EC^4: ECN and Credit-Reservation Converged Congestion Control

Zihao Wei*, Dezun Dong*†, Shan Huang and Liquan Xiao
College of Computer, National University of Defense Technology
{weizihao17,dong,huangshang12,xiaoliquan}@nudt.edu.cn

***Abstract*—Bursty traffic and thousands of concurrent flows incur inevitable congestion in data center networks (DCNs) and then affect the overall performance. Various transport protocols are developed to mitigate the network congestion, including reactive and proactive protocols. Reactive schemes to handling congestion after congestion arises are common to current DCNs. However, with the growth of scale and link speed, reactive schemes such as DCTCP encounter the significant problem of slow responding to congestion. On the contrary, proactive protocols are designed to avoid congestion, and they have the advantages of zero data loss, fast convergence and low buffer occupancy (e.g., credit-reservation protocols). But in actual deployment scenario, it is hard to guarantee one protocol to be deployed in every server at one time. When credit-reservation protocol is deployed to DCNs step-by-step, the network is converted to multi-protocol state and faces the following fundamental challenges: (i) unfairness, (ii) high buffer occupancy, and (iii) heavy tail delay. Therefore, we propose** EC^4**, which is for converging ECN-based and credit-reservation protocols with minimal modification. To the best of our knowledge,** EC^4 **is the first to address how to harmonize proactive and reactive congestion control. Targeting the common ECN-based protocol–DCTCP,** EC^4 **leverages the Forward Explicit Congestion Notification (FECN) to deliver real-time congestion information and redefines feedback control. After evaluation, the results show that** EC^4 **effectively addresses the unfair link allocation. Furthermore, even workloads at 0.6 does not cause buffer overflow, thus largely eliminating the timeouts problem.**

***Index Terms*—Congestion Control, Data Center Network (DCN), Explicit Congestion Notification (ECN), Credit Reservation, Multi-protocol Network.**

I. INTRODUCTION

Data centers provide high-quality services to global users and have attained great achievements [1].With the increase of users, intra-datacenter traffic is continually becoming heavier and small latency-sensitive messages take up a large part of it [2]. As the response time of the applications greatly determines the user experience, the data center networks (DCNs) are pressured to be more efficient. However, the current data centers with shallow buffered switches cannot meet the requirement of ultra-low latency and high bandwidth [3]. The data queueing and buffer overflow, which will cause high queueing delay and retransmission delay, are ubiquitous when congestion arises [4]. Therefore, an efficient congestion control is critical.

A large body of works are proposed to address this challenge, and they can be classified into reactive congestion control and proactive schemes [5]. Reactive congestion control uses Backward Explicit Congestion Notification (BECN) [6], [7], Forward Explicit Congestion Notification (FECN) [8], [9] or Round-Trip Time (RTT) [10] as the feedback signal to manage congestion. However, reactive schemes have problems in current high-speed networks, they are hard to timely respond to network congestion. Proactive congestion control manages the sources to inject packets into the network within the network capacity before congestion arises, and it has a significant advantage of fast convergence that helps it gain high applicability in high-speed environment [11].

We argue that hop-by-hop credit-reservation is one of most promising technology of proactive congestion control. After comparing with other technologies, we find the following superiorities of hop-by-hop credit-reservation congestion control over other proactive schemes.

- Unlike centralized approached (e.g., Fastpass [12] and Flowtune [13]), credit-reservation protocols can be implemented without modifications on the commercial switch chips.
- Without relying on the Link-layer Flow Control (LFC), it can achieve near zero congestion packet loss as well. Hence, the unfairness, Head of Line (HoL) blocking, and deadlock can be avoided [14], [15].
- Hop-by-hop credit-reservation protocols has the highlight of fast convergence, low overhead, and high utilization. Using testbed experiments and simulations, ExpressPass [16], the outstanding representative of credit-reservation protocols even converges 80 times faster than DCTCP [8] (the most common reactive protocol in the DCNs) in 10Gbps links, and this gap will increase as the link speed increases.

Due to its attractive features, credit-reservation protocols, especially ExpressPass is held in highly recognized by academia [17], [18]. However, ExpressPass is not deployed in the real data centers, and current DCNs (e.g., Google, Microsoft, and Amazon) still mainly deploy the FECN-based protocol (e.g., DCTCP and DCQCN [9]). Since ExpressPass must be incrementally deployed to the DCNs to save costs [19], [20], many fundamental challenges to the fairness of bandwidth allocation will be brought. We show in §*II* that simply mixing ExpressPass with the current existing protocols brings serious trouble for the data centers. To solve these

* These authors contributed equally to this work.
† Dezun Dong is the corresponding author.

practical problems, we propose EC^4 instead of ExpressPass, which can perfectly converge with DCTCP. As far as we know, EC^4 is the first effort for incremental deployment of proactive congestion control for data centers.

Due to the different ways of propagating congestion information, it is challenging for the incremental deployment of ExpressPass in the DCTCP environment. ExpressPass picks up congestion information from the credit queue, while DCTCP responds to the congestion according to whether the length of data queue exceeds the threshold. As Fig. 1 shows, there is obvious physical isolation between the data queue and the credit queue. Since DCTCP traffic makes no difference on the credit queue, ExpressPass traffic will always be transmitted in the data channel at full throttle. As a result, the DCTCP traffic will be constantly throttled by the sender with the increase of data queue, until its bandwidth occupancy is close to zero. We make some multi-protocol experiments to testify the aggressiveness of the ExpressPass when coexists with the popular reactive protocols, and the results show ExpressPass is unfriendly to not only DCTCP but to the other reactive schemes such as CUBIC (drop-based TCP) and TIMELY [10] (delay-based). To address this problem, therefore, isolation errors must be eliminated.

Faced these challenges, this paper proposes EC^4, aimed for realization of the symbiosis between the credit reservation congestion control and the ECN-based protocol–DCTCP. The key points of EC^4 are as follows.

- The ECN-marking mechanism is implemented at the switch data queue to process both the EC^4 and the DCTCP traffic so as to break up the isolation between the credit-reservation protocol's credit limiter and the data queue.
- In order to make a trade-off between the throughput and latency in a multi-protocol network, we adjust the threshold of random early detection (RED) ECN marking mechanism at data queue.
- An ECN-based feedback control mechanism is developed to replace the old mechanism of ExpressPass so that EC^4 can leverage the congestion information which reflects the real-time data queue status.

Our ECN-based feedback control detects congestion in the first few RTTs and responds in a timely manner without affecting the performance of traditional protocol stack. EC^4 guarantees fairness between different protocols, cross-protocol convergence reaches to the millisecond level, and average buffer occupancy is reduced to less than 100 KB. Moreover, when there is few traffic based on other congestion control in the network, the transmission can be completed at a high convergence speed without introducing additional overhead. Simulating the partition/aggregation mode common to the current DCNs shows that EC^4 not only ensures that the average Flow Completion Time (FCT) is unaffected in the multi-protocol network but also greatly reduces the tail delay. EC^4 guarantees the quality of service (QoS) of all applications and bring great economic benefits.

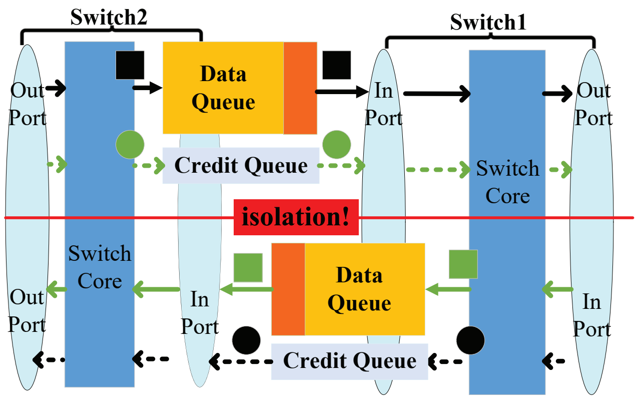


Fig. 1. Reasons for Congestion Information Isolation on the Link: ● represents credit packets and ■ represents data packets. The same color packets reflect the congestion in the same direction (the credit reflects the congestion of the reverse link), i.e., for ECN-based and credit-reservation protocols, congestion information is delivered based on different channels.

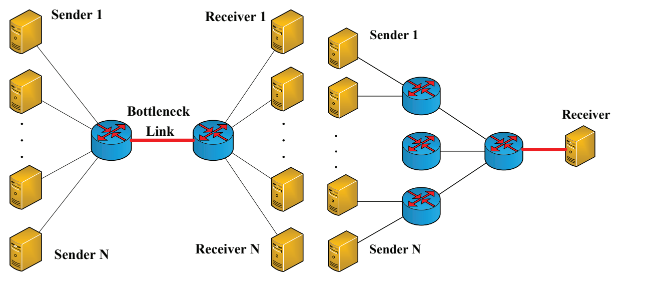
II. MOTIVATION

DCTCP plays a significant role in the last decade, as it achieves high throughput while guaranteeing low buffer occupancy in 10 Gbps DCNs [21]. Until now, many data centers still deploy DCTCP to prevent a meltdown of the network under heavy and bursty traffic. Arjun et al. from Google proclaim that they refer to DCTCP and enable ECN on switches and modify the host stack based on ECN signals [3]. Moreover, ref. [22] regard DCTCP as one of the most promising technique for deployment in DCNs.

However, due to the shallow buffered commercial switches and the increase in link speed (from 10 Gbps to 100 Gbps), the buffer size provided for per Gbps link speed is decreasing [23]. DCTCP result in high bandwidth occupancy and unfairness due to the slow response time of reactive schemes. This makes DCTCP be in straits. Moreover, the experimental result shows that when there is a large number of concurrent flows, DCTCP cannot efficiently deal with the incast problem and the instantaneous queue length is much higher than the maximum queue capacity [24].

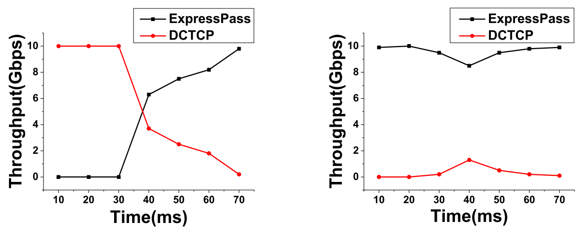
However, credit-reservation protocols (e.g., ExpressPass) is characterized by the merits of fast convergence, bounded-queue, and high throughput, which give it strong applicability in high-speed DCNs. To make the context of our paper more coherent, here we briefly introduce ExpressPass which is reported in the conference of SIGCOMM'17. ExpressPass is a credit-reservation decentralized hop-by-hop proactive congestion control. Before sending packets, the source delivers a credit request to inform receiver that there are packets waiting to be transmitted. When receives the request, the receiver begin to send credit packets to the sender. Combining the rate limiter, the credit packets are guaranteed not to exceed 5% of the bandwidth per hop. As Fig. 1 says, credit packets which is throttled to 5% bandwidth[1] ensure the flow

[1] ExpressPass specifies that a credit packet (84B) corresponds to a maximum transmission unit (1538B), 84/(84+1538) $\approx$ 5%.



(a) Dumbell topology: link frequently becomes the bottleneck resource. (b) Incast topology: receiver frequently becomes the bottleneck resource.

Fig. 2. When the red link becomes the bottleneck resource, how to allocate the bottleneck resource becomes a key issue.



(a) ExpressPass insert network. (b) DCTCP insert network.

Fig. 3. Problem of Multi-protocol in DCNs. ExpressPass occupy most bandwidth, but DCTCP host can send packets after ExpressPass completes its transmission.

is transmitted within the link capacity. ExpressPass requires only a simple configuration at the switch and achieves lossless transmission at a low bandwidth cost (5%). However, it has not been deployed in DCNs. To be incrementally deployed in actual DCNs, ExpressPass has to be improved to coexist with DCTCP that has already been commonly deployed.

Fig. 2 shows two simple situations. In both Fig. 2a and Fig. 2b, we assume Flow 1 to N are scheduled by different protocols, including ExpressPass and DCTCP. In these topologies, when the senders simultaneously send data to the receivers, the red link becomes the bottleneck resource. Then we raise a question: *Can bottleneck resource be allocated in a fair way?*

A multi-protocol network overview of ExpressPass' incremental deployment to the DCN is shown in Fig. 1. The credit queue for ExpressPass is isolated from the data queue for DCTCP. In theory, the transmitting rate of the ExpressPass flow will not be decreased as the sender does not receive any real-time congestion information from the data queue. But the DCTCP flow will be throttled, as the congestion information would be delivered to the sender by the ECN mechanism. Consequently, the ExpressPass flow preempts all resources of the bottleneck link and the DCTCP flow can only wait.

To validate our analysis, we design two experiments based on the topology shown in Fig. 2a. We use OMNeT++ simulator [2] to perform our simulation.

In the first experiment, ExpressPass traffic is inserted into the running DCTCP flows at time t. In the second experiment, DCTCP traffic is inserted into the running ExpressPass flows at time t. We set the flow size large enough so that they can be continuously transmitted, besides, the link speed is set to 10Gbps and the propagation delay is set to 5 μs. The results show that ExpressPass quickly occupies bottleneck bandwidth in both of these experiments, while the other flows can only wait. Given that, ExpressPass must be improved to coexist with DCTCP, but it is a challenging job.

[2] OMNeT++ (Objective Modular Network Testbed in C++) is a modular,component-based C++ simulation library and framework. OMNeT++ itself is a simulation framework without models for network protocols. The main computer network simulation models are available in several external frameworks. Our experiment used one is INETwhich offers a variety of models for all kind of network protocols and technologies like for IPv6, BGP, etc. Many works use it for simulation.

III. EC^4 Design

EC^4 makes a great success by choosing an appropriate congestion signal for proactive schemes, developing an ingenious feedback control scheme, and deciding a reasonable network configuration.

We choose the FECN to deliver congestion information. Furthermore, EC^4 modifies the switches and hosts based on functionalities supported by commodity data centers. After balancing a trade-off between efficiency and accuracy, we abandon other options and inherit the ExpressPass' separate queue on the switch to transmit credits. Besides, we use a single-threshold marking scheme at switches to mark the packets with ECN label as soon as the buffer queue exceeds a fixed threshold. The receiver reacts by reducing the credit by two factors, one depending on the fraction of marked packets and the other depending on the credit loss rate[3].

We must note that feedback control law itself is not the key contribution. It is the behavior of *combining multi congestion information provided by congestion experienced (CE) codepoint (data queue) and credit loss rate (credit queue).* Information that other feedback control laws obtain is incomplete (reactive congestion control obtains real-time congestion information, but proactive schemes get the theoretical link capacity). However ECN-based feedback control makes full use of both kinds of information to achieve great success.

A. Switch and End-Host Design

(I)Link Bandwidth Allocation of Switch: Compared with ExpressPass, there is no difference in the design of the credit packet size and the separate queue on the switch. The credit packet of the ~84 B Ethernet frame (minimum size) triggers the sender to transmit up to a ~1538 B size Ethernet frame (maximum size). So 5% of the link capacity is for credit rate-limiting and the remaining 95% of the link capacity is allocated to packet forwarding. Although the credit channel is underutilized in a multi-protocol network, the design of the switch and host NIC remains unchanged. We also considered to dynamically adjust the size of the separate queue, but the following reasons shelve this idea: (i) The traffic of DCNs is changing rapidly. Of course, we

[3] The ExpressPass has provided an interface to calculate the credit loss rate by carrying the sequence number in the credit packets.

Algorithm 1: ECN-based Feedback Control at Receiver

```
$ECN\_\alpha \leftarrow 0$, $\omega \leftarrow \omega_{init}$, $cur\_rate \leftarrow initial\_rate$, $MODE\_FLAG \leftarrow 0$;
while not at end of this flow do
    if $ECN\_ratio \leqslant target\_ECN\_ratio$ then
        ▹ (increasing phase);
        if $MODE\_FLAG \leqslant mode\_threshold$ then
            ▹ (LOW mode);
            $ECN\_\alpha = (1-g) * ECN\_\alpha + g * ECN\_ratio$;
            $tmp\_rate = cur\_rate * (1 + (target\_ECN\_ratio + ECN\_\alpha)/2)$;
            $MODE\_FLAG++$;
        else
            ▹ (HIGH MODE);
            Use credit-based feedback control of ExpressPass;
            return;
    else
        ▹ (decreasing phase);
        $ECN\_\alpha = (1-g) * ECN\_\alpha + g * ECN\_ratio$;
        $tmp\_rate = cur\_rate * (1 - (target\_ECN\_ratio + ECN\_\alpha)/2)$;
        $MODE\_FLAG \leftarrow 0$;
    if $credit\_Loss \leqslant target\_loss$ then
        ▹ (increasing phase);
        $\omega = (\omega + \omega_{mid})/2$;
        $cur\_rate = (1-\omega) * tmp\_rate + \omega * max\_rate$;
    else
        ▹ (decreasing phase);
        $cur\_rate = tmp\_rate * (1 - credit\_loss\_rate)$;
        $\omega = max(\omega_{min}, \omega/2)$;
```

can introduce deep reinforcement learning like AuTO [25] to analyze protocol composition for a period of time, but high overhead can only bring limited benefits (little bandwidth for packets forwarding), and this departure from the principle of minimal modification costs. (ii) Although EC^4 is designed for optimization during the incremental deployments, remaining in the multi-protocol state long-term is unexpected for network operators. Dynamic separate queues will be meaningless as the deployment of EC^4 grows.

(II) Marking at Switch: We refer to the RED scheme which is first proposed in ref. [26] when setting the threshold of commercial switches: we use the instantaneous queue length as the congestion metric to mark packets. If the queue length exceeds threshold K, it is immediately marked with the CE codepoint by the switch as soon as the packet arrives; otherwise, it is not marked. Doing so allows the receiver to quickly detect congestion on the switch.

(III) Controller at Receiver: There is a significant d-ifference between credit-based congestion control and other congestion control where the reaction location of credit-based congestion control is at the receiver. Controller at receiver gives us great convenience because we can accurately confirm whether a packet is marked with the CE codepoint. Compared with using ACKs to inform the sender to cut the congestion window (CWnd), reducing the credit sending rate on the receiver reduces the load on the link.

B. ECN-based Feedback Control

Employing ECN is not a rare occurrence in reactive schemes. However, to the best of our knowledge, we are the first to apply ECN to proactive congestion control. ECN-based feedback control focuses on two questions: (i) How to quickly detect the congestion. (ii) How to deal with the relationship between two kinds of congestion information we obtain.

First, we need to protect the fine performance of credit-reservation protocols. When there is little or no other traffic in the network environment, the credits can be sent at a high transmission rate to achieve high convergence. For this, we designed the **HIGH** mode. With the initial few RTTs, EC^4 still enables traffic tend to send packets at the link capacity, so few other traffic can cause queuing. Therefore, after several RTTs, if no packets marked with CE codepoint are received, receiver controller can consider that there is few other traffic in the network, enter the HIGH mode, and take the more aggressive approach.

We make a trade-off between easy packet loss and convergence here. When hosts send credits at a high speed as in Algorithm 1 before the network environment is determined, it is easy to cause packet loss. We can design the phase to send packets using slow start technology, which sacrifices the convergence of ExpressPass. The reasons we send credit at a high rate are as follows.

(i) Interactive, soft real-time workloads, such as the ones seen in search engines, social networking, and retail generate a large number of small requests and responses across the data centers that are then stitched together to perform a user-requested computation [27]. For these small flows, transmitting only one RTT is enough to transmit, and if started at a low speed, this may cause wasted bandwidth. (ii) By starting at a high rate we can quickly detect if there is other traffic in the network. (iii) The high convergence of credit-reservation protocols is guaranteed when there is little other traffic in the DCNs.

Moreover, when there is other traffic in the network, some packets will be quickly marked and the **LOW** mode will be triggered. The ECN_ratio recursively converges to the *target_ECN_ratio* by the increasing and decreasing phase.

Finally, we are also willing to make the best of the information of theoretical link capacity provided by separate credit queue. We design a variant version of the credit-based increase and decrease phase. ω floats between the smaller ω_{min} and ω_{mid}, and can achieve a fast decrease and slow self-increase, which is suitable for multi-protocol networks.

Based on this, we have designed an ECN-based feedback control. The specific algorithm is shown in Algorithm 1. Through this algorithm, when EC^4 is deployed incrementally in DCNs, it allocates bottleneck link bandwidth in a fair way so that it does not become "rogue" and disrupts other traffic. When EC^4 is deployed in a large number of DCNs, it can guarantee the advantages of high convergence and bounded-queue of credit-reservation protocols through the HIGH mode.

C. Parameter Choice

First of all, we choose the appropriate *mode_threshold* to make a trade-off between accuracy and convergence. However, it has been found through experiments that credit-based traffic is initially injected at the maximum link capacity. If there is other traffic in the network, the buffer will soon exceed the threshold. Therefore, if in four or fewer RTTs controller do not receive the tagged packets, we can consider that there is no other traffic in the network. In our experiments, the *mode_threshold* is chosen to be 3.

Second, to keep pace with the other ECN-based protocols, K and g should be chosen to synchronize with the target threshold. DCTCP [8] has proved that if we use C to indicate the link capacity, then the threshold values K and the parameter g should be guaranteed as shown in equations (1) and (2). However, one must make allowances for bursts in practice when selecting the value of K. For the 10 Gbps Ethernet, k = 20 packets can satisfy the equation(1), but we know that there are many factors for bursty traffic (e.g., architecture details, unbalanced load, and interrupt moderation). In addition, EC^4 is particularly prone to porn packets into the data centers at link capacity. We observe an excess of 10Gbps at data mining [28] workload at load 0.6 and find it to be worse than we thought in a multi-protocol environment. We notice that some transients reach more than 100 packets. We set K to 100 MTUs (146.5KB) to balance throughput and latency. g is set to 1/16 in our experiments.

$$K > (C \times RTT)/7 \tag{1}$$

$$g < \frac{1.386}{\sqrt{2(C \times RTT + K)}} \tag{2}$$

Third, for the variant version of the credit-based increase and decrease phase, a lower *target_loss* (only for the variant version) should be readjusted to reduce the aggressiveness in the early stages of deploying EC^4. In order to fit the current multi-protocol network environment, we choose 0 as the *target_loss* at present.

Fourth, the current *target_ECN_ratio* should be 0 in current shallowed-buffer switch environment. This will ensure that the average queue length is around K. However, we provide an interface by defining the *target_ECN_ratio* to improve the feedforward compatibility for DCNs. We can control the switch buffer usage by modifying the parameters on the host without modifying the threshold at the switch.

In addition, for EC^4, the selection of the aggression factor ω is particularly significant. EC^4 does not always utilize all the bandwidth, and thus in most cases, credit_loss is smaller than the target_loss, which means ω is getting closer to ω_{mid} in the multi-protocol network. In order to make EC^4 reduce its strong encroachment in such DCNs, we choose a smaller ω_{mid}. After parallel flow experiments between EC^4 and DCTCP, it is found that ω_{mid} should be 0.04–0.06 to ensure fairness. The specific evaluation of the range of ω_{mid} continues in the following section. Here, ω_{mid} is set to 0.05.

IV. EVALUATION

We use the OMNeT++ simulator to evaluate three key aspects of EC^4 (The ratio of EC^4 (ExpressPass) and DCTCP in all of experiments is 1:1.): (i) the utilization, convergence speed, fairness, and queuing of EC^4 with DCTCP; (ii) the effectiveness under heavy incast traffic patterns; and (iii) the performance under the actual data centers workload.

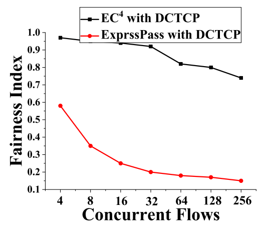
A. Microbenchmark

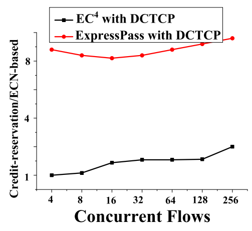
(I)Utilization: We first measure the utilization. One of the benefits of using ECN is high utilization. However, since EC^4 must provide 5% link bandwidth for credit transmission, the utilization of EC^4 is close to 95%. We compare EC^4 in the multi-protocol network, ExpressPass under same environment, ExpressPass, and DCTCP. The result shows that both EC^4 (ExpressPass) and DCTCP can make full use of network resources. The network in which the credit-reservation protocol and the ECN-based protocol coexist, the network utilization is 95%, while the utilization of only ECN-based protocol is up to 100%.

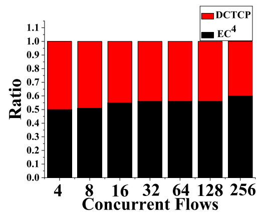
(II)Fairness: Fairness is a highlight of our work. We mainly conduct two evaluations on the fairness of EC^4 in multi-protocol networks. (i) We calculate the average bandwidth for each flow over a 100 ms interval and use the Jain's fairness index [29] to measure fairness. This is presented in Fig. 4a. Due to the aggressiveness of the credit-reservation congestion control, and the fairness being poor. As the concurrent flows increases, i.e., when there is bursty traffic, ExpressPass suffers a large amount of packet loss, and fairness deteriorates further. In contrast, EC^4 has shown great advantages due to ECN-based feedback control. (ii)The fairness between two types of traffic is evaluated by the ratio of the EC^4 and DCTCP as the Fig. 4c and 4d. The closer the ratio is to 1, the better the fairness between the two types of traffic. This method of judging ignores the internal unfairness caused by packet loss, and only focuses on the unfairness between traffic. From Fig. 4b, we note that the internal unfairness caused by packet loss is shielded, but the unfairness between traffic is still immense.

(III) Convergence: Another advantage of EC^4 is the high convergence. We designed a series of experiments where EC^4 (or ExpressPass) coexists with DCTCP. As shown in Fig. 5a and 5e, ω_{mid} is too small or too large to completely convergence. However, as seen from Fig. 5b, 5c and 5d , when ω_{mid} ranges from 0.04 to 0.06, EC^4 can converge with DCTCP. Besides, the smaller the value, the worse convergence, but the better the stability. When the aggression factor is set to 0.06 as shown in Fig. 5d, the convergence speed is 5 ms faster than that of DCTCP (as shown in Fig. 5g). Furthermore, Fig. 5g shows that ExpressPass with DCTCP in the multi-protocol network cannot achieve convergence. Finally, Fig. 5h shows that both protocols have the same performance with high convergence when all hosts deploy EC^4.

(IV)Queue Length: The last major indicator is buffer occupancy. For shallow buffered switch, low buffer occupancy can avoid as much packet loss as possible. Our experiments assume that the buffer is infinite, measuring the maximum queue length of ExpressPass in multi-protocol networks. Through experiments, we determine that EC^4 effectively reduces the queue length, and can guarantee a queue length less than 200KB even in the case of 256 concurrent flows, thus greatly reducing the risk of packet loss and retransmission. We evaluate two indicators of the queue. The first indicator is the maximum queue length, which characterizes the


(a) Jain's Fairness Index


(b) Credit-reservation/ECN-based


(c) EC^4 : DCTCP

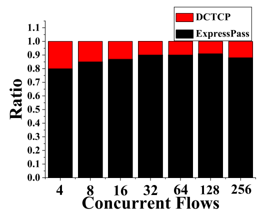
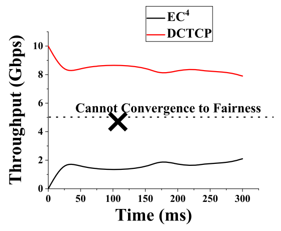
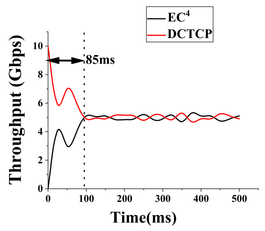
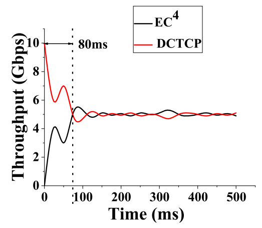
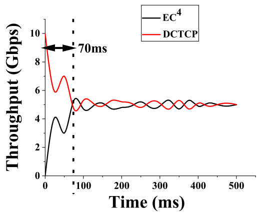

(d) ExpressPass : DCTCP

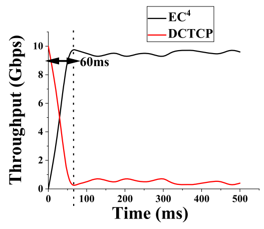
Fig. 4. Fairness Behavior: EC^4 shows great advantages due to ECN-based congestion control, conversely ExpressPass cannot guarantee fairness.

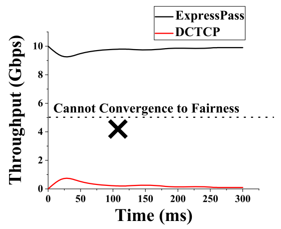

(a) EC^4 with DCTCP (ω_{mid}=0.03)

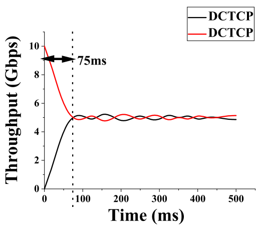

(b) EC^4 with DCTCP (ω_{mid}=0.04)


(c) EC^4 with DCTCP (ω_{mid}=0.05)


(d) EC^4 with DCTCP (ω_{mid}=0.06)


(e) EC^4 with DCTCP (ω_{mid}=0.07)


(f) ExpressPass with DCTCP


(g) DCTCP

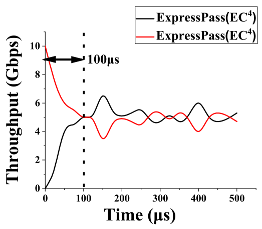
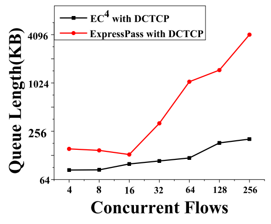

(h) EC^4 (ExpressPass)

Fig. 5. Convergence behavior: when ω_{mid} ranges from 0.04-0.06, EC^4 can converge to fairness with DCTCP, while ExpressPass cannot converge to fairness.


(a) Max Queue Length

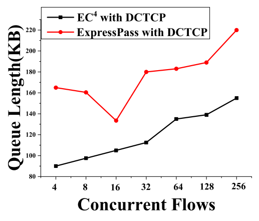

(b) Average Queue Length

Fig. 6. The maximum queue length represents the ability to cope with bursty traffic. The average queue length represents the average occupancy of the buffer. EC^4 avoids bursty traffic and reduces the buffer occupancy.

ability to react to bursts. The second indicator is the average queue length, which characterizes the queue under general circumstances. It can be seen from the Fig. 6 that the EC^4 avoids a large amount of packet loss in response to bursty traffic and reduce buffer occupancy in the general case.

B. Heavy Incast

To demonstrate the advantages of EC^4 in the case of heavy incast, we use the traffic pattern of the shuffle step in MapReduce [30] for evaluation. In a shuffle workload pattern, a ToR has forty servers that deploy EC^4 and DCTCP in a 1:1 ratio. These servers broadcast a 5.12 MB message to other servers. These messages flow out randomly without interruption. Each server has a corresponding host, and it sends a response message of about 16 KB to the source node. This process is repeated 30 times.

By measuring the FCT of the mini flow, we find that the medium FCT of the EC^4 is slightly improved. The 99th percentile and max FCT are essentially the same as ExpressPass. Note that due to the improvement of fairness, performance will be slightly improved, but the impact on mini-flow is not obvious.

However, as can be seen from Fig. 7, the median FCT of ExpressPass is better than that of EC^4. This is because the unfairness causes the ExpressPass to occupy most bandwidth, and the transmission is completed first; therefore, the FCT is 0.9 s, which is slightly better than EC^4's 1.18 s. However, this results in a heavy tail delay. Since the EC^4 guarantees fairness, the 99th percentile delay is only 2.25 s; that of ExpressPass is 8.7s. Furthermore, the maximum delay of EC^4 is only 2.35 s where that of ExpressPass is 20.3s.

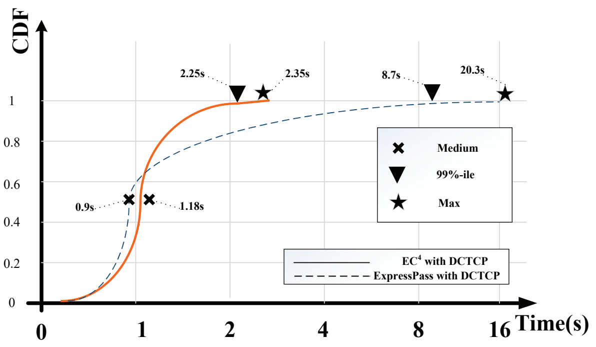


Fig. 7. CDF of FCT to the heavy flow. The impact of EC^4 on heavy-flow under shuffle workload embodies that the 99th percentile delay of EC^4 is only 2.35 s; that of ExpressPass is 8.7s.

TABLE I
Flow Size Distribution of Realistic Workload

	Data Mining [28]	Web Search [8]	Cache Follower [2]	Web Server [2]
0-10 KB (S)	78%	49%	50%	63%
10 KB-100 KB(M)	5%	3%	3%	18%
100 KB-1 MB (L)	8%	18%	18%	19%
1 MB- (XL)	9%	30%	29%	
Average flow size	7.41 MB	1.6 MB	701 KB	64 KB

C. Performance under Actual Data Centers Workload

To make EC^4 more realistic, we test it under the four workloads shown in Table I. These four workloads cover all flow sizes. The average flow size ranges from 64 KB to 7.4 MB. We test three loads of 0.2, 0.4 and 0.6.

We use a fat tree topology consisting of 8 core switches, 16 aggregator switches, 32 ToR switches and 192 nodes. This allows ToR to have a 3:1 over-subscription. Our network is set to 40 Gbps. The maximum queue length is set to 384.5 KB (250 MTUs). The link and host delays are set to 1 μs. The ratio of EC^4 (ExpressPass) and DCTCP deployed in nodes is 1:1. We measure the FCT for different sizes of flows for each workload, and measure the queue occupancy of the switch.

(I) Flow Completion Time: We test the three workloads shown in Table 2, and determine the average and 99th percenfile FCT. The FCT counted here is the FCT of all traffic in the network. For the average FCT, ExpressPass with DCTCP is worse than DCTCP due to packet loss and retransmission. The average FCT of EC^4 and DCTCP under the multi-protocol environment is almost the same as for DCTCP only. For the 99th percentile FCT in the multi-protocol network, EC^4 is particularly better than ExpressPass for heavy flow. The results are shown in Fig. 8.

(II) Queue Length: By measuring the queue length at the bottleneck link switch, we conclude that heavy flow has a high tail queue length when ExpressPass is deployed into

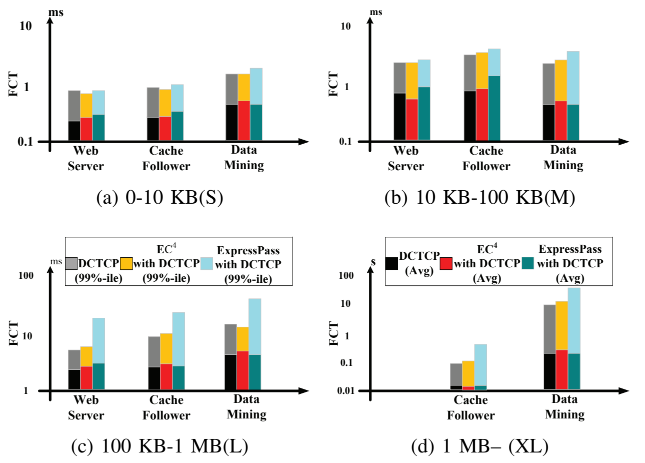


Fig. 8. Average/99th percentile FCT for realistic workload (40 Gbps, load 0.6). EC^4 greatly reduces 99th percentile FCT for heavy flow.

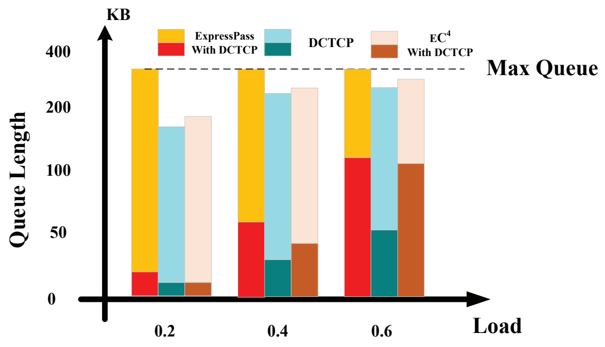


Fig. 9. Average/99th percentile queue occupancy (web search workload). EC^4 avoid bursty traffic and optimize the average queue length.

the multi-protocol network. Moreover, the maximum queue length reaches 373.5 KB regardless of the load. Using EC^4, the performance in the multi-protocol network can be almost the same as the performance of DCTCP. We conduct a simulation under the web search workload and the results are shown in Fig. 9.

V. Related Work

Delay-based congestion control: RTT, most applications provide an interface to extract is an effective congestion signal without any requirement of the switch feedback. The delay of a packet can embody the queuing delay of all the channels it passes through [10], and therefore it can reflect the ene-to-end congestion status of the network. Many works have explored congestion control making use of delay (e.g. DX [31] and TIMELY [10]). We once attempt to use delay as the congestion signal to inform the receiver to lower the credit sending rate. However, the feedback period of delay-based feedback control is not synchronized with ECN-based schemes and they are hard to converge. When the bursty traffic arrives, it often fails to react in time, causing the bandwidth balance to break down. More practical delay-based algorithm remains to be explored.

Other Credit-based Feedback Control: Credit-reservation congestion control in data centers is inspired by credit-based flow control for other interconnected systems [32]. ExpressPass uses a similar idea like TVA [33] that performs rate-limit requests at the router and EC^4 inherite this method. Furthermore, in high-performance networks, proactive congestion control uses grants for congestion control [34]–[36].Unlike EC^4, these schemes (SRP, SMSRP, and CRP) use speculative packets on a grant-based basis to avoid wasting preparing the data transmission. Although they are difficult to implement and have extra preprocessing time overhead, and these trade-offs are hard to balance, they provide an idea for the credit-based protocol in the DCNs. We look forward to finding a reasonable compatibility solution for other credit-based congestion controls. Moreover, compared with EC^4, end-to-end credit-scheduled congestion control focus on incast problems based on receiver. these transmission controls add an extra control layer to make sure senders only transmit according to some quota assigned to them. Examples include [37] and [38].

VI. Conclusion

In this work, we propose EC^4, an ECN-friendly credit-scheduled congestion control for incremental deployment in current data centers. To coexist with DCTCP, EC^4 uses FECN to detect the congestion in data queue to break the isolation between data queue and credit rate limiter. To guarantee high performance of EC^4 without interfering with DCTCP, we develop an efficient ECN-based feedback control to control the credit sending rate.

The evaluation results show that EC^4 can (i) achieve better performance than ExpressPass in a multi-protocol network, (ii) guarantee high utilization and fairness even at a high load in a multi-protocol network, and (iii) inherit the fine performance of ExpressPass under an ideal network environment. Therefore, EC^4 has high applicability in the incremental deployment of credit-reservation congestion control.

Acknowledgment

We gratefully acknowledge members of Tianhe interconnect group at NUDT for many inspiring conversations. This project was partially supported by the National Science and Technology Major Projects on Core Electronic Devices, High-End Generic Chips and Basic Software under grants No. 2018ZX01028101 and No.2017ZX01038104-002.

References

[1] A. Kalia *et al.*, "Using RDMA efficiently for key-value services," in *Proceedings of SIGCOMM'15*, pp. 295–306.

[2] A. Roy *et al.*, "Inside the social network's (datacenter) network," in *Proceedings of SIGCOMM'15*, pp. 123–137.

[3] A. Singh *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in Google's datacenter network," in *Proceedings of SIGCOMM '15*, pp. 183–197.

[4] P. Cheng *et al.*, "Catch the whole lot in an action: Rapid precise packet loss notification in data center," in *Proceedings of NSDI'14*, pp. 17–28.

[5] S. Huang *et al.*, "Congestion control in high-speed lossless data center networks: A survey," *Future Generation Computer Systems*, vol. 89, pp. 360–374, 2018.

[6] A. Kabbani *et al.*, "AF-QCN: Approximate fairness with quantized congestion notification for multi-tenanted data centers," in *Proceedings of HOTI'10*, pp. 58–65.

[7] M. Gusat *et al.*, "R^3C^2: reactive route and rate control for CEE," in *Proceedings of HOTI'10*, pp. 50–57.

[8] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," in *Proceedings of SIGCOMM '10*, pp. 63–74.

[9] Y. Zhu *et al.*, "Congestion control for large-scale RDMA deployments," in *Proceedings of SIGCOMM'15*, pp. 523–536.

[10] R. Mittal *et al.*, "TIMELY: RTT-based congestion control for the datacenter," in *Proceedings of SIGCOMM'15*, pp. 537–550.

[11] L. Jose *et al.*, "High speed networks need proactive congestion control," in *Proceedings of HotNets'15*, pp. 14:1–14:7.

[12] J. Perry *et al.*, "Fastpass: A centralized zero-queue datacenter network," in *Proceedings of SIGCOMM'14*, pp. 307–318.

[13] ——, "Flowtune: Flowlet control for datacenter networks," in *proceedings NSDI'17*, pp. 421–435.

[14] D. R. Pannell, "Network switch with head of line input buffer queue clearing," Jan. 21 2003, uS Patent 6,510,138.

[15] D. Lee, S. J. Golestani, and M. J. Karol, "Prevention of deadlocks and livelocks in lossless, backpressured packet networks," Feb. 22 2005, uS Patent 6,859,435.

[16] I. Cho *et al.*, "Credit-scheduled delay-bounded congestion control for datacenters," in *Proceedings of SIGCOMM'17*, pp. 239–252.

[17] Y. Zhang *et al.*, "BDS: a centralized near-optimal overlay network for inter-datacenter data replication," in *Proceedings of EuroSys'18*, p. 10.

[18] R. Mittal *et al.*, "Revisiting network support for RDMA," in *Proceedings of SIGCOMM'18*, pp. 313–326.

[19] N. Farrington and A. Andreyev, "Facebook's data center network architecture," in *Proceedings of OI'13*, pp. 49–50.

[20] N. Farrington *et al.*, "Data center switch architecture in the age of merchant silicon," in *Proceedings of HOTI'13*, pp. 93–102.

[21] Alizadeh, Mohammad *et al.*, "Analysis of DCTCP: stability, convergence, and fairness," in *SIGMETRICS '11*, pp. 73–84.

[22] G. Judd, "Attaining the promise and avoiding the pitfalls of TCP in the datacenter," in *Proceedings of NSDI'15*, pp. 145–157.

[23] A. Bechtolsheim *et al.*, "Why big data needs big buffer switches," *Arista White Paper*, 2016.

[24] P. Sreekumari *et al.*, "Transport protocols for data center networks: a survey of issues, solutions and challenges," *Photonic Network Communications*, vol. 31, no. 1, pp. 112–128, 2015.

[25] L. Chen *et al.*, "AuTO: scaling deep reinforcement learning for datacenter-scale automatic traffic optimization," in *Proceedings of SIGCOMM'18*, pp. 191–205.

[26] A. Kuzmanovic, "The power of explicit congestion notification," in *Proceedings of SIGCOMM'05*, pp. 61–72.

[27] M. Alizadeh *et al.*, "pFabric: minimal near-optimal datacenter transport," in *Proceedings of SIGCOMM'13*, pp. 435–446.

[28] A. Greenberg *et al.*, "VL2: a scalable and flexible data center network," in *Proceedings of SIGCOMM'09*, pp. 51–62.

[29] R. Jain *et al.*, "Throughput fairness index: An explanation," in *ATM Forum contribution*, vol. 99, no. 45, 1999.

[30] G. Ananthanarayanan *et al.*, "Reining in the outliers in Map-Reduce clusters using mantri," in *Proceedings of OSDI'10*, pp. 265–278.

[31] C. Lee *et al.*, "Accurate latency-based congestion feedback for datacenters," in *Proceedings of ATC '15*, pp. 403–415.

[32] H. T. Kung *et al.*, "Credit-based flow control for ATM networks: Credit update protocol, adaptive credit allocation, and statistical multiplexing," in *Proceedings of SIGCOMM'94*, pp. 101–114.

[33] X. Yang *et al.*, "A dos-limiting network architecture," in *Proceedings of SIGCOMM'05*.

[34] J. Nan *et al.*, "Network congestion avoidance through speculative reservation," in *Proceedings of HPCA'12*, pp. 443–454.

[35] G. Michelogiannakis *et al.*, "Channel reservation protocol for over-subscribed channels and destinations," in *Proceedings of HPCA'13*, pp. 52:1–52:12.

[36] J. Nan *et al.*, "Network endpoint congestion control for fine-grained communication," in *Proceedings of SC'15*, pp. 35:1–35:12.

[37] B. Montazeri *et al.*, "Homa: A receiver-driven low-latency transport protocol using network priorities," in *Proceedings of SIGCOMM'18*, pp. 221–235.

[38] M. Handley *et al.*, "Re-architecting datacenter networks and stacks for low latency and high performance," in *Proceedings of SIGCOMM'17*, pp. 29–42.