CQRD: A Switch-based Approach to Flow Interference in Data Center Networks

Guo Chen †, Dan Pei * †, Youjian Zhao †
† Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
Email: {chen-g11@mails, peidan@, zhaoyoujian@}.tsinghua.edu.cn

***Abstract*—Modern data centers need to satisfy stringent low-latency for real-time interactive applications (e.g. search, web retail). However, short delay-sensitive flows often have to wait a long time for memory and link resource occupied by a few of long bandwidth-greedy flows because they share the same switch output queue (OQ). To address the above flow interference problem, this paper advocates more fine-grained flow separation in the switches than traditional OQ. We propose CQRD, a simple and cost-effective queue management scheme for data center switches, through only minor changes to the buffering and scheduling scheme. No change to the transport layer or coordination among switches is required. Simulation results show that CQRD can reduce the FCT of short flows by 20-44% in a single switch and 8-30% in a multi-stage data center switch network, only at the cost of a minor goodput decrease of large flows.**

I. Introduction

As people and business increasingly rely on the Internet in their daily life and work, the performance requirement on the Data Center Networks (DCN), where most of the Internet applications are hosted, has become more stringent. However, recent studies have shown that short delay-sensitive flows from the real-time interactive applications (e.g. search, web retail), although contributing to majority of flows in DCNs [1], often have to wait a long time at switches for buffer and bandwidth resources occupied by a few of long bandwidth-greedy flows (e.g., backup, replication etc.). This causes the collapse of the throughput, and dramatically increases the flow completion time (FCT) of most short flows [2].

As analyzed in many recent studies [1–4], the fundamental reason for the above mentioned performance degradation is that the commodity DCN switches' traditional and coarse (Output Queue, or OQ) queue management schemes are not suited well for the DCN traffic characteristics, causing unnecessary **flow interference**. We define that two flows **are interfered** by each other when they contend for some shared resources at switches, such as queue memory or link capacity. Many transport layer solutions [2, 3, 5] and preemptive flow scheduling architectures [1, 4] have been proposed to get around the coarse queue management problem by optimizing flows' rate assignment and scheduling to keep the switch queues near empty. However, all these approaches need to modify end host's protocol stack and face significant deployment hurdles.

* Dan Pei is the corresponding author.

Different from these previous approaches, we address the DCN flow interference problem by directly tackling its root cause: coarse switch queue management schemes. Hence, we argue that the DCN flow interference (between large number of small delay-sensitive flows and a small number of giant flows) calls for a more fine-grained queue management than the current output queue (OQ) in the commodity DCN switches. As such, we propose a simple and cost-effective queue management scheme, crosspoint-queue with random-drop (CQRD), where a separate queue is assigned to each pair of input and output port. The proposed approach only requires a minor revision to the switches' queue management scheme, without any coordination among switches, and without any modification to end hosts. Through simulations, we show that CQRD significantly reduces the flow completion time of short flows by 20-44% in a single switch and 8-30% in a multi-stage data center switch network, only potentially at the cost of a minor goodput decrease for large flows. Furthermore, CQRD is complementary to transport layer approaches, and a hybrid of them could potentially achieve even better performance.

The rest of the paper is organized as follows. We review the related works in Section II. In Section III, we use analysis and simulation to study flow interference and its performance impact in traditional OQ, HCF (state-of-the-art switch queue management for DCN) [6], and CQ. In Section IV, we present our CQRD approach and analyze how it can alleviate flow interference in DCN. In Section V, we use simulation experiments to show that CQRD greatly improves the overall DCN performance over both OQ and HCF. Finally we conclude in Section VI.

II. Related Works

Long delay of short delay-sensitive flows due to flow interference is a well known problem in data center network. We describe several solutions below and illustrate the difference between CQRD and them.

A. Transport Layer Rate Control

A major direction of prior work uses transport layer rate control to reduce flow completion time of short flows. D-CTCP [2] and HULL [5] apply adaptive rate control schemes based on ECN [7] and packet pacing, to control the rate of giant flows. By keeping the queue size of switches near empty, they improve the overall FCT of short flows. D^2TCP [8] and D^3 [3] use the deadline information for rate control.

978-1-4799-3780-6/14/$31.00 ©2014 IEEE

They allocate the rate of each flow according to their deadline information.

All these methods require a modification to end hosts' TCP stack to implement their rate control schemes, thus are not well compatible with legacy TCP. In addition to TCP stack change, some of them further require a significant change of end hosts' NIC [5] and/or switch hardware [3, 5]. Therefore, it is hard to deploy these approaches in real DCN. Moreover, precise rate control is a great challenge due to the bursty traffic in DCN.

B. Preemptive Flow Scheduling

Another direction to solve this problem is the preemptive flow scheduling. Recent work such as PDQ [4] and pFabric [1] try to implement optimal flow scheduling to minimize the FCT of short flows. However, these solutions are almost clean-slate architecture that requires new end host protocol stack and switch hardware design, which can be far from getting deployed in reality. Furthermore, implementing PDQ is quite complex, and starvation of large flows is a big concern in pFabric.

C. Switch Based Solutions

There are also many queue management schemes in the literature for switches/routers to provide fairness for TCP flows, such as DRR [9] and SFQ [10]. However, they have been designed for traditional routers and LANs, and not applicable for (quite different) traffic characteristics in DCN. Furthermore, recent work in [6] has shown that HCF outperforms them in DCN environment.

The most closely related work to CQRD and the state-of-the-art approach in this space is HCF [6] (Hashed Credits Fair). Similar to CQRD, HCF tries to address the flow interference problem by providing switch queue management scheme that is more fine-grained than OQ. HCF sets two separate queues, one high-priority (HP) and one low-priority (LP), at each output. It hashes all the incoming flows into several bins and assigns each bin a credit. Packets belonging to bins which have credit left will be stored in the HP queue, otherwise in the LP queue. Switches serve the HP queue if it is not empty. When the HP queue becomes empty, HCF resets all the credits to the initial and HP queue is swapped with LP queue. However, it is challenging to hash flows uniformly using static hash function. Therefore, HCF needs to change its hash function periodically, which increases the cost on hardware. Furthermore, as will be shown in Section III-C, HCF is not fine-grained enough to solve the DCN flow interference problem very well.

III. Flow Interference: Causes and Impact

In this section, we discuss the causes and performance impact of flow interference. We study different switch queue management schemes, including traditional output-queue (O-Q), the state-of-the-art DCN fairness queue management scheme (HCF [6]), and classic crosspoint-queue (CQ) [11]. Through analysis and a toy example, we will show that CQ is more promising to solve the flow interference problem.

A. Performance Metrics and Definitions

Following the convention in [1], we consider two main performance metrics—**flow completion time** and **goodput**. Flow completion time (FCT) is an important metric for short delay-sensitive flows, and reflects how fast the flow has been successfully transmitted. Goodput equals the flow size divided by its FCT, which is crucial to large bandwidth greedy flows.

Similar to prior work [1], we define flows smaller than 100KB as **small/short flows**, flows larger than 100KB as **large/long flows**, and flows larger than 1MB as **giant flows** (special case of large flows).

At a given switch, when two flows have the same output port and they overlap in time, we say these two flows **output port contending** or simply **output contending**. Then, we define a **switch path** as the pair of input port and output port on the same switch. When two flows on the same switch path overlap in time, we say these two flows are **switch path contending** or simply **path contending**, which is a special case of output contending. For flows that go through multiple switches, we define two flows as path contending when they are path contending at any of these switches.

In this section, we only focus on the output contending (but excluding the path contending) interference.

B. OQ Switches

Typically, commodity switches in data centers apply OQ with tail-drop scheme [2, 12]. Packets from output contending flows are stored in the same queue at the specific output port. This exploits statistical multiplexing and saves memory resources to achieve a certain packet loss rate. However, it may also cause a strong interference of flows which contend for the same output.

Fig. 1(a) shows an toy example with an 8 port OQ switch connecting to server 1-8 with each port respectively. Note that each port consists of an input link and an output link shown in the figure. Assume that there are seven flows coming from inputs I1 to I7 respectively. All the flows are destined to the same output O8. Flows from I6 and I7 are *giant* flows and the other five flows are short delay-sensitive flows. While contending for the same output, the large flows quickly occupy the majority of shared memory resource and the capacity of output link. Packets of the small flows have to wait unnecessarily, queuing behind the packets from the long flow. This leads to a significant increase of packet delay for small flows and impairs the upper-layer applications. As the output contention lasts for a while, the output buffer will be filled up and begin to drop packets of all short flows. This further results in an overall performance degradation.

We use NS2 [13] for simulation and measure the goodput and FCT of each flow in this toy example. Each port has a bidirectional link rate of 10Gbps and a one-way link delay of $4\mu s$. Each port has a small output queue with a typical small size of 36KB [2] (*i.e.*, 288KB in total). Assume servers 6 and 7 are generating long file-backup traffic to server 8. At time 0, both servers 6 and 7 start to send a 100MB file using TCP (flows 6-7) to server 8. Five milliseconds later, servers 1-5

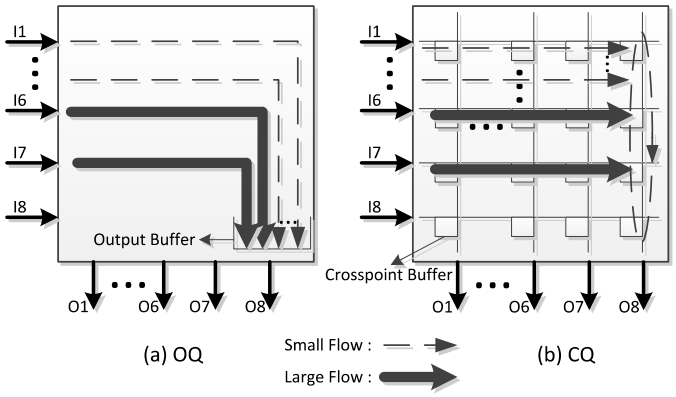
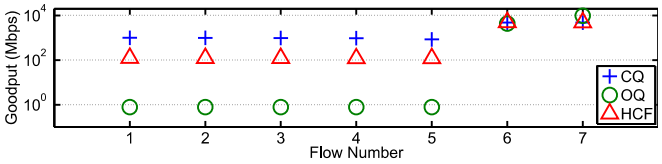


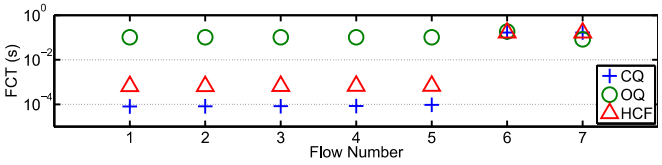
Fig. 1. Flow interference in OQ switch and CQ switch

start 5 delay-sensitive tasks and each sends a 10KB TCP flow (flow 1-5) to server 8 at time 0.005s. TCP SACK [14] are used for all the flows.

The goodput and FCT of each flow are shown in Fig. 2. During this situation, the output queue of port 8 is quickly filled up with packets of flow 6 and 7. Thus, packets of flow 1-5 are continuously dropped when they reach the switch. As a result, their flow completion time soar up to hundreds of milliseconds, while the theoretical ideal FCT should be as low as tens of microseconds. Also, their goodput fall down to lower than 1Mbps. Apparently, this will cause a dramatic performance degradation of upper-layer applications.



(a) Goodput (Log scale)



(b) Flow Completion Time (Log scale)

Fig. 2. A toy example of flow interference.

C. HCF Switches

Recently, a hash-based based queue management algorithm for DCN switches called HCF (Hashed Credits Fair) [6] has been proposed. Through hash and assigning credit, flows from different bins can share the HP queue fairly. And when HCF schedules packets out of HP queue, flows from different bins could fairly share the link capacity. That provides a relative good bandwidth fairness. However, it does not provide enough buffer fairness between small flows and giant flows. Many flows coming from different inputs still have to contend for the same buffer resource. While large flows quickly consume the credits of their bins, they fill up the LP queue quickly. If small flows are unluckily hashed to the same bins, they would be dropped. In addition, even if they are not in the same bins, as the number of small flows grows larger, they will also consume their credits and be dropped at the tail of LP queue. That causes packet loss of many small flows with inputs different from the few giant flow.

We simulate HCF switch in the same toy example, with the same 288KB total memory. All the parameters of HCF are set as the recommended in their paper (2 queues with same length, 1 credit for each of the 20 bins, and periodical XOR hash function). As we can see in Fig. 2, HCF greatly reduces the FCT of short flows and serves two large flows fairly. However, the packets of small flows also have been dropped. That increases their FCT to around $1ms$, which should be less than $100\mu s$ without loss.

D. CQ Switches

Recently, the decade-old CQ switching scheme, once considered infeasible for commodity switches when first proposed [15], has been shown to become very feasible using modern semiconductor technologies [11]. CQ offers full flow separation for the flows that are output contending but not path contending. As shown in Fig. 1(b), CQ switch reserves separate memory resources for each pair of inputs and outputs. Packets arrived at each input are first buffered into the crosspoint-buffer (XB). Then each output port, without coordinating with any other ports, picks one of the XBs in its column and schedules the head packet out of the switch. Thus, flows from different input ports have separated buffers. Also, by using simple Round-Robin (RR) scheduling manner for each output, each flow from different inputs destined to the same output shares approximately the same output link capacity. Therefore, CQ switch can provide a performance separation for flows coming from different inputs.

We simulate CQ switch in the same toy example. With 288KB memory in total, the same as in OQ switch, each XB of CQ switch has a capacity of 4.5KB. As shown in Fig. 2 , with CQ switch, the FCT of delay-sensitive flows can be 3 orders of magnitude lower than the OQ switch and 1 order of magnitude lower than the HCF switch. Also, the goodput of those flows are 3 orders of magnitude and 1 order of magnitude higher than the OQ and HCF switch. Although there is only 4.5KB buffer resource for flows 6 and 7 in CQ switch, the FCT and goodput of these two long flows are almost the same as in OQ and HCF switches. Meanwhile, CQ switch achieves a very good fairness among flows.

E. Summary

Overall, the performance of flows that are output contending with the giant flows is severely degraded by interference by the giant flows in OQ. Although to a less extent than in OQ, HCF still suffers from the same problem. Through separating queues for flows that are output contending but not path contending, a classic CQ switch can achieve much better performance than OQ and HCF in our toy example. We will present how to

design a CQ-based scheme for more complex real-world DCNs in the next section.

IV. Crosspoint-queue with Random-Drop Scheme

In the previous section, we have shown that CQ performs much better than OQ and HCF in the toy example where there are only very small number of flows and the primary interference is output contending only (excluding path contending). In the real world DCNs, there may be thousands of flows come and go, a large number of flows can overlap in time, and a flow may go through multiple switches. As such, flow interference becomes more complex, and both output contending and path contending can happen. In this section, we will first make an observation about DCN flow characteristics, which motivates our CQRD approach. Then we present CQRD approach and how it addresses the challenges faced in real world DCNs.

A. DCN Flow Characteristics

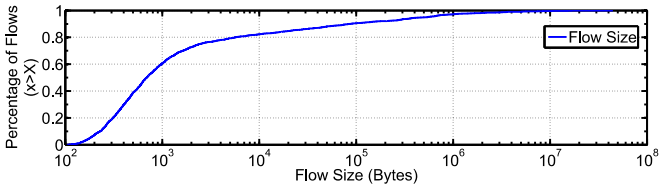


Fig. 3. Traffic workload derived from real data center.

The workload from the characteristics of real operation data centers [16] is shown in Fig. 3. In this workload, about 90% flows are small/short flows, and very few (about 3%) of flows are giant flows. Based on the measurement results of real operational large data centers in [2, 17], we make the following observation.

Observation 1. *Long bandwidth-greedy flows traverse a few of switch paths in a DCN switch, while most of the switch paths are transmitting short delay-sensitive flows.* There are always a large number (more than one thousand [17]) of active flows in DCN. However, very few of concurrent flows [2] are larger than 1MB. In a data center running data mining jobs, over 80% flows are less than 10KB [1]. As a result, while some long bandwidth-greedy flows passing a few of *switch paths* (defined in Section III), the majority of switch paths are transmitting short flows.

The above observation explains the significant flow interference in current DCNs. For example, a 24×24 aggregation switch has 24×23 switch paths in total (assuming no flow is destined to its coming input port). Assume that there are two giant flows passing two switch paths. Meanwhile, there may be hundreds of short delay-sensitive flows passing the other 550 switch paths. In a commodity OQ switch, 2×23 out of these 550 switch paths have the same output queues as the two giant flows, which interfere (and contend for paths) with the hundreds short flows going through the same 46 switch paths.

B. CQRD Overview

We argue that the above observation calls for a queue management approach that is more fine-grained than OQ to reduce the probability of DCN flow interference. However, it is desirable to maintain a good balance between queue management granularity and the overhead/cost. In the ideal and most fine-grained approach, if we could reserve dedicated buffer and link capacity large enough for every single flow, the flow interference is entirely eliminated. However, the memory and link capacity needed for this ideal approach to deal with the large number of overlapping flows are prohibitive in practice. Moreover, it's hard to dynamically allocate physical resource according to the flow's various needs (e.g. buffer size or bandwidth), because these information is not available to the switch, without big modification to its packet parsing procedure or current network protocol stack.

We thus present Crosspoint-Queue with Random Drop (CQRD), a cost-effective queue management approach that is fine-grained enough to achieve desirable flow separation. The basic idea of CQRD are two-fold:

- Complete separation between flows on different switch paths, because a separate buffer is allocated to each switch path, and packets destined to the same output port but on different crosspoint buffers are scheduled in round-robin fashion.
- When a crosspoint queue is full, random-drop is used to alleviate the flow interference within the same switch path.

Like original CQ switches shown in Fig. 1(b), CQRD allocates separate crosspoint buffers (XB) with the same capacity for each pair of inputs and outputs. Arriving packets are first stored in the crosspoint buffer and wait to be scheduled out by the output scheduler. CQRD uses Round-Robin (RR) scheduling for each output to schedule the packets. It ensures that each XB is fairly served. With separated buffers, flows in different switch paths will not contend for the queue with each other, although they might be destined to the same output. Also, with RR scheduling, flows going to the same output port is allocated with almost the same link bandwidth. This addresses the output contending but not path contending interference, and achieves *switch path separation*.

Secondly, when the corresponding crosspoint buffer is full, CQRD takes random-drop scheme upon packet arrival. Instead of simply dropping the tail (as in classic CQ), if the XB does not have enough space for the coming packet, CQRD will randomly choose a packet in this XB and drop it. A flow's packets will be more likely to be dropped if this flow occupies most of the XB, and vice versa. As such, if several small flows are contending for the same XB with a large flow, packets of those small flows still have a reasonable chance to get into the buffer even if the buffer are currently filled up by the large flows. That helps to alleviate interference within the same paths.

C. Implementation

One might wonder about whether it is simple and cost-effective to implement CQRD switch with crosspoint buffers large enough by modern technology. In fact, to build a typical data center switch with 24 ports, it is easy to implement crosspoint buffer with size of more than 10 kilobytes, only using on-chip memory of a commodity FPGA chip [18, 19]. Moreover, [11] showed that a crosspoint buffer could store over 3 kilobytes packets for a switch with more than a hundred of ports, built with application-specific integrated circuit chips (ASIC), which is much easier and cheaper today.

V. Performance Evaluation

To evaluate CQRD's performance, we implement and simulate CQRD in NS2 [13], and compare its performance with two other switch-based approaches, OQ and HCF. Because we assume no modification to end host systems, the transport layer and high-level flow scheduling approaches are not compared. The evaluation is based on the following two experiments. In *experiment 1*, we simulate a single DCN aggregation/core switch. In *experiment 2*, a classic multi-stage DCN switching topology with 480 servers has been simulated. We first describe the traffic workloads, simulation parameters, and performance metrics, followed by the detailed results of the two experiments.

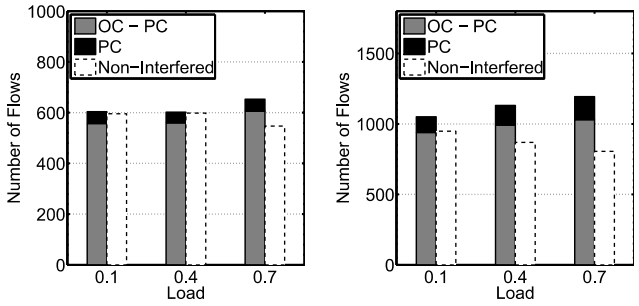
Since giant flows are the triggers of DCN's performance degradation [2], we are mainly interested in the flows interfered by the giant flows. We will show that OQ, HCF, CQRD perform differently because they differ in queue management granularity and how they deal with the output contending and path contending flows.

A. Experiment Setup

Traffic Workloads: We derive our workloads from the characteristics of real operation data center traffic [16, 17] as shown earlier in Fig. 3. During the simulations, source and destination of the flows are randomly chosen among all the switch ports (in experiment 1) or among all hosts (in experiment 2). The inter-arrival time of the flows obeys the log-normal distribution [17]. We scale the flow inter-arrival time to simulate the moderate (0.1), heavy (0.4), and extreme (0.7) loads in the network.

Fig. 4 shows the ratio of flows that are output contending but not path contending (OC-PC), path contending (PC), and not interfered (Non-Interfered) with giant flows (as defined in Section IV) in all the simulations. This figure shows that the giant flows interfere with more than 50% of all flows although they contribute to less than 3% of all flows according to Fig. 3. And most of the interfered flows are in different switch paths with the giant flows.

Simulation Parameters: All the parameters of HCF are set as the recommended in HCF paper [6] (same as Section III-C). During all the simulations, we use the SACK [14] version of TCP, which has already been implemented in most of the Linux release version. The initial window size and min retransmission-time-out (RTO) are set to be 4 and $200\mu s$ respectively, which is typical in DCN [1].



(a) Experiment 1 (b) Experiment 2

Fig. 4. The number of flows that are output contending but not path contending (OC-PC), path contending (PC), and not interfered (Non-Interfered) with giant flows in the experiments.

Performance Metrics: Following the convention in [1], we consider two main performance metrics—flow completion time for short flows and goodput for large flows. These two metrics reflect the key performance of these two kinds of flows.

B. Experiment 1: Single Aggregation/Core Switch

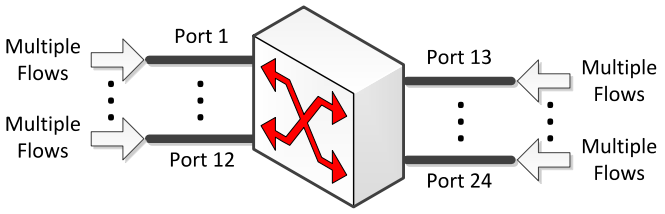
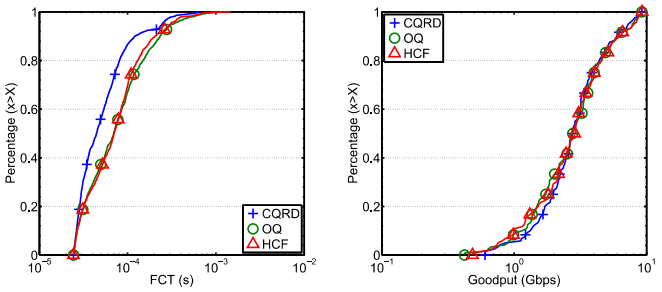


Fig. 5. Experiment 1: A 24×24 switch used in simulations

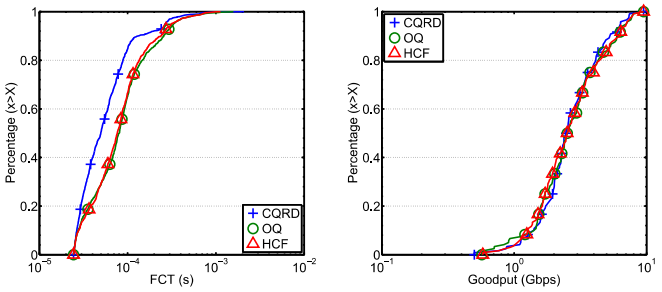
Setup: Our first experiment is to simulate the aggregation and core switches with huge amount of flows passing by, and compare different approaches in this situation. We simulate a 24×24 switch (a typical DCN switch) shown in Fig. 5. Each port has a 10Gbps link rate and $4\mu s$ link delay. This generates a $\sim 16\mu s$ end-to-end round-trip time (RTT) without queueing, which is realistic in the real DCN environment according to [1]. In all three schemes, we assume the on-chip memory for packet buffers are 5MB, which can be easily implemented by commodity FPGAs. As a result, each output buffer has $\sim$210KB in OQ, which conforms to the convention [1]; HP and LP queue each has $\sim$105KB in HCF; each crosspoint queue buffer has $\sim$9KB in CQRD.

During the simulation, 1200 flows are generated to 24 ports. The workload is as described before.

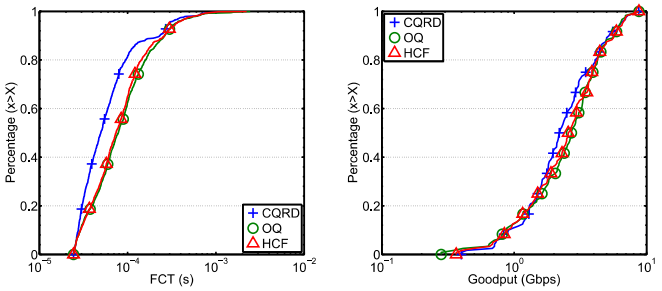
Results: Fig. 6 shows the overall FCT of all short flows and goodput of all large flows at various loads of 10%, 40% and 70%, which represent moderate, heavy and extreme loads. We observe that, with any load, CQRD has a much better overall FCT of short flows, which contribute to about 90% of all flows according to Fig. 3. Almost all the percentile of CQRD's FCT



(a) Load=0.1



(b) Load=0.4



(c) Load=0.7

Fig. 6. Experiment 1: Overall CDF of all short flows' FCT and all large flows' goodput, at various loads.

is greatly shorter than HCF or OQ's. The median of CQRD's FCT is ∼35% shorter than other schemes at all the loads. Also, the 99th percentile of CQRD's FCT is still ∼5-10% better. As the load grows higher (e.g ≥40%), more and more concurrent flows come into the switch and flow interference becomes severe. In this situation, it gets harder to provide good flow separation and the performance of these three schemes tends to get similar at the tail. However, CQRD still performs the best at almost all the percentiles as we can see in Fig. 6. This is because that there are several giant flows with sizes of more than 1MB among the 1200 generated flows. In OQ switch, they occupy most of the buffer resource and lead to packets of many other short flows dropped, which greatly increase the overall FCT. Although HCF sets two separate queues at each output and tries to fairly serve all the flows using hash and credit schemes, it does not provide enough buffer separation for different flows. Many flows are output contending but not path contending, and in HCF they still have to contend for the same buffer resource. This results in the packet losses for short flows that are output contending but not path contending with giant flows, which is avoided in CQRD.

As for flows larger than 100KB, CQRD has almost the same overall performance (i.e., goodput) as HCF and OQ (the right half of Fig. 6). These results show that the small size of separated buffer in CQRD will not significantly impair the large flows' goodput.

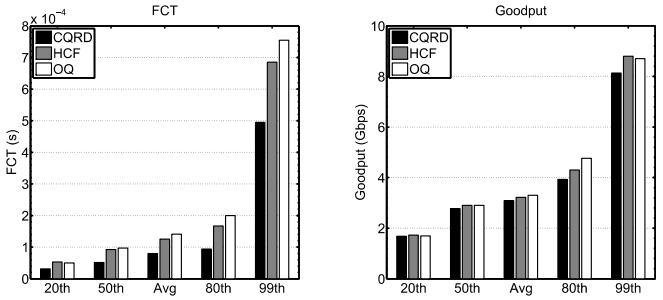
So far, we have shown that the overall performance of all short flows in CQRD greatly outperforms the other two approaches, at the cost of only a minor goodput decrease of very few large flows. This is because CQRD alleviates the performance degradation of the flows interfered by a few giant flows. As Fig. 4(a) shows, although less than 3% of all flows are giant flows, they interfere with more than 50% of all flows. And most of the interfered flows are in different switch paths with the giant flows. CQRD uses separated buffers and random-drop schemes respectively, to guarantee the performance of those interfered flows having the same outputs or paths with giant flows. We now investigate all the flows (including giant flows) interfered by giant flows and show how CQRD improves their performance. Those flows which are not interfered by giant flows perform well and similarly in all the three schemes, so we omit the results here due to page limit.

In Fig. 7 we show the average and 20th to 99th percentile FCT of short flows and goodput of large flows which are interfered by giant flows, at a typical moderate load. We have also done these simulations at other loads from 10% to 80% and the comparison results are similar. We only present the results at typical moderate load (10%) here due to page limit.

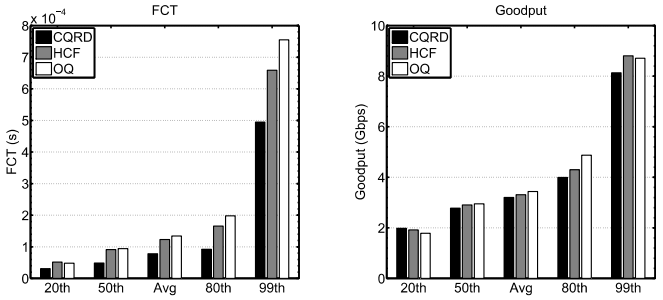
As Fig. 7(a) shows, for all interfered short flows, FCT in CQRD is about 27% to 44% lower than both HCF and OQ switches. On the other hand, the goodput of all interfered large flows is only a little lower than the other two approaches. For example, the 99th percentile in CQRD is only 7% lower than HCF's. These results show that CQRD is able to deal with flow interference much better than the other two approaches.

Recall that, as discussed in Section IV, CQ provides 1) complete separation for *output contending (excluding those path contending) flows*, by using separated buffers and RR scheduling, and 2) interference alleviation *for path contending flows* by using random-drop when queue is full. We now show how well CQRD performs for these two types of flows.

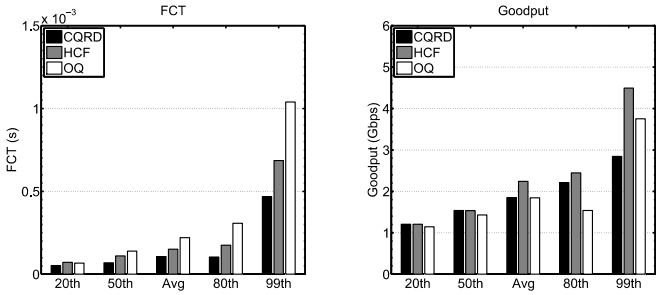
Fig. 7(b) shows the results for flows output contending (but not path contending) with giant flows. CQRD greatly exceeds others' performance for these flows just as we expect. These flows contribute to the majority of all interfered flows as shown in Fig. 4(a), therefore, their performance dominates the performance of all interfered flows. As for flows path contending with giant flows, CQRD successfully reduces the FCT of short flows significantly (left part of Fig. 7(c)), by random-drop scheme. However, due to separated allocation manner, each crosspoint buffer in CQRD has $1/N$ size of OQ's. When small and large flows in the same switch path contend for buffer, their packet loss rate are roughly proportional to their flow sizes, thanks to small buffer size and random-drop. This increases the chance of small flows to be stored



(a) All Interfered



(b) Output Contending But Not Path Contending



(c) Path Contending

Fig. 7. Experiment 1: Average and various percentile FCT of all short flows and goodput of all large flows **interfered** by the giant flows at moderate load.

in the buffer, but leads to a lower goodput of large flows. As we can see in Fig. 7(c), the 99th percentile (representing the flows with highest goodput) of CQRD's goodput is lower than that of HCF. However, the average and percentiles from 20th to 80th goodput of CQRD are similar to HCF. That shows CQRD successfully alleviates the interference among flows path contending with giant flows.

C. Experiment 2: Multi-stage DCN Switching Fabric

Setup: In the second experiment, we simulate a two layer multi-root topology with full bisection bandwidth (see Fig. 8). This topology is one of the most commonly used topologies in large-scale DCN[16, 20]. The network consists of 480 end hosts allocated in 24 racks, which are interconnected by 2 aggregation switches and 24 top-of-rack (ToR) switches. Aggregation switches have 5MB memory for packet buffer as before, and twenty-four 10Gbps ports connected to each ToR switch. Each ToR switch has less memory with size of 4MB, and two 10Gbps ports connected with 2 aggregation switches, and twenty 1Gbps ports connected to 20 hosts respectively.

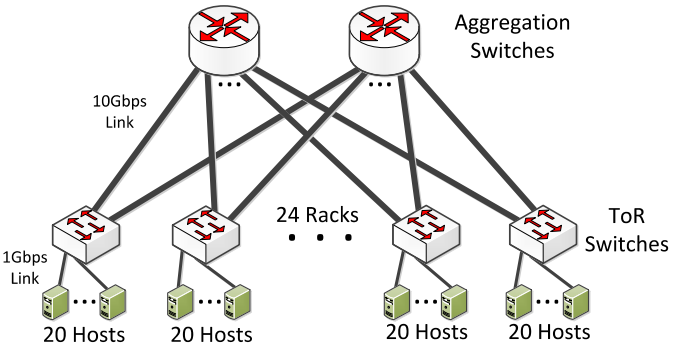
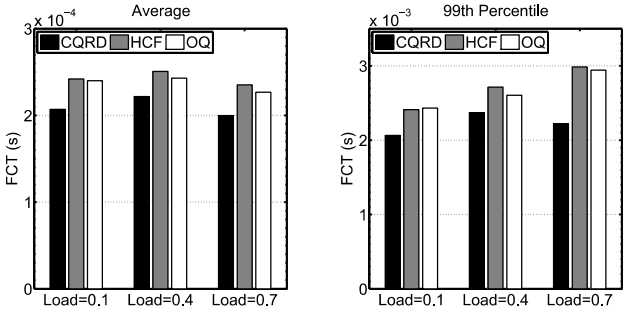
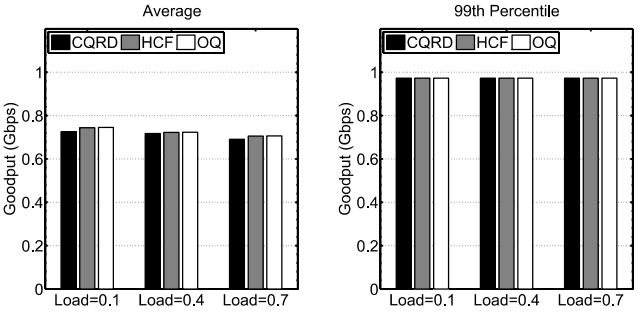


Fig. 8. Experiment 2: A multi-stage DCN topology used in simulations

These are typical parameters of ToR switches [21]. The delay of each link is $2\mu s$, which means a $\sim 16\mu s$ end-to-end RTT across racks and a $\sim 8\mu s$ RTT within a rack. We use Equal Cost Multi-path (ECMP) for network load-balancing, which is the de facto routing algorithm[22] in modern data centers. CQRD (HCF, OQ) are used in all switches in the topology when simulating CQRD (HCF, OQ)'s performance. During the experiment, 2000 flows are generated according to the work load described in Section V-A.



(a) FCT

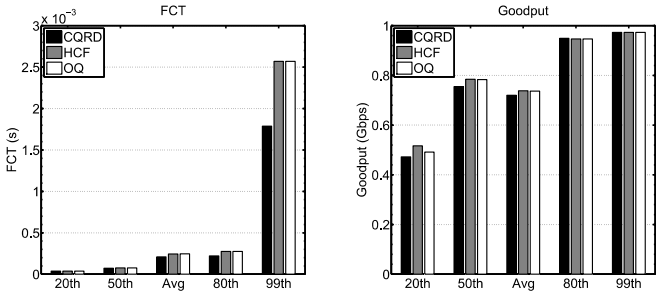


(b) Goodput

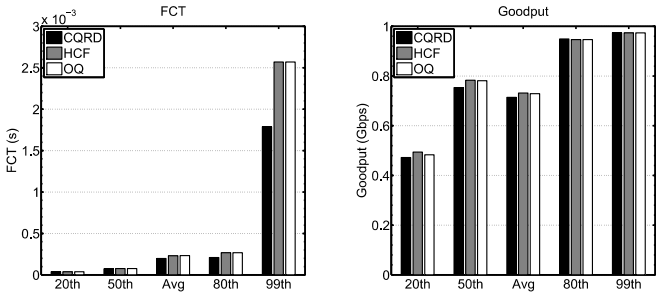
Fig. 9. Experiment 2: Overall FCT of all short flows and goodput of all large flows at various loads.

Results: First, we show the overall FCT of all short flows and goodput of all large flows at various loads. As shown in Fig. 9(a), CQRD has the best FCT performance at various loads. The average FCT of all short flows in CQRD are about 10% to 24% lower than HCF and OQ at moderate (10%), heavy (40%) and extreme (70%) loads. And the 99th

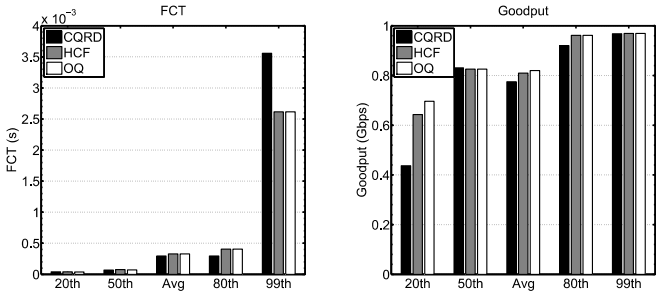
percentile of CQRD's FCT is also about 8% to 30% lower than others at all loads. In addition, Fig. 9(b) shows that, in CQRD, the average and 99th percentile goodput of all large flows perform almost the same as in HCF and OQ. These results show that a DCN built with CQRD switches has a much better overall performance, only at the cost of a minor goodput decrease of very small portion large flows. Using separated buffer and random-drop scheme, CQRD not only provides flow separation for a single switch environment as shown in the former subsection, but also performs well in a multi-stage switch data center network.



(a) All Interfered



(b) Output Contending But Not Path Contending



(c) Path Contending

Fig. 10. Experiment 2: Average and various percentile FCT of all short flows and goodput of all large flows **interfered** by the giant flows at moderate load.

As shown in Fig. 4(b), giant flows interfere with more flows in multi-stage DCN than in a single switch. This is because flows pass longer paths in this topology and more flows tend to intersect in ToR and aggregation switches. As load grows higher, flow interference becomes severe and more flows are affected by the giant flows. Next, we also dive into the performance of the flows interfered by giant flows. Those flows which are not interfered by giant flows are omitted here similar to experiment 1, because they almost have the same low delay and high goodput. Results at various loads are similar, so we only present the results at moderate load here due to page limit.

Fig. 10(a) shows the overall performance of all the interfered flows at moderate load. The average and 99th percentile FCT in CQRD are 14% and 30% lower than HCF and OQ. CQRD also has a much lower FCT for all interfered short flows at other percentiles. On the other hand, the goodput of large flows in CQRD is almost the same as other methods. This shows that in data center, for a multi-stage switch network built by CQRD, the flow interference caused by giant flows can be significantly alleviated.

We repeat the same simulation as experiment 1, to reveal the performance of the flows that are output contending with giant flows, in multi-stage switch DCN environment. As shown in Fig. 4(b), output but not path contending flows contribute to about 90% of all interfered flows. Thus, as we can see in Fig. 10(b), the average and various percentile of FCT and goodput of these flows are almost the same as those of all interfered flows. CQRD greatly improves the performance of these flows by switch path separation.

As for flows path contending with giant flows shown in Fig. 10(c), CQRD has the best FCT for average and all percentile except for 99th. As analyzed before, for those flows unluckily share the same switch path with giant flows, have to contend for a crosspoint buffer in CQRD. This buffer is much smaller than output buffer in HCF and OQ. In these situations, a few unlucky small flows may be dropped by random-drop schemes. This leads to a higher 99th percentile FCT and lower 20th percentile goodput. However, our random-drop scheme achieves a better overall performance as shown in Fig. 10(c). In addition, the flows path contending with giant flows only contribute to a very small portion of all the interfered flows. CQRD significantly improves the overall performance of DCN at the cost of a little portion of these flows. How to handle this small part of flows more gracefully is interesting and worth further studying. For example, nowadays the on-chip memory is cheap in price, thus increasing the buffer size in CQRD is a feasible approach to greatly improve the 99th-tile performance of CQRD. Furthermore, we can even employ HCF at each crosspoint if the buffer size is large enough. However, a more detailed study on this improvement is beyond the scope of this paper, and is left as our future work.

VI. Conclusion

In this paper, we advocate that modern DCN flow characteristics call for fine-grained queue management in switches. Along this direction, we propose a switched based solution called CQRD to address DCN flow interference problem, without any modification to end hosts or any coordination among different switches. CQRD is cost-effective, and is more fine-grained than traditional OQ scheme and current state-of-arts HCF scheme. It only requires some minor changes to the buffering and scheduling scheme in DCN switches. We show through NS2 simulations that, when flow interference happens, CQRD can improve the flow completion time of short and

delay-sensitive flows by up to $\sim 44\%$, at the cost of only a minor goodput decrease of large flows.

We believe fine-grained queue management is a very promising direction for DCN performance improvement. In the future, we plan to further investigate its performance under different scenarios, such as incremental deployment, larger buffer sizes, and a hybrid approach that combines switch-based and transport layer ones.

Acknowledgment

This work has been supported in part by the National High Technology Research and Development Program of China (863 Program) under Grant No. 2013AA013302, the National Key Basic Research Program of China (973 program) under Grant No. 2013CB329105 and the State Key Program of National Natural Science of China under Grant No. 61233007.

We would like to thank the anonymous reviewers for their valuable comments. We greatly thank Zhiyan Zheng for the packet trace of real operational network that he provided us. Also, we thank Kai Chen for his useful discussions regarding the architecture of actual data centers and Jilei Yang, Juexing Liao for their proofreading on this paper.

References

[1] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," in *Proc. of SIGCOMM*, pp. 435–446, ACM, 2013.

[2] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center tcp (dctcp)," *Proc. of SIGCOMM*, vol. 40, no. 4, pp. 63–74, 2010.

[3] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," in *Proc. of SIGCOMM*, vol. 41, pp. 50–61, ACM, 2011.

[4] C.-Y. Hong, M. Caesar, and P. Godfrey, "Finishing flows quickly with preemptive scheduling," *Proc. of SIGCOMM*, vol. 42, no. 4, pp. 127–138, 2012.

[5] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: trading a little bandwidth for ultra-low latency in the data center," in *Proc. of NSDI*, pp. 19–19, USENIX Association, 2012.

[6] A. Shpiner, I. Keslassy, G. Bracha, E. Dagan, O. Iny, and E. Soha, "A switch-based approach to throughput collapse and starvation in data centers," *Computer Networks*, vol. 56, no. 14, pp. 3333–3346, 2012.

[7] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification (ecn) to ip," tech. rep., RFC 2481, January, 1999.

[8] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter tcp (d2tcp)," *Proc. of SIGCOMM*, vol. 42, no. 4, pp. 115–126, 2012.

[9] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *Networking, IEEE/ACM Transactions on*, vol. 4, no. 3, pp. 375–385, 1996.

[10] P. E. McKenney, "Stochastic fairness queueing," in *Proc. of INFOCOM*, pp. 733–740, IEEE, 1990.

[11] Y. Kanizo, D. Hay, and I. Keslassy, "The crosspoint-queued switch," in *INFOCOM*, pp. 729 –737, april 2009.

[12] "Cisco catalyst 4500-x series fixed 10 gigabit ethernet aggregation switch data sheet." http://www.cisco.com/en/US/prod/collateral/switches/ps10902/ps12332/data_sheet_c78-696791.html.

[13] K. Fall and K. Varadhan, "The network simulator (ns-2)," *URL: http://www. isi. edu/nsnam/ns*, 2007.

[14] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "Tcp selective acknowledgment options," tech. rep., RFc 2018, October, 1996.

[15] P. Goli and V. Kumar, "Performance of a crosspoint buffered atm switch fabric," in *INFOCOM*, pp. 426–435, IEEE, 1992.

[16] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "Vl2: a scalable and flexible data center network," in *Proc. of SIGCOMM*, vol. 39, pp. 51–62, ACM, 2009.

[17] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. of IMC*, pp. 267–280, ACM, 2010.

[18] Altera Corp., *Stratix V Device Handbook*, 2012.

[19] K. Saban, "Xilinx stacked silicon interconnect technology delivers breakthrough fpga capacity, bandwidth, and power efficiency," *Xilinx White paper: Vertex-7 FPGAs*, 2013.

[20] C. D. C. Infrastructure, "2.5 design guide," 2007.

[21] "IBM rack switch." http://www-03.ibm.com/systems/networking/switches/rack.html.

[22] A. Dixit, P. Prakash, Y. C. Hu, and R. R. Kompella, "On the impact of packet spraying in data center networks," in *Proc. of INFOCOM*, pp. 2130–2138, IEEE, 2013.