Traffic-Aware Buffer Management in Shared Memory Switches

Sijiang Huang, Mowei Wang, *Graduate Student Member, IEEE*, and Yong Cui, *Member, IEEE*

***Abstract*— Switch buffer serves an important role in the modern internet. To achieve efficiency, today's switches often use on-chip shared memory. Shared memory switches rely on buffer management policies to allocate buffer among ports. To avoid waste of buffer resources or excessive buffer occupation by a few ports, existing policies tend to maximize overall buffer utilization and pursue queue length fairness. However, blind pursuit of utilization and misleading fairness definition based on queue length lead to buffer occupation with no benefit to throughput but extends queuing delay and undermines burst absorption of other ports. With analysis of current dynamic threshold policies, we demonstrate that meaningless buffer occupation can potentially impair the absorption capability of shared buffer, whereas none of the existing policies have addressed this problem. We contend that a buffer management policy should proactively detect port traffic and adjust buffer allocation accordingly. In this paper, we propose Traffic-aware Dynamic Threshold (TDT) policy. On the basis of the classic dynamic threshold policy, TDT proactively raises or lowers port threshold to absorb burst traffic or evacuate meaningless buffer occupation. We present detailed designs of port control state transition and state decision module that detect real-time traffic and change port thresholds accordingly. Simulation and DPDK-based real testbed demonstrate that TDT simultaneously optimizes for throughput, loss and delay, and reduces up to 50% flow completion time.**

***Index Terms*— Shared memory switches, buffer management, dynamic threshold, traffic-aware design.**

I. Introduction

SWITCH buffer is used to absorb burst traffic and improve the overall performance of the switch. Insufficient buffer results in reduced port throughput [1], thereby impairing the network quality of service [2]–[4]. To achieve high buffer efficiency, the majority of today's switches adopt on-chip shared memory, instead of private memory that exclusively allocated for each port [2], [5]–[7].

Shared memory switches rely on specific buffer management policies to allocate buffer among different ports [2]. Without buffer management policies, a few ports can occupy as much as the entire shared buffer space, blocking other ports from benefitting from the shared memory, resulting in severe unfairness between switch ports [7]. To avoid unfairness, the simplest way is to split the total buffer evenly and allocate it as private buffer for each port. However, the problem is that ports only have access to their exclusive buffer space. This limits buffer utilization when only a few ports are active, which is against the basic principle of shared memory that buffer should be dynamically shared among ports. For a long time, researchers have believed that the ideal buffer management should be somewhere in between complete sharing (no control) and complete partitioning (evenly split), with high buffer utilization as well as port fairness [7]–[9].

Many buffer management policies have been proposed in the past three decades [7]–[15] to allocate buffer in shared memory switches. Although the design concepts and methods of different schemes are diverse, the design goals of these schemes are basically consistent, that is, pursuing the highest possible buffer utilization, reducing packet loss, maximizing throughput, while ensuring queue length fairness between ports. Since being proposed over twenty years ago, the dynamic threshold (DT) [7] policy has been used as the default buffer management scheme by switch manufacturers [2] and in various congestion control related researches [5], [16]. Besides several variants of DT [8]–[10], no significant progress had been made in this research area.

Existing buffer management policies fail to make full use of the shared buffer due to misleading optimization goals: buffer utilization [8] and queue length fairness [7]. More specifically, when long-lived over-line-speed traffic arrives at a switch port, blind pursuit of buffer utilization leads to buffer occupation that does not contribute to port throughput but extends queuing delay. On the other hand, misleading fairness definition based on queue length impedes meaningless buffer occupation from being evacuated, impairing burst absorption capability of other ports through additional buffer. Besides, queue length fairness ignores the demand difference between different ports. In other words, fairness in queue length is not equivalent to fairness in throughput.

Analysis of current dynamic threshold policies further demonstrates that buffer occupied by long-lived traffic not only is non-beneficial but potentially impairs the absorption capability of shared buffer, which is the fundamental function of buffer. When port traffic arriving rate is sufficiently fast, as a result of line rate limitation, buffer occupation might not

Manuscript received 23 June 2021; revised 22 January 2022, 10 April 2022, and 30 April 2022; accepted 4 May 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Caesar. Date of publication 17 May 2022; date of current version 20 December 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1800303 and in part by NSFC Project of China under Grant 62132009 and Grant 61872211. An earlier version of this paper was presented in part at the IEEE INFOCOM 2021 [DOI: 0.1109/INFOCOM42981.2021.9488849]. *(Corresponding author: Yong Cui.)*

The authors are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: cuiyong@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TNET.2022.3173930

be evacuated in time, causing performance degradation of the shared memory.

In light of the limitations of existing buffer management policies, we aim to propose a buffer management policy that optimizes for metrics that actually have impacts on the quality of service, i.e., throughput, loss and delay. To design such policies, we first put forward the design principles of traffic-aware buffer management that a policy should be able to allocate buffer based on the real-time traffic status of the ports. Designing a traffic-aware buffer management policy can be challenging because it requires switch ports to proactively detect port traffic and adjust buffer allocation accordingly. More specifically, switch ports have to determine the type of traffic it is transmitting in a timely manner with limited port-level information.

In this paper, we propose Traffic-aware Dynamic Threshold (TDT), a buffer management policy that controls the buffer allocation of shared memory switches by detecting port traffic status in real-time. TDT can fully utilize the shared buffer to absorb burst traffic, avoid meaningless buffer occupation through proactive evacuation, and ensure throughput fairness among ports at the same time. By using buffer only when needed, TDT is friendly to loss-sensitive burst traffic, throughput-sensitive long-lived traffic, and delay-sensitive short traffic in different switch ports simultaneously.

TDT uses a set of port-wise control states to differentiate the status of different ports, and impose different thresholds to individual ports accordingly. The transition between different port control states is determined by a state decision module that detects port traffic in real-time. The state decision module of TDT is only composed of several counters, comparators and triggers, avoiding dependency on the assumption of time variables. When port traffic changes, TDT proactively raises or lowers port threshold according to port control state to allocate buffer among different ports in a dynamic manner.

We evaluate TDT with ns-3 simulation and a DPDK-based switch prototype, comparing TDT with existing buffer management policies. Simulation results demonstrate that when long-lived over-line-speed traffic exists, TDT can absorb ∼60% additional burst traffic while maintaining a similar overall throughput and reducing overall queuing delay in the switch, compared with current policies. Experiments on real DPDK testbed show that TDT reduces 12% average flow completion time (FCT). Specifically, TDT outperforms DT over 80% of the time and has an up to 50% reduction in FCT. In summary, we make the following contributions:

- Propose the design principles of traffic-aware buffer management. To the best of our knowledge, it is the first time the idea that buffer management of shared memory switches should consider proactive evacuation has been proposed (§IV).
- Perform elaborate analysis on current dynamic threshold policies to verify the adverse effect of meaningless buffer occupation (§III).
- Design TDT, our proposed version of traffic-aware buffer management and its detailed design of threshold determination based on port-wise control states and state decision module (§V).

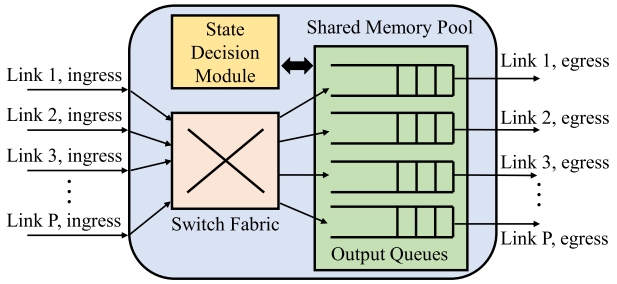


Fig. 1. The architecture of a shared memory switch.

- Conduct comprehensive evaluations on TDT, along with existing buffer management policies with large-scale simulation and real DPDK testbed implementation (§VI).

The rest of this paper is organized as follows: §II introduces the background of our research and limitations of current buffer management policies. §III analyzes the deficiencies of current policies to emphasize the need for traffic-aware buffer management. §IV overviews the key design ideas of traffic-aware buffer management. §V presents the design details of TDT, our proposed version of traffic-aware buffer management. Evaluations based on simulation and DPDK testbed are presented in §VI. §VII discusses the advantages and limitations of this study. Previous researches related to our work are introduced in §VIII and §IX concludes this paper.

II. Background and Motivation

In this section, we first introduce shared memory switches and current buffer management policies, with a special focus on widely used dynamic threshold policies. Then, we pinpoint the limitations of existing strategies, i.e., blind pursuit of utilization and misleading definition of fairness, which motivate our research.

A. Background

Shared Memory Switch: Figure 1 shows the architecture of a typical shared memory switch. In the context of this paper, we focus on output-queued shared memory switches with high speedup switch fabric that can process packets from any input port in negligible time and send them to the designated output port queue [6]. To achieve high buffer efficiency, every output port can access the shared memory pool which means theoretically any single port can use as much as the entire buffer space. In practice, shared memory switches rely on specific buffer management policies deployed on a control module (e.g., state decision module in Figure 1) to allocate buffer among output ports.

Buffer Management Policies: Several buffer management policies [7]–[15] have been designed to allocate buffer among ports in shared memory switches. Current policies have two main design goals. First, buffer should be fairly distributed among different ports, i.e., no port is "starved" because a few ports have occupied too much buffer space [7]. And second, buffer utilization should be as high as possible, i.e., free buffer space when packets are dropped should be as small as possible [8].

In general, existing buffer management policies can be divided into two categories: preemptive policies and non-preemptive policies. For preemptive policies [13], [15], packets that are already in the memory can be overwritten, or "pushed out", by newly arrived packets. Preemptive policies have been proved to be optimal in certain circumstances [13], [15] whereas they are too hard to implement in practice due to hardware limitations [7], [8]. On the other hand, non-preemptive policies [7]–[12], [14] that only allow packets to be dropped before entering the port queue, are more suitable to be deployed on existing devices. To achieve fairness, non-preemptive policies often use threshold(s) to restrict the amount of buffer each port can access. In this paper, we only consider non-preemptive policies in design for practicality.

Specifically, in data center networks, buffer management policies are essential in utilizing buffer to absorb microburst traffic. Previous works [17], [18] have demonstrated that fast and short microburst is ubiquitous in data center networks. Microburst traffic may be caused by TCP synchronization and/or fan-in traffic pattern [19]. Typical microburst traffic exhibits an ON-OFF pattern in the order of microseconds to milliseconds. This type of traffic often carries delay-sensitive short messages [20] therefore a buffer management policy should strive to absorb microbursts as much as possible.

Dynamic Threshold Policies: Among a wide variety of non-preemptive policies, dynamic threshold policies [7], [8], [10] are most widely used by switch vendors [2] due to their simplicity of deployment. In dynamic threshold policies, the queue length of every switch port is restrained by its corresponding threshold, which dynamically changes with switch status indicators (e.g., remaining buffer space). Classic dynamic threshold scheme (DT) [7] sets its threshold, shared by all ports, proportional to the current amount of unoccupied buffer space. More specifically, the threshold at time t can be calculated by (1).

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

where $T(t)$ is the threshold at time t, B is the total buffer size, $Q_i(t)$ is the queue length of port i at time t, and α is a control function normally set to a constant value (usually a power of 2) for simplicity. To avoid unfairness when traffic changes, DT reserves a certain amount of buffer in the "stable" state (i.e., the state of port when its queue length is equal to the port threshold).

On the basis of the classic dynamic threshold scheme, in recent years, researchers have proposed several variants to cope with specific problems in networking. A typical one is the Enhanced Dynamic Threshold (EDT) [8]. EDT analyzes the buffer waste of DT when transmitting microburst traffic and improves the burst absorbing ability of DT by temporally relaxing the threshold restraint. EDT defines ports that are under the restriction of shared threshold as "controlled" and ports that are not as "uncontrolled". With this definition, port thresholds of EDT can be calculated by (2). For design details of EDT, please refer to [8].

$$T(t) = \begin{cases} \alpha \cdot (B - \sum_i Q_i(t)), & \text{if port is "controlled"} \\ B/n', & \text{if port is "uncontrolled"} \end{cases} \tag{2}$$

where $T_i(t)$ is the threshold of port i at time t and n' is the number of "uncontrolled" ports.

Although dynamic threshold policies have made huge successes in the past decade, we will demonstrate the limitations of existing dynamic threshold policies in the next part.

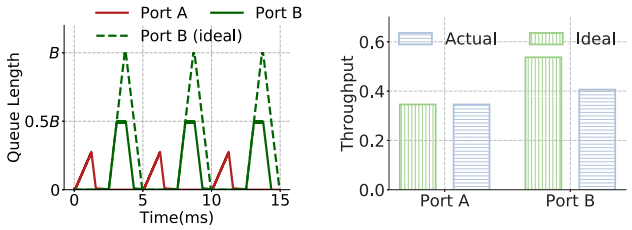
B. Motivation

The optimization goal of current buffer management strategies is to maximize total throughput, minimize overall packet loss while maintaining port fairness [7], [8], [10], [15]. In order to optimize the metrics above, current policies focus on maximizing buffer utilization and fairness based on queue length, for the reasons that "drop packet only when inevitable" intuitively means fewer packet losses, and "different queue has similar chances to reach the same queue length" intuitively equals to fair buffer distribution among different port queues.

Unfortunately, neither of these intuitions holds. In the following, we show why blindly pursuing buffer utilization or using queue length to measure fairness results in meaningless buffer occupation, which further leads to performance degradation.

Blind Pursuit of Utilization: Existing buffer management schemes contend that buffer should be as full as possible when switch port is overloaded, otherwise buffer is not fully utilized [7], [8], [12]. This opinion comes from an ancient cognitive habit that if resources are not used, they are wasted. However, resources not wasted are not equivalent to resources that are beneficial. Due to the line rate limitation of the switch ports, switch buffer management is a problem in which the law of diminishing marginal utility [21] is very obvious. For an overloaded port (i.e., a port transmitting an aggregate flow whose rate is over port line rate), throughput is limited by port line rate instead of available buffer. In that case, additional buffer allocated for this port brings little benefit to overall throughput. Although high buffer utilization in this particular case does not directly lead to throughput degradation, when multiple ports are competing for buffer, high utilization potentially impairs the absorption capacity of ports that became active later. We will further demonstrate this kind of performance degradation caused by meaningless buffer occupation is indeed possible in §III. On the other hand, maintaining high buffer utilization results in longer port queues, which prolongs the queueing delay of delay-sensitive packets.

Misleading Definition of Fairness: Fairness is one of the most important considerations when designing a buffer management policy. Current policies either use descriptive definitions of fairness [7] or use the queue length of competing ports to measure fairness [8]. A buffer management scheme is considered fair if different ports that start active at different times can achieve similar queue lengths. Under this definition



(a) Queue length fairness. (b) Throughput unfairness.

Fig. 2. Queue length fairness is not equivalent to throughput fairness. Equal queue length threshold shared by ports with different throughput demand causes unfairness in throughput.

of fairness, the fairest allocation strategy is evenly split, i.e., the shared buffer pool is equally distributed among all output ports as private buffer. However, the above fairness description ignores the difference in demand between ports and only limit the fairness of the queue length, an intermediate variable, without considering the fairness of metrics that actually make a difference in terms of quality of service (e.g., throughput). For example, Figure 2 shows a simple example generated by simulation where two ports with different flow rates compete for the whole buffer space. With the fairness definition based on queue length, the resource (buffer) should be equally divided between these two ports. Port A receives full additional throughput with the help of buffer while port B can only partially benefit from the buffer. Obviously, relative to the ideal throughput, port B is affected more than port A, as a result of buffer allocation. Therefore, in terms of throughput, fairness measurement based on queue length is not always fair. This misleading definition of fairness has caused unnecessary restrictions that impede necessary buffer evacuation in the design of existing schemes. In light of the reasons above, in this paper we will directly use throughput as a measure of fairness. Specifically, we define the throughput fairness as the fairness of how much additional throughput different ports gain from the shared buffer. We will explain this notion in detail in §VI.

Meaningless Buffer Occupation: As a result of blind pursuit of utilization and misleading definition of fairness, current strategies maintain relatively longer queues in active ports, keeping average queue length at a high level. However, as discussed above, buffer occupation is not a sufficient condition for neither high throughput nor fairness. On the contrary, maintaining long port queues undermines the overall capacity of the switch to absorb potential burst traffic, and extends the overall queuing delay of the switch [5]. Therefore, reducing meaningless buffer occupation is a vital step toward better buffer management policies. In order to achieve that, a buffer management policy needs to be able to determine whether or not the buffer occupation at a specific time is beneficial. In conclusion, there is an urgent need for a buffer scheduling strategy that can perceive changes in the traffic status of switch ports and take actions accordingly.

III. Analysis of Dynamic Threshold Policies

As discussed in §II-B, without traffic-aware ability, current dynamic threshold policies suffer from meaningless buffer occupation. We have argued that such sort of buffer occupation is non-beneficial. To further emphasize the need for traffic-aware buffer management, in this section, we use analytical methods to illustrate how meaningless buffer occupation can cause performance degradation by impairing the fundamental function of buffer, that is, absorbing burst traffic.

TABLE I
Notations Used in This Paper

Notation	Description
P	Total number of ports in the switch
B	Total buffer size
C	link capacity
R_i	Traffic arriving rate of port i
$Q_i(t)$	Queue length of port i at time t
$T_i(t)$	Threshold of port i at time t
d_i	duration of burst traffic of port i

In this section, consider the following scenario. A switch with P output ports sharing a total amount of B buffer space. Before $t = t_0$, there are N ports with the same transmitting rate R in "stable" state, i.e., queue length of those ports is equal to their corresponding threshold thus remains constant. At $t = t_0^+$, another M previously inactive ports start transmitting burst traffic, also with the same rate R. We want to analyze the switch's capability to absorb the burst traffic transmitting over the M newly active ports under different dynamic threshold policies, namely DT [7] and EDT [8] introduced in §II-A. Notations used in the analysis are summarized in Table I.

For dynamic threshold policies, when new ports become active, queue lengths of existing "stable" ports decrease. Attributed to the fact that the maximum queue reduction rate is limited by link capacity, there are two different scenarios distinguished by port traffic arriving rate R_i. As demonstrated in Figure 3 (where M and N are set to 1 for simplicity), when traffic arriving rate is relatively small, the queue length reduction of existing queues (Q_n) matches the reduction of threshold, which is proportional to the queue length increase of newly active ports (Q_m). Conversely, if traffic arriving exceeds a certain value, the queue length reduction rate is restrained by port line rate so that it cannot match the threshold reduction rate. In the following, we will discuss these two scenarios case-by-case.

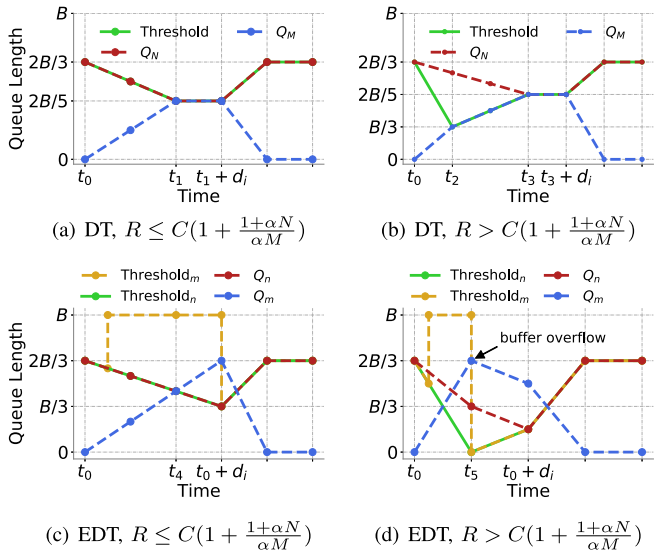
1) $R \leq C\,(1+\frac{1+\alpha N}{\alpha M})$

In this case, queue lengths of the existing N active ports will decrease alongside the threshold. For DT [7], queue lengths of newly active ports will not be restricted by the threshold until t_1 and therefore the maximum duration of burst traffic that can be absorbed by DT in this case is t_1. Then, according to [7], t_1 can be calculated by (3),

$$t_1 - t_0 = \frac{\alpha B}{[1+\alpha(M+N)]\,(R-C)} \tag{3}$$

making the sufficient condition for burst traffic to be fully absorbed by DT follows (4).

$$(R-C)\cdot d_i \leq \frac{\alpha B}{1+\alpha(M+N)} \tag{4}$$



(a) DT, $R \leq C(1+\frac{1+\alpha N}{\alpha M})$ (b) DT, $R > C(1+\frac{1+\alpha N}{\alpha M})$

(c) EDT, $R \leq C(1+\frac{1+\alpha N}{\alpha M})$ (d) EDT, $R > C(1+\frac{1+\alpha N}{\alpha M})$

Fig. 3. Queue length and threshold evolution of dynamic threshold policies under different traffic rate R.

With DT, the absorbing capacity is limited by the shared threshold to make sure a proportion of headroom of buffer will always be reserved.

On the other hand, without the restriction from the shared threshold, EDT [8] allows ports transmitting burst traffic access as much as the entire buffer space. Therefore, the complete absorption condition for EDT is

$$(R-C)\cdot d_i \leq \frac{B}{M} \tag{5}$$

Clearly, in this case EDT always outperforms DT, as

$$\frac{\alpha B}{1+\alpha(M+N)} \leq \frac{B}{M+N} \leq \frac{B}{M} \tag{6}$$

In fact, B/M is the ideal upper bound of burst absorption capacity in this case, as all the buffer space has been leveraged to absorb burst traffic.

We can conclude that for the first case, notwithstanding the fact that DT is subject to the negative effect from shared threshold restriction, EDT performs nearly impeccably. No performance degradation has been manifested so far, and this leaves us with the second case.

2) $R > C\,(1+\frac{1+\alpha N}{\alpha M})$

In this case, the rate of queue reduction is unable to match that of threshold reduction. This means buffer occupied by the N active ports starting to play an important role.

In this scenario, queue lengths and threshold evolution of DT is shown in Figure 3(b) and the queue lengths of M newly active ports reach threshold in time t_2. [7] has given the expression of t_2 as

$$t_2 - t_0 = \frac{\alpha B}{(1+\alpha N)\left[(1+\alpha M)(R-C)-\alpha NC\right]} \tag{7}$$

Then the sufficient condition for a burst with duration d_i to be fully absorbed by DT in this case can be calculated as follows.

$$(R-C)\cdot d_i \leq \frac{\alpha B(R-C)}{(1+\alpha N)\left[(1+\alpha M)(R-C)-\alpha NC\right]} \tag{8}$$

For EDT, buffer that can be used by ports transmitting burst traffic is still not under the constraint of the shared threshold. However, this does not mean that EDT can use the entire buffer space as earlier. The buffer access of newly active N ports might be blocked by not yet evacuated buffer occupation and those ports will be compelled to drop packets in bursts on account of buffer overflow. The sufficient condition for packet dropping by buffer overflow is

$$M(R-C)d > \frac{B}{1+\alpha N} + NCd \tag{9}$$

where $M(R-C)d$ is the queue length increase in burst duration d, $\frac{B}{1+\alpha N}$ is the remaining free buffer space when there are N ports in "stable" state. And NCd is the queue length reduction during d with port line rate C, i.e., the extra buffer space from evacuation.

Now we verify that condition in (9) is possible in practice. The following inequality relations

$$R > C(1+\frac{1+\alpha N}{\alpha M}) > C(1+\frac{N}{M}) \tag{10}$$

which can be deformed as

$$M(R-C)-NC > 0 \tag{11}$$

indicating that the condition in (9) is possible.

The total amount of buffer also imposes restriction on burst duration. Free buffer space can never exceed the total amount of shared buffer B.

$$\frac{B}{1+\alpha N} + NCd < B \tag{12}$$

In conclusion, with the overflow flow condition from (9) combing with the additional restriction of total buffer from (12), the sufficient condition for burst traffic to be absorbed by EDT are

$$\left[MR-(M+N)C\right]d \leq \frac{B}{1+\alpha N} \tag{13}$$

and

$$d \leq \frac{\alpha B}{(1+\alpha N)C} \tag{14}$$

Remarks: Although EDT is a policy specially designed for absorbing burst traffic, and it does perform better than DT in both cases above, in the second case we have found that it is not always able to fully utilize the shared buffer space in some circumstances because of pre-existing buffer occupation. When the traffic rate of a burst is sufficiently large, which is often the case as "incast" scenarios are common in networking, buffer occupation caused by the restriction of line rate plays a non-negligible role and buffer overflow occurs before complete absorption. Recall from §II-B that buffer occupation in such circumstances is non-beneficial, analysis in this section has further demonstrated that none of the current dynamic threshold policies can efficiently address the meaningless buffer occupation problem that undermines the functionality of buffer. It seems logical that a better buffer management policy should be traffic-aware, with the ability to detect burst traffic that arrives and departs as well as long-lived traffic causing buffer occupation, and then make buffer allocation decisions accordingly.

IV. Traffic-Aware Buffer Management

Discussion in §II-B and analysis in §III have shown that the key factor causing performance degradation is buffer occupied by ports overwhelmed by long-lived traffic. Solving this problem requires a buffer management policy that can perceive and determine whether a switch port is transmitting long-lived flows or short bursty flows. In this section, we present the key design principles of traffic-aware buffer management. By real-time traffic detection, a traffic-aware buffer management policy is expected to fully utilize buffer when absorbing burst traffic (§IV-A) or proactively evacuate meaningless buffer occupation (§IV-B) when traffic changes.

A. Burst Absorption

In the context of this paper, burst traffic refers to fast (transmitting at over the port line rate) but short (no longer than switch buffering time[1]) flows that arrive at switch ports [18]. In other words, we only consider bursts that can be fully absorbed given that they can monopolize all the buffer. Otherwise, buffer management policies are futile for this flow because no possible allocation can achieve lossless transmission. Burst traffic caused by "incast" scenario often carries deadline-sensitive short messages [5], [20]. Retransmissions triggered by packet loss will result in the service missing its deadline [4], [22]. It can be concluded that this sort of burst is loss-sensitive, therefore the optimization goal should be lossless transmission. Once packet loss occurs the absorption fails and the number of packet losses, except zero, is irrelevant.

For a single burst, buffer management policy should leverage as much as possible buffer space to avoid packet loss. In this paper, we share a similar basic design idea to absorb burst traffic as EDT [8]. When burst traffic arrives at a port, port queue length should not be restrained by the dynamic threshold. If the burst traffic transmission is completed or it exceeds the buffer capacity, buffer management should quickly react to traffic changes by putting the corresponding ports back under the threshold control.

The most challenging part of the design is to determine whether a port has started or finished transmitting burst and anticipate whether buffer can fully absorb that burst. Traffic detection is expected to be sensitive and accurate, otherwise, the buffer management policy will not be able to change the threshold in a timely fashion, resulting in inability to fully absorb burst traffic or a single port occupying too much buffer for too long.

Based on the above reasoning, we propose several conditions for traffic status judgment. First, we define several traffic states to represent typical port traffic status. When a port is idle or transmitting under-line-speed traffic, it is "underloaded". An "Overloaded" port refers to a port that transmits short burst flows. And a port transmitting long-lived over-line-speed flows is called an "overwhelmed" port. Conditions for judgment of port traffic state are listed as follows:

- A port becomes "overloaded" when burst traffic arrives at this port. When a port becomes "overloaded", its queue length increases while no packets are dropped [8].
- A port becomes "underloaded" when over-line-speed traffic transmission has completed. When a port becomes "underload", there are consecutive packet dequeue events between packet enqueue events.
- A port becomes "overwhelmed" when over-line-speed traffic exceeds buffer capacity. Either buffer overflow or consecutive packet drops can indicate a port being "overwhelmed".

Burst absorption only applies to "overloaded" ports. Because extra buffer can only be of help when there is a possibility to fully absorb burst traffic. Extra buffer is needless to "underloaded" ports and useless to "overwhelmed" ports. Burst absorption should be terminated as soon as an "overloaded" port becomes "underloaded" or "overwhelmed". *Port traffic state* will be used to determine *port control state*. Detail designs of port control states will be shown in §V-A.

B. Proactive Evacuation

When a port is transmitting long-lived over-line-speed traffic, no amount of buffer can absorb the part of the traffic that exceeds line speed. It is worth pointing out that typical long-lived over-line-speed traffic is not common in switch as the purpose of congestion control algorithm (CC) [5], [16], [23]–[25] is to avoid continuous overload scenarios. As a very recent census [26] have shown, TCP CUBIC [25] remains the dominant TCP variant on the internet and undocumented TCP variants have occupied a non-negligible proportion on the internet, some of which may focus more on maximizing link utilization when facing competition instead of maintaining a relatively short queue like TCP BBR [24], making buffer occupancy by long-lived traffic a reasonable concern. In the context of this paper, long-lived traffic is a relative concept, which refers to **aggregated** flows whose duration exceeds switch buffering time. It is possible that when multiple large flows share a bottleneck link, their aggregated fits the description of long-lived traffic. The rationality of this description is that if a flow exceeds the buffer capacity of the switch, even if its absolute duration is relatively short, from the perspective of the switch, this flow is sufficient to be considered long-lived.

When a port is transmitting long-lived over-line-speed traffic, its serving rate cannot match the arriving rate, causing rapid growth in port queue length. Once the queue length matches the port threshold, it will remain stable until transmission of the whole flow has completed in which case consecutive packet drop will occur due to threshold limitation. However, as discussed in §II-B, buffer occupation in this scenario does not benefit port throughput but significantly reduces the amount of buffer that can be used by other ports and extends the overall queuing delay. Therefore, if a port is transmitting long-lived over-line-speed traffic, its port queue should be proactively evacuated, for the purpose of avoiding meaningless buffer occupation.

Proactive evacuation can be achieved with a simple mechanism. When a switch port determines that the traffic it is transmitting is long-lived, it proactively lowers its port

[1]Switch buffering time is the time switch can buffer packets arriving at the line speed of the port, e.g 1MB buffer on a 16-port switch can be equivalently expressed as 500us buffer per port with 1Gbps line rate.

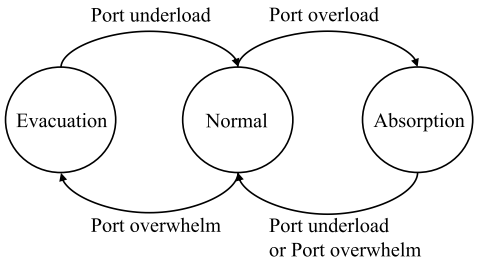


Fig. 4. Port control state transition diagram of TDT.

threshold in order to reduce its queue length. It is worth noting that queue length reduction in this circumstance does not affect port throughput. When the transmission is over, buffer management policy should quickly restore its threshold to the normal threshold for fairness consideration.

Proactive evacuation only applies to "overwhelmed" ports. For the reason that additional buffer to "overwhelmed" ports does not bring extra throughput, whereas long-term buffer occupied by these ports potentially impedes other ports from leveraging the same buffer space to absorb bursts, and increases the delay caused by queuing. Proactive evacuation should never be activated unless there is high confidence that a port will remain in the "overwhelmed" state for a relatively long time thus port throughput is not affected by the reduction of queue length. Once a port is no longer in the "overwhelmed" state, proactive evacuation should be terminated accordingly to avoid throughput loss.

V. Traffic-Aware Dynamic Threshold

In this section, we present a possible design of traffic-aware buffer management policies, Traffic-aware Dynamic Threshold (TDT). On the basis of the classic dynamic threshold [7], TDT controls port thresholds by assigning a traffic state indicator to each port and detecting traffic changes in a time-independent way. Our design controls the thresholds on the port level, without maintaining flow-level information [14], which gives our design natural scalability. Moreover, we show that parameter tuning of TDT is convenient, because it only considers the parameters of the switch itself, and avoids the use of network state-related parameters (e.g., burst duration as in EDT [8]). At the end of this section, a simple but typical example is given to show how TDT works.

A. Port-Wise Control State

In TDT, each switch port has three possible control states. When the traffic is relatively mild, the port is in **"normal" state** and controls its queue length by the shared dynamic threshold as in equation (1). When a port is transmitting burst traffic, the port state transits to "absorption", temporarily raising the threshold to absorb burst traffic as much as possible. On the other hand, when a port is transmitting long-lived over-line-speed traffic, the port turns to the "evacuation" state that proactively evacuates the queue by lowering the port threshold. State transition diagram is shown in Figure 4.

"Absorption" State: In the "absorption" state, the port threshold is the total amount of buffer divided by the total number of ports in "absorption" state. The "absorption" state is triggered by port becoming "overloaded", which indicates that the port is transmitting burst traffic. Therefore, the trigger condition for the "absorption" control state is the same as the judgment condition of port "overload", that is: *port queue length increases while no packets are dropped.*

For fairness consideration, a port should quickly lower the threshold and return to the "normal" state when additional buffer is no longer needed. Under two circumstances additional buffer is no longer beneficial: first, the port has finished transmitting burst traffic and second, the size of traffic exceeds buffer capacity. In other words, the switch should cease extra buffer allocation when a port has transited from "overloaded" state to either of the other two states.

For the former condition, two possible scenarios exist: a) After the burst traffic transmission is completed, no traffic continues to be transmitted on this port, or the rate of traffic that continues to be transmitted is low, which is manifested as *consecutive packets dequeue from the port queue* (packets dequeued between two enqueued packets); b) After the burst traffic transmission is completed, there is still traffic of a certain rate (within the line speed of the port) being transmitted in the port. In that case, queue length will decrease slowly, resulting in the port occupying excessive buffer for a long time after the completion of the burst traffic. *Cumulative packets dequeue from the port queue* (packet dequeued regardless of packet enqueued) should be restricted to ensure a timely return to the "normal" state.

For the latter condition, even the entire buffer is not enough to avoid packet drop, which means that this "burst" traffic is actually long-lived, at least from the perspective of switch buffer, and the port is actually in the "overwhelmed" state. In that circumstance, *buffer overflow event* will occur, indicating a state transition back to the "normal" state.

"Evacuation" State: In the "evacuation" state, port threshold is the total amount of buffer divided by the total number of ports (both active and inactive), which means that the evacuation threshold is equal to the "evenly split" share of the total buffer. The "evacuation" state is triggered by *cumulative drop events*, which indicates the port is transmitting traffic that cannot be absorbed by the buffer, i.e., long-lived over-line-speed traffic. A port should return from the "evacuation" state to the "normal" state once the long-term traffic has finished transmitting and port traffic state becomes "underloaded". *Consecutive packets dequeue from the port queue* can indicate the port being "underloaded" for most cases. We use a lower bound on queue length as "safety reassurance" in case over-line-speed traffic is followed by traffic transmitting at slightly below port line rate so that there might not be consecutive packets dequeue.

B. Time-Independent State Decision

To be directly deployed on high-speed switches, buffer management policies should be able to directly interact with port signals. This means circuit level design is needed for a practical buffer management policy. TDT uses a state decision module in each output port to detect traffic state and decide port control state. State decision module leverages signals of

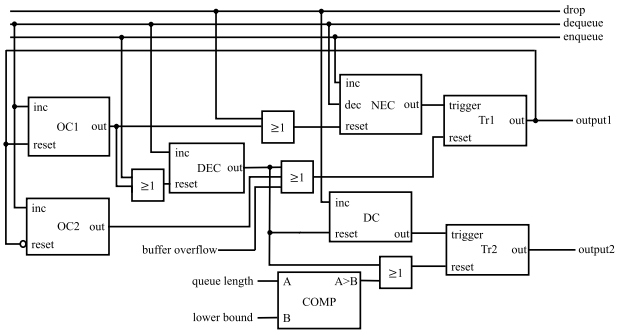


Fig. 5. Circuit logic of state decision module.

packets enqueue, dequeue and drop in the port, as well as state information such as queue length and buffer overflow signal. Two-bit output is generated by the state decision module whenever any of the above signals trigger a state decision. Next state of the port is determined by the two-bit output and the following mapping rules: (0, 0) → "normal" state; (1, 0) → "absorption" state; (0, 1) → "evacuation" state. The state decision module consists of three parts: absorption decision, evacuation decision and restoration decision. The circuit logic design of the port state decision module is shown in Figure 5. The port state decision module is composed of several counters, triggers and comparators. Notice that timers are not part of the state decision module. This is an important design choice in TDT that we will explain later.

Absorption Decision: The core part of the absorption decision is the Net Enqueue Counter (NEC) that keeps tracking the amount of net enqueue packets. When NEC reaches its threshold, it will trigger the first trigger (Tr1) to inform a state transition from "normal" to "absorption". The function of NEC is to detect rapid queue length increasing in the condition that no packets are dropped, which means there should be several reset conditions for NEC if the above requirements are not met. First, if a packet drop event occurs, the remaining buffer is not sufficient to absorb burst traffic, thus NEC should be reset for future burst detection. Second, if the first output counter (OC1) reaches its threshold, indicating at least a certain amount of time has passed before NEC reaches its threshold, it can be assumed that the arriving rate of traffic does not meet the speed requirement of burst traffic, thus NEC should also be reset. Notice that OC1 serves a similar role of determining a "time out" event, but we use an output counter instead of a timer.

Evacuation Decision: The Drop Counter (DC) is the core of the evacuation decisions. The value of DC keeps adding with packet drop events in each port. When the value of DC reaches its threshold, a signal will be sent to trigger the second trigger (Tr2) to transit port control state to "evacuation". "Evacuation" state should only be triggered by consecutive packet drop events instead of long-term accumulation of packet dropping. Therefore, when Dequeue Counter (DEC) detects consecutive packet dequeue events, it will send a signal to reset DC to restart the counting.

Restoration Decision: When a port is in the "absorption" state or "evacuation" state, it is important to restore its state to "normal" when traffic changes. When a port is in the "absorption" state, there are three possible conditions indicating state restoration: 1) DEC reaches its threshold, indicating burst traffic has finished transmitting; 2) the second output counter (OC2) reaches its threshold, indicating the port has already transmitted burst traffic of the maximum possible size allowed by the switch; 3) buffer overflow, which is the sign that the buffer is unable to fully absorb the burst traffic.

On the other hand, if a port is in the "evacuation" state, there are two conditions for state restoration: 1) DEC reaches its threshold, indicating that the port is no longer "overwhelmed"; 2) queue length is less than a lower bound. The latter condition serves the role of a "safety reassurance" if the over-line-speed traffic is followed by traffic transmitting at a rate slightly lower than the line speed. In that case, DEC might not reach its threshold but evacuation should be terminated.

Time-Independent Design: As mentioned above, unlike existing methods [8], TDT avoids the use of timers in the design of the state decision module. Instead, TDT uses two output counters to indicate "time out" events. Two major reasons are behind this design choice. First, output counters can adjust the trigger interval according to the change of port traffic. More specifically, if a port is overloaded, the port queue cannot remain empty, in this case, the output port transmits at maximum speed, i.e., port line speed. Output counter is equivalent to a timer in that case. Second, the timer timeout setting usually needs to consider the changes of the network environment which means parameters of the timer often need retuning when network condition changes. On the contrary, the parameter of output counters is directly related to switch buffer settings, which means using output counters can help achieve designs independent of the often inaccurate assumptions on specific network environments.

C. Convenient Parameter Setting

In TDT, several parameters need tuning before deployment. As discussed above, TDT avoids the use of time-related parameters. Only thresholds of several counters and one input of comparator require setting. We assume that the line rate of all output ports is equal in the same switch for simplicity. The switch has a buffer size of B and n ports.

NEC and OC1 together make absorption decisions thus their thresholds should be set together. NEC should detect rapid queue length growth, thus a smaller NEC threshold is more sensitive to small bursts and a larger NEC threshold would mean otherwise. In practice, the evenly split threshold B/n is good enough for most cases. OC1 limit the rate of burst traffic. Burst traffic with rate less than a certain value cannot trigger NEC because OC1 will reset NEC in advance. If we set that the rate of burst traffic should be at least k-times the line speed of the port, then the threshold of OC1 should be set to $\frac{1}{k-1}$ of the NEC threshold. In practice, if we believe that burst rates from incast scenarios should be at a minimum twice the port line-speed, we can set OC1 threshold to B/n.

Evacuation decision relies on DC. DC is used to detect cumulative packet drop events. Cumulative packet drops can be a result of long-live over-line-speed traffic or temporal traffic changes. Therefore, the threshold should be large enough to avoid frequent triggers. Evacuation should not be triggered

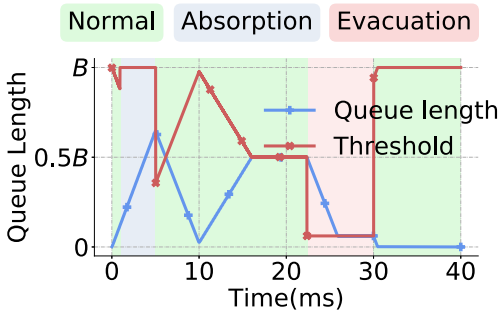


Fig. 6. An example of queue length and threshold evolution of TDT.

by periodical queue length variation caused by TCP behavior. More specifically, packet drops during congestion control response time should not be enough to trigger evacuation. When α of DT is set to 1, at most half of the total buffer is reserved when a port reaches its threshold [7]. The number of cumulative packet drops exceeds half of the total buffer indicates the traffic cannot be absorbed even with the whole buffer and for the switch, this traffic is enough to be considered as long-lived. Base on that we conservatively set the threshold of DC to $B/2$.

Restoration decision is based on DEC, OC2, and the input of the comparator. DEC detects consecutive packet dequeue events, which is a characteristic behavior rather than quantitative behavior. Therefore, DEC can be set to a small number such as 3 or 5. OC2 limits the maximum size of bursts. For burst traffic of twice the port line rate, the whole buffer can absorb burst with size up to twice the buffer space. The number of packets in a burst that the buffer can absorb decreases as the traffic rate increases. For most scenarios, we can set the threshold of OC2 to $2B$. Finally, the lower bound of queue length is a "safety reassurance" for state restoration, which should not be too sensitive considering in most cases the end of long-lived traffic should be signaled by DEC instead of the comparator. Setting it to half the evacuation threshold, i.e., $1/2 \times B/n = B/2n$ is sufficient for most situations.

D. Putting It All Together

In this part, we will use a simple but typical example to show how TDT works. First, a burst starts at time $t = 0$ and finishes at $t = 5ms$, then a long-lived slightly over-line-speed traffic starts at $t = 10ms$ and continues over-line-speed transmission until $t = 30ms$. Figure 6 shows the evolution of queue length and corresponding threshold.

Port starts at the "normal" control state, threshold decreases as queue length increases. When the state decision module detects burst traffic by rapid queue length growth, it instructs the port to change its state to "absorption" to raise its threshold to the total amount of buffer, ensuring the absorption of traffic burst. After the burst transmission has finished, the state decision module makes a restoration decision and puts the port control back to "normal". When the second flow arrives at $t = 10ms$, it does not trigger the "absorption" state because its arriving rate does not meet the rate requirement of a burst. Then this queue length of this flow is controlled by the dynamic threshold as in "normal" state. When queue length converges with the threshold, it stops increasing and packet drop occurs. When cumulative packet drop has reached a certain amount, state decision module decides that this port is in "overwhelm" traffic state, thus control state should transit to "evacuation" and threshold undergo a cliff-like drop while queue length decreases gradually due to the restriction of port transmission speed. After the long-lived over-line-speed traffic has finished transmitting, the state decision module quickly lets the "normal" state retake control and raises the threshold for future traffic.

VI. Evaluation

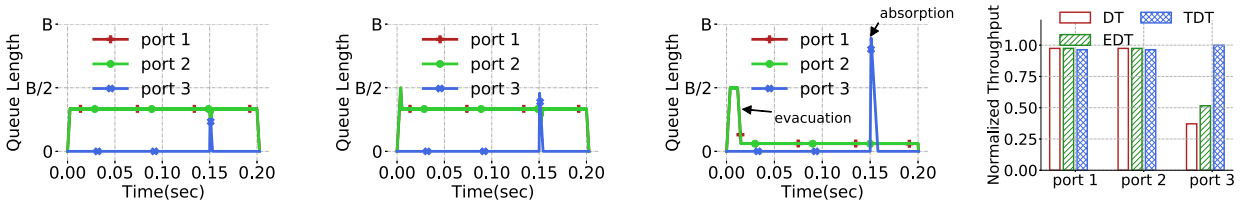
In this section, we compare the performance of TDT and existing buffer management policies with large-scale simulation based on the ns-3 network simulator[2] [27] and deep dive into the reasons behind performance gain of TDT. We use DT [7] and EDT [8] as comparison schemes of TDT. Evenly Split (ES) and Complete Sharing (CS) are also used as auxiliary baselines. Stability and convergence of congestion control behavior with TDT enabled in the switch is tested. We also implement TDT and its comparison dynamic threshold policies in a real DPDK testbed and compare their performance of flow completion times.

A. Micro-Benchmarks

In this section, we consider a 16-port shared memory switch with 1MB buffer and 1Gbps port line speed [8], [28]. Packet size is set to 1500 bytes which makes the total buffer size 667 packets. We use 1 as the control factor α of DT, EDT and the "normal" state of TDT as suggested in [7]. Parameters of EDT are set according to [8] as $C_1 = 3$, $C_2 = 8$, $TM_1 = 2.1ms$, $TM_2 = 10ms$. As for TDT, threshold of NEC, DEC, DC, OC1 and OC2 are set to 42, 3, 333, 42 and 1344, respectively, according to the suggested parameter setting in §V-C. Note that this section only serves as a demonstration of potential, ideal performance gain of TDT, thus all experiments are carefully designed for simplicity and better understanding. Results in this section are not representative of overall performance of TDT, which will be shown in §VI-B and §VI-C.

Burst Absorption: First, we use a simple scenario with long-lived traffic and burst traffic to demonstrate how TDT improves the absorption capacity of burst traffic by proactive evacuation. Queue length evolution of TDT and its comparison policies are shown in Figure 7. The first two ports transmit 2Gbps traffic throughout the entire $0.2s$ experiment. The third port starts transmitting 8Gbps burst traffic at $t = 0.15s$ and finishes $1ms$ later. As shown in Figure 7(a), due to strict queue length fairness restriction of DT, packet drop of burst occurs early. It should be pointed out that packets start dropping even before queue lengths converge, the same as the case in Figure 3(b) EDT improves its burst absorption capacity by relaxing threshold restriction for port 3, as shown

[2]Vanilla ns-3 simulator does not support shared memory switch architecture and threshold control. We implement total buffer control and port queue threshold control by adding static functions and buffer-related variables to ns-3 source code using C++ programming.



(a) DT. (b) EDT. (c) TDT. (d) Throughput.

Fig. 7. Queue length evolution of two ports transmitting long-lived over-line-speed traffic and burst traffic arrives at the third port at 0.15s.

in Figure 7(b). Unfortunately, burst absorption of EDT stops just after a while when buffer overflow occurs, as a consequence of buffer occupation of port 1 and port 2. As for TDT in Figure 7(c), early proactive evacuation of queues of port 1 and port 2 pays off when burst traffic arrives at port 3. TDT allows ports that transmit bursty traffic to use up to the entire buffer without being limited by the evacuation rate of ports occupying the buffer.

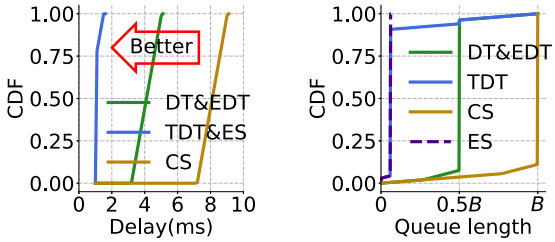
It may come as a surprise that proactive evacuation makes TDT a "fairer" policy, in terms of throughput of different ports. As shown in Figure 7(d), TDT achieves nearly the optimal throughput in all three ports whereas its comparison policies fail to bring similar additional throughput for port 3 compared with the first two ports. By getting rid of meaningless queue length restrictions, TDT can fairly provide as much additional throughput as possible for different kind of ports.

Results above can be mathematically interpreted using the analytical results from §III. We can substitute $R = 8$Gbps, $N = 2$, $M = 1$, $B = 1$MB, $\alpha = 1$ into (8) and (13). Theoretically, the max burst duration for complete absorption are $222us$ and $533us$ for DT and EDT respectively. This means that DT/EDT first drop a packet from the burst flow when its queue length is 0.19MB/0.47MB. As for TDT, the max burst duration is still $\frac{B}{M(R-C)} = 1.14ms$ which means a packet can only be dropped when the burst fully occupied the buffer. The theoretical results matches the experimental results of Figure 7 perfectly.

Delay Reduction: We use a different micro-benchmark to demonstrate TDT's improvements on delay-sensitive traffic and the reasons behind. Figure 8 shows the delay results of single active port transmitting 2Gbps traffic from $t = 0$ to $t = 0.2s$ and a delay-sensitive flow starts right after $t = 0.2s$, with 0.8Gbps arriving rate and lasts for $10ms$. We focus on the delay of the second flow. As shown in Figure 8(a), TDT has an identical delay performance as ES, which outperforms DT and EDT by a large margin. CS is considered as the upper bound for delay. The reason behind a lower delay of TDT is straightforward, reduction of average queue length. Queuing delay is determined by the total amount of packets in queue when a packet arrives at a port. As shown in Figure 8(b), by proactive evacuation of meaningless buffer occupation, TDT can reduce queue length in ports transmitting long-term traffic without affecting their throughput.

B. TDT With Homogeneous Traffic

We test TDT and its comparison policies in large-scale stochastic scenarios. In the context of this section, homogeneous

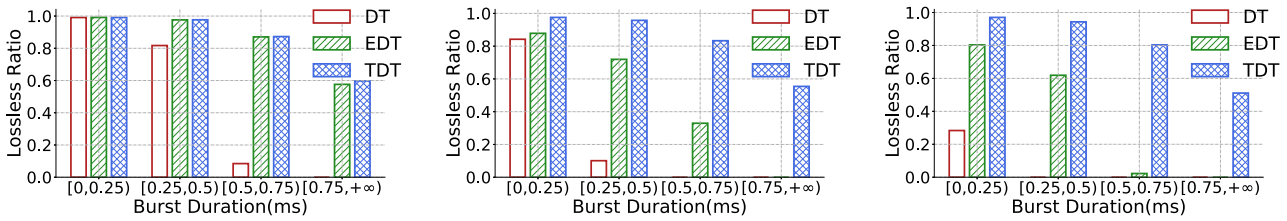


(a) CDF of delay. (b) CDF of queue length.

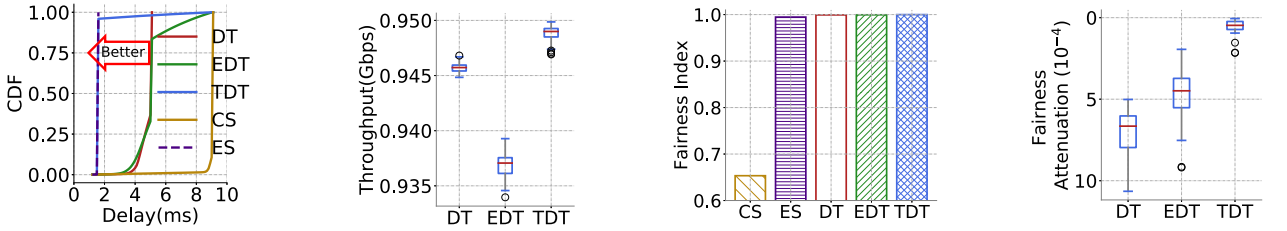
Fig. 8. TDT reduce delay by keeping a relatively short queue length. $A\&B$ indicates that scheme A and B have identical delay performance in this scenario.

traffic means that except for the background traffic, each switch port can transmit either short burst traffic or long-lived traffic. In this part, we consider unresponsive UDP traffic and leave response TCP traffic evaluations in the following parts. Switch and parameter settings are identical to §VI-A. Guided by the experiment settings in [7] and [8], we consider the following scenario: Among 16 1Gbps output ports, 8 of them are transmitting traffic composed of poisson background traffic with 20% average load and burst traffic with 8Gbps arriving rate, $250us$ average "on" time and $19.75ms$ average "off" time. The total average traffic load is 30%. We test performance of TDT and its comparison policies in terms of the absorption capability of loss-sensitive burst, the throughput of long-lived traffic, and the delay of light load short traffic with 0,1 or 2 additional port(s) transmitting long-lived over-line-speed traffic.

Loss-Sensitive: We count the lossless ratio of burst traffic within different duration intervals and show the results in Figure 9(a)-9(c). When no port is transmitting long-lived over-line-speed traffic (Figure 9(a)), TDT can achieve a similar lossless ratio as EDT given hardly any port needs evacuation. DT achieves poor absorption performance due to the strict threshold restriction. However, as shown in Figure 9(b) and Figure 9(c), when overwhelmed ports exist, the performance of EDT drops dramatically while the performance of TDT remains basically unchanged. Especially, in the case of two overwhelmed ports, almost all burst traffic longer than $0.5ms$ cannot be absorbed using DT or EDT, whereas TDT achieves an over 80% lossless ratio when bursts are no longer than $0.75ms$ and an over 50% lossless ratio otherwise. In general, the overall lossless ratios of Figure 9(c) are 5.5%, 57.8%, 92.7%, respectively, which means TDT has a $\sim$60% performance gain than the best performing existing policies when it comes to absorption of loss-sensitive burst traffic.



(a) Burst absorption with no overwhelmed port. (b) Burst absorption with one overwhelmed port. (c) Burst absorption with two overwhelmed ports.



(d) Delay of background traffic.

(e) Throughput of long-lived traffic.

(f) Overall fairness.

(g) Zooming in on fairness attenuation.

Fig. 9. Performance of TDT and its comparison policies with homogeneous traffic.

Delay-Sensitive: Figure 9(d) focuses on the delay distribution of background traffic. Notice that in Figure 9(d) only delay of background traffic in overwhelmed ports is presented, since delay in ports absorbing burst traffic cannot be optimized simultaneously as burst absorption capability on the port level. For DT and EDT, most of the time queue length is equal to the port dynamic threshold, i.e., half the total buffer space when only one port is overwhelmed and α is 1, causing a half-buffering-time queuing delay for packets. CS always allows a single port to occupy as much as the entire buffer. For an overwhelmed port, this means queue length is as long as possible within the total buffer limit, causing a queuing delay close to buffering time. With proactive evacuation, TDT can keep the average queue length similar to ES, resulting in less than $2ms$ queuing delay for more than 90% packets.

Throughput-Sensitive: As discussed in §IV-B, proactive evacuation of TDT should not affect the throughput of long-lived over-line-speed traffic. Figure 9(e) shows the throughput of the overwhelmed port. All three dynamic threshold policies have similar throughput results. EDT achieves a slightly lower throughput than DT and TDT due to its time-dependent design which potentially leads to relatively long-term buffer occupation of ports transmitting burst traffic that further causes throughput degradation of overwhelmed ports.

Fairness: TDT is fair among different ports in the long run because when multiple ports are active, it ensures each port has a similar chance to obtain additional throughput. We use the throughput fairness combined with Jain's Fairness Index [29] to measure fairness numerically. The Jain's Fairness Index is defined as (15).

$$\textit{Fairness Index} = \frac{(\sum x_i)^2}{n \sum {x_i}^2} \tag{15}$$

where x_i is the ratio of real throughput to ideal throughput (the maximum throughput that can be achieved when a port can monopolize the buffer space) of port i and n is the number of total ports. This notion of throughput fairness is much more reasonable compared with queue length fairness as it takes port demand information into consideration. Specifically, not giving buffer to a port that does not need buffer is not considered unfair anymore.

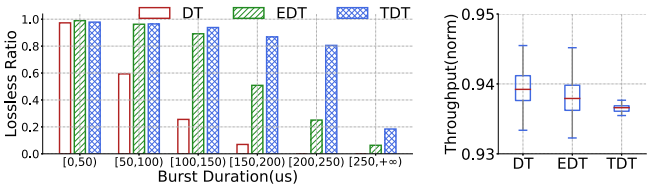
Figure 9(f) shows the overall fairness index of different ports. All policies have similar fairness indexes except CS that has no queue length restriction and is considered the most "unfair" policy. Let us zoom in on the fairness attenuation[3] of three dynamic threshold policies. TDT is "fairer" than DT and EDT because it can achieve closer to the ideal throughput on ports with burst traffic while achieving nearly ideal throughput on ports transmitting long-lived traffic.

C. TDT With Mixed Traffic

In this part, we consider a more general scenario in which each port in the switch transmits mixed traffic composed of background traffic, short burst traffic and long-lived traffic. To demonstrate that TDT performs well with different line rates, we consider a 16 port shared memory switch with a 10Gbps line rate and 3MB shared buffer. Among the 16 ports, 8 are active and transmit traffic with a 40% average load. Specifically, traffic arriving at each port consists of background traffic with a 20% average load, burst traffic with an 80Gbps rate and $200us$ average "on" time and long-lived traffic with a 15Gbps rate and $100ms$ average "on" time. The ratio between burst traffic and long-lived traffic is 9:1.

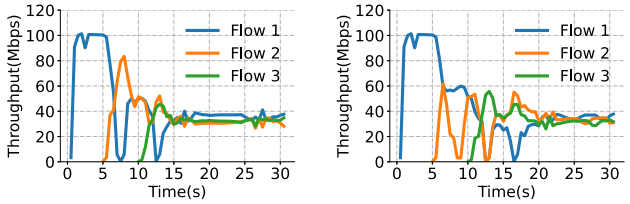
The experimental results are shown in Figure 10. In this scenario, TDT still evidently outperforms other dynamic threshold policies in terms of burst absorption. Overall, TDT can absorb over 140% additional burst traffic compared to DT and 22% compared to EDT. TDT achieves slightly (less than 1% in average) less throughput of long-lived traffic compared with

[3]This metric is proposed in [30] based on the concept of measuring relative performance degradation.



(a) Burst absorption ratio. (b) Throughput of long-lived traffic.

Fig. 10. Performance of TDT and its comparison policies with mixed traffic.



(a) DT. (b) TDT.

Fig. 12. Convergence of TCP CUBIC under different buffer management policies.

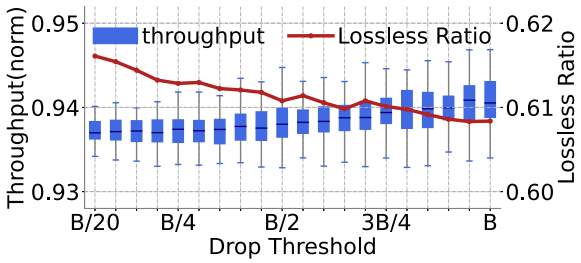


Fig. 11. Parameter sensitivity of TDT.

DT and EDT. This makes sense because the throughput difference derives from the buffer that DT and EDT allow long-lived traffic clinging onto, which TDT proactively evacuates in order to absorb more bursts. When the duration of long-lived traffic is sufficiently long, this difference is negligible.

D. Parameter Sensitivity of TDT

We test the sensitivity of TDT to model parameters. We consider one essential parameter of TDT, i.e., the threshold of the Drop Counter (DC) that determines whether and when to conduct proactive evacuation. Intuitively, if the threshold of DC is set too small, frequently triggered evacuations can cause throughput degradation. Conversely, a too large DC threshold may hinder necessary evacuation, resulting in less burst traffic being absorbed. The experimental setup is identical to §VI-C except the average duration of long-lived traffic is set to $10ms$ to make it even harder for the Drop Counter to detect whether a port is transmitting long-lived traffic.

We report the lossless ratio of burst traffic and throughput of long-lived traffic given different DC thresholds in Figure 11. Normalized throughputs are shown in box plots and lossless burst ratios are in a line plot. Generally, smaller DC thresholds lead to premature evacuations and better absorption capability whereas larger DC thresholds yield delayed evacuations and better throughput of long-lived traffic. It is worth noting that the performance variation in terms of average lossless ratio and throughput is less than 2%, indicating that TDT can perform consistently well under a wide range of DC thresholds. In practice, our suggested DC threshold $B/2$ will work just fine.

E. Stability and Convergence

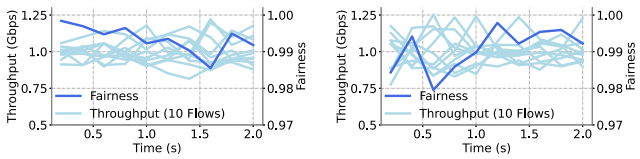
The advantages of TDT derive from its traffic-aware features. Therefore, directly discussing the stability of a buffer management policy itself may not be a good way to evaluate a highly dynamic, time-varying scheme. Nonetheless, we believe the influence of different buffer management policies on the stability of congestion control behavior should be taken into consideration. We consider two widely deployed TCP congestion control schemes, CUBIC [25], a typical loss-based congestion control scheme in wide area networks, and DCTCP [5], one of the most commonly used congestion control scheme in data center networks. We configure the switch with DT or TDT to observe the convergence properties of end host congestion control. Specifically, we use the convergence of throughput to indicate the stability of congestion control under the influence of buffer management policies.

TCP CUBIC: is a loss-based congestion control scheme that makes decisions according to whether and when a packet is dropped. However, TDT changes that by proactively raising or lowering port thresholds. Potentially, TDT can affect the stability of TCP CUBIC. To get an accurate grasp of this influence, we construct a simple scenario to test the throughput convergence of TCP CUBIC under the effect of DT and TDT. We have 4 hosts connected to a shared-memory switch through a 100Mbps link, 3 of which act as senders and 1 as a receiver. Three senders start data transmission 5 seconds apart. The shared buffer size is 2MB, base α of DT and TDT is 1 and other TDT parameters are set based on the guide from §V-C.

Results from Figure 12 show that with both DT and TDT, 3 flows can converge to their fair share of throughput (33Mbps). Indeed, TDT converges slower, we find that reasonable given the fact on a single port DT is actually equivalent to a static threshold and TDT dynamically adjusts the threshold. That being said, the results demonstrate that compared with DT, TDT does not make TCP CUBIC unstable as it can still converge.

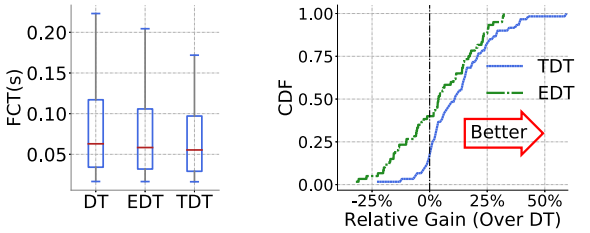
DCTCP: leverages switch ECN to avoid long-standing queues. Buffer management policies often work at the beginning of many-to-one incast scenarios. We consider a 10-to-1 incast where 10 DCTCP flows from different senders to one receiver start simultaneously and report their individual throughput and overall fairness. To construct a high-bandwidth, shallow-buffered data center scenario, we set link bandwidth to 10Gbps, the shared buffer size of the switch to 1MB and the ECN threshold to 0.72 times the bandwidth-delay product (BDP) following [31].

In Figure 13 we plot the throughput of 10 flows as well as the jain's fairness index within the first 2 seconds. It can be observed that due to threshold variation at the beginning, TDT encounters larger fluctuation than DT, resulting in a



(a) DT. (b) TDT.

Fig. 13. Convergence of DCTCP under different buffer management policies.



(a) Mean FCT. (b) CDF of relative gain over DT.

Fig. 14. TDT reduces FCT on DPDK testbed.

0.98 fairness index at around 0.5s. But TDT does not affect the overall stability of DCTCP. Both DT and TDT can achieve an average fairness index above 0.99.

F. TDT in a DPDK-Based Switch

We implement TDT and its comparison buffer management policies on a DPDK [32] testbed with 4 host servers connecting to an emulating switch with four ports. The server emulating the switch has a 28-core Intel Xeon E5-2660 2.00GHz CPU, along with 32GB of memory, a hard disk with 2TB storage, and four Intel 82599ES 10-Gigabit Ethernet NICs. We use Ubuntu 16.04 LTS GNU/Linux kernel 4.15.0 as the operating system. The other four servers have a 4-core Intel i3-3220 3.30GHz CPU and an Intel 82599ES 10-Gigabit Ethernet NIC. The buffer size of the switch server is set to 256KB and we use a rate limiter to limit the sending rate of each port to 1Gbps. TCP CUBIC [25] is used as the congestion control algorithm [33]. We use an empirical traffic generator [34] to generate traffic of simple client/server application. In our experiment, we use the WebSearch flow size distribution given in [5]. Two active ports are set to have different offered loads. A port has a 40% average load while the other is overwhelmed over a 100% average load. We focus on the flow completion time (FCT) [35] of the first port under the influence of the overwhelmed port. Using different random seeds, we repeat the experiment 50 times and report the average FCT. We have generated over 20,000 flows in total.

The parameters of EDT and TDT are adjusted according to the port number and buffer size. The threshold of dequeue counter (C1 in EDT, DEC in TDT) is set a little larger because TCP data flow has relatively larger fluctuations. Mean FCT results of TDT and its comparison policies are shown in Figure 14(a). On average, TDT reduces the flow completion time by 12% compared to DT, and 8% compared to EDT. As shown in Figure 14(b), more than 80% of the time, TDT performs better than DT, and the maximum gain can reach up to 50%. EDT also outperforms DT, but not as much as TDT. The reason for TDT's better performance is its time-independent feature and proactive evacuation design. TDT reserves more buffer for burst absorption by accurately determining port traffic type in a timely manner. On the contrary, the time-dependent design makes it difficult for EDT to accurately determine the arrival and departure of burst traffic, and the buffer occupied by the overwhelmed port limits the other port's ability to absorb burst traffic.

VII. Discussion

Scalability: TDT only leverages port-level traffic information of aggregated flows. As a result, the complexity of TDT only depends on the number of ports, not the number of concurrent flows. TDT scales easily when the number of ports or the line rate of ports increases. For example, when the number of switch ports increases from 16 to 96 or 128, TDT only needs to add the corresponding number of state decision modules, and each module only takes information from its own port, along with little global information such as total buffer usage and statistical information of global port control states. TDT is naturally scalable to port line rate because its design is irrelevant to the absolute value of port line rate. Not only can TDT scale to scenarios with larger port numbers and line rates, we believe that the performance gain of TDT can even be amplified in such scenarios. The logic behind is that as the number of ports and port line rate increases, typically the degree of sharing increases as well, implying that the average buffer for each port shrinks. Buffer occupation in such cases incurs even greater performance degradation, calling for "dynamic" use of buffer in a more extreme manner, which is what TDT provides with "absorption" and "evacuation" designs. Finally, when there exists more than one type of port (e.g. mix of 1Gbps and 10Gbps ports), simple readjustment of parameters is enough while the core design remains unchanged.

Necessity of an Intermediate "Normal" State: In the design of TDT, we use an intermediate "normal" state to control buffer usage of newly arrived traffic according to the original dynamic threshold. One might wonder if a "normal" is necessary as two states seem enough to achieve absorption and evacuation. In fact, we contend that a "normal" state is an important design choice and an essential part of TDT. The core of TDT is its traffic-aware feature so that it only allows **short** and **fast** burst traffic to have access to additional buffer. As port-level control cannot predict the duration of traffic, an intermediate can assure that TDT neither too aggressive nor conservative. Specifically, by imposing a dynamic threshold on ports whose traffic state is unclear, TDT can keep potential long-term traffic at bay while preventing unnecessary evacuation.

Trade-Offs of TDT: TDT trades off a small part of the throughput of long-lived traffic for absorption of short burst traffic. Therefore, TDT performs better in certain scenarios. It is possible that TDT may not be the best fit for scenarios where there is little burst traffic. That being said, as previous

works [18], [36] have demonstrated, bursty traffic is common in data center networks. Besides, one primary rule in parameter tuning of TDT (§V-C) is to avoid premature evacuation and only ports transmitting traffic too long to be called bursts will be evacuated. To such traffic, additional one-buffer-size throughput is marginal. In conclusion, TDT does have trade-offs in its design, but we believe it is a good bargain.

Restrictions of Port-Level Control: Port-level control provides TDT with excellent scalability but also restrictions. TDT, as any other buffer management policy based on port level control, cannot distinguish specific flows with different types of service in the same port. This mean TDT can only support different type of service in different ports whereas it cannot simultaneously optimize for flows that are sensitive to different metrics within the same port. If the aggregated traffic to one port actually contains flows with more than one type of service, differential service using flow-level information and port priority queues is inevitable. Theoretically, it is possible to expand the design of TDT to multi-priority scenarios. We leave it as our future work.

Relationship With Congestion Control: The performance of buffer management policies depends on the traffic pattern, therefore the interaction between buffer management and congestion control needs to be further explored. Current congestion control related researches [5], [16], [23] from the cloud providers' perspective, who may not have enough control over switches, often adopt classic dynamic threshold [7] as a prerequisite. Buffer management related researches, including this paper, from the switch vendors' perspective, due to the lack of access to the congestion control on end hosts, try to design more general policies insusceptible to congestion control. Nevertheless, when designing buffer management policies, one should take the traffic pattern and dynamic of congestion control into consideration. For example, previous work [1] have pointed out that for n flows sharing the same link, minimum buffer size to achieving full throughput is $BDP/\sqrt{n}$. In a scenario where n is relatively small, one should be extra careful when evacuating buffer to avoid potential throughput degradation.

From another point of view, buffer management policies like TDT can be surprisingly helpful in maintaining fairness in scenarios in which end hosts are not under the control of a control command and can potentially misbehave by using more aggressive congestion control algorithms. At the end of the day, we believe the best way is to co-design congestion control with buffer management. We leave that as our future work.

VIII. Related Work

Besides EDT [8], there are several other variants of the classic dynamic threshold policy [7]. The dynamic neural sharing [9] scheme uses neural networks to allocate buffer by predicting future traffic patterns based on the self-similar model. However, the dynamic neural sharing suffers from the inaccurate assumptions and predictions of traffic. Flow-aware Buffer Sharing (FAB) [14] leverages flow-level information to allocate different control parameters to different flows. By composing different thresholds on flows with different sizes, FAB can achieve a similar "evacuation" effect as TDT. The problem with FAB is that in order to enable flow-level buffer management, the switch inevitably needs to maintain information of every flow. Even with carefully designed simplification methods, FAB still faces severe scalability issues. Classic dynamic threshold has also been expanded scenarios with priority [10] where each port has several priority queues. Priority queues add complexity, as well as new challenges, to buffer management. But as discussed in §VII, priority queues are a possible, perhaps the most straightforward way for a switch port to support different types of service simultaneously, and [10] could be the foundation for designing a multi-priority version of TDT. Recently, FB (Flexible Buffer Management) [37] has been proposed to address the minimum buffer guarantee and burst-tolerance guarantee problems of shared memory switches, whose main focus is on multi-priority scenarios.

There has been a recent work that designs buffer-aware Explicit Congestion Notification (ECN) called BCC [31]. BCC considers the effect of shared buffer on congestion control and active queue management (AQM), and uses Dynamic Threshold [7] as a prerequisite for buffer management when designing ECN. BCC cannot be directly compared with TDT, yet we believe co-designing buffer management with congestion control and active queue management is a very promising research direction.

IX. Conclusion

In this paper, we demonstrate that buffer occupation caused by blind pursuit of buffer utilization and misleading definition of port fairness does not contribute to throughput but undermines other metrics. With analytic methods, we further show that meaningless buffer occupation undermines the absorption capability of existing buffer management policies. To avoid that, we propose the design principles of traffic-aware buffer management policy, along with a specific policy, TDT, that simultaneously optimize for loss-sensitive burst traffic, throughput-sensitive long-lived traffic and delay-sensitive short traffic. By utilizing the buffer as much as possible to absorb burst traffic and proactively evacuate meaningless buffer occupation, TDT can fully exploit the benefit of buffer. We test TDT on the ns-3 simulator and a real DPDK switch prototype. Experimental results show TDT outperforms existing policies in terms of several practical metrics, while maintaining throughput fairness among ports.

References

[1] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 281–292, Aug. 2004.

[2] S. Das and R. Sankar, "Broadcom smart-buffer technology in data center switches for cost-effective performance scaling of cloud applications," Broadcom, San Jose, CA, USA, White Paper, 2012. [Online]. Available: https://docs.broadcom.com/docs/12358325

[3] *Intelligent Buffer Management on Cisco Nexus 9000 Series Switches White Paper*. Accessed: Aug. 15, 2020. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-s%eries-switches/white-paper-c11-738488.html

[4] M. Alizadeh *et al.*, "CONGA: Distributed congestion-aware load balancing for datacenters," in *Proc. ACM Conf. SIGCOMM*, Aug. 2014, pp. 503–514.

[5] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 63–74, Aug. 2010.
[6] U. Cummings, A. Lines, P. Pelletier, and R. Southworth, "Shared-memory switch fabric architecture," U.S. Patent 7 814 280, Oct. 12, 2010.
[7] A. K. Choudhury and E. L. Hahne, "Dynamic queue length thresholds for shared-memory packet switches," *IEEE/ACM Trans. Netw.*, vol. 6, no. 2, pp. 130–140, Apr. 1998.
[8] D. Shan, W. Jiang, and F. Ren, "Analyzing and enhancing dynamic threshold policy of data center switches," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 9, pp. 2454–2470, Sep. 2017.
[9] H. Yousefi'Zadeh and E. A. Jonckheere, "Dynamic neural-based buffer management for queuing systems with self-similar characteristics," *IEEE Trans. Neural Netw.*, vol. 16, no. 5, pp. 1163–1173, Sep. 2005.
[10] E. L. Hahne and A. K. Choudhury, "Dynamic queue length thresholds for multiple loss priorities," *IEEE/ACM Trans. Netw.*, vol. 10, no. 3, pp. 368–380, Jun. 2002.
[11] A. Kesselman and Y. Mansour, "Harmonic buffer management policy for shared memory switches," *Theor. Comput. Sci.*, vol. 324, nos. 2–3, pp. 161–182, Sep. 2004.
[12] G. Ascia, V. Catania, and D. Panno, "An evolutionary management scheme in high-performance packet switches," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 262–275, Apr. 2005.
[13] S. X. Wei, E. J. Coyle, and M.-T. Hsiao, "An optimal buffer management policy for high-performance packet switching," in *Proc. IEEE Global Telecommun. Conf., Countdown New Millennium, Conf. Rec.*, Dec. 1991, pp. 924–928.
[14] M. Apostolaki, L. Vanbever, and M. Ghobadi, "FAB: Toward flow-aware buffer sharing on programmable switches," in *Proc. Workshop Buffer Sizing*, Dec. 2019, pp. 1–6.
[15] I. Cidon, L. Georgiadis, R. Guerin, and A. Khamisy, "Optimal buffer sharing," *IEEE J. Sel. Areas Commun.*, vol. 13, no. 7, pp. 1229–1240, Sep. 1995.
[16] Y. Zhu *et al.*, "Congestion control for large-scale RDMA deployments," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 523–536, Sep. 2015.
[17] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 92–99, 2010.
[18] Q. Zhang, V. Liu, H. Zeng, and A. Krishnamurthy, "High-resolution measurement of data center microbursts," in *Proc. Internet Meas. Conf.*, Nov. 2017, pp. 78–85.
[19] D. Shan, F. Ren, P. Cheng, R. Shu, and C. Guo, "Micro-burst in data centers: Observations, analysis, and mitigations," in *Proc. IEEE 26th Int. Conf. Netw. Protocols (ICNP)*, Sep. 2018, pp. 88–98.
[20] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 50–61, 2011.
[21] H. R. Varian, *Intermediate Microeconomics With Calculus: A Modern Approach*. New York, NY, USA: WW Norton & Company, 2014.
[22] B. Vamanan, J. Hasan, and T. N. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 115–126, Sep. 2012.
[23] Y. Li *et al.*, "HPCC: High precision congestion control," in *Proc. ACM Special Interest Group Data Commun.*, 2019, pp. 44–58.
[24] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based congestion control," *Queue*, vol. 14, no. 5, pp. 20–53, 2016.
[25] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *ACM SIGOPS Operating Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.
[26] A. Mishra, X. Sun, A. Jain, S. Pande, R. Joshi, and B. Leong, "The great internet TCP congestion control census," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 3, no. 3, pp. 1–24, Dec. 2019.
[27] G. F. Riley and T. R. Henderson, "The *ns*-3 network simulator," in *Modeling and Tools for Network Simulation*. Berlin, Germany: Springer, 2010, pp. 15–34.
[28] *Packet Buffers*. Accessed: Jun. 29, 2020. [Online]. Available: https://people.ucsc.edu/~warner/buffer.html
[29] R. Jain, A. Durresi, and G. Babic, "Throughput fairness index: An explanation," *ATM Forum contribution*, vol. 99, no. 45, pp. 1–13, 1999.
[30] D. C. Reeve, "A new blueprint for network QoS," Ph.D. dissertation, Comput. Lab., Univ. Kent, Canterbury, Kent, U.K., 2003. [Online]. Available: http://www.cs.kent.ac.uk/pubs/2003/1892
[31] W. Bai, S. Hu, K. Chen, K. Tan, and Y. Xiong, "One more config is enough: Saving (DC)TCP for high-speed extremely shallow-buffered datacenters," *IEEE/ACM Trans. Netw.*, vol. 29, no. 2, pp. 489–502, Apr. 2021.
[32] Linux Foundation. (2015). *Data Plane Development Kit (DPDK)*.[Online]. Available: http://www.dpdk.org
[33] B. Levasseur, M. Claypool, and R. Kinicki, "A TCP CUBIC implementation in ns-3," in *Proc. Workshop ns-3 (WNS3)*, 2014, pp. 1–8.
[34] *Empirical-Traffice-Generator*. Accessed: Jun. 29, 2020. [Online]. Available: https://github.com/datacenter/empirical-traffic-gen.git
[35] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 59–62, Jan. 2006.
[36] R. Kapoor, A. C. Snoeren, G. M. Voelker, and G. Porter, "Bullet trains: A study of NIC burst behavior at microsecond timescales," in *Proc. 9th ACM Conf. Emerg. Netw. Exp. Technol.*, Dec. 2013, pp. 133–138.
[37] M. Apostolaki, V. Addanki, M. Ghobadi, and L. Vanbever, "FB: A flexible buffer management scheme for data center switches," 2021, *arXiv:2105.10553*.
[38] S. Huang, M. Wang, and Y. Cui, "Traffic-aware buffer management in shared memory switches," in *Proc. IEEE Conf. Comput. Commun.*, May 2021, pp. 1–10.

Sijiang Huang received the B.E. degree in information engineering from the Beijing University of Posts and Telecommunications, Beijing, China. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University, Beijing. His current research interests include data center networks and digital twins networks.

Mowei Wang (Graduate Student Member, IEEE) received the B.E. degree in communication engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2017. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University, China. His research interests include data center networks and data-driven networks.

Yong Cui (Member, IEEE) received the B.E. and Ph.D. degrees in computer science and engineering from Tsinghua University, China, in 1999 and 2004, respectively. He is currently a Full Professor with the Department of Computer Science, Tsinghua University. His major research interests include mobile cloud computing and network architecture. He served or serves on the Editorial Board of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS (TPDS), IEEE TRANSACTIONS ON CLOUD COMPUTING (TCC), and the IEEE INTERNET COMPUTING.