Fair Packet Enqueueing and Marking in Multi-Queue Datacenter Networks

Sultan Alanazi and Bechir Hamdaoui

School of EECS, Oregon State University, Corvallis, Oregon, USA
{alanazsu,hamdaoui}@oregonstate.edu

***Abstract*—Recently, Explicit Congestion Notification (ECN) has been leveraged by most Datacenter Network (DCN) protocols for congestion control to achieve high throughput and low latency. However, the majority of these approaches assume that each switch port has one queue while current industry trends towards having multiple queues per switch port. To this end, we propose ML-ECN, a fairness-aware packet enqueueing and multi-level probabilistic ECN marking scheme for DCNs enabled with multiple-service, multiple-queue switch ports. The main design of ML-ECN relies on the separation between small, medium, and large flows by dedicating multiple queues for each flow class to ensure fair enqueueing. ML-ECN employs one ECN marking threshold for the small queue class and multiple thresholds with a probabilistic marking for the medium and large queue classes to achieve low latency for mice (small) and high throughput for elephant (large) flows. In addition, ML-ECN performs fairness-aware ECN marking that ensures that packets of short flows are not getting marked due to buffer buildups caused by longer flows. Large-scale ns-2 simulations show that ML-ECN outperforms existing approaches at different performance metrics.**

***Index Terms*—Datacenter networks, congestion control, queue management, explicit congestion notification, fairness.**

I. Introduction

Improving datacenter network (DCN) performance has recently been the focus of great deal of research in both academia and industry [1–5]. Cloud datacenters host a variety of applications and services with stringent performance needs and conflicting requirements. For instance, MapReduce [6] and Hadoop [7] require high throughput, whereas group video calls [8] and web services require low latency. Moreover, some services such as online data-intensive applications [9] require both high throughput and low latency. Thus, it is important to have datacenter transport-layer protocols that can deliver both low latency and high throughput simultaneously. However, the Internet's dominant transport layer protocol, TCP, cannot satisfy DCN's requirements [10].

To meet these requirements, many DCN transport-layer proposals such as those proposed in [11–14] have employed ECN (explicit congestion notification) marking [15] and show that such a marking can deliver both low latency and high throughput simultaneously with a proper tuning of the ECN marking threshold [13]. ECN-based transport mechanisms such as DCTCP [10] and DCQCN [14] have been broadly used in industry because of their simplicity. Although DCTCP and other transport schemes can deliver low latency and high throughput, their ECN marking schemes are designed for switches with only one single queue per switch port. However, current industry trends towards manufacturing switches with up to 8 classes of service queues per port [16–18].

This work was supported in part by US National Science Foundation (NSF) under awards No. 1923884 and No. 2003273.

Multi-Service Multi-Queue ECN (MQ-ECN) [19] has been proposed to optimize the ECN marking scheme of DCTCP in multi-queue scenarios. MQ-ECN periodically adjusts the ECN marking threshold for each queue independently based on the queue weight and the round-trip time. However, imprecise measurement is considerable in MQ-ECN since data traffic in DCN is bursty in nature. Moreover, setting the threshold dynamically requires periodic round-trip time measurement which incurs non-negligible overhead for switches [19]. DemePro [20], on the other hand, decides to measure the total queue length in the port buffer instead of measuring the round-trip time of each queue periodically. DemePro marks the head packet in the most congested queue when the total queue length becomes greater than per-port threshold. A-ECN [21] designs an adaptive ECN marking scheme that does not consider time interval, as in MQ-ECN, to update the marking threshold. Instead, A-ECN uses the number of enqueue packets as the interval to update the marking threshold. MM-ECN [22] designs a mix marking scheme that combines front-end ECN marking and random ECN marking to provide best burst tolerance and short-flow friendliness simultaneously.

Although the ECN marking schemes mentioned above can deliver low latency and high throughput, none of them consider fairness among flows when performing ECN marking. Therefore, in this paper, we propose ML-ECN, a multi-level ECN marking mechanism that provides fair ECN marking among flows while delivering high throughput and low latency simultaneously. We employ multiple thresholds to perform probabilistic ECN marking. The minimum threshold triggers ECN marking, with low probability, on the largest flow sizes indicated by the stage value tagged in the packet header of each flow. This ensures no queue underflow in the presence of micro-burst, thereby resulting in a good network performance. Then, as the threshold level increases, ML-ECN triggers ECN marking, with low probability, on smaller flow sizes compared to previous threshold levels. Furthermore, ML-ECN increases the marking probability on the larger flows marked on previous threshold levels. The idea behind these marking decisions is to reduce queues buildups without negatively impacting the performance of smaller flows. The maximum threshold triggers ECN marking on all incoming packets with maximum probability, except that flows with the smallest stage value receive lower marking probability. The aggressive ECN marking performed on the maximum threshold level mitigates packets drop at the tail of the queue. To this end, the key features of ML-ECN are as follows.

1) It prevents excessive queueing delays of short flows when sharing the same queue with larger flows. ML-ECN does so by classifying incoming flows into small, medium, and large service queues based on the flows' sizes.
2) It employs a single marking threshold on small service queues to ensure low latency for mice flows.
3) It employs both per-port and per-queue packet marking on medium and large service queues to avoid frequent packet drops when multiple queues are concurrently active and to maintain weighted fair sharing among same-class queues.
4) It ensures fairness-aware ECN marking that only marks large flows at lower threshold levels. This guarantees fairness among flows as marking is applied on the flows that contribute the most to the queue buildup.

We conduct extensive experiments using ns-2 simulator to show the performance of ML-ECN. We run five different workloads, a web search workload [10], a cache workload [7], a data mining workload [23], a Hadoop workload [7] and a mixture of these four workloads on Leaf-Spine topology. The results show that ML-ECN can significantly reduce the Flow Completion Time (FCT) for short flows while maintaining high throughput performance for large flows as compared with existing schemes. For example, the FCT performance reduction of ML-ECN, compared to the other schemes, exceeds 39%, 55%, 60%, 40% and 44% when running mix, web search, cache, data mining, and Hadoop workloads respectively. The experiments also show that ML-ECN ensures fairness marking for short flows and can further reduce the packet drop rates compared to the other schemes.

The paper is organized as follows. Sections II and III motivates and presents ML-ECN, repectively. Section IV evaluates ML-ECN on a large-scale network simulator. Section V outlines related work. Finally, Section VI concludes this work.

II. MOTIVATION

A. Why Multi-Level ECN-Marking Thresholding?

The widely used ECN-based active queue management protocols like DCTCP [10] and DCQCN [14], designed specifically for switches with single-queue ports, fall short when applied to today's commodity switches enabled with multi-queue ports. To the best of our knowledge, MQ-ECN [19] is the first to explore the unsuitability of single, fixed thresholding in multi-queue scenarios and does so as follows.

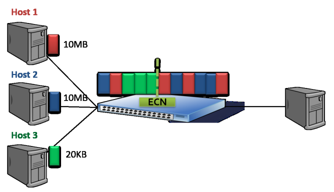


Fig. 1: Hosts 1,2 send large flows; Host 3 sends small flow; all sent to same receiver.

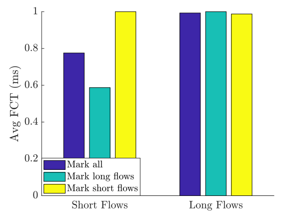


Fig. 2: Impact of ECN marking mechanism on short flows and large flows.

- Per-queue ECN with standard threshold, which sets the standard threshold for each queue independently. An arrival packet gets ECN marked if the per-queue buffer occupancy exceeds the per-queue standard threshold. This ensures high throughput (benefiting large flows) but can lead to poor latency (for small flows) when multiple queues are concurrently active. The standard threshold K can be calculated as[24, 25]

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

where C is link capacity, RTT is average round-trip time, and λ is a variable parameter related to the congestion control algorithms. For example, DCTCP sets $\lambda = 0.17$ while the approach in [13] sets $\lambda = 1$.
- Per-queue ECN with minimum thresholds, which divides the standard threshold among all the queues statically according to their weights. Although low latency is guaranteed, the achievable throughput could be degraded when only a few queues are active.
- Per-port ECN with standard thresholds, which uses the same standard threshold defined for the per-queue on each port buffer (instead of individual queues). An arrival packet gets ECN marked if the per-port buffer occupancy exceeds the per-port standard threshold. Per-port ECN can maintain both high throughput and low latency, but can't guarantee weighted fair sharing across different queues.

From these above scenarios as well as other findings using single thresholding approaches [20], we conclude that it is hard to fine-tune a single, fixed threshold values that can achieve both high throughput and low latency in the presence of network flows with varying sizes.

Motivated by the above observations, this paper proposes a multi-threshold-per-queue marking scheme that strives to deliver low latency for small flows while achieving high throughput for large flows.

B. Why Fairness-Aware ECN Marking?

Existing ECN marking schemes leverage instantaneous queue length and a single ECN marking threshold to achieve high throughput and low latency. They mark all packets when the instantaneous queue length exceeds the single threshold. However, these schemes perform poorly in fairness, since their ECN marking neglects the size of the flows. In current marking schemes, a short flow can be penalized due to queue pressure caused by multiple large flows. So packet marking should also consider flow sizes in order to maintain fairness among multiple flows. Figure 1 shows a scenario where fairness is not achieved among multiple flows. In this scenario, there are 3 flows between 3 different hosts and one receiver. Hosts 1 and 2 are sending large flows F1 and F2 (Red and Blue) with 10 MB sizes while Host 3 is sending a short flow F3 (Green) with 20 KB in size. We can observe that flows F1 (Red) and F2 (Blue) are sending more packets (putting more pressure on the queue), as they are large flows, causing the queue to build up quickly and reach the marking threshold. Upon their arrivals, flow F3 (Green) packets get ECN marked. As a result, Host 3 reacts to the packet marking by reducing the sending rate of the flow, leading to poor performance.

To quantify this impact, we conduct three experiments by running web search workload [10] on one of the widely deployed topologies in datacenter networks, Leaf-Spine topology. We generate 50K flows on each simulation. In the first experiment, we perform ECN marking on all packets when a queue length exceeds the marking threshold. For the second experiment, we only mark packets that belong to large flows. In the last experiment, we only mark packets that belong to short flows. Figure 2 shows the performance impact of ECN marking on short and large flows. Observe that the average FCT of short flows, when only large flow packets are marked, has decreased by about 25% compared to their performance when all packets get ECN marked. On the other hand, there is no performance degradation for large flows. Therefore, flow size information needs to be carefully accounted for to achieve fairness. That is said, flow size information may not always be available beforehand. For example, database systems (e.g., Microsoft SQL Servers [26]) do not necessarily wait until the end of the query execution to publish the query outcome. Instead, they send partial query results as they are created. For this scenario, the total flow size will not be available before the flow starts. Another example is HTTP content transfer, where content is divided into chunks and the chunks are dynamically transferred as they are created in real time. For DC networks, multiple DC applications use chunked transfer to dump database content into OpenStack Object Storage [27], and for such applications, flow sizes again are not available beforehand.

III. ML-ECN: The Proposed Multi-Queue Fair Packet Marking Scheme

A. Design Decision & Goals

ML-ECN is proposed to further enhance the performance of the ECN marking scheme of DCTCP in datacenters with multiple queues per switch port. In designing ML-ECN, we targeted a solution with three key design goals:

- **Low latency:** Ensure low queueing delay for small, latency-sensitive flows.
- **High throughput**: Ensure full utilization of the network bandwidth for large, throughput-sensitive flows.

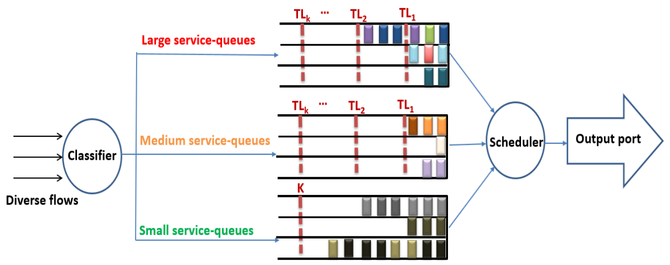


Fig. 3: ML-ECN Overview

- **Fairness:** Differentiate between flows during the enqueueing and ECN marking process. Smaller flows should not get penalized by ECN marking due to a high buffer pressure caused by larger flows.

B. ML-ECN Design Details

We now present ML-ECN, our proposed fairness-aware Multi-Level probabilistic ECN marking scheme for datacenter networks with multi-queue switch ports. At its core, ML-ECN has two components: packet tagging and switch design.

1) Packet Tagging: Maintaining per-flow state at switches requires counting and storing the number of bytes sent for each flow, which is not supported in existing commodity switches. Therefore, ML-ECN follows the work in [28] and distributes packet tagging (indicating a flowâĂŹs sent size) to end hosts as it is easy for TCP senders to count the number of bytes sent when packets are delivered. Thus, switches can leverage the tagged information to perform fairness-aware packet enqueueing and ECN marking. ML-ECN utilizes Weighted Round Robin (WRR) for scheduling among different queues. However, it is difficult to carry the accurate flow size information in the packet header, for switches to utilize. Storing accurate values of flow size information requires many reserved bits in the packet header, which is not practical.

To minimize the overhead (i.e., assigned packet header bits) associated with carrying exact flow sizes, ML-ECN divides flows into several stages, each representing a flow size range, and tags each packet with its associated stage. Embedding the stage ID instead of the exact flow size requires a lesser number of header bits. Therefore, a flow in ML-ECN can go through K stages S_i, $1 \leq i \leq K$, during its lifetime. Thus, there are $K-1$ elevation thresholds α_j, $1 \leq j \leq K-1$, and we assume that $S_1 < S_2 \ldots < S_k$ and $\alpha_1 < \alpha_2 \ldots < \alpha_{K-1}$.

At end hosts, when a new flow is initialized, its packets gets tagged with the first stage, S_1, and as more bytes are sent, packets are tagged with increasing stages S_j $(2 \leq j \leq K)$ and are buffered in a lower weighted queue based on the queueing mechanism used by the network switches. A threshold α_{j-1} elevates a flow from stage S_{j-1} to S_j.

2) Switch Design: As shown in Figure 3, ML-ECN switch design consists of three main components: a classifier, an ECN marker, and a scheduler. First, the classifier enqueues arriving packets into one of the three queues, small, medium or large, based on the stage value tagged in the IP DSCP packet header field. Then, the ECN marker decision is made based on the ECN-marking mechanism applied for each queue class. Finally, the scheduler dequeues packets using Weighted Round Robin (WRR) scheduling.

Algorithm 1 Classifier

```
1:  Processing received packet p
2:  $flow_{size} \leftarrow p.dscp$
3:  $flow_{id} \leftarrow p.flowid$
4:  $Q_s$ total number of small queues
5:  $Q_m$ total number of medium queues
6:  $Q_l$ total number of large queues
7:  $T_{small}$ threshold for small flows
8:  $T_{large}$ threshold for large flows
9:  if $flow_{size} \leq T_{small}$ then
10:     $q_{number} \leftarrow flow_{id} \mod Q_s$
11:     $q_{number}.enque(p)$
12: else if $flow_{size} \geq T_{large}$ then
13:     $q_{number} \leftarrow flow_{id} \mod Q_l$
14:     $q_{number}.enque(p)$
15: else
16:     $q_{number} \leftarrow flow_{id} \mod Q_m$
17:     $q_{number}.enque(p)$
18: end if
```

Algorithm 2 ECN Marking in Medium Queues

```
1:  Classify packet p into $(P_{small}, P_{medium}, P_{large})$
2:  $Pr_1 \leftarrow TL_1 \,/\, K$
3:  $Pr_2 \leftarrow TL_2 \,/\, K$
4:  $Pr_3 \leftarrow TL_3 \,/\, K$
5:  if $Port_{length} > K$ then
6:      if $Q_{length} \leq TL_1$ then
7:          Don't mark packet
8:      else
9:          Mark packet
10:     end if
11: else
12:     if $TL_1 \geq Q_{length} < TL_2$ then
13:         Mark $P_{medium}$ packet with probability $Pr_1$
14:         Mark $P_{large}$ packet with probability $Pr_2$
15:     else if $TL_2 \geq Q_{length} < TL_3$ then
16:         Mark $P_{small}$ packet with probability $Pr_1$
17:         Mark $P_{medium}$ packet with probability $Pr_2$
18:         Mark $P_{large}$ packet with probability $Pr_3$
19:     else
20:         Mark $P_{small}$ packet with probability $Pr_2$
21:         Mark $P_{medium}$ packet with probability $Pr_3$
22:         Mark $P_{large}$ packet with probability $1$
23:     end if
24: end if
```

a) Classifier: Current commodity datacenter switches support up to 8 queues per port. ML-ECN arranges these queues, based on the type of flows they serve, into small, medium, and large service queues. Upon receiving an incoming packet, the classifier puts the packet into one of the queues by comparing the stage value tagged in the IP DSCP header field to a set of thresholds determined by the datacenter provider as shown in Algorithm 1. That is, if the stage value is smaller than or equal to T_{small}, an arriving packet is enqueued to one of the small service queues, and if it is greater than T_{small} but smaller than or equal to T_{medium}, the packet is enqueued to one of the medium service queues. Otherwise, the packet is enqueued to one of the large service queues.

Separating flows into small, medium and large flows allows ML-ECN to minimize the average Flow Completion Time (FCT) among flows. Furthermore, the separation helps ML-ECN in maintaining fairness for smaller flows by allowing the scheduler to easily prioritize them over larger flows, to emulate the shortest job first scheduling, which results in lower latency for small flows. It also prevents short flows from experiencing large queueing delays when sharing the same queue with larger flows. The reason is that large flows have many more packets arriving at the queue that perform First In First Out (FIFO) scheduling, compared to small flows. The scheduler also prevents penalizing short flows by traditional ECN marking due to high pressure on queue caused by large flows. For large flows, instead of putting them in lower weighted queues at the beginning of their transmission, ML-ECN allows them to start at the highest weighted queues (in small service queues) and demotes them to lower weighted queues as they grow in size. This ensures high throughput for large flows.

b) ECN Marker: As discussed in Section II, ECN-marking based on a single threshold value cannot always guarantee high throughput, low latency, and fairness in the presence of flows with various sizes. To overcome the limitations of a single ECN marking threshold, we leverage the classification of queues into different classes and design a specific ECN-marking scheme for each queue class as follows:

- Small service queues: We employ per-queue standard threshold K for each queue in this class. Packets are marked only when their corresponding queue exceeds the standard threshold. We design these queues to be the starting point of all flows until they grow in size and get migrated to medium service queues. Therefore, employing the standard threshold ensures that short flows are not marked at early queue buildup. It also ensures that the queues can absorb traffic burst.
- Medium service queues: In contrast to the ECN marking in small service queues, where the marking is based on the length of each queue independently, the marking design in medium service queues is based on both the current queue length and the total length of all queues in the port buffer as shown in Algorithm 2. We first classify packets based on the current sent size of their flows into small, medium, and large (line 1). Then we generate different marking probabilities to trigger packet marking on each threshold level (line 2 to line 4). The algorithm triggers packet marking if the total length of

all queues exceeds the standard threshold K (line 5). However, instead of marking all enqueued packets and violating the weighted fair sharing among the queues in this class, we impose a condition on each queue that prevents marking packets if the queue length is less than the minimum threshold (line 6 to line 9). Therefore, performing ECN marking based on the total length of all queues mitigates frequent packet drops when multiple queues are concurrently active. On the other hand, if the total length is is less than K then we apply ECN marking on each queue independently based on multi-level thresholds as follows.

$$TL_i = i \times \frac{K}{L}, i \in [1, L] \tag{2}$$

where TL_i, K, and L are respectively the i-th, standard thresholds and the total number of thresholds. Furthermore, for each threshold level employed on the service queues, we perform fairness-aware probabilistic ECN marking on a selective range of flows to ensure that smaller flows are not marked (penalized) at the early stages of queue build-up. Therefore, at the first (minimum) threshold (line 12 to line 14), ML-ECN only performs ECN marking, with lower probabilities, on medium and large flows. As the threshold level increases, ECN marking is also performed, with low probability, on small flows while increasing the marking probability on the medium and large flows (line 15 to line 22). Thus, packet marking in ML-ECN is proportional to the queue length.

- Large service queues: We employ similar ECN-marking as in the medium service-queue but with the following change. Instead of applying selective marking, based on flow sizes, when queue length reaches each threshold level, the scheme in large service queues marks all packets at each threshold level with increasing marking probability.

c) Scheduler: In today's datacenters, each port of commodity switches has up to 8 queues. To dequeue packets from these queues, the following packet schedulers supported in most commodity switches are used:

- Weighted Round Robin (WRR): This scheduler assigns a weight for each queue. The number of packets dequeued from a queue is proportional to the weight of the queue. Thus, WRR can provide guaranteed bandwidth.
- Deficit Weighted Round Robin (DWRR): This type of scheduler uses a deficit counter to provide more desired bandwidth allocation than WRR.
- Strict Priority: Packets are dequeued strictly based on their priorities, so packets from lower priority queues are dequeued only after the highest priority queues have dequeued all their packets.

With packets enqueued into different queues according to their current stage level, ML-ECN's switches perform WRR scheduling, which is a built-in function on existing commodity switches. ML-ECN leverages the classification of queues into small, medium, and large service queues and assigns weights to the queues with one objective in mind, achieving low latency for small flows without negatively impacting the throughput of large flows. Therefore, ML-ECN assigns higher weights to the queues in the small service queue category with all the queues in this category being assigned the same weight. On the other hand, all queues in the medium service queue category are assigned lower weight compared to the ones in the small service queues. Moreover, queues in the large service category are assigned the lowest weight compared to other queues in the other categories.

In ML-ECN, we choose the round robin scheduler over strict priority scheduling to avoid the starvation problem associated with strict priority queueing. For example, if the number of flows in high-priority queues becomes excessive, flows in lower-priority queues may starve and trigger TCP timeouts, which result in performance degradation.

C. Features of ML-ECN

We now explain how ML-ECN achieves the targeted design goals described in Section III-A.

1) Low latency for small flows:

- ML-ECN migrates flows from small service queues as the queues increase in size to prevent short flows from experiencing large queueing delays when sharing the same queue with larger flows.
- ML-ECN assigns high weights to small service queues to help maintain shallow buffers and ensure short FCTs for short flows.
- ML-ECN avoids marking small flows at low buffer occupancy to ensure high sending rates for small flows. End-hosts react to marked packets by slowing down the sending rate of flows.

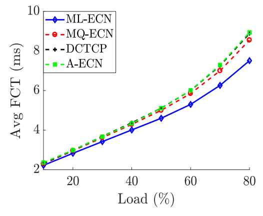
2) High throughput for large flows:

- ML-ECN allows large flows to begin at small service queues, so they can leverage the high weight assigned to small service queues. As they grow in size, they will be demoted to medium service queues and leverage their weights before being placed at lower weighted queues.
- ML-ECN employs WRR scheduling, which ensures bandwidth allocation for large flows without experiencing resource starvation.
- ML-ECN performs no marking on large service queues if the queue length is less than the minimum threshold even if the total length of all queues exceeds the standard threshold.

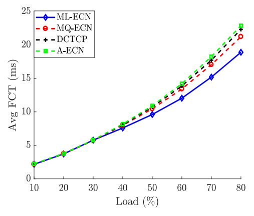
3) Fairness: To maintain fairness for short flows, ML-ECN only marks packets in small service queues if the queue length exceeds the standard threshold K. For medium service queues, ML-ECN performs packet marking on shorter flows with lower probability compared to longer flows.

IV. Performance Evaluation

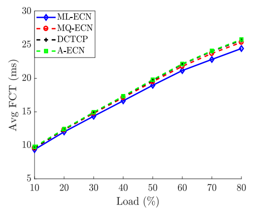
In this section, we evaluate the performance of ML-ECN in large-scale datacenter networks using ns-2 simulation. We



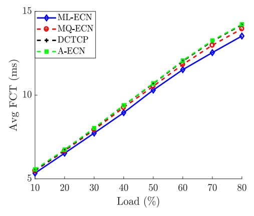
(a) WRR:Mix



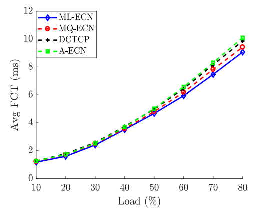
(b) WRR:Web search



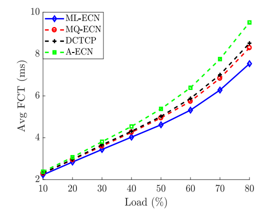
(c) WRR:Data mining



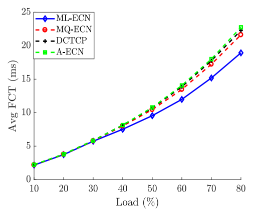
(d) WRR:Hadoop



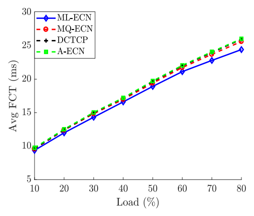
(e) WRR:Cache



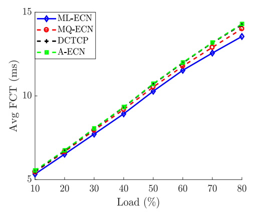
(f) DWRR:Mix



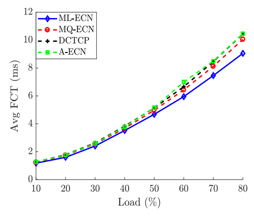
(g) DWRR:Web search



(h) DWRR:Data mining



(i) DWRR:Hadoop



(j) DWRR:Cache

Fig. 4: Overall FCT statistics across different workload

compare the performance of ML-ECN with MQ-ECN [19], MM-ECN [22], A-ECN [21], and the ECN marking scheme of DCTCP [10]. To ensure fair comparison between ML-ECN, MQ-ECN and other schemes, we use the exact code and the same settings as those in the MQ-ECN [19].

Topology: We conduct the performance evaluation based on 144-host Leaf-Spine topology. The network has 12 leaf (Top-of-Rack (ToR)) switches, and 12 spine (Core) switches. Each Leaf switch is connected to 12 hosts through 10Gbps downlink ports and 12 Spine switches through 10Gbps uplink ports, forming a non-blocking network. The end-to-end RTT latency across spines (4 hops) is $85.2\mu sec$. We employ the widely used ECMP [29] for network load balancing.

Transport: We use DCTCP as a transport protocol with initial window size set to 16 packets and both initial and minimum values of TCP RTO set to 5ms.

Switch: Current commodity switches support up to 8 queues per port, so we set 8 queues for each switch port and we map these queues into small, medium, and large categories. We utilize Weighted Round Robin (WRR) and Deficit Weighted Round Robin (DWRR) for scheduling flows among different queues in each switch port. We set the number of threshold levels on each queue to 4 and set the weight ratio of small, medium, and large queues to 4:2:1 respectively and the quantum for each queue to 1.5KB.

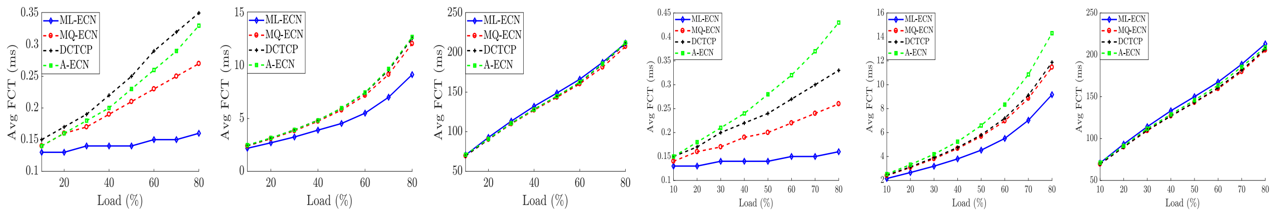
Workloads: In our previous work [30], we evaluate the performance of the proposed scheme using only one workload which is a mix of four different flow distributions (mix workload). In this paper, we extend the evaluation and run all the schemes under 4 different workloads, a web search [10], a data mining [23], a cache [7], and a Hadoop [7] with 914 KB, 1671 KB, 4149 KB, and 7495 KB of mean flow size, respectively. We use FCT, throughput, number of ECN marking, and packet drop rate as performance metrics in our simulation. We divide all flows based on their size into three classes including short flows $(0, 100KB]$, medium flows $(100KB, 10MB]$, and large flows $(10MB, \infty)$. We consider the results of overall flows and the break down across different flow sizes independently (short, medium, and long). All simulations last for 50K flows.

A. *Flow Completion Time*

FCT is the time from when the SYN packet of a flow is sent until when the last packet is received by the destination. Figure 4 shows the average FCTs (averaged over all flows) achieved under ML-ECN (proposed), MQ-ECN, A-ECN, MM-ECN, and DCTCP when running the schemes across different workloads. Observe that ML-ECN outperforms all other schemes for all workloads when using both WRR and DWRR packet schedulers. Running the mix workload in Figures 4a and 4f, we can see that the proposed ML-ECN reduces the overall FCT by 20% Compared to A-ECN , MM-ECN and DCTCP respectively. Moreover, ML-ECN outperforms MQ-ECN by up to 15%. For individual workloads, observe that the improvement of ML-ECN differs based on the type of workload. It is higher for Web search and cache workloads compared to Data mining and Hadoop workloads. The reason is that Web search and cache have smaller mean flow size compared to Data mining and Hadoop workloads, and ML-ECN is more beneficial for short flows as shown in Figures (5a ,5d). Therefore, the improvement is more significant for workloads with smaller mean flow size. Next, we present the breakdown of FCT across different flow sizes for each individual workload.

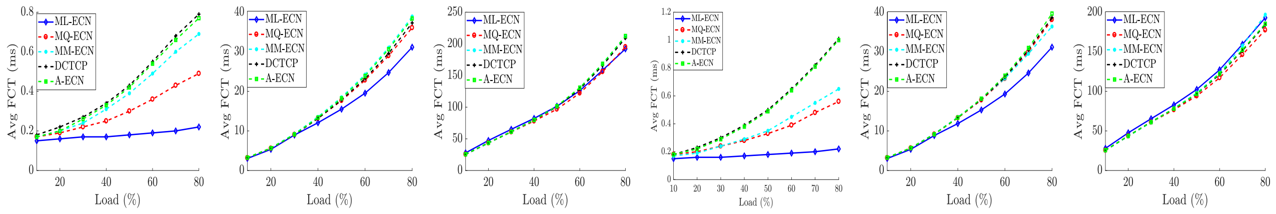
1) *Mix Workload:*

a) Short flows: In this section, we take a deep dive into small $(0, 100KB]$ flows and compare the average FCT of ML-ECN to other schemes. Compared to the ECN marking scheme of DCTCP shown in Figures 5a and 5d, ML-ECN has about 54% reduction in average FCT of small flows. DCTCP suffers in this scenario because, at high load with standard threshold,

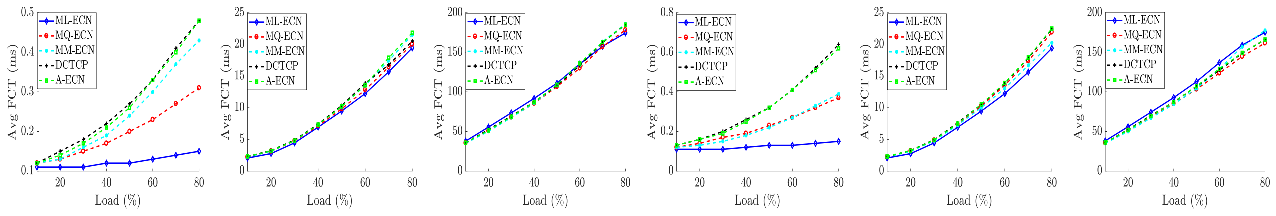


(a) WRR-Short (b) WRR-Medium (c) WRR-Long (d) DWRR-Short (e) DWRR-Medium (f) DWRR-Long

Fig. 5: FCT statistics across different flow sizes for mix workload

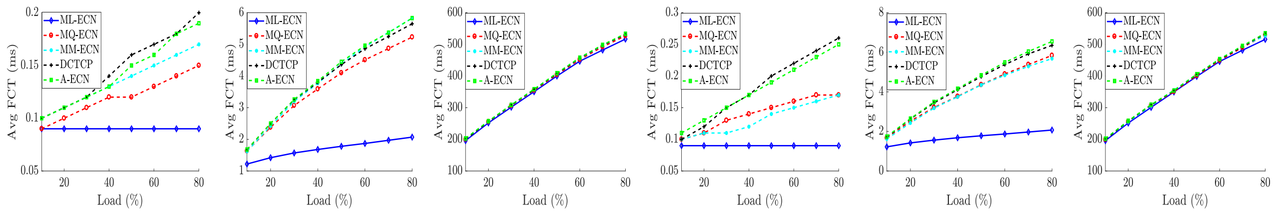


(a) WRR-Short (b) WRR-Medium (c) WRR-Long (d) DWRR-Short (e) DWRR-Medium (f) DWRR-Long

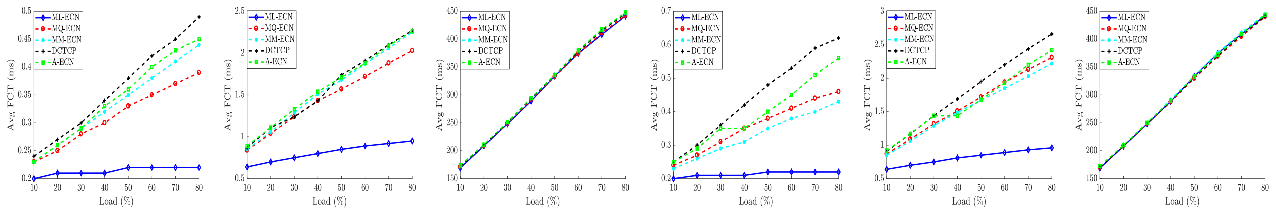


(g) WRR-Short (h) WRR-Medium (i) WRR-Long (j) DWRR-Short (k) DWRR-Medium (l) DWRR-Long

Fig. 6: FCT statistics across different flow sizes for Web search and Cache workloads



(a) WRR-Short (b) WRR-Medium (c) WRR-Long (d) DWRR-Short (e) DWRR-Medium (f) DWRR-Long



(g) WRR-Short (h) WRR-Medium (i) WRR-Long (j) DWRR-Short (k) DWRR-Medium (l) DWRR-Long

Fig. 7: FCT statistics across different flow sizes for Data mining and Hadoop workloads

multiple queues are concurrently active, causing congestion at the switch port, which in turn results in a high queueing delay for small flows. Compared to A-ECN, MM-ECN and MQ-ECN, ML-ECN reduces its average FCT by up to 51% , 51% and 39% respectively. We attribute the improvement of ML-ECN to its enqueueing and dequeueing mechanism that isolates small flows from other flows and assigns more weight to small service-queues.

b) Medium flows: For medium $(100KB, 10MB]$ flows, ML-ECN achieves notable reduction in FCT compared to all other schemes as shown in Figures 5b and 5e. Two factors contribute to the superior performance of ML-ECN. First, demoting large flows, as they grow in size, from medium service-queues to large service-queues and assigning medium service-queues with more weight compared to large service-queues. Second, employing multiple thresholds with increasing probability of ECN marking slows down the rapid increase of queue length in medium service-queues without throttling throughput for majority of the flows.

c) Large flows: Figures 5c and 5f show the average FCT of large $(10MB, \infty)$ flows when comparing ML-ECN to other schemes. Observe that all flows achieve similar performance. This shows the efficiency of ML-ECN in reducing the FCT for small and medium flows without jeopardizing the performance of large flows.

2) Web-Search and Cache Workloads:

a) Short flows: Figure 6 shows the FCT statistics across different flow sizes for Web search and Cache workloads. ML-ECN has a significant improvement in FCT for short flows exceeding 55% for web search compared to the other schemes as shown in Figures 6a and 6d. Similarly, ML-ECN performance exceeds 60% when running cache workload as shown in Figures 6g and 6j. We attribute the remarkable improvement of ML-ECN to its enqueueing, dequeueing and ECN-marking mechanisms as follows. First, ML-ECN dedicates a couple of queues to serve short flows only. Once a flow grows in size, it is migrated to medium service queues. This avoids buildup of small service-queues that causes long queueing delay. Second, ML-ECN assigns higher weight to small service-queues. Finally, instead of marking short flows when the total length of all queues reaches the standard threshold, ML-ECN marks small flows only if the length of their corresponding queues exceeds the standard threshold.

b) Medium flows: Figures 6a and 6d evaluate the improvement of ML-ECN for medium flows $(100KB, 10MB]$. Running Web search workload, ML-ECN reduces the average FCT by around 22%,22%,20%, and 18% compared to A-ECN, MM-ECN, DCTCP, and MQ-ECN respectively as shown in Figures 6b and 6e. On the other hand, Figures 6h and 6k show that ML-ECN achieves about 18%,18%,17%, and 15% reduction compared to A-ECN, MM-ECN, DCTCP, and MQ-ECN respectively when running the schemes under Cache workload.

c) Large flows: Similar to the Mix workload, ML-ECN achieves comparable performance for large flows compared to the other schemes. Figures 6f and 6l show that the performance of ML-ECN has dropped by around 7% compared to the other schemes.

3) Data-Mining and Hadoop Workloads:

a) Short flows: Figure 7 further confirms the superior performance of ML-ECN for short flows when running the schemes under both Data mining and Hadoop workloads. The performance gap of ML-ECN, as shown in Figures 7b and 7e, reaches 40% in WRR and 47% in DWRR compared to MQ-ECN for Data mining workload. Running Hadoop workload, Figures 7h and 7k show that the performance improvement reaches 44% in WRR and 52% in DWRR. Compared to A-ECN, MM-ECN, and DCTCP, the performance gap of ML-ECN exceeds 44% using WRR and 52% using DWRR for both workloads.

b) Medium flows: Figure 7 shows interesting results about the FCT of medium flow sizes when running Data mining and Hadoop workloads. Observe that the performance gain of ML-ECN becomes more significant compared to Web search and Cache workloads in Figure 6. The reason is that in Data mining and Hadoop, the majority of the bytes are from small percent of the flows. For example, over 95% of bytes in Data mining are from about 3.6% flows that are larger than 35MB [31]. Therefore, ML-ECN migrates these large flows from medium to large service-queues once they grow beyond a certain size. This results in a remarkable improvement in the average performance of medium flows.

c) Large flows: For large flows, Figure 7 shows that all schemes provide similar performance when running both Data mining and Hadoop workloads. This shows that ML-ECN is efficient and incurs no performance degradation for large flows.

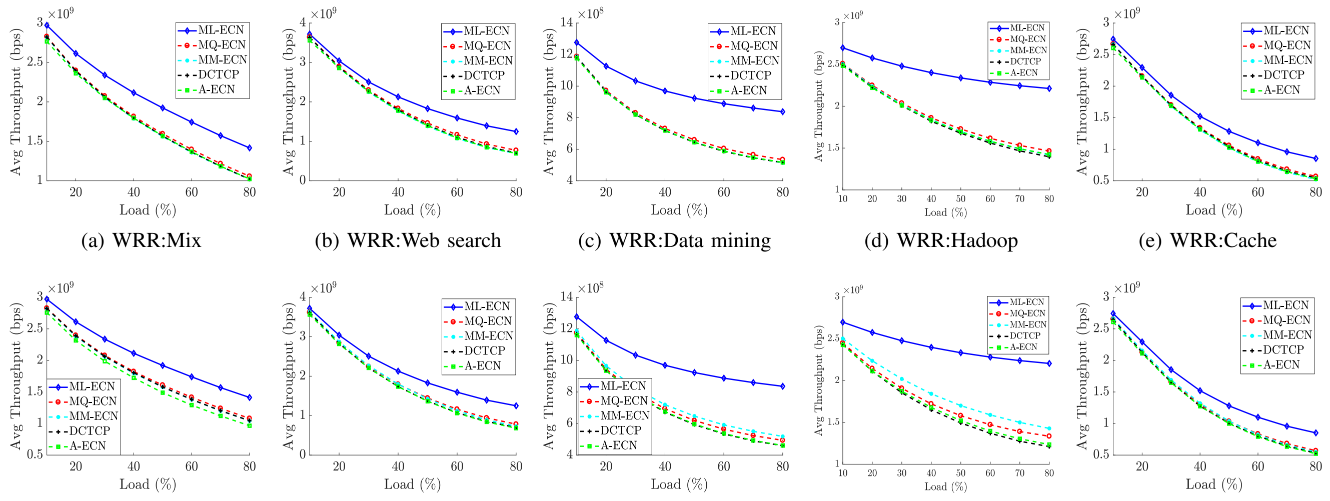
B. Throughput

The achievable average rate is calculated by first dividing each flow size by its measured FCT and then averaging the overall flows. Figure 8 shows the average throughput achieved under the studied schemes when running the schemes across different workloads. Observe that ML-ECN outperforms the other schemes in all the workloads for both schedulers. More specifically, the throughput performance of ML-ECN, compared to the other schemes, exceeds 24%, 38%, 33%, 36% and 34% when running Mix, Web search, Cache, Data mining, and Hadoop workloads respectively. Next, we show the breakdown of throughput across different flow sizes for each individual workload.

1) Mix Workload:

a) Short flows: Figures 9a and 9d show the average throughput for short flows. Observe that ML-ECN achieves about 35%, 33%, 37%, and 32% gain in throughput performance using the WRR scheduler compared to A-ECN, MM-ECN, DCTCP, and MQ-ECN. On the other hand, it increases the throughput performance by 45%, 34%, and 30% For the DWRR scheduler.

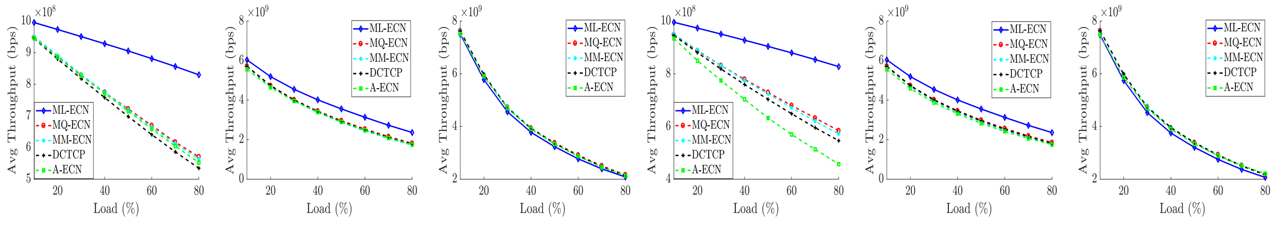
b) Medium flows: For medium flow sizes, the performance gain of ML-ECN exceeds 26%, , 25% 25%, and 21%



(a) WRR:Mix (b) WRR:Web search (c) WRR:Data mining (d) WRR:Hadoop (e) WRR:Cache

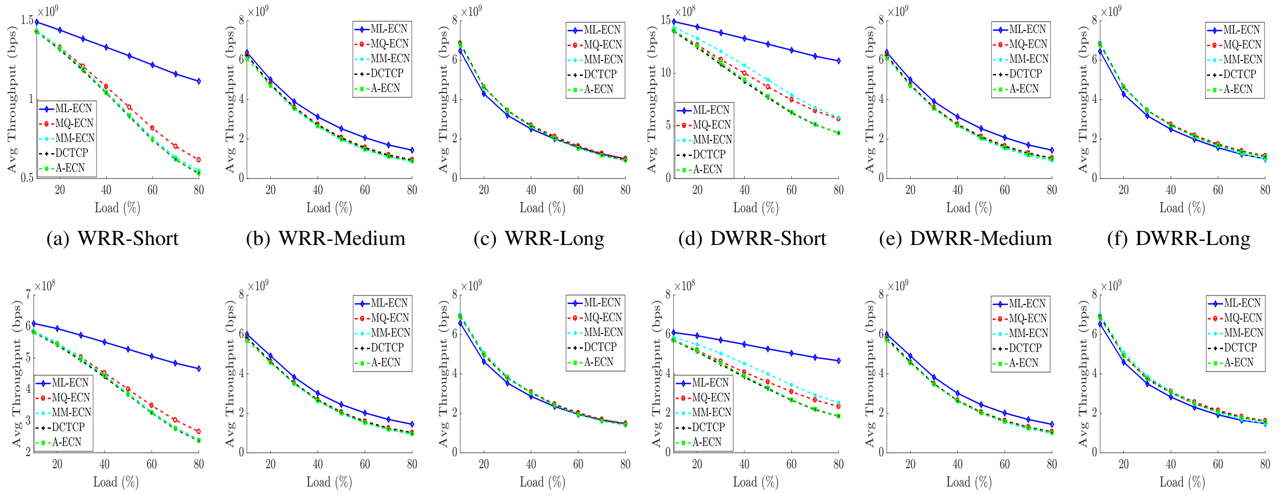
(f) DWRR:Mix (g) DWRR:Web search (h) DWRR:Data mining (i) DWRR:Hadoop (j) DWRR:Cache

Fig. 8: Overall throughput statistics across different workload



(a) WRR-Short (b) WRR-Medium (c) WRR-Long (d) DWRR-Short (e) DWRR-Medium (f) DWRR-Long

Fig. 9: Throughput statistics across different flow sizes for Mix workload



(a) WRR-Short (b) WRR-Medium (c) WRR-Long (d) DWRR-Short (e) DWRR-Medium (f) DWRR-Long

(g) WRR-Short (h) WRR-Medium (i) WRR-Long (j) DWRR-Short (k) DWRR-Medium (l) DWRR-Long

Fig. 10: Throughput statistics across different flow sizes for Web search and Cache Workloads

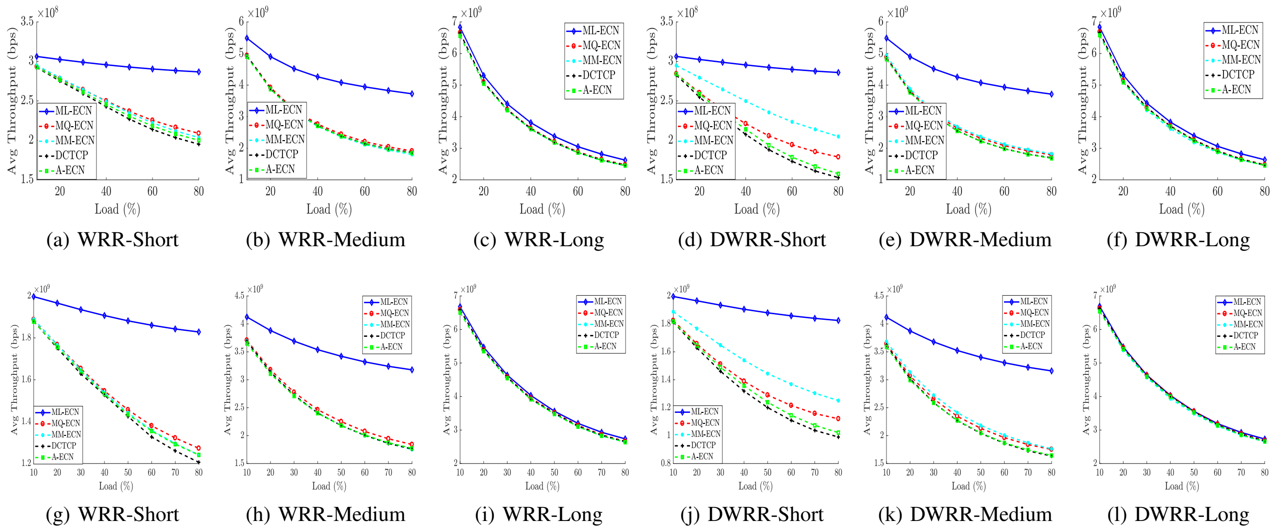


Fig. 11: Throughput statistics across different flow sizes for Data mining and Hadoop Workloads

compared to A-ECN, MM-ECN, DCTCP, and MQ-ECN as shown in Figures 9b and 9f.

c) Large flows: Figures 9c and 9f show that ML-ECN maintains comparable performance compared to the other schemes. This confirms the efficiency of ML-ECN in achieving better latencies for short flows while ensuring the throughput of large flows is not affected.

2) Web Search and Cache Workloads: Figure 10 shows the throughput statistics across different flow sizes for Web search and Cache Workloads. The results show similar performance improvement for ML-ECN over the other schemes as achieved in the Mix workload.

3) Data Mining and Hadoop Workloads: Figure 11 shows the throughput statistics across different flow sizes for Data mining and Hadoop Workloads. Observe that the throughput performance of ML-ECN compared to the other schemes has been further increased for medium flows when running the schemes under Data mining and Hadoop workloads. For large flows, we can see that ML-ECN slightly outperforms the other schemes in Data mining workload. More specifically, it achieves about 6% throughput gain compared to other schemes as shown in Figures 11c and 11f. The reason for the further increase in the performance of large flows is that ML-ECN decides not to mark packets, if the queue length is less than minimum threshold, in large queues even if the total length of all queues exceeds the standard threshold K.

C. ECN Marking

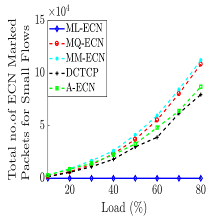
1) Mix Workload: Figure 12 shows the total number of ECN-marked packets across different flow sizes. Compared to other schemes, ML-ECN guarantees fairness for small flows by performing no ECN-marking on packets belonging to these flows as shown in Figures 12a and 12d. Although other schemes achieve lower marking for medium and larger flows, they incur more queueing delay, which results in higher FCT as shown in Figure 5.

2) Web-Search and Cache Workloads: Figure 13 shows the marking performance of the studied schemes when running under Web search and Cache workloads. Notice that ML-ECN has lower ECN marking for large flows in contrast to the other schemes. There are two reasons for that: First, the size of the large flows in Web search and Cache is not as big as the size of large flows in Data mining and Hadoop. Thus, there is low chance they co-exist in large queues for long period of time given that they start on small queues then get demoted to medium queues before they grow more in size and be placed on large queues. Second, ML-ECN insures good throughput for large flows by not marking their packets when the length of their corresponding queues is below the minimum threshold TL_1.

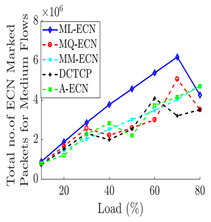
3) Data-Mining and Hadoop Workloads: Figure 14 shows the marking performance of the studied schemes when running under Data mining and Hadoop workloads. Observe that ML-ECN incurs no packet marking on short flows and more markings on medium and large flows. Compared to Web search and Cache workloads, ML-ECN marking performance on large flows is inferior to the other schemes. This is because the size of large flows in Data mining and Hadoop is significant, which results in more pressure on large queues and more marking.

D. Packet Drop Rate

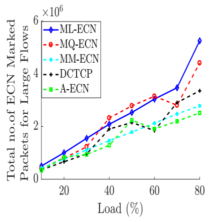
1) Mix Workload: Figures 15a and 15f show the average packet drop rate, due to queue overflow, achieved under different schemes in Mix workload. Observe that DCTCP has the highest packet drop rate among the schemes. The reason is that DCTCP applies the standard threshold in all queues, which makes it difficult to prevent queue buffers from being filled out and dropping packets. It is clear that ML-ECN provides significant reduction in packet drop rate compared to A-ECN and DCTCP across all loads. Compared to MQ-ECN and MM-ECN, observe that ML-ECN achieves slightly



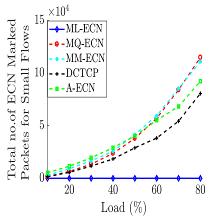
(a) WRR-Short



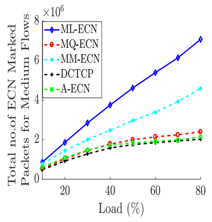
(b) WRR-Medium



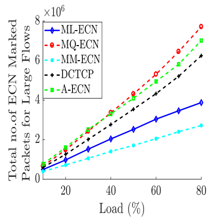
(c) WRR-Long



(d) DWRR-Short

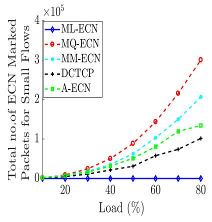


(e) DWRR-Medium

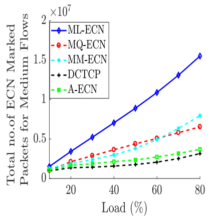


(f) DWRR-Long

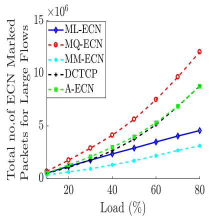
Fig. 12: ECN-marked packets across different flow sizes for mix workload



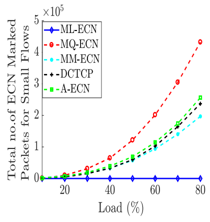
(a) WRR-Short



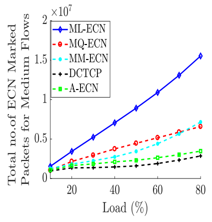
(b) WRR-Medium



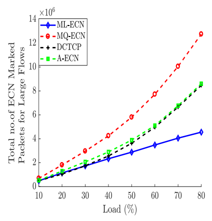
(c) WRR-Long



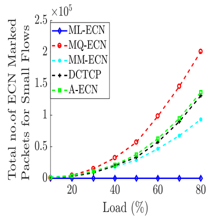
(d) DWRR-Short



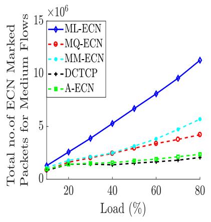
(e) DWRR-Medium



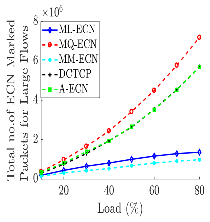
(f) DWRR-Long



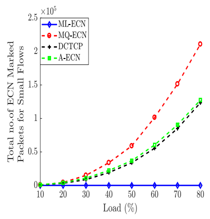
(g) WRR-Short



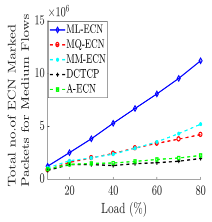
(h) WRR-Medium



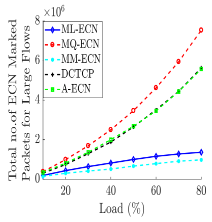
(i) WRR-Long



(j) DWRR-Short

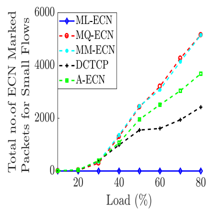


(k) DWRR-Medium

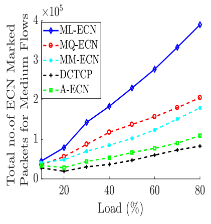


(l) DWRR-Long

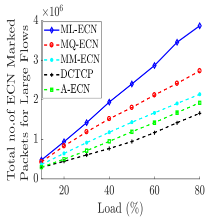
Fig. 13: ECN-marked packets across different flow sizes for Web search and cache Workloads



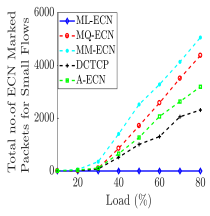
(a) WRR-Short



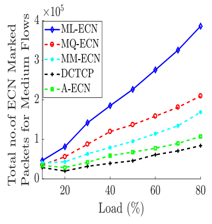
(b) WRR-Medium



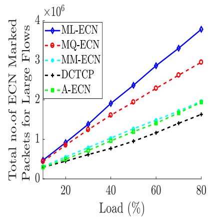
(c) WRR-Long



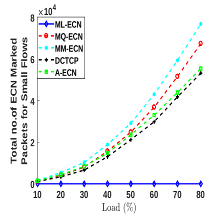
(d) DWRR-Short



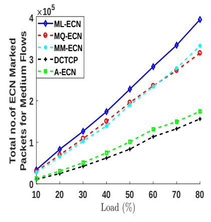
(e) DWRR-Medium



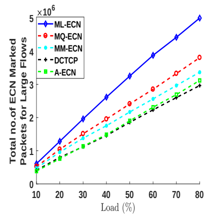
(f) DWRR-Long



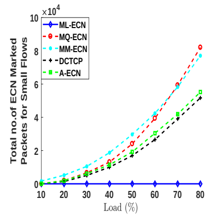
(g) WRR-Short



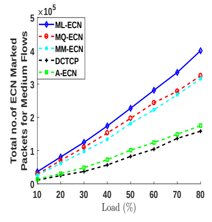
(h) WRR-Medium



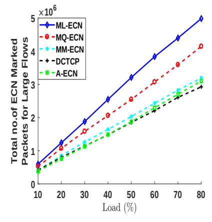
(i) WRR-Long



(j) DWRR-Short



(k) DWRR-Medium



(l) DWRR-Long

Fig. 14: ECN-marked packets across different flow sizes for Data mining and Hadoop Workloads

higher drop rate at 80%. This is because, at high load, MQ-ECN and MM-ECN sets the ECN-marking threshold at each queue to a lower value and starts marking all packets when the queue length exceeds this threshold. Although MM-ECN reduces the drop rate, it degrades the FCT of small flows as they share the same queues with large flows.

2) Web-Search and Cache Workloads: Figure 15 shows that ML-ECN maintains the superior performance compared to A-ECN, MM-ECN and DCTCP in Web search and Cache workloads. On the other hand, ML-ECN provides inferior performance compared to MQ-ECN. This is because MQ-ECN dynamically lowers the marking threshold in all queues, which marks all packets at the early stages of queue buildup while ML-ECN performs no early marking on packets in the small service queues to achieve low FCT for short flows.

3) Data-Mining and Hadoop Workloads: ML-ECN, MM-ECN, and MQ-ECN provide low packet drop rates for Data mining and Hadoop workloads. The reason is that in those workloads, the majority of bytes come from a very small number of large flows. Thus applying early ECN-marking on packets belonging to those flows reduces queue buildup and lowers the packet drop rate.

V. Related Work

Single-Queue ECN Marking: Tons of literature have been proposed to minimize the flow latency in datacenters [32–37]. These schemes are designed with the assumption that a switch port has only one queue and mainly consider how to perform ECN/RED marking for a single queue. For space limitation, we only introduce part of this literature. DCTCP [10] marks packets, during the enqueuing process, when the instantaneous queue length exceeds a static threshold. ECN* [38] chooses to mark packets based on a static threshold but during the dequeuing process. CEDM [39] goes further and decided to mark packets during both the enqueing and dequeuing process by using two static thresholds.

Multi-queue ECN Marking: The performance of single-queue schemes is drastically degraded in the presence of multi-queue switch ports, thereby prompting researchers to propose schemes for multi-service multi-queue switches. Multi-Service Multi-Queue ECN (MQ-ECN) [19] has been proposed to optimize the ECN marking scheme of DCTCP in multi-queue scenarios. MQ-ECN periodically adjusts the ECN marking threshold for each queue independently based on the queue weight and the round-trip time. However, imprecise measurement is considerable in MQ-ECN since data traffic in DCN is bursty in nature. Moreover, setting the threshold dynamically requires periodic round-trip time measurement which incurs non-negligible overhead for switches [19]. DemePro [20], on the other hand, decides to measure the total queue length in the port buffer instead of measuring the round-trip time of each queue periodically. DemePro marks the head packet in the most congested queue when the total queue length becomes greater than per-port threshold. A-ECN [21] designs an adaptive ECN marking scheme that does not consider time interval, as in MQ-ECN, to update the marking threshold. Instead, A-ECN uses the number of enqueued packets as the interval to update the marking threshold. TCN [40] decided to mark packets according to the sojourn times and a static threshold rather than the instantaneous length of the queue. DC-ECN [41] uses a machine learning-based classifier to separate mice and elephant flows through dual-coupled queues with independent ECN marking thresholds. Moreover, it dynamically tunes the ECN-marking threshold for the elephant queue to absorb micro-burst mice traffic, thereby lowering latency without impacting throughput. PMSB [42] uses a per-queue threshold to revoke or cancel the ECN marking of victim flows even if they qualify the per-port threshold. Although the schemes mentioned above can deliver low latency and high throughput, none of them consider fairness enqueueing and marking among flows.

Congestion Control: Much research has been devoted to congestion control for wide-area networks and data center networks. For instance, Cubic [43] achieves high scalability and proportional RTT-fairness by using the cubic time function to grow the congestion window. TCP Wave [44], based on proactive burst transmission, replaces traditional window-based transmission paradigms. BBR and BBRp [45, 46] are new congestion control mechanisms proposed to enhance the network performance in WANs. DCTCP [10], designed primarily for datacenters, leverages ECN to detect congestion and react in proportion to the measured congestion level.

VI. Conclusion

We propose ML-ECN for multi-service multi-queue DCNs to attain both high throughput and low latency simultaneously while maintaining fairness marking among flows. ML-ECN separates small, medium, and large flows into their desired queues with a different number of ECN thresholds. Afterward, instead of marking all flows when a queue length hits the lower threshold levels, ML-ECN only marks flows that contribute the most to the queue length increase. Simulation results show the superiority of ML-ECN in delivering lower FCT, maintaining high throughput, ensuring fairness marking among flows when compared to existing approaches.

References

[1] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang, “Understanding data center traffic characteristics,” *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 1, pp. 92–99, 2010.

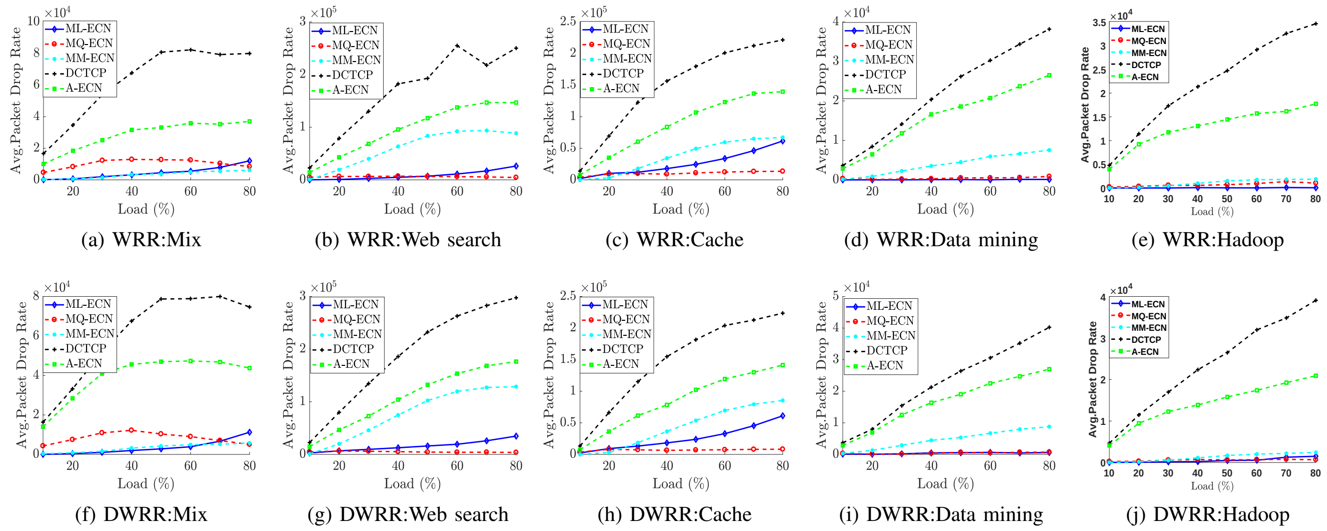
[2] Ting Wang, Zhiyang Su, Yu Xia, and Mounir Hamdi, “Rethinking the data center networking: Architecture, network protocols, and resource sharing,” *IEEE access*, vol. 2, pp. 1481–1496, 2014.

[3] Jarallah Alqahtani, Sultan Alanazi, and Bechir Hamdaoui, “Traffic behavior in cloud data centers: A survey,” in *2020 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2020, pp. 2106–2111.

[4] Sultan Alanazi and Bechir Hamdaoui, “Caft: Congestion-aware fault-tolerant load balancing for three-tier clos data centers,” in *2020 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, 2020, pp. 1746–1751.

[5] Jarallah Alqahtani, Hassan H Sinky, and Bechir Hamdaoui, “Clustered multicast source routing for large-scale cloud data centers,” *IEEE Access*, vol. 9, pp. 12693–12705, 2021.

[6] Jeffrey Dean and Sanjay Ghemawat, “Mapreduce: simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.



(a) WRR:Mix (b) WRR:Web search (c) WRR:Cache (d) WRR:Data mining (e) WRR:Hadoop

(f) DWRR:Mix (g) DWRR:Web search (h) DWRR:Cache (i) DWRR:Data mining (j) DWRR:Hadoop

Fig. 15: Packet drop rate comparison across different workload

[7] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C Snoeren, "Inside the social network's (datacenter) network," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 123–137.

[8] Ting Wang, Lu Wang, and Mounir Hamdi, "A cost-effective low-latency overlaid torus-based data center network architecture," *Computer Communications*, vol. 129, pp. 89–100, 2018.

[9] Keon Jang, Justine Sherry, Hitesh Ballani, and Toby Moncaster, "Silo: Predictable message latency in the cloud," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 435–448.

[10] Mohammad Alizadeh, Albert Greenberg, David A Maltz, Jitendra Padhye, Parveen Patel, Balaji Prabhakar, Sudipta Sengupta, and Murari Sridharan, "Data center tcp (dctcp)," in *Proceedings of the ACM SIGCOMM 2010 Conference*, 2010, pp. 63–74.

[11] Mohammad Alizadeh, Abdul Kabbani, Tom Edsall, Balaji Prabhakar, Amin Vahdat, and Masato Yasuda, "Less is more: Trading a little bandwidth for ultra-low latency in the data center," in *9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, 2012, pp. 253–266.

[12] Balajee Vamanan, Jahangir Hasan, and TN Vijaykumar, "Deadline-aware datacenter tcp (d2tcp)," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 115–126, 2012.

[13] Haitao Wu, Jiabo Ju, Guohan Lu, Chuanxiong Guo, Yongqiang Xiong, and Yongguang Zhang, "Tuning ecn for data center networks," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, 2012, pp. 25–36.

[14] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang, "Congestion control for large-scale rdma deployments," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 523–536, 2015.

[15] Kadangode Ramakrishnan, Sally Floyd, David Black, et al., "The addition of explicit congestion notification (ecn) to ip," 2001.

[16] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker, "pfabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 435–446, 2013.

[17] Wei Bai, Li Chen, Kai Chen, Dongsu Han, Chen Tian, and Hao Wang, "Information-agnostic flow scheduling for commodity data centers," in *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, 2015, pp. 455–468.

[18] Ali Munir, Ghufran Baig, Syed M Irteza, Ihsan A Qazi, Alex X Liu, and Fahad R Dogar, "Friends, not foes: synthesizing existing transport strategies for data center networks," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 491–502.

[19] Wei Bai, Li Chen, Kai Chen, and Haitao Wu, "Enabling {ECN} in multi-service multi-queue data centers," in *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, 2016, pp. 537–549.

[20] Chengxi Gao, Victor CS Lee, and Keqin Li, "Demepro: Decouple packet marking from enqueuing for multiple services with proactive congestion control," *IEEE Transactions on Cloud Computing*, 2017.

[21] Shuo Wang, Jiao Zhang, Tao Huang, Tian Pan, Jiang Liu, and Yunjie Liu, "A-ecn minimizing queue length for datacenter networks," *IEEE Access*, vol. 8, pp. 49100–49111, 2020.

[22] Yifei Lu, Xu Ma, and Zhengzhi Xu, "Choose a correct marking position: Ecn should be freed from tail mark," *Computer Networks*, vol. 197, pp. 108329, 2021.

[23] Albert Greenberg, James R Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A Maltz, Parveen Patel, and Sudipta Sengupta, "Vl2: A scalable and flexible data center network," in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, 2009, pp. 51–62.

[24] Yawen Pan, Chen Tian, Jiaqi Zheng, Gong Zhang, Hengky Susanto, Bo Bai, and Guihai Chen, "Support ecn in multi-queue datacenter networks via per-port marking with selective blindness," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 33–42.

[25] Mohammad Alizadeh, Adel Javanmard, and Balaji Prabhakar, "Analysis of dctcp: stability, convergence, and fairness," *ACM SIGMETRICS Performance Evaluation Review*, vol. 39, no. 1, pp. 73–84, 2011.

[26] "Microsoft sql server," https://www.microsoft.com/en-us/sql-server, Accessed: 2021-04-30.

[27] "Openstack object storage," https://docs.openstack.org/api-ref/object-store/index.html, Accessed: 2021-04-30.

[28] Wei Bai, Li Chen, Kai Chen, Dongsu Han, Chen Tian, and Hao Wang, "Pias: Practical information-agnostic flow scheduling for commodity data centers," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 1954–1967, 2017.

[29] C Hopps, "Rfc2992: analysis of an equal-cost multi-path algorithm," 2000.

[30] Sultan Alanazi and Bechir Hamdaoui, "Ml-ecn: Multi-level ecn marking for fair datacenter traffic forwarding," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 2726–2731.

[31] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, et al., "Conga: Distributed congestion-aware load balancing for datacenters," in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 503–514.

[32] Li Chen, Shuihai Hu, Kai Chen, Haitao Wu, and Danny HK Tsang, "Towards minimal-delay deadline-driven data center tcp," in *Proceed-*

ings of the Twelfth ACM Workshop on Hot Topics in Networks, 2013, pp. 1–7.

[33] Sally Floyd and Van Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on networking*, vol. 1, no. 4, pp. 397–413, 1993.

[34] Kadangode Ramakrishnan, Sally Floyd, David Black, et al., "The addition of explicit congestion notification (ecn) to ip," 2001.

[35] Balajee Vamanan, Jahangir Hasan, and TN Vijaykumar, "Deadline-aware datacenter tcp (d2tcp)," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 115–126, 2012.

[36] Haitao Wu, Jiabo Ju, Guohan Lu, Chuanxiong Guo, Yongqiang Xiong, and Yongguang Zhang, "Tuning ecn for data center networks," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, 2012, pp. 25–36.

[37] Yibo Zhu, Haggai Eran, Daniel Firestone, Chuanxiong Guo, Marina Lipshteyn, Yehonatan Liron, Jitendra Padhye, Shachar Raindel, Mohamad Haj Yahia, and Ming Zhang, "Congestion control for large-scale rdma deployments," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 523–536, 2015.

[38] Haitao Wu, Jiabo Ju, Guohan Lu, Chuanxiong Guo, Yongqiang Xiong, and Yongguang Zhang, "Tuning ecn for data center networks," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, 2012, pp. 25–36.

[39] Danfeng Shan and Fengyuan Ren, "Ecn marking with micro-burst traffic: Problem, analysis, and improvement," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1533–1546, 2018.

[40] Wei Bai, Kai Chen, Li Chen, Changhoon Kim, and Haitao Wu, "Enabling ecn over generic packet scheduling," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, 2016, pp. 191–204.

[41] Akbar Majidi, Nazila Jahanbakhsh, Xiaofeng Gao, Jiaqi Zheng, and Guihai Chen, "Dc-ecn: A machine-learning based dynamic threshold control scheme for ecn marking in dcn," *Computer Communications*, vol. 150, pp. 334–345, 2020.

[42] Yawen Pan, Chen Tian, Jiaqi Zheng, Gong Zhang, Hengky Susanto, Bo Bai, and Guihai Chen, "Support ecn in multi-queue datacenter networks via per-port marking with selective blindness," in *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2018, pp. 33–42.

[43] Sangtae Ha, Injong Rhee, and Lisong Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[44] A Abdelsalam, M Luglio, C Roseti, and F Zampognaro, "Tcp wave resilience to link changes," in *SCITEPRESS-Science and Technology Publications, Lda*, 2016, pp. 72–79.

[45] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson, "Bbr: congestion-based congestion control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[46] Carlo Augusto Grazia, Natale Patriciello, Martin Klapez, and Maurizio Casoni, "Bbr+: improving tcp bbr performance over wlan," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.