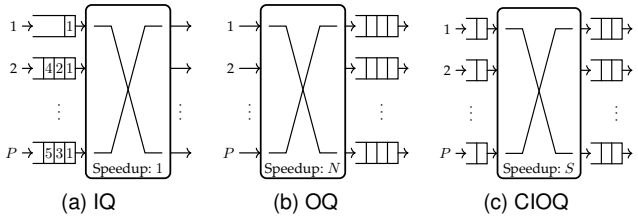
Micro-burst is a common traffic pattern in modern data center networks, and has been brought to public attention recently [1], [2], [3], [4], [5], [6], [7]. Generally, it refers to bursty traffic with very small time-scale. It is usually generated by distributed data center applications and appears in the switch when packets from multiple concurrent flows are destined to the same output port. For example, in data centers deploying online services, the divide and conquer computing paradigm is widely used, thus large-scale concurrent flows may travel across networks. Micro-burst appears in a switch port when results are aggregated from multiple nodes [8], [9]. Packet dropping caused by micro-burst traffic is usually unacceptable because micro-burst traffic is comprised of several delay-sensitive short flows, and the triggered timeouts always extend the flow completion time, which lowers the user experience and thus revenue [9], [10], [11], [12].

Packet dropping in a switch is directly related to its buffer architecture and buffer management policy. Today the majority of switches employ on-chip shared memory to reduce latency by avoiding packet readings and writings to and from external memory. The on-chip packet buffer is dynamically shared across ports by statistical multiplexing [10], [13], [14]. However, shared memory switches might suffer from the fairness problem that few output ports could occupy the majority of the shared buffer, starving other output ports. In order to overcome the problem, many buffer management policies were proposed to restrict the queue length on each output port [15], [16], [17], [18], [19], [20], [21], [22].

Among various policies, *Dynamic Threshold* (DT) [17] has been widely used by switch vendors [14], [23], [24], [25]. In this policy, the queue length is restricted by a dynamic threshold shared by all output ports, which is proportional to the current amount of free buffer space. However, because DT needs to reserve a fraction of buffer so that the newly overloaded ports won't be starving, packets from micro-burst traffic may be dropped even when there is free buffer space in the switch.

In this paper, we theoretically deduce the sufficient conditions for packet dropping caused by micro-burst traffic and quantitatively estimate the corresponding free buffer size in DT switches. The analysis result tells that the free buffer size when packets are dropped is negatively correlated to the number of overloaded ports. Particularly, when the number of overloaded ports is small, the amount of wasted buffer would be especially large. If these buffer can be utilized by the overloaded ports, additional 50% - 100% micro-burst traffic can be absorbed. Furthermore, to ensure fair sharing of memory, the queue length of each overloaded port is restricted by the same threshold. As a result, when several ports are overloaded, packets from micro-burst traffic will be dropped even through the micro-burst traffic size is much smaller than the buffer size. However, it is of great importance to avoid packet dropping of micro-burst traffic in data center networks. On the other hand, when more buffer is temporarily allocated to the ports transmitting micro-bust traffic, there will be few effects on the fairness in the long run because the time-scale of micro-burst is quite short and the allocated buffer will be freed immediately. Therefore, the fairness constraint of DT can be relaxed to absorb micro-burst traffic.

- *D. Shan and F. Ren (corresponding author) are with the Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China.*
  E-mail: sdf13@mails.tsinghua.edu.cn, renfy@tsinghua.edu.cn.
- *W. Jiang is with the School of Information Science and Engineering, Central South University, Changsha, 410083, China.*
  E-mail: jiangwc@csu.edu.cn.



(a) IQ (b) OQ (c) CIOQ

Fig. 1. Packet queueing architecture

In light of these, we propose the *Enhanced Dynamic Threshold* (EDT) policy, which can absorb micro-burst traffic as much as possible through fully utilizing the buffer and temporarily relaxing the fairness constraint when micro-burst traffic arrives at a port. EDT has three advantages: (1). Buffer is fully used to absorb micro-burst traffic. (2). Buffer is fairly shared among output ports transmitting long-lived flows and micro-burst traffic, respectively. (3). EDT is simple enough to be implemented in high-speed switches, as it is comprised by several counters and timers.

We evaluate DT and EDT on ns-2 platform [26]. The results show that in the worst case 50% of buffer remains unused when micro-burst traffic causes packet dropping in DT switches. In comparison, packets will not be dropped until there is no free buffer space in EDT switches. As a result, EDT switches can absorb ∼300% additional micro-burst traffic. Moreover, although EDT temporarily relaxes the fairness constraint, buffer is also fairly shared among output ports in the long run. We also implement a software prototype of EDT using DPDK [27]. The experiments on real testbed show that with EDT the 99th flow completion time for small flows can be reduced by 68.2% compared to that with DT.

The rest of the paper is organized as follows: Section 2 presents background about packet queueing and related work. In Section 3, we deduce the sufficient conditions for packet dropping caused by micro-burst traffic and estimate the corresponding free buffer size is estimated. Section 4 describes the design of EDT. Evaluation is presented in Section 5. Finally, the paper concludes in Section 6.

2 Background and Related Work

2.1 Packet Queueing Architecture

In a switch, packets need to be queued when multiple packets from different input ports are destined to the same output port. According to the location of queueing, there mainly are three approaches. We describe each of them in this part.

Input-queueing (IQ) (Fig.1a): With IQ, packets are queued at input ports, waiting for access to switch fabric. IQ switches suffer from the well-known *head-of-line blocking* problem. Specifically, since only the packet at the head of input queue is contending for switch fabric, packets behind the head packet are blocked. For example, in Fig.1a, if packet from input queue 1 is allowed to send to port 1, packets in other input queues, which are destined to other output ports (port 2, 3, 4, 5), are blocked. Many efforts have been made to solve the problem [28], [29], [30], [31], [32], [33]. Despite the head-of-line blocking problem, IQ switches has lower complexity, as it only needs to have a speedup[1] of 1.

Output-queueing (OQ) (Fig.1b): With OQ, after arriving at the switch, a packet is put to the queue at the destination port. OQ switches are work-conserving as they can provide 100% throughput. However, in a switch with P ports, the fabric needs to operate P times as fast as the line rate (i.e, speedup is P). Therefore, OQ switch is considered to be complex and expensive.

Combined Input and Output Queueing (CIOQ) (Fig.1c): CIOQ switches are designed to achieve larger throughput with lower complexity. CIOQ switches have a speedup of S ($1 < S < P$) and packets need to be queued at both input ports and output ports. With proper output scheduling algorithms [34], [35], [36], [37], [38], [39], [40], CIOQ switches can approach the throughput of OQ switches with low speedup (e.g., $2 \sim 5$).

2.2 Shared Memory

Shared memory is a technique for building an OQ switch. With shared memory switches, packets are written into a centralized memory as they arrive from various input ports, and *linked* to the appropriate output queues after output port lookup. As a result, the centralized memory is shared among all output queues. Shared memory switch is very simple and low-cost [41], [42]. However, its capacity is limited by the memory bandwidth, because a packet needs to access memory twice (written into and read from) and all switch ports may access the memory simultaneously.

In data centers, shared memory switches utilize fast on-chip memory to increase the memory bandwidth and achieve high line rate [14], [23]. Specifically, rather than using external memory, the memory is embedded into switch ASIC. As the internal logic can be clocked much faster than external memories [41], the memory bandwidth can be significantly increased and high switch capacity can be achieved (e.g., 10Gbps for a 64-port switch [14]). Therefore, shared memory switches can be widely used in data centers [10], [43], [44], [45].

2.3 Buffer Management Policies

To ensure fair sharing of memory across different ports, lots of buffer management policies have been proposed in the literature. Generally, they can be divided into two categories: non-preemptive policies and preemptive policies. For non-preemptive policies [15], [16], [17], [18], [19], a packet will not be dropped once it has entered into the buffer and newly arrived packets will be dropped when the memory is full. For preemptive policies [20], [21], [22], a packet in the buffer can be overwritten by newly arrived packets if the memory is full. Preemptive policies are proved to be optimal. However, they are too difficult to be implemented in the hardware. Among these policies, DT [17] is widely used in commodity switches [14], [23], [45] because it is both adaptive and easy to be implemented. Therefore, we consider DT in this paper.

DT is a threshold-based non-preemptive buffer management policy, in which the queue lengths of all ports are

1. The speedup of a switch is defined as the ratio of a switch's internal bandwidth to its line rate [34].

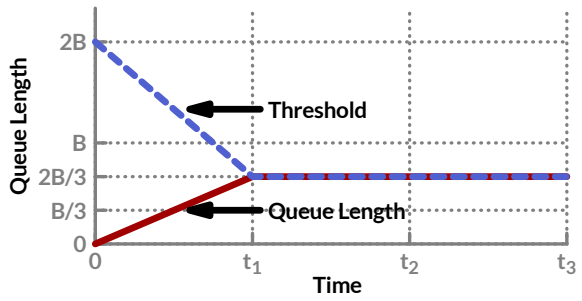


Fig. 2. Queue length and threshold evolutions

constrained by a threshold. Packets are not allowed to enter into the queue whenever the queue length exceeds or equals to the threshold. The key idea of DT is that the threshold is proportional to the current amount of unused buffer space. More precisely, let $Q_i(t)$ be the queue length of port i at time t and B be the shared buffer size, then the threshold $T(t)$ can be given by

$$T(t) = \alpha \cdot \left(B - \sum_i Q_i(t) \right) \tag{1}$$

where α is a control parameter. DT reserves a fraction of buffer all the time such that other ports won't be starved.

To understand the mechanism of DT, consider the following scenario. Assume that the switch buffer is empty and the k-th output port becomes overloaded at time $t = 0$, then $\sum_i Q_i(t) = Q_k(t)$ when $t = 0^+$. Let $\alpha = 2$, then $T(t) = 2 \cdot (B - Q_k(t))$. At time $t = 0$, $Q_k(0) = 0$ and $T(0) = 2B$, thus $Q_k(0) < T(0)$. Packets are allowed to enter into the buffer, and $Q_k(t)$ will increase until $Q_k(t) = T(t) = 2B/3$, as illustrated in Fig. 2. Once $T = Q_k$, the port is not allowed to occupy additional buffer and the queue length will not increase any longer. The reserved buffer size in this case is $B/3$.

2.4 Micro-burst in Data Centers

In data center networks, micro-burst has been studied recently. Traces from ten data centers are examined in [3], and authors find that traffic exhibits an ON/OFF pattern in the order of milliseconds. [5] studies bursts in NIC at packet-level. In [7], authors find that short-lived congestion caused by micro-burst brings challenges to load balancing systems. In multi-tenant datacenters, some algorithms are proposed to achieve bandwidth guarantees [46] and fairness [47] with bursty traffic. Several tools [4], [6] are developed to detect micro-burst. However, none of them considered the buffer management policy in switches.

3 Analysis of Dynamic Threshold

In this part, we will analyze the DT policy following a specific-to-general way, so that the analysis is clear and thus readable. Specifically, we'll first analyze the case when the arriving rate of micro-burst traffic to any port is constant and the same. Then, we'll analyze the case when the arriving rate of micro-burst traffic to each port can be different. Finally, we extend the analysis to the most general case — the arriving rate varies with time.

TABLE 1
Notations

Not.	Description
R_i	the arriving rate of traffic to i-th port
C	link capacity
$Q_i(t)$	queue length of i-th port at time t
B	total buffer size
$T(t)$	threshold at time t
$F(t)$	free buffer size at time t
d_i	duration of micro-burst in i-th port
P	the number of ports in the switch
$f'(x)$	the derivative of $f(x)$

For the convenience of expression, we give the following names about the status of a switch output port.

1) **Overloaded and Underloaded State:** A port is in overloaded state if and only if the arriving rate of traffic to this port is larger than the port's transmitting rate. Otherwise, the port is in underloaded state. More precisely, let the arriving rate of traffic to the i-th output port be R_i. Let C denote the link capacity. Then port i is overloaded if and only if $R_i > C$.
2) **Steady State:** When a port is in the **overloaded** state, it reaches steady state if and only if its queue length is equal to the threshold and the queue length, as well as the threshold, will not change for a while. More precisely, port i reaches steady state at time t if and only if $T(t) = Q_i(t)$ and $T'(t) = Q_i'(t) = 0$[2].

Consider a switch with P output ports and buffer size B. At time $t = 0$, the queues of port $1, \cdots$, port M are empty, and port $(M+1), \cdots$, port $(M+N)$ have reached their steady states[3]. At time $t = 0^+$, micro-burst traffic arrives at port $1, \cdots$, port M at the same time[4], and these ports become overloaded. Let R_i be the arriving rate of micro-burst traffic to port i and d_i be the duration of micro-burst traffic in port i. The free buffer size at time t is denoted by $F(t)$. These notions are summarized in TABLE 1 for the sake of terseness.

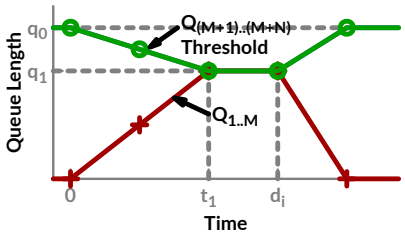
3.1 $R_i\,(i = 1, 2, \cdots, M)$ is constant and $R_1 = R_2 = \cdots = R_M = R$

The evolutions of queue lengths and threshold in this case have been analyzed in [17] in detail. However, for the convenience of explaining the following cases, we'll briefly show the analysis.

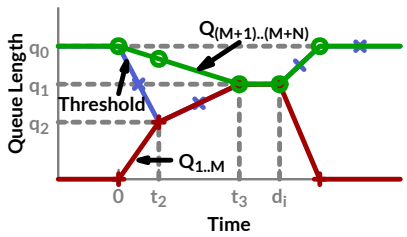
At time $t = 0^+$, as micro-burst traffic arrives at port $1, \cdots$, port M, the unused buffer will be occupied. Thus, the threshold will decrease, which makes $Q_{M+1}, \cdots, Q_{M+N}$ decrease. The maximum decreasing rate of queue length is C, when no packets are entering into the queue and packets in the queue are transmitted at a rate of C (the port transmitting rate). Therefore, there are two cases.

a). $R \leqslant C\left(1 + \frac{1+\alpha N}{\alpha M}\right)$

2. In this paper, we let $f'(x)$ denote the derivative of $f(x)$.
3. $M + N \leqslant P$. Because some ports may be idle.
4. When several micro-bursts arrive at a port at the same time, we can simply consider them as an aggregated micro-burst, whose arriving rate is the sum of all separate micro-bursts.



(a) $R \leqslant C\left(1+\frac{1+\alpha N}{\alpha M}\right)$



(b) $R > C\left(1+\frac{1+\alpha N}{\alpha M}\right)$

Fig. 3. Evolutions of queue lengths and threshold (Theorem 1)

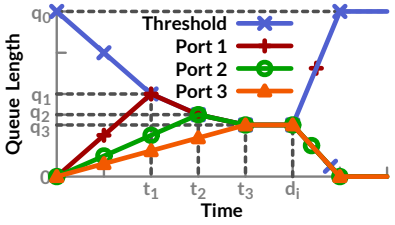


Fig. 4. Evolutions of queue lengths and threshold when $M = 3$ (Theorem 2)

In this case, $|T'(0^+)| \leqslant C$. Therefore, at time $t = 0^+$, $Q_{M+1}, \cdots, Q_{M+N}$ will decrease at the same rate as that of threshold, as is illustrated in Fig. 3a. Meanwhile, $Q_1, \cdots, Q_M$ will increase at a rate of $(R - C)$, until $Q_1, \cdots, Q_M$ hit the threshold at time $t = t_1$. According to [17], time t_1 is given by

$$t_1 = \frac{\alpha B}{[1 + \alpha(M + N)](R - C)} \tag{2}$$

Then packets are dropped since port $1, \cdots$, port M are not allowed to acquire additional buffer. Therefore, the sufficient condition for packet dropping in port i is

$$d_i > t_1 \tag{3}$$

According to [17],

$$T(t_1) = \frac{\alpha B}{1 + \alpha(N + M)} \tag{4}$$

Therefore, the free buffer size when packets are dropped is

$$F(t_1) = \frac{T(t_1)}{\alpha} = \frac{B}{1 + \alpha(N + M)} \tag{5}$$

b). $R > C\left(1 + \frac{1+\alpha N}{\alpha M}\right)$

In this case, $|T'(0^+)| > C$. Therefore, at time $t = 0^+$, $Q_{M+1}, \cdots, Q_{M+N}$ will decrease at a rate of C, which is lower than the decreasing rate of threshold, as is illustrated in Fig. 3b. Meanwhile, $Q_1, \cdots, Q_M$ will increase at a rate of $(R - C)$, until $Q_1, \cdots, Q_M$ hit the threshold at time $t = t_2$. According to [17], time t_2 is given by

$$t_2 = \frac{\alpha B}{(1 + \alpha N)[(1 + \alpha M)(R - C) - \alpha NC]} \tag{6}$$

Then packets are dropped since the increasing rate of $Q_1, \cdots, Q_M$ is limited by DT. At the same time, $Q_{M+1}, \cdots, Q_{M+N}$ will keep decreasing. Thus, the threshold and $Q_1, \cdots, Q_M$ will increase at the same rate until all of the ports reach the steady state. In this case, the sufficient condition for packet dropping in port i is

$$d_i > t_2 \tag{7}$$

And according to [17],

$$T(t_2) = \frac{\alpha(R - C)B}{(1 + \alpha N)[(1 + \alpha M)(R - C) - \alpha NC]} \tag{8}$$

Therefore, the free buffer size when packets begin to be dropped is

$$F(t_2) = \frac{T(t_2)}{\alpha} = \frac{(R - C)B}{(1 + \alpha N)[(1 + \alpha M)(R - C) - \alpha NC]} \tag{9}$$

Considering these two cases, we can summarize the sufficient conditions for packet dropping and free buffer size while the packets from micro-burst traffic are dropped into the following theorem.

Theorem 1. *When $R_1 = R_2 = \cdots = R_M = R$, the packets from micro-burst traffic will be dropped in port k ($k = 1, 2, \cdots, M$) if*

$$d_k > \begin{cases} \dfrac{\alpha B}{[1 + \alpha(M + N)](R - C)}, \\ \quad \textit{if } R \leqslant C\left(1 + \frac{1+\alpha N}{\alpha M}\right) \\ \dfrac{\alpha B}{(1 + \alpha N)[(1 + \alpha M)(R - C) - \alpha NC]}, \\ \quad \textit{if } R > C\left(1 + \frac{1+\alpha N}{\alpha M}\right) \end{cases} \tag{10}$$

and the free buffer size when packets are dropped is

$$F = \begin{cases} \dfrac{B}{1 + \alpha(M + N)}, \\ \quad \textit{if } R \leqslant C\left(1 + \frac{1+\alpha N}{\alpha M}\right) \\ \dfrac{(R - C)B}{(1 + \alpha N)[(1 + \alpha M)(R - C) - \alpha NC]}, \\ \quad \textit{if } R > C\left(1 + \frac{1+\alpha N}{\alpha M}\right) \end{cases} \tag{11}$$

Remarks:

When $R \leqslant C\left(1 + \frac{1+\alpha N}{\alpha M}\right)$, equation (10) can be rewritten as

$$R \cdot d_k - C \cdot d_k > \frac{\alpha B}{1 + \alpha(M + N)} \tag{12}$$

If the micro-burst traffic size (i.e., $R \cdot d_k$) is fixed, then the condition (12) can be easily satisfied for small d_k or larger R. This theoretically explains why micro-burst traffic readily results in packet dropping.

Besides, when the packets are dropped, the free buffer size is negatively correlated to the number of overloaded ports (i.e., $M + N$). Particularly, when the number of overloaded ports is small, the free buffer size would be very large (e.g. $B/2$ if $M + N = 1$ and $\alpha = 1$). DT reserves this fraction of memory for two reasons. First, it provides a cushion for newly overloaded ports, so that these ports will not starve for memory. Secondly, because the threshold of DT is proportional to the amount of unused memory, the action that the reserved memory is occupied can be used to notify DT to change the threshold. However, the reserved buffer should be utilized when a port is transmitting micro-burst traffic, because on the one hand, the time-scale of micro-burst traffic is quite short. Occupying reserved buffer will only last for relatively short time and is worthwhile

since it contributes to absorbing the micro-burst traffic. On the other hand, DT can be simply implemented by using a shift register and a free buffer size counter if α is a power of two. The actions that a packet enters into and departs from the buffer can be used to inform DT of adjusting threshold instead.

Moreover, from Fig. 3a, we have the following observation. To ensure fair buffer sharing among overloaded ports, the packets from micro-burst traffic will be dropped after the queue lengths of newly overloaded ports reach the queue lengths of other ports. As a result, packets may be dropped even though the micro-burst traffic size is far smaller than the buffer size. However, avoiding packet dropping caused by micro-burst traffic is of great importance. In addition, it has few effects on the fairness among ports transmitting long-lived flows that more shared buffer is allocated to the ports transmitting micro-burst traffic because the time-scale of micro-burst traffic is quite short compared to the durations of long-lived flows. Therefore, the fairness constraint of DT could be temporarily relaxed to absorb micro-burst traffic.

The similar insights can be obtained in the case $R > C\left(1+\frac{1+\alpha N}{\alpha M}\right)$.

3.2 $R_i\,(i = 1, 2, \cdots, M)$ is constant and $R_1 \geqslant R_2 \geqslant \cdots \geqslant R_M$

In this case, the sufficient conditions for packet dropping caused by micro-burst traffic and the corresponding free buffer size can be given by the following two theorems.

Theorem 2. *When $\sum_{i=1}^{M}(R_i - C) \leqslant \frac{(1+\alpha N)C}{\alpha}$, packets will be dropped in port k $(k = 1, 2, \cdots, M)$ if*

$$d_k > t_k \tag{13}$$

where

$$\begin{cases} t_k &= \dfrac{\alpha\left[F_{k-1} + \alpha F_{k-1}(N+k-1) + G_k t_{k-1}\right]}{(R_k - C)[1+\alpha(N+k-1)] + \alpha G_k} \\ F_k &= F_{k-1} - \dfrac{G_k(t_k - t_{k-1})}{1+\alpha(N+k-1)} \\ G_k &= \sum_{i=k}^{M}(R_i - C) \end{cases} \tag{14}$$

Time t_k denotes the first time when the queue length Q_k hits the threshold; $t_0 = 0$. And F_k denotes the free buffer size at time $t = t_k$. At $t = 0$, $F_0 = B/(1+\alpha N)$. Next, we'll use mathematical induction to prove this theorem. As an example, the evolutions of queue lengths and threshold when $M = 4$ are illustrated in Fig. 4.

Proof:

a). Basis: inequality (13) and equation (14) hold for port 1 (i.e., $k = 1$)

At $t = 0$, only port $(M+1)$, $\cdots$, port $(M+N)$ are overloaded and they have reached their steady states, therefore, we have

$$\begin{cases} T(0) &= \alpha F_0 \\ F_0 &= B - \sum_{i=M+1}^{M+N} Q_i(0) \\ Q_i(0) &= 0, \quad i = 1, 2, \cdots, M \\ Q_i(0) &= T(0), \quad i = M+1, M+2, \cdots, M+N \end{cases} \tag{15}$$

Solving F_0 from (15), we get

$$F_0 = \frac{B}{1+\alpha N} \tag{16}$$

Port 1, $\cdots$, port M become overloaded at time $t = 0^+$; the traffic arriving rate in port i is R_i. Thus, at time $t = 0^+$, $Q_1, \cdots, Q_M$ will increase at a rate of $(R_i - C)$. As port 1, $\cdots$, port M occupy the free buffer, the free buffer size will decrease, which causes the decreasing of the threshold, and then $Q_{M+1}, \cdots, Q_{M+N}$ will exceed the threshold and decrease. Let D denote the decreasing rate of $Q_{M+1}, \cdots, Q_{M+N}$ $(D < 0)$, Then, at $t = 0^+$, the free buffer size will change as

$$F(t) = F_0 - G_1 \cdot t - ND \cdot t \tag{17}$$

Thus, the dynamic threshold will change as

$$T(t) = \alpha\left(F_0 - G_1 \cdot t - ND \cdot t\right) \tag{18}$$

Differentiating both sides of (18), we have

$$T'(t) = -\alpha G_1 - \alpha N D, \quad t = 0^+ \tag{19}$$

When $G_1 \leqslant \frac{(1+\alpha N)C}{\alpha}$, the decreasing rate of threshold at time $t = 0^+$ is no larger than C, namely,

$$T'(t) \geqslant -C \tag{20}$$

We can prove this by contradiction. The maximum decreasing rate of queue length is C. Thus, if $T'(t) < -C$, $Q_{M+1}, \cdots, Q_{M+N}$ will decrease at a rate of C. Meanwhile, since $G_1 \leqslant \frac{(1+\alpha N)C}{\alpha}$, we have

$$T'(t) \geqslant -C - \alpha N(C + D) \tag{21}$$

Substituting $D = -C$ into (21), we have $T'(t) \geqslant -C$, which contradicts with the previous hypothesis.

Inequality (20) means that the threshold will decrease at a rate lower than the port transmitting rate. Therefore, $Q_i\,(i = M+1, M+2, \cdots, M+N)$ will decrease at the same rate as that of threshold, namely, $D = T'(t)$. Combining (19), we have

$$D = T'(t) = -\frac{\alpha G_1}{1+\alpha N} \tag{22}$$

Substituting (22) into (18), we yield

$$T(t) = \alpha\left(F_0 - \frac{G_1}{1+\alpha N} \cdot t\right), \quad t = 0^+ \tag{23}$$

Equation (23) will hold until the queue length in port 1 hits the threshold at time $t = t_1$, then the packets in port 1 are dropped, namely,

$$T(t_1) = (R_1 - C) \cdot t_1 \tag{24}$$

Solving t_1 from (24), we get

$$t_1 = \frac{\alpha F_0(1+\alpha N)}{(R_1 - C)(1+\alpha N) + \alpha G_1} \tag{25}$$

Therefore, in port 1, packets are dropped if $d_1 > t_1$.

At time t_1, the free buffer size reduces to

$$F_1 = F_0 - \frac{G_1 t_1}{1+\alpha N} \tag{26}$$

Thus, inequality (13) and equation (14) hold for $k = 1$.

b). Inductive step:

We assume that inequality (13) and equation (14) hold for port i $(1 \leqslant i \leqslant M-1)$.

After the queue length of port i hits the threshold at time t_i, the evolutions of queue lengths and threshold are the same as those at time $t = 0^+$, except that the free buffer size is F_i, and there are $N_i = N + i$ output ports whose queue lengths decrease at the same rate as that of threshold. Equation (23) can be rewritten as

$$T(t) = \alpha \cdot \left[F_i - \frac{G_{i+1}}{1+\alpha N_i} \cdot (t - t_i) \right], \quad t = t_i^+ \tag{27}$$

Equation (27) holds until the queue length Q_{i+1} hits the threshold at $t = t_{i+1}$, namely, $T(t_{i+1}) = (R_{i+1} - C) \cdot t_{i+1}$. Then the packets in port $(i+1)$ are dropped. Solving t_{i+1}, we have

$$t_{i+1} = \frac{\alpha \left[F_i(1+\alpha N_i) + G_{i+1} t_i \right]}{(R_{i+1} - C)(1+\alpha N_i) + \alpha G_{i+1}} \tag{28}$$

Therefore, the packets in port $i+1$ will be dropped if $d_{i+1} > t_{i+1}$.

At time t_{i+1}, the free buffer size reduces to

$$F_{i+1} = F_i - \frac{G_{i+1}(t_{i+1} - t_i)}{1+\alpha N_i} \tag{29}$$

Thus, the inequality (13) and equation (14) hold for $k = i+1$.

In conclusion, the inequality (13) and equation (14) hold for $k = 1, 2, \cdots, M$. □

We also have the following theorem when $\sum_{i=1}^{M}(R_i - C)$ is larger than $\frac{(1+\alpha N)C}{\alpha}$:

Theorem 3. *When $\sum_{i=1}^{M}(R_i - C) > \frac{(1+\alpha N)C}{\alpha}$, packets in port k $(k = 1, 2, \cdots, L)$ will be dropped if*

$$d_k > t_k \tag{30}$$

where

$$\begin{cases} t_k = \dfrac{\alpha \left\{ F_{k-1} + [G_k - (N+k-1)C] t_{k-1} \right\}}{\alpha [G_k - (N+k-1)C] + R_k - C}, \\ F_k = F_{k-1} - [G_k - (N+k-1)C](t_k - t_{k-1}), \\ G_k = \sum_{i=k}^{M}(R_i - C) \end{cases} \tag{31}$$

L is the largest k such that $G_k > \frac{[1+\alpha(N_k-1)]C}{\alpha}$ and $L \leqslant M$.

The denotations and initial values of t_k and F_k are the same as those in Theorem 2. Again, we use mathematical induction to prove this theorem.

Proof:

a). Basis: inequality (30) and equation (31) hold for port 1 (i.e., $k = 1$)

In this case, since $G_1 > \frac{(1+\alpha N)C}{\alpha}$ and $T'(t) = -\alpha G_1 - \alpha N D$,

$$T'(t) < -(1+\alpha N)C - \alpha N D \tag{32}$$

Because $D \geqslant -C$, we can get

$$T'(t) > -C \tag{33}$$

In other words, the decreasing rate of threshold is larger than the port transmitting rate. Therefore, at time $t = 0^+$, Q_k $(k = M+1, M+2, \cdots, M+N)$ will decrease at a rate of C. Meanwhile, Q_k $(k = 1, 2, \cdots, M)$ will increase at a rate of $(R_k - C)$. Thus, at time $t = 0^+$, the free buffer size will change as

$$F(t) = F_0 - (G_1 - NC) \cdot t \tag{34}$$

Therefore, the threshold will change as

$$T(t) = \alpha \left[F_0 - (G_1 - NC) \cdot t \right] \tag{35}$$

where F_0 is given in (16).

Equation (35) holds until $t = t_1$ when Q_1 hits the threshold and the packets in port 1 are dropped, namely,

$$T(t_1) = (R_1 - C) t_1 \tag{36}$$

Solving t_1 from (36), we have

$$t_1 = \frac{\alpha F_0}{\alpha(G_1 - NC) + (R_1 - C)} \tag{37}$$

Thus, the packets in port 1 will be dropped if $d_1 > t_1$.

The free buffer size at time t_1 is given by

$$F_1 = F_0 - (G_1 - NC) t_1 \tag{38}$$

Thus, inequality (30) and equation (31) hold for $k = 1$

b). Inductive step:

We assume that inequality (30) and equation (31) hold for port i $(1 \leqslant i \leqslant L-1)$.

After Q_i hits the threshold at time t_i, the evolutions of queue lengths and threshold are the same as those at time $t = 0^+$, except that the free buffer size is F_i and there are $N_i = N + i$ output ports whose queue lengths decrease at the rate of C. Thus, equation (35) can be rewritten as

$$T(t) = \alpha \cdot \left[F_i - (G_{i+1} - N_i C) \cdot (t - t_i) \right], \quad t = t_i^+ \tag{39}$$

Equation (39) holds until the queue length Q_{i+1} hits the threshold at $t = t_{i+1}$, namely, $T(t_{i+1}) = (R_{i+1} - C) \cdot t_{i+1}$. Then the packets in port $(i+1)$ begin to be dropped. Solving t_{i+1}, we have

$$t_{i+1} = \frac{\alpha [F_i + (G_{i+1} - N_i C) t_i]}{\alpha(G_{i+1} - N_i C) + R_{i+1} - C} \tag{40}$$

Thus the packets in port $(i+1)$ will be dropped if $d_{i+1} > t_{i+1}$.

At time t_{i+1}, the free buffer size reduces to

$$F_{i+1} = F_i - (G_{i+1} - N_i C)(t_{i+1} - t_i) \tag{41}$$

Therefore, the inequality (30) and equation (31) hold for $k = i+1$.

In conclusion, the inequality (30) and equation (31) hold for $k = 1, 2, \cdots, L$. □

3.3 $R_i (i = 1, 2, \cdots, M)$ varies with time

As is shown in the previous analysis, according to the decreasing rate of threshold, the evolutions of queue lengths in port $M+1$, $\cdots$, port $M+N$ can be divided into two cases: the queue length decreases at the same rate as the threshold and the queue length decrease at a rate of C. When the arriving rate varies with time, the evolutions can be either case at any time. Therefore, it's impossible to deduce a general sufficient condition here. Fortunately, the way to get the sufficient condition is similar. Therefore, in this part, we only present the analysis of a common scenario

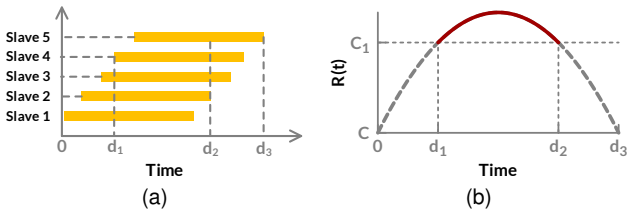


Fig. 5. The scenario considered in analysis: (a) The responding time of each slave; (b) Traffic arriving rate in the output port.

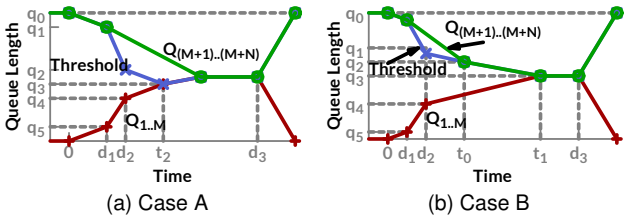


Fig. 6. Evolutions of queue lengths and threshold can be divided into two scenarios (Theorem 4)

in data centers. Sufficient conditions for other scenarios can be deduced in a similar way.

We consider the case that a master server sends queries to multiple slave servers and each slave server responds with a message. Micro-burst occurs when response messages from multiple slaves arrive at the switch simultaneously. Because of asynchrony among slaves, some of slaves may start responding and finish responding ahead of others. Therefore, the aggregated arriving rate of micro-burst traffic increases at the beginning as more and more slaves begin to respond. After a while, the arriving rate decreases as more and more slaves finish responding. More precisely, let $R_1(t) = R_2(t) = \cdots = R_M(t) = R(t)$. As depicted in Fig.5b, the arriving rate $R(t)$ can be divided into 3 parts:

$$\begin{cases} R(t) \leqslant C\left(1+\frac{1+\alpha N}{\alpha M}\right) & 0 \leqslant t \leqslant d_1 \\ R(t) > C\left(1+\frac{1+\alpha N}{\alpha M}\right) & d_1 \leqslant t \leqslant d_2 \\ R(t) \leqslant C\left(1+\frac{1+\alpha N}{\alpha M}\right) & d_2 \leqslant t \leqslant d_3 \end{cases} \tag{42}$$

Therefore, when $0 \leqslant t \leqslant d_1$ and $d_2 \leqslant t \leqslant d_3$, the threshold will decrease at a rate smaller than C. When $d_1 \leqslant t \leqslant d_2$, the threshold will decrease at a rate larger than C. In this scenario, the sufficient condition and corresponding free buffer size can be given by the following theorem:

Theorem 4. *Let t_1 and t_2 be given by*

$$\begin{cases} (1+\alpha M+\alpha N)\left(\int_0^{t_1} R(\tau)d\tau - Ct_1\right) = \alpha B \\ (1+\alpha M)\int_0^{t_2} R(\tau)d\tau - (1+\alpha M+\alpha N)Ct_2 \\ \qquad = \frac{\alpha B+\alpha N\left[\alpha M\int_0^{d_1} R(\tau)d\tau-(1+\alpha M+\alpha N)Cd_1\right]}{1+\alpha N} \end{cases} \tag{43}$$

then we have following 4 sufficient conditions:

a). Packets will be dropped if

$$d_1 > t_1 \tag{44}$$

The free buffer size when packets are dropped is

$$F(t_1) = \frac{T(t_1)}{\alpha} = \frac{B - M\int_0^{t_1} R(\tau)d\tau + MCt_1}{1+\alpha N} \tag{45}$$

b). When $d_1 \leqslant t_1$, packets will be dropped if

$$d_2 > t_2 \tag{46}$$

The free buffer size when packets are dropped is

$$F(t_2) = \frac{B + N\left[\alpha M\int_0^{d_1} R(\tau)d\tau - (1+\alpha M+\alpha N)Cd_1\right]}{1+\alpha N} + (M+N)Ct_2 - M\int_0^{t_2} R(\tau)d\tau \tag{47}$$

c). When $d_1 \leqslant t_1$, $d_2 \leqslant t_2$, and $\alpha M\int_{d_1}^{t_3} R(\tau)d\tau - (1+\alpha M+\alpha N)(t_3-d_1)C > 0$, packets will be dropped if

$$d_3 > t_2 \tag{48}$$

The free buffer size when packets are dropped is

$$F(t_2) = \frac{B + N\left[\alpha M\int_0^{d_1} R(\tau)d\tau - (1+\alpha M+\alpha N)Cd_1\right]}{1+\alpha N} + (M+N)Ct_2 - M\int_0^{t_2} R(\tau)d\tau \tag{49}$$

d). When $d_1 \leqslant t_1$, $d_2 \leqslant t_2$, and $\alpha M\int_{d_1}^{t_3} R(\tau)d\tau - (1+\alpha M+\alpha N)(t_3-d_1)C \leqslant 0$, packets will be dropped if

$$d_3 > t_1 \tag{50}$$

The free buffer size when packets are dropped is

$$F(t_1) = \frac{T(t_1)}{\alpha} = \frac{B - M\int_0^{t_1} R(\tau)d\tau + MCt_1}{1+\alpha N} \tag{51}$$

Proof:

At $t = 0^+$, the queue length of port $1, \cdots$, port M increases at a rate of $R(t) - C$, and the queue length of port $M+1, \cdots$, port $M+N$ evolves as the threshold, namely,

$$Q_i(t) = \begin{cases} \int_0^t R(\tau)d\tau - Ct & i = 1, 2, \cdots, M \\ T(t) & i = M+1, M+2, \cdots, M+N \end{cases} \tag{52}$$

Substituting (52) into (1), we can get the threshold by

$$T(t) = \frac{\alpha\left(B - M\int_0^t R(\tau)d\tau + MCt\right)}{1+\alpha N} \tag{53}$$

Let t_1 be the time when $Q_i(t_1) = T(t_1)(i = 1, 2, \cdots, M)$. Then t_1 can be given by

$$(1+\alpha M+\alpha N)\left(\int_0^{t_1} R(\tau)d\tau - Ct_1\right) = \alpha B \tag{54}$$

Then packet dropping will happen if $Q_i(d_1) > T(d_1)(i = 1, 2, \cdots, M)$, which is equivalent to

$$d_1 > t_1 \tag{55}$$

If the condition (55) is met, then the free buffer size when packets are dropped is given by

$$F(t_1) = \frac{T(t_1)}{\alpha} = \frac{B - M\int_0^{t_1} R(\tau)d\tau + MCt_1}{1+\alpha N} \tag{56}$$

If $Q_i(d_1) \leqslant T(d_1)(i = 1, 2, \cdots, M)$ (i.e., packets are not dropped before time $t = d_1$), then at $t = d_1^+$, $Q_1, \cdots, Q_M$ will keep increasing at a rate of $R(t) - C$. Meanwhile,

$Q_{M+1}, \cdots, Q_{M+N}$ will decrease at a rate of C. Thus, at time $t = d_1^+$, the queue length evolution of each port can be given by

$$Q_i(t) = \begin{cases} \int_0^t R(\tau)d\tau - Ct, & i = 1, 2, \cdots, M \\ Q_i(d_1) + Cd_1 - Ct, & i = M+1, \cdots, M+N \end{cases} \tag{57}$$

where

$$Q_i(d_1) = T(d_1) = \frac{\alpha\left(B - M\int_0^{d_1} R(\tau)d\tau + MCd_1\right)}{1+\alpha N} \quad (i = M+1, M+2, \cdots, M+N) \tag{58}$$

Substituting (57) into (1), we can get

$$T(t) = \frac{\alpha B + \alpha N\left[\alpha M\int_0^{d_1} R(\tau)d\tau - (1+\alpha M+\alpha N)Cd_1\right]}{1+\alpha N} + \alpha(M+N)Ct - \alpha M\int_0^t R(\tau)d\tau \tag{59}$$

Let t_2 be the time when $Q_i(t_2) = T(t_2)(i = 1, 2, \cdots, M)$. Then t_2 can be given by

$$(1+\alpha M)\int_0^{t_2} R(\tau)d\tau - (1+\alpha M+\alpha N)Ct_2 = \frac{\alpha B + \alpha N\left[\alpha M\int_0^{d_1} R(\tau)d\tau - (1+\alpha M+\alpha N)Cd_1\right]}{1+\alpha N} \tag{60}$$

In this period, packet dropping will happen if $Q_i(d_2) > T(d_2)(i = 1, 2, \cdots, M)$, namely,

$$d_2 > t_2 \tag{61}$$

If the condition (61) is met, then the free buffer size when packets are dropped is given by

$$F(t_2) = \frac{B + N\left[\alpha M\int_0^{d_1} R(\tau)d\tau - (1+\alpha M+\alpha N)Cd_1\right]}{1+\alpha N} + (M+N)Ct_2 - M\int_0^{t_2} R(\tau)d\tau \tag{62}$$

If $Q_i(d_2) \leqslant T(d_2)(i = 1, 2, \cdots, M)$, then at time $t = d_2^+$, $Q_1, \cdots, Q_M$ will increase at a rate of $R(t)$. Meanwhile, $Q_{M+1}, \cdots, Q_{M+N}$ will decrease at a rate of C. Thus at time $t = d_2^+$, evolutions of queue lengths and threshold are the same as those in $t \in [d_1, d_2]$. Therefore, the queue length evolution of each port can be given by (57) and the threshold can be given by (59). At time $t \in [d_2, d_3]$, there are two cases, which are depicted in Fig.6a and Fig.6b, respectively. Recall that t_2 is the time when $Q_i(t_2) = T(t_2)(i = 1, 2, \cdots, M)$, where $Q_i(t)$ and $T(t)$ is given by (57) and (59), respectively. Therefore, the queue length evolves as case A if $Q_i(t_2) > T(t_2)(i = M+1, M+2, \cdots, M+N)$. The inequality $Q_i(t_2) > T(t_2)(i = M+1, M+2, \cdots, M+N)$ can be rewritten as

$$\alpha M\int_{d_1}^{t_2} R(\tau)d\tau - (1+\alpha M+\alpha N)(t_2 - d_1)C > 0 \tag{63}$$

In this case, packet dropping happens if

$$d_3 > t_2 \tag{64}$$

The free buffer size when packets are dropped is given by (62).

In the case B, we have the inequality $Q_i(t_2) \leqslant T(t_2)(i = M+1, M+2, \cdots, M+N)$, namely,

$$\alpha M\int_{d_1}^{t_2} R(\tau)d\tau - (1+\alpha M+\alpha N)(t_2 - d_1)C \leqslant 0 \tag{65}$$

In this case, the threshold and queue length of port $M+1, \cdots$, port $M+N$ will meet at time t_0, where t_0 is given by $Q_i(t_0) = T(t_0)(i = M+1, M+2, \cdots, M+N)$, namely

$$\alpha M\int_{d_1}^{t_0} R(\tau)d\tau - (1+\alpha M+\alpha N)(t_0 - d_1)C = 0 \tag{66}$$

At time t_0^+, the queue length of port $1, \cdots$, port M increases at a rate of $R(t) - C$, and the queue length of port $M+1, \cdots$, port $M+N$ evolves the same as the threshold. The evolutions of queue length and threshold are the same as those in duration $t \in [0, d_1]$. Thus, the queue length evolution of each port can be given by (52) and the threshold can be given by (53). Recall that t_1 is the time when $Q_i(t_1) = T(t_1)(i = 1, 2, \cdots, M)$. Then packets will be dropped if $Q_i(d_3) > T(t_1)$, which is equivalent to

$$d_3 > t_1 \tag{67}$$

If condition (67) is met, then the free buffer size when packets are dropped can be given by (45). □

Remarks: By substituting (54) into (45), we can simplify the free buffer size as

$$F(t_1) = \frac{B}{1+\alpha M+\alpha N} \tag{68}$$

Therefore, the free buffer size is negatively correlated to the number of overloaded ports, which is the same as that in Theorem 1. Furthermore, from Fig.6, we also have the same observation as that from Fig.3a that packets may be dropped even though the micro-burst traffic size is far smaller than the buffer size. Therefore, the insights from Theorem 1 can also be obtained in this general case.

4 EDT Policy

Analysis results indicate that the switch buffer should be fully utilized and the fairness constraint of DT should be temporarily relaxed to absorb micro-burst traffic. Therefore, in this section, we propose the *Enhanced Dynamic Threshold* (EDT) policy to absorb micro-burst traffic.

4.1 Basic Idea

Intuitively, increasing the buffer utilization can be simply achieved by setting a larger α in DT policy. However, it may result in fairness problem [17]. Specifically, if α is very large, one output port can occupy the majority of memory, starving newly overloaded ports. While ensuring the inter-port fairness is also necessary when these ports are only transmitting long-lived traffic or transmitting micro-burst traffic.

According to the above considerations, we propose the EDT policy. EDT allows an output port to aggressively occupy buffer in a relatively short interval when the port

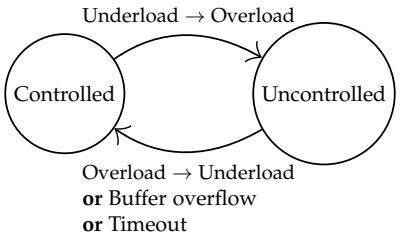


Fig. 7. State transition diagram of EDT in each port

becomes overloaded. Specifically, for each port, EDT has two states: controlled state and uncontrolled state. Two thresholds are used in each state: T_1 and T_2. Specifically, in the controlled state, T_1 is used to restrict queue length according to DT, namely, $T_1 = \alpha(B - \sum_i Q_i(t))$. In the uncontrolled state, T_2 is used to enable a port to acquire more buffer to absorb micro-burst traffic, while ensuring fair buffer sharing among all ports in the uncontrolled state as well. Therefore, port threshold is set to $T_2 = B/n$, where n is the number of uncontrolled ports.

Fig. 7 depicts the state transition diagram of EDT in each port. At the beginning, EDT is in controlled state. It turns into uncontrolled state when bursty traffic arrives and the port becomes overloaded. When a port is in uncontrolled state, the port will return to controlled state if one of following three conditions is met: the port becomes underloaded, buffer overflows, and timeout occurs. We now explain the reason why a port should return to controlled state if these conditions are met. *First*, if the port becomes underloaded, then micro-burst traffic has left and thus the port should return to the controlled state. *Second*, if the buffer overflows, there are two cases: (1). The port is transmitting long-lived flows. (2). The switch is not able to absorb the micro-burst traffic. For either case, the queue length should be controlled. *Finally*, the last condition (timeout) is in case of the following scenario that might happen in practice. There is a background flow in the port whose rate is just equal to the link capacity. After a port has finished transmitting micro-burst traffic, the buffer occupied by the port is not freed immediately. So we use a timer to prevent the long-time buffer occupancy in this case.

EDT has three advantages:

1) The output port can occupy every piece of available buffer when it becomes overloaded. Thus packets from micro-burst traffic are dropped only when it is inevitable.
2) Buffer could be fairly shared among output ports transmitting long-lived flows because the period over which EDT stays in uncontrolled state is very short. Buffer can also be fairly shared if there are multiple ports transmitting micro-burst traffic.
3) EDT is simple enough to be implemented in high-speed switches, as it only requires several additional timers and counters.

The main challenge of EDT is how to recognize that the output port becomes overloaded. From analysis, we observe that when the output port becomes overloaded, *its queue length is increasing and no packets are dropped* at the beginning, so we use this characteristic for recognition.

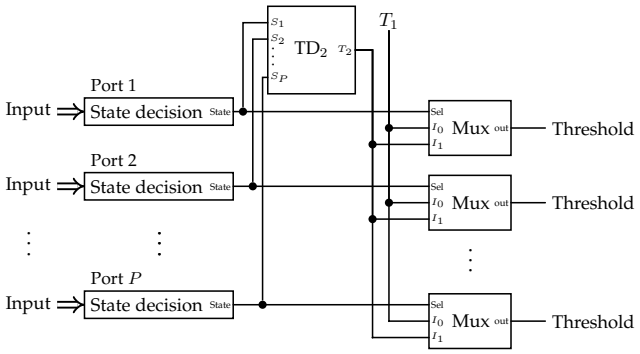


Fig. 8. Structure of EDT

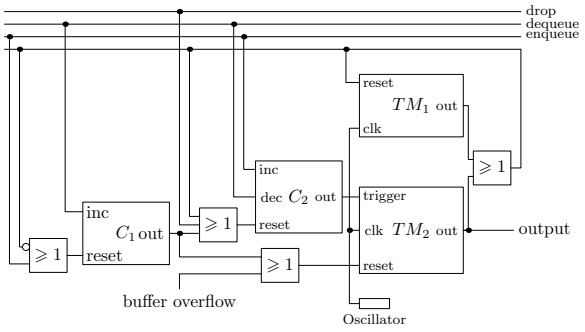


Fig. 9. Circuit diagram of state decision module

4.2 Design of EDT

Fig.8 shows the structure of EDT. We assume that there are P ports in the switch. For each port, there is a state decision module used for deciding which state the port is currently in. The details of state decision module will be shown in the next part. The module TD_2[5] takes the states of each port as input and determine the threshold T_2. The logic is simple: let $T_2 = B/n$, where n is the number of ports in uncontrolled state[6]. The last module of EDT is a Multiplexer. It selects a threshold from T_1 and T_2 according to the port's state.

Fig. 9 illustrates the circuit diagram of state decision module. Inputs of this diagram are enqueue signal, dequeue signal, and packet dropping signal generated by each logic output queue of the port. A pulse is generated on them whenever a packet is enqueued, dequeued, and dropped, respectively, from the output queue. There is also a buffer overflow signal, on which a pulse is generated whenever total shared buffer overflows. The output of the module determines whether EDT is in controlled state or uncontrolled state. TM_1 and TM_2 are countdown timers. Each of them begins to count down from a specific time interval once they are enabled. C_1 and C_2 are counters. They increase (decrease) their values for every enqueue (dequeue) signal. Next, we'll show the details of these timers and counters one by one.

5. TD is short for threshold determining.

6. Division in hardware is slow and might bring latency in high speed switches. Fortunately, as there are only a limited number of ports, we can simply implement module T_2 by storing the value $B, B/2, \cdots, B/n$ into registers, and the module just chooses one of them according to the number of ports in uncontrolled state.

TM_2 begins to work when EDT turns to uncontrolled state and determines the time interval over which EDT stays in the uncontrolled state. It begins to count down when its trigger pin receives a pulse signal, and stops either when it receives a reset signal or it expires. When TM_2 is counting down, its output pin is set to 1, which indicates that EDT is in uncontrolled state. When it expires (or it is reset), its output pin is set to 0 to signal EDT to return to the controlled state, then it will stop working until it receives another trigger signal. TM_2 can be reset by two signals: buffer overflow and output of C_1. The output of C_1 implies that the port becomes underloaded, which will be explained in detail in the following part. The counting time of TM_2 should be longer than the duration of most micro-burst traffic (e.g. 10ms).

C_2 is used for identifying that the output port becomes overloaded. It works when EDT is in the controlled state. Specifically, it increases for every pulse on enqueue signal and decreases for every pulse on dequeue signal. Therefore, its value represents the queue length increment. When it reaches its counting number, EDT will change to the uncontrolled state, and a pulse will be generated to notify TM_2 to start counting, then C_2 will stop working until EDT returns to the controlled state. The counting number influences the sensitivity of identifying overloaded state. On the one hand, if this value is too large, TM_2 will not be triggered until the packets from micro-burst traffic are dropped. On the other hand, if this value is too small, TM_2 will be triggered frequently, which results in unfairness among output ports transmitting long-lived flows. Thus C_2 should obey the following three rules:

- Rule 1: C_2 works only when the port becomes overloaded.
- Rule 2: C_2 reaches its counting number before packets are dropped.
- Rule 3: The counting number should be as large as possible on the premise of following Rule 2.

From Fig. 3, we notice that when a port becomes overloaded, its queue length is increasing and no packets are dropped at the beginning. Therefore, we let C_2 reset itself whenever a packet is dropped to obey Rule 1. Let the counting number of C_2 be cn_2. Then cn_2 should satisfy the following inequality to obey Rule 2:

$$cn_2 \leqslant \begin{cases} (R-C)\cdot t_1, & R \leqslant C\left(1+\frac{1+\alpha N}{\alpha M}\right) \\ (R-C)\cdot t_2, & R > C\left(1+\frac{1+\alpha N}{\alpha M}\right) \end{cases} \tag{69}$$

Meanwhile,

$$(R-C)\cdot t_1 \geqslant (R-C)\cdot t_2 \tag{70}$$

and

$$\begin{aligned} (R-C)\cdot t_2 &= \frac{\alpha B(R-C)}{(1+\alpha N)\left[(1+\alpha M)(R-C)-\alpha NC\right])} \\ &> \lim_{R\to\infty} \frac{\alpha B(R-C)}{(1+\alpha N)\left[(1+\alpha M)(R-C)-\alpha NC\right])} \\ &= \frac{\alpha B}{(1+\alpha N)(1+\alpha M)} \\ &\geqslant \frac{4\alpha B}{(2+\alpha P)^2} \end{aligned} \tag{71}$$

where P is the number of switch ports. Thus cn_2 should satisfy inequality

$$cn_2 \leqslant \frac{4\alpha B}{(2+\alpha P)^2} \tag{72}$$

To obey Rule 3, we can set

$$cn_2 = \frac{4\alpha B}{(2+\alpha P)^2} \tag{73}$$

TM_1 is used for making sure that TM_2 is only triggered by bursty traffic. Because if the arriving rate of micro-burst traffic is too small, no packets will be dropped. Uncontrolling queue length in such scenario is unnecessary and may cause unexpected results. Therefore, it's essential to add bursty traffic detection to EDT. TM_1 works as follows. When C_2 begins to increase, TM_1 begins to count down from its initial value as well. If the value of C_2 has not reached its counting number yet when TM_1 reaches 0, a pulse is sent to C_2 to notify it to reset itself. If the value of C_2 reaches its counting number before TM_1 reaches 0, TM_1 is reset and counts down from the default value again. In this way, TM_2 is triggered only by bursty traffic. Unlike TM_2, TM_1 keeps working all the time. Its default value is given as follows. No packets are dropped when the duration of arriving traffic (denoted by d) satisfies the following inequality:

$$d \leqslant t_1 = \frac{\alpha B}{\left[1+\alpha(M+N)\right](R-C)} \tag{74}$$

Equation (74) can be rewritten as

$$R-C \leqslant \frac{\alpha B}{\left[1+\alpha(M+N)\right]\cdot d} \tag{75}$$

Meanwhile,

$$\frac{\alpha B}{\left[1+\alpha(M+N)\right]\cdot d} \geqslant \frac{\alpha B}{(1+\alpha P)\cdot d} \tag{76}$$

Thus, the packets will not be dropped if

$$R-C \leqslant \frac{\alpha B}{(1+\alpha P)\cdot d} \tag{77}$$

If the period over which C_2 increases from 0 to cn_2 is denoted by t_{c2}, then packets will not be dropped if

$$t_{c2} \geqslant \frac{cn_2}{\alpha B/\left[(1+\alpha P)\cdot d\right]} = \frac{4(1+\alpha P)}{(2+\alpha P)^2}\cdot d \tag{78}$$

where d is longer than the duration of most micro-burst traffic. Thus the default value of T_1 should be set to

$$\frac{4(1+\alpha P)}{(2+\alpha P)^2}\cdot d$$

C_1 is used for identifying that the output port returns to the underloaded state. On the one hand, the queue length will not keep increasing all the time when the output port is overloaded, since the port is transmitting packets at the same time. Thus the shape of queue length evolution curve is like a sawtooth, as is shown in Fig.10. On the other hand, if a few packets are dequeued without any new arrivals, EDT should be able to judge that the port becomes underloaded. Therefore, we use C_1 to record the number of successively dequeued packets. Specifically, it increases when a packet leaves from the queue and resets itself when a packet enters into the queue. When C_1 reaches its counting number, a pulse is generated to reset C_2.

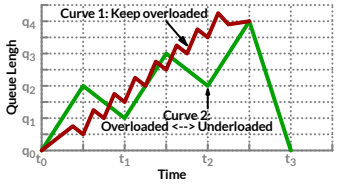


Fig. 10. Different appearances of queue length evolution curve[8]

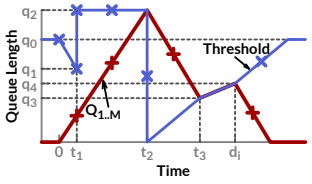
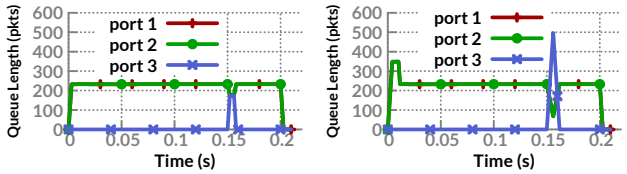


Fig. 11. Evolutions of queue length and threshold with EDT



(a) DT (b) EDT

Fig. 12. Evolutions of queue lengths when $N = 2,\ M = 1$. Micro-burst occurs at time 0.15s.

4.3 Analysis of EDT

In this part, we analyze EDT and give a formal comparison between DT and EDT.

Consider a scenario when queue length is empty at port $1, \cdots$, port M, and port $(M+1), \cdots$, port $(M+N)$ have reached their steady states; micro-burst traffic arrives at port $1, \cdots$, port M with arriving rate R simultaneously. To simplify our analysis, we assume that $M \leqslant N\frac{R}{R-C}$[9].

The sufficient condition for packet dropping caused by micro-burst traffic with EDT policy can be given by the following theorem:

Theorem 5. *The packets from micro-burst traffic will be dropped in port k ($k = 1, 2, \cdots, M$) if*

$$d_k > \frac{B}{M(R-C)} \tag{79}$$

Proof: The evolutions of queue length and threshold are depicted in Fig.11. Assume that micro-burst traffic arrives at port $1, \cdots$, port M at $t = 0$, then at $t = 0^+$, the queue length of port $1, \cdots$, port M $(Q_1, \cdots, Q_M)$ will increase at a rate of $(R-C)$, and the queue length threshold will decrease. At time $t = t_1$, $Q_1, \cdots, Q_M$ have increased for $\frac{4\alpha B}{2+\alpha P}$ packets, and port $1, \cdots$, port M will turn into uncontrolled state, and the threshold will temporarily increase to B/M. After that, if $Q_1, \cdots, Q_M$ continue to increase until buffer overflows at time $t = t_2$, port $1, \cdots$, port M will turn into controlled state and packets from micro-burst traffic are dropped. In summary, the packets from micro-burst traffic will be dropped in port k if

$$d_k > t_2 = \frac{B}{M(R-C)} \tag{80}$$

□

Comparison between DT and EDT: When $R \leqslant C\left(1 + \frac{1+\alpha N}{\alpha M}\right)$, compared to equation (10), additional $\left(\frac{1}{\alpha M} + \frac{N}{M}\right)$ micro-burst traffic can be absorbed with EDT. Even when $\alpha \to +\infty$ (i.e., DT achieves the maximum efficiency), EDT can still absorb more micro-burst traffic than DT. This performance improvement, however, comes at the price of fairness. Specifically, between $[t_1, t_2]$, EDT allows port $1, \cdots$, port M to occupy more buffer than port $(M+1), \cdots$, port $(M+N)$. However, as is discussed in §3.1, we argue that this is worthwhile as the duration of unfairness time is very small ($t_2 - t_1 < \frac{B}{M(R-C)}$). In addition, EDT is also carefully designed to ensure that this duration is not too long (using C_1 and TM_2).

4.4 Parameter Settings

Before deploying EDT, three parameters need to be determined: α, d, and the counting number of C_1. The setting of α has been discussed in [17], and we don't repeat here.

The parameter d is used for setting the counting time of TM_1 and TM_2. As is mentioned before, when setting the counting time of TM_1, we should make sure that parameter d is larger than the duration of most micro-burst traffic. On the other hand, since the timer TM_2 should not expire before micro-burst traffic ends or buffer overflows, parameter d should be longer than the time for the micro-burst traffic to fill the buffer. In summary, the parameter d should meet the inequality

$$d \geqslant \max\{B/(R_{min} - C), d_{mb}\} \tag{81}$$

where R_{min} is the arriving rate of the smoothest micro-burst traffic, and d_{mb} is the duration of micro-burst traffic. In general, $B/(R_{min} - C) > d_{mb}$, therefore, we can only consider the value of $B/(R_{min} - C)$. When the micro-burst occurs in the switch, there are multiple (more than 2) input ports sending packets to the same output port. Therefore, we can roughly let $R_{min} = 2C$. Then (81) can be rewritten as $d \geqslant B/C$.

The counting number of C_1 (cn_1) should not be too large, otherwise, EDT could not detect that the port has become underloaded. Besides, the parameter should be larger than the number of successively dequeued packets when the port is transmitting micro-burst traffic. Actually, if there is no batching scheme in the port, there are usually only one successively dequeued packet when micro-burst occurs. Therefore, we can simply set cn_1 to 3.

5 Evaluation

In this section, we compare the performances of DT and EDT by simulations on ns-2 platform [26][10]and experiments on a real testbed.

8. If the queue length evolves as Curve 1, then the port is in the overloaded state. If the queue length evolves as Curve 2, then the port is changing between underloaded and overloaded state.

9. When $M > N\frac{R}{R-C}$, buffer may overflow before queue length of port $1, \cdots$, port M reaches the threshold of uncontrolled state. The expressions are more complex but the analysis method is the same.

10. By default, ns-2 does not support shared memory architecture. We implement it by enabling an output queue to get the queue length in all other ports of the same switch (by using static member of a C++ class).

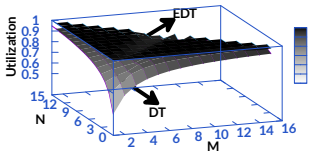


Fig. 13. Buffer utilization when packets from micro-burst traffic begins to be dropped

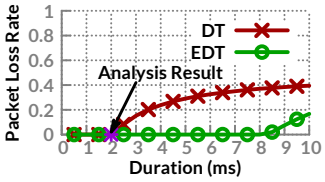


Fig. 14. Packet loss rate of micro-burst traffic as a function of its duration when $N = 2$ and $M = 1$

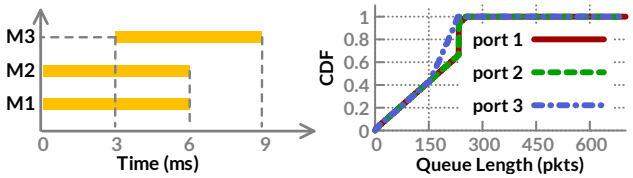


Fig. 15. Fairness between ports transmitting micro-burst traffic: (a) Simulation settings; (b) Distribution of queue length in each port.

5.1 Deterministic Scenario

We consider a 16-port 1Gbps switch with 1MB shared memory. As suggested by [17], we set α to 1 so that DT performs well. The counting number of C_1 is set to 3. The counting time of TM_2 is set to 10ms. According to above parameters, the counting number of C_2 is 8 and counting time of TM_1 is 2.1ms. In deterministic scenario, N output ports are overloaded and have reached their steady states. Meanwhile, M output ports begin to transmit micro-burst traffic and become overloaded. When a port is overloaded, the arriving rate of traffic is 2Gbps.

First, to show how EDT works, we set the duration of micro-burst traffic to 6ms and let $N = 2$, $M = 1$. The queue length evolution of each output port is shown in Fig. 12. Port 3 begins to transmit micro-burst traffic at $t = 0.15$s and finishes transmission 6ms later. In DT switches, packets in port 3 are dropped immediately after the arriving of micro-burst traffic. In comparison, in EDT switches, packet droppings can be avoided in port 3 by temporarily letting the port take over as much buffer as possible and letting other ports make way for it.

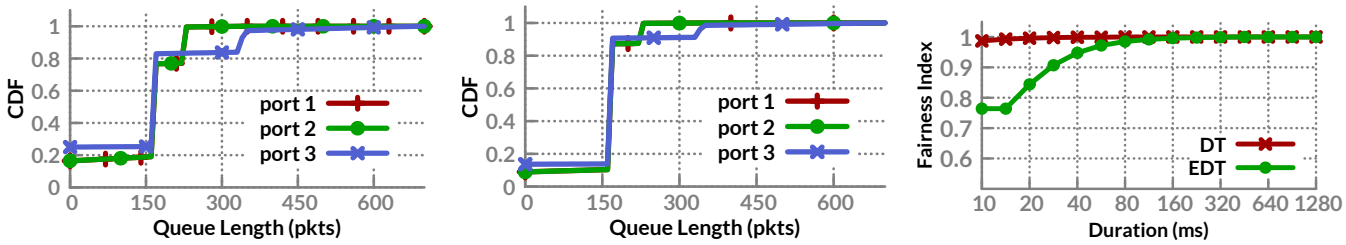
The buffer utilization when packets from micro-burst traffic are dropped is shown in Fig. 13. In DT switches, the utilization decreases as the number of overloaded ports decreases. In the worst case, the utilization is only 50.0%. Compared to it, in EDT switches, the utilization is almost 100% for all Ns and Ms, which implies that packets are dropped only when it is inevitable.

Fig. 14 illustrates the packet loss rate of micro-burst traffic as a function of its duration when $N = 2$ and $M = 1$. Apparently, the condition given by Theorem 1 in §3 agrees with the simulation result. Moreover, in DT switches, packet dropping caused by micro-burst traffic happens when the duration of micro-burst traffic reaches 2ms. While in EDT switches packet dropping won't happen until the duration is longer than 8ms. Note that when the duration is 2ms, the traffic size is 2ms $\times$ 2Gbps $=$ 0.5MB and it only needs 0.25MB switch memory, while packets are dropped in DT switches with 1MB buffer in this scenario. On the other hand, when the duration is 8ms, the traffic size is 8ms $\times$ 2Gbps $=$ 2MB and it needs 1MB switch memory. Packet dropping is inevitable in this scenario. In summary, compared with DT, additional 300% micro-burst traffic can be absorbed by EDT switch.

In the next, we use two simulations to show whether the fairness achieved DT can also be achieved by EDT.

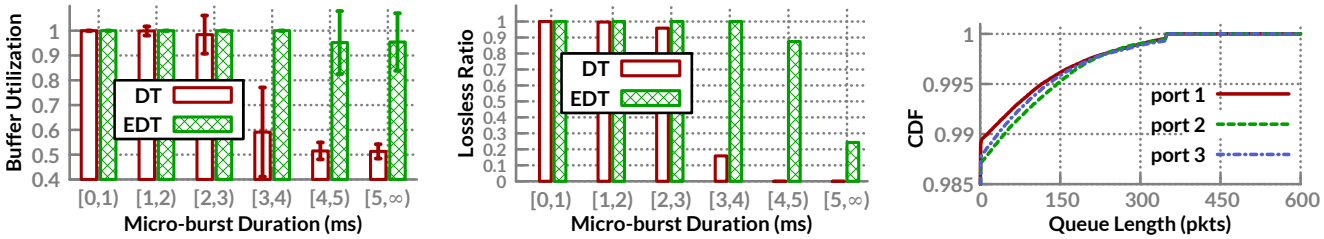
First, we evaluate the fairness when multiple output ports are transmitting micro-burst traffic. In this simulation, there is 1 output port transmitting long-lived flows and there are 3 output ports transmitting micro-burst traffic. As is shown in Fig.15a, the 1st micro-burst (in port 1) and the 2nd micro-burst (in port 2) arrive simultaneously, while the 3rd micro-burst (in port 3) arrives 3ms later. The durations of micro-bursts are 6ms. The queue length CDFs of 3 ports are depicted in Fig.15b. We have two observations from it. First, when micro-burst traffic arrives at different ports at the same time (micro-bursts in port 1 and port 2), EDT can make sure that the buffer is strictly fairly shared. Second, when one micro-burst arrives later than another (micro-bursts in port 1 and port 3), the port transmitting later arrived micro-burst will not starve for buffer. In addition to graphical impression, we also use *Jain's fairness index* [48] to quantitatively estimate the fairness. The fairness index ranges from $1/n$ (worst case, in this case n=3) to 1 (best case). In this case, the fairness index of DT and EDT is 0.992 and 0.998, respectively. Thus, EDT can achieve almost the same fairness as DT. In summary, EDT can ensure fair sharing of buffer among ports transmitting micro-burst traffic.

We now evaluate the fairness among output ports transmitting long-lived flows. The unfairness may happen when an output port becomes overloaded while other ports have reached their steady states because the port can occupy more buffer than other ports at that time. Therefore, we consider a scenario that a port becomes overloaded after other ports have reached their steady states. Specifically, port 1 and port 2 is in overloaded state at $t = 0$. Port 3 begins to transmit long-lived flows and becomes overloaded at $t = 0.15$s. After a while, port 3 finishes transmitting long-lived flows and becomes underloaded, while port 1 and port 2 are still in overloaded state. When these ports are in overloaded state, the traffic arriving rate in each port is 2Gbps. We sample the queue length of each port during the overloading period of port 3. The CDF of queue length in each port is shown in Fig. 16a and Fig.16b, where port 3 corresponds to the newly overloaded port. Port 3 can only acquire a little more buffer than port 1 and port 2 when the duration of long-lived flows are 500ms and 1000ms, respectively. Therefore, fairness is well promised in these cases. We also use *Jain's fairness index* [48] to compare the fairness between DT and EDT. We range the duration of long-lived flows from 10ms to 1000ms. As shown in Fig.16c, EDT can achieve the same fairness as DT as long as the duration of long-lived flow is larger than 80ms, which is only 8 times larger than the parameter d.



(a) duration: 500ms (b) duration: 1000ms (c) fairness index

Fig. 16. Fairness of EDT: (a,b) Queue length CDFs with different durations of the long-lived flows in port 3; (c) *Jain's fairness index* with different durations of the long-lived flows in port 3, where the fairness is better when fairness index is closer to 1.



(a) (b) (c)

Fig. 17. Performance of EDT in stochastic scenario: (a) Average buffer utilization when packets from micro-burst traffic begin to be dropped; (b) The ratio of micro-bursts that can avoid packet dropping; (c) Queue length distributions for port 1, port 2, and port 3.

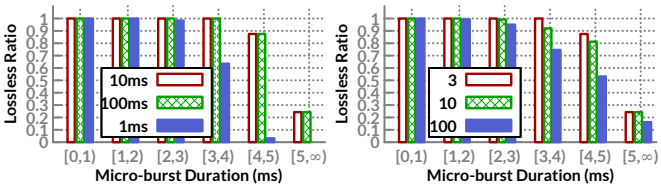
5.2 Stochastic Scenario

5.2.1 *Unresponsive ON-OFF traffic*

Workload: In this scenario, there are two kinds of traffic in each output port: background traffic and micro-burst traffic. We use Poisson model to simulate background traffic. As for the micro-burst traffic, we use the measurement results from real data center traffic in [3]. Specifically, we use Lognormal ON/OFF model to simulate micro-burst traffic. In Lognormal ON/OFF model, packets are generated at a constant rate of 2Gbps during "ON" periods and no packets are generated during "OFF" periods. Both "ON" and "OFF" intervals follow lognormal distribution. The average "ON" and "OFF" durations are set to 2ms and 58ms, respectively, and the standard deviations of them are equal to their averages. The average arriving rate of background traffic is 0.133Gbps, so that average load of each output port is 20% and the amount of background traffic is 2 times that of micro-burst traffic. Switch settings are the same as that in deterministic scenario.

Buffer utilization: First, we evaluate the buffer utilization when micro-bursts cause packet droppings. Fig. 17a illustrates the average buffer utilization for different micro-burst durations. In DT switches, buffer is fully utilized only when the duration of micro-burst traffic is shorter than 2ms. The buffer utilization is no larger than 60% when the duration is longer than 3ms. In comparison, in EDT switches, buffer is fully utilized for almost all micro-bursts.

Micro-burst absorption: Enabling buffer to be fully utilized could make more micro-bursts absorbed. Fig. 17b illustrates the ratio of lossless micro-bursts. In DT switches, few micro-bursts can avoid packet dropping when their durations are longer than 3ms. In comparison, in EDT switches, over 85% of micro-bursts can be absorbed when their durations are shorter than 5ms.



(a) $d = 10\text{ms}, 100\text{ms}, 1\text{ms}$ (b) $cn_1 = 3, 10, 100$

Fig. 18. The ratio of micro-bursts that can avoid packet dropping with different parameter settings

Fairness: In this part, we evaluate the fairness among switch ports. We choose 3 ports and illustrate their queue length CDFs in Fig. 17c, which shows that the queue lengths have similar distributions. The queue length CDFs of other ports are similar. Besides, the fairness index is 0.977 with DT policy and 0.964 with EDT policy, which quantitatively shows that the fairness achieved by DT policy can also be promised by EDT policy.

Parameter sensitivity: Fig.18a shows the ratio of lossless micro-bursts with $d = 10\text{ms}, 100\text{ms}, 1\text{ms}$, respectively. We have shown (in Section 4) that the parameter d should meet the inequality $d \geqslant B/C$. In this scenario, $B/C = 8\text{ms}$. From the Fig.18a, we can observe that the parameter d will not influence the lossless ratio when $d > 8\text{ms}$. Specifically, lossless ratio only slightly decreases when the parameter d increases from 10ms to 100ms. Therefore, as long as $d > B/C$, the performance of EDT is not sensitive to the parameter d. However, if we set $d = 1\text{ms}$, the lossless ratio dramatically drops, because TM_2 will expire before the buffer is fully used to absorb micro-burst traffic. Fig.18a shows the ratio of lossless micro-bursts with $cn_1 = 3, 10, 100$, respectively. As is discussed in Section 4, EDT performs well by simply

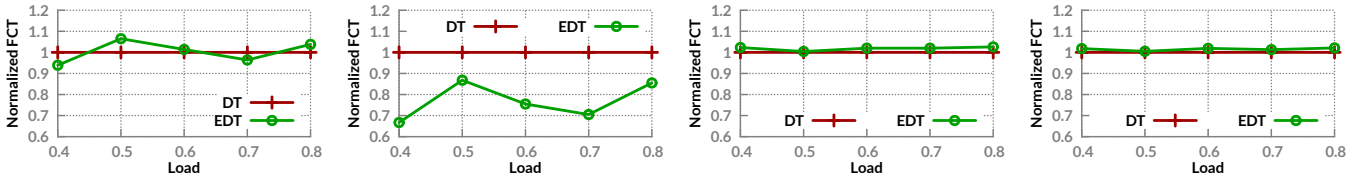


Fig. 19. Flow Completion Time (FCT) across different flow sizes with bursty traffic.

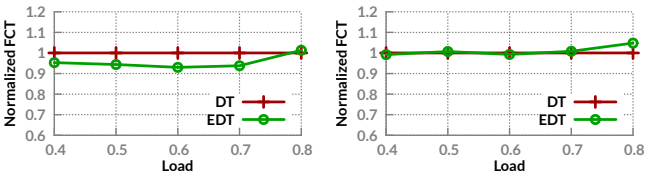


Fig. 20. Flow Completion Time (FCT) with uniform traffic

setting the cn_1 to 3. And increasing cn_1 to 10 will not influence EDT's performances. However, cn_1 should not be too large, otherwise, EDT will not be able to detect that the port has become underloaded.

5.2.2 Responsive traffic

In this part, we evaluate DT and EDT using responsive flows (i.e., TCP flows). Two scenarios are considered. One is with bursty traffic, in which a host will generate a query message to all other hosts and other hosts will send a response message to it simultaneously. The other is with uniform traffic, where flows follow a one-to-one pattern. Source host and destination hosts are randomly chosen. In both scenarios, we use flow size distributions from a real production data center supporting web search [10] to generate TCP flows. Following [49], [50], [51], [52], [53], flow arrivals are according to a Poisson process.

Flow completion time (FCT) is used as the performance metric. We also normalize the FCTs to the values achieved by DT for clear comparison.

Bursty traffic: Fig.19 shows flow completion time (FCT) across different flow sizes. With EDT, more packet droppings can be avoided for small flows, which improves their flow completion times. Specifically, the 99th percentile of flow completion time can be reduced by 14%-34% compared to that with DT (Fig.19b). On the other hand, EDT can achieve similar results as DT when flow size so large that switch is unable to absorb the bursty traffic (Fig.19c).

Uniform traffic: When traffic is not bursty, EDT can achieve similar performance as DT. Fig.20 shows flow completion time with uniform traffic. The overall flow completion time with EDT is almost the same as that with DT.

5.3 Implementation and Evaluation

Beyond simulations, we also implement EDT policy. As we cannot program switch chips, we emulate the switch with a server with multiple NICs instead, and implement DT and EDT policies on top of it using Intel DPDK [27] — a framework for fast packet processing on user space. After that, we evaluate DT and EDT on a real testbed.

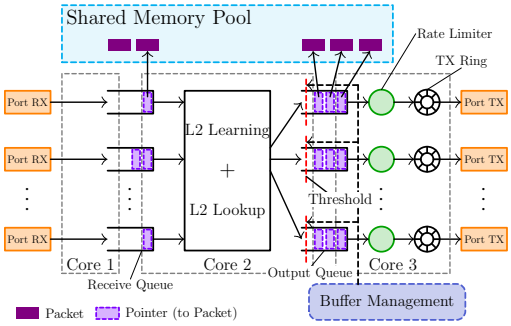


Fig. 21. Structure of switch implemented by DPDK

5.3.1 Implementation

Fig.21 shows the structure of shared memory switch implemented by DPDK. We use a pipeline model for packet processing. Specifically, packet processing is divided into 3 parts: retrieve packets from receiving ports (**receive**), forwarding packets to output queues (**forward**), and delivering packets to the transmit ring buffers (**send**). Each part is processed by different CPU cores.

Receive: In this part, Core 1 polls each receive port for packets. If a packet is received, it is delivered to a receive queue, by which the packet is passed to Core 2 for further processing. The receive queue is implemented by DPDK's Ring Library, which is lock-free.

Forward: In this part, Core 2 will poll each receive queue and fetch packets from it. For each packet, source and destination MAC addresses are extracted from it. Thereafter, the switch will add (update) the forwarding table according to the source MAC address and source port, and look up the destination port from forwarding table according to the destination MAC address. If the destination port is not found, the packet is flooded to all output queues. The forwarding table is implemented as a hash table using DPDK's Hash Library. After the destination output port is found, the packet is enqueued into the proper output queue. Note that the CPU processing rate is much higher than the line rate, thus most packets will be in the output queues rather than receive queues.

Send: In this part, Core 3 will poll each output queue and fetch a packet (if any) from it, and deliver the packet to the transmit ring (TX Ring). Packets in the ring will be sent out to the network by DPDK. However, if too many packets are in TX Ring, then the buffer occupancy in output queue cannot reflect the real buffer occupancy. Therefore, we introduce a rate limiter into this part. The rate limiter is very simple. Assume that we want to limit the sending rate to R, and we have sent out a packet of size B, then next

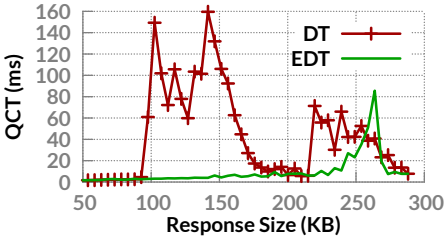
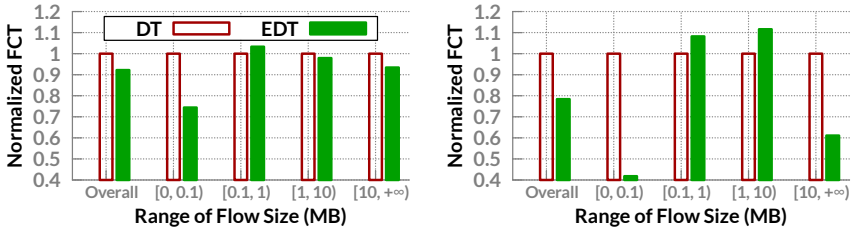


Fig. 22. Query Completion Time (QCT) with fan-in traffic



(a) Average (b) 99th percentile

Fig. 23. Flow Completion Time (FCT) with benchmark traffic. The FCTs are normalized to the values achieved by DT for clear comparison.

packet in the output queue is not allowed to be sent until B/R later.

Buffer Management: Each output queue has a threshold, and packets to be enqueued are dropped if the queue length is larger than the threshold. Buffer management policies are implemented by controlling the thresholds.

Shared Memory Pool: Rather than processing real packet data, packet delivery is achieved by pointers. Specifically, once a packet enters into the switch, it is placed into a pre-allocated shared memory pool, and a pointer to the packet is returned when we fetch a packet from a receive port. To send out a packet, we put the pointer into the TX Ring, and DPDK's driver will get packet data from shared memory pool before sending the packet to the network.

5.3.2 Evaluation

Testbed setup: We build a small testbed with 4 servers connected to a server emulating a switch with 4 ports. The server emulating the switch is a Dell OptiPlex 7010 desktop with a 4-core Intel® Core™ i3-3240 3.40GHz CPU, a 4GB memory, a 500GB hard disk, and four Intel® 82576 Gigabit Ethernet NICs, running CentOS 7.2 with GNU/Linux kernel 3.10.0. The emulated switch has 256KB shared memory for all ports, and the sending rate of each port is limited to 999Mbps by the rate limiter. Other servers are Dell Optiplex 780 desktops; each server has 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. These hosts use TCP NewReno as their transport protocols, and Delayed ACK is disabled as is suggested in [54]. In EDT, the counting numbers of C_1 and C_2 are set to 10 and 20, respectively. The counting times of TM_1 and TM_2 are set to 5.56ms and 10ms, respectively. Parameter α is set to 1 as before.

Fan-in traffic: First, we use a fan-in traffic pattern to show the ability of EDT to absorb micro-burst traffic. In this experiment, a client will send a query to other servers, and these servers will send a response message to the client, leading to fan-in bursty traffic in the switch. This scenario is common in many data center applications, such as partition-aggregation structure in online user-facing services [9], [10], [11], [12], synchronized reads in cluster-based storage [55], shuffle stage in MapReduce [56].

We repeat each experiment for 100 times, and the average query completion time with different response sizes is shown in Fig.22. With EDT, the query completion time is much smaller than that with DT when the response size is in $[100\text{KB}, 170\text{KB}]$. This is because EDT can absorb fan-in bursts and thus most response flows can avoid TCP retransmission timeouts[11]. For example, when the response size is 150KB, with DT 64% of response flows suffer from TCP timeouts, while with EDT only 10% of flows experience timeouts.

Benchmark traffic: In this experiment, a client (running on all hosts) is fetching data from servers (running on other hosts). Following [49], [50], [51], [52], [53], we use the flow size distribution from the real data center supporting web search service [10], and generate the fetching events according to a Poisson process. $\text{RTO}_{\min}$ is set to 10ms. The network load is 50%. The experiment lasts for 5 minutes and over 67,000 flows are generated.

Fig.23 shows the flow completion time (FCT) across different flow sizes. With EDT, the FCT of small flows can be greatly reduced. Specifically, with EDT the average and 99th percentile FCTs of flows whose sizes are within $[0, 0.1\text{MB})$ are reduced by 25.6% and 68.2%, respectively. Meanwhile, the FCT of larger flows is not severely penalized. The average FCT for flows in $[0.1\text{MB}, 1\text{MB})$ is 3.3% larger with EDT than that with DT. The average FCT for flows in $[1\text{MB}, 10\text{MB})$ and $[10\text{MB}, +\infty)$ is 2.1% and 6.6% smaller with EDT than that with DT.

6 Conclusion

In this paper, we theoretically deduce the sufficient conditions for packet dropping caused by micro-burst traffic and estimate the corresponding free buffer size. The analysis shows that the free buffer size is negatively correlated to the number of overloaded ports. And in order to ensure fair sharing of switch buffer among all ports, packets are dropped even when the micro-burst traffic size is far smaller than the buffer size. Thus, we propose the EDT policy guided by the conclusions obtained from theoretical analysis. EDT can absorb micro-burst traffic as much as possible by fully utilizing the buffer and temporarily relaxing the fairness constraint. We evaluate DT and EDT on ns-2 platform. We also implement a software prototype of EDT using DPDK and evaluate DT and EDT on a real testbed. The evaluations show that EDT can absorb more micro-bursts than DT and improve flow completion time for small flows.

Acknowledgments

The authors gratefully acknowledge the anonymous reviewers for their constructive comments. This work is supported

11. The default $\text{RTO}_{\min}$ is 200ms in GNU/Linux.

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. Part of this work was presented at IEEE INFOCOM [57], Hong Kong, April, 2015.

REFERENCES

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

[2] "What are microbursts?" White Paper, Arista. [Online]. Available: https://www.arista.com/assets/data/pdf/TechBulletins/AristaMicrobursts.pdf

[3] T. A. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding Data Center Traffic Characteristics," in *ACM SIGCOMM Workshop: Research on Enterprise Networking*, 2009.

[4] F. Uyeda, L. Foschini, F. Baker, S. Suri, and G. Varghese, "Efficiently measuring bandwidth at all time scales," in *Proc. USENIX NSDI*, 2011.

[5] R. Kapoor, A. C. Snoeren, G. M. Voelker, and G. Porter, "Bullet Trains: A Study of NIC Burst Behavior at Microsecond Timescales," in *Proc. ACM CoNEXT*, 2013, pp. 133–138.

[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 *Proc. ACM SIGCOMM 2014*, 2014, pp. 3–14.

[7] S. Ghorbani, B. Godfrey, Y. Ganjali, and A. Firoozshahian, "Micro Load Balancing in Data Centers with DRILL," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XIV, 2015.

[8] D. Meisner, C. Sadler, L. Barroso, W. Weber, and T. Wenisch, "Power management of online data-intensive services," in *Proc. ACM/IEEE ISCA*, 2011, pp. 319–330.

[9] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware Datacenter TCP (D2TCP)," in *Proc. ACM SIGCOMM*, 2012, pp. 115–126.

[10] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center TCP (DCTCP)," in *Proc. ACM SIGCOMM*, 2010, pp. 63–74.

[11] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better Never Than Late: Meeting Deadlines in Datacenter Networks," in *Proc. ACM SIGCOMM*, 2011, pp. 50–61.

[12] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, "DeTail: Reducing the Flow Completion Time Tail in Datacenter Networks," in *Proc. ACM SIGCOMM*, 2012, pp. 139–150.

[13] U. Cummings, P. P. Andrew Lines, and R. Southworth, "Shared-memory switch fabric architecture," U.S. Patent 7 814 280, Oct. 12, 2010.

[14] S. Das and R. Sankar, "Broadcom smart-buffer technology in data center switches for cost-effective performance scaling of cloud applications," *Broadcom White Paper*, 2012. [Online]. Available: https://www.broadcom.com/collateral/etp/SBT-ETP100.pdf

[15] M. Irland, "Buffer Management in a Packet Switch," *Communications, IEEE Transactions on*, vol. 26, no. 3, pp. 328–337, Mar 1978.

[16] F. Kamoun and L. Kleinrock, "Analysis of shared finite storage in a computer network node environment under general traffic conditions," *Communications, IEEE Transactions on*, vol. 28, no. 7, pp. 992–1003, Jul 1980.

[17] A. Choudhury and E. Hahne, "Dynamic queue length thresholds for shared-memory packet switches," *Networking, IEEE/ACM Transactions on*, vol. 6, no. 2, pp. 130–140, Apr 1998.

[18] A. Kesselman and Y. Mansour, "Harmonic buffer management policy for shared memory switches," in *Proc. IEEE INFOCOM*, vol. 3, 2002, pp. 1615–1622 vol.3.

[19] G. Ascia, V. Catania, and D. Panno, "An evolutionary management scheme in high-performance packet switches," *Networking, IEEE/ACM Transactions on*, vol. 13, no. 2, pp. 262–275, April 2005.

[20] A. Thareja and A. Agrawala, "On the Design of Optimal Policy for Sharing Finite Buffers," *Communications, IEEE Transactions on*, vol. 32, no. 6, pp. 737–740, Jun 1984.

[21] S. Wei, E. Coyle, and M.-T. Hsiao, "An optimal buffer management policy for high-performance packet switching," in *Proc. IEEE GLOBECOM*, Dec 1991, pp. 924–928 vol.2.

[22] I. Cidon, L. Georgiadis, R. Guerin, and A. Khamisy, "Optimal buffer sharing," *Selected Areas in Communications, IEEE Journal on*, vol. 13, no. 7, pp. 1229–1240, Sep 1995.

[23] "Congestion management and buffering in data center networks," White Paper, Extreme Networks, Dec. 2013. [Online]. Available: http://learn.extremenetworks.com/rs/extreme/images/Congestion-Management-and-Buffering-wp.pdf

[24] A. V. Bechtolsheim and D. R. Cheriton, "Per-flow dynamic buffer management," U.S. Patent 6 515 963, Feb. 4, 2003.

[25] V. Alaria, D. Bergamasco, M. Caramello, and C. Piglione, "Flexible and hierarchical dynamic buffer allocation," U.S. Patent 8 149 710, Apr. 3, 2012.

[26] "ns-2." [Online]. Available: http://www.isi.edu/nsnam/ns/

[27] "Intel DPDK," Intel. [Online]. Available: http://dpdk.org/

[28] M. Karol, M. Hluchyj, and S. Morgan, "Input Versus Output Queueing on a Space-Division Packet Switch," *IEEE Transactions on Communications*, vol. 35, no. 12, pp. 1347–1356, 1987.

[29] M. J. Karol, K. Y. Eng, and H. Obara, "Improving the performance of input-queued atm packet switches," in *INFOCOM*, 1992.

[30] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High-speed switch scheduling for local-area networks," *ACM Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319–352, Nov. 1993.

[31] N. W. McKeown, "Scheduling algorithms for input-queued cell switches," Ph.D. dissertation, University of California at Berkeley, 1995.

[32] B. Prabhakar, N. McKeown, and R. Ahuja, "Multicast scheduling for input-queued switches," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 5, pp. 855–866, 1997.

[33] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1260–1267, Aug 1999.

[34] N. McKeown, B. Prabhakar, and M. Zhu, "Matching output queueing with combined input and output queueing," in *PROCEEDINGS OF THE ANNUAL ALLERTON CONFERENCE ON COMMUNICATION CONTROL AND COMPUTING*, vol. 35. UNIVERSITY OF ILLINOIS, 1997, pp. 595–603.

[35] B. Prabhakar and N. McKeown, "On the speedup required for combined input and output queued switching," *Automatica*, vol. 35, no. 12, pp. 1909 – 1920, 1999.

[36] I. Stoica and H. Zhang, "Exact emulation of an output queueing switch by a combined input output queueing switch," in *IWQoS*, 1998.

[37] A. Charny, P. Krishna, N. Patel, and R. Simcoe, "Algorithms for providing bandwidth and delay guarantees in input-buffered crossbars with speedup," in *IWQoS*, 1998.

[38] S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1030–1039, Jun 1999.

[39] P. Krishna, N. S. Patel, A. Charny, and R. J. Simcoe, "On the speedup required for work-conserving crossbar switches," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 6, pp. 1057–1066, Jun 1999.

[40] A. Firoozshahian, V. Manshadi, A. Goel, and B. Prabhakar, "Efficient, Fully Local Algorithms for CIOQ Switches," in *INFOCOM*, 2007.

[41] R. Seifert and J. Edwards, *The All-New Switch Book: The Complete Guide to LAN Switching Technology*. John Wiley & Sons, 2008.

[42] S. Iyer and N. McKeown, "Techniques for fast shared memory switches," Stanford High Performance Networking Group, Tech. Rep. TR01-HPNG-081501, 2001.

[43] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, S. Boving, G. Desai, B. Felderman, P. Germano, A. Kanagala, J. Provost, J. Simmons, E. Tanda, J. Wanderer, U. Hölzle, S. Stuart, and A. Vahdat, "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network," in *SIGCOMM*, 2015.

[44] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the Social Network's (Datacenter) Network," in *SIGCOMM*, 2015.

[45] C. Guo, H. Wu, Z. Deng, G. Soni, J. Ye, J. Padhye, and M. Lipshteyn, "RDMA over Commodity Ethernet at Scale," in *SIGCOMM*, ser. SIGCOMM '16, 2016.

[46] J. Guo, F. Liu, X. Huang, J. C. S. Lui, M. Hu, Q. Gao, and H. Jin, "On efficient bandwidth allocation for traffic variability in datacenters," in *INFOCOM*, 2014.

[47] J. Guo, F. Liu, J. C. S. Lui, and H. Jin, "Fair Network Bandwidth Allocation in IaaS Datacenters via a Cooperative Game Approach," *IEEE/ACM Transactions on Networking*, vol. 24, no. 2, pp. 873–886, April 2016.
[48] R. Jain, D.-M. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer system," DEC Research Report TR-301, Sep 1984.
[49] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," in *SIGCOMM*, 2013.
[50] W. Bai, L. Chen, K. Chen, and H. Wu, "Enabling ecn in multi-service multi-queue data centers," in *NSDI*, 2016.
[51] K. He, E. Rozner, K. Agarwal, Y. J. Gu, W. Felter, J. Carter, and A. Akella, "Ac/dc tcp: Virtual congestion control enforcement for datacenter networks," in *SIGCOMM*, 2016.
[52] L. Chen, K. Chen, W. Bai, and M. Alizadeh, "Scheduling mix-flows in commodity datacenters with karuna," in *SIGCOMM*, 2016.
[53] Y. Zhu, M. Ghobadi, V. Misra, and J. Padhye, "Ecn or delay: Lessons learnt from analysis of dcqcn and timely," in *CoNEXT 2016*, 2016.
[54] M. Yu, A. Greenberg, D. Maltz, J. Rexford, L. Yuan, S. Kandula, and C. Kim, "Profiling Network Performance for Multi-tier Data Center Applications," in *NSDI*, 2011.
[55] 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.
[56] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.
[57] D. Shan, W. Jiang, and F. Ren, "Absorbing micro-burst traffic by enhancing dynamic threshold policy of data center switches," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, April 2015, pp. 118–126.

Danfeng Shan received the B.E. degree in computer science and technology from Xi'an Jiaotong University, Xi'an, China, in 2013, and is currently pursuing Ph.D. degree in computer science and technology at Tsinghua University, Beijing, China. His research interests include congestion control and data center network.

Wanchun Jiang received the Ph.D. degree and the B.E. degree in computer science and technology from Tsinghua University, Beijing, China in 2014 and 2009, respectively. He is currently an assistant professor in the School of Information Science and Engineering, Central South University. His research interests include congestion control, data center networks and the application of control theory in computer networks.

Fengyuan Ren (M'04) is a professor of the Department of Computer Science and Technology at Tsinghua University, Beijing, China. He received his B.A and M.Sc. degrees in Automatic Control from Northwestern Polytechnic University, China, in 1993 and 1996 respectively. In Dec. 1999, he obtained Ph.D. degree in Computer Science from Northwestern Polytechnic University. From 2000 to 2001, he worked at Electronic Engineering Department of Tsinghua University as a post doctoral researcher. In Jan. 2002, he moved to the Computer Science and Technology Department of Tsinghua University. His research interests include network traffic management, control in/over computer networks, wireless networks and wireless sensor networks. He (co)-authored more than 80 international journal and conference papers. He is a member of the IEEE, and has served as a technical program committee member and local arrangement chair for various IEEE and ACM international conferences.