Designing Buffer Capacity of Crosspoint-Queued Switch

Guo Chen, Dan Pei*, Youjian Zhao, and Yongqian Sun

Department of Computer Science and Technology, Tsinghua University, Beijing
{chen-g11,sunyq12}@mails.tsinghua.edu.cn,
{peidan,zhaoyoujian}@tsinghua.edu.cn

Abstract. We use both theoretical analysis and simulations to study crosspoint-queued(CQ) buffer size's impact on CQ switch's throughput and delay performance under different traffic models, input loads, and scheduling algorithms. In this paper, 1) we present an exact closed-form formula for the CQ switch's throughput and a non-closed-form but convergent formula for its delay using static non-work-conserving random scheduling algorithms with any given buffer size under independent Bernoulli traffic; 2) we show that the above results can serve as a conservative guidance on deciding the needed buffer size in pure CQ switches using work-conserving algorithms such as random, under independent Bernoulli traffic. Furthermore, our simulation results under real-trace traffic show that simple round-robin and random work-conserving algorithms can achieve quite good throughput and delay performance with feasible crosspoint buffer size. Our work reveals the impact of buffer size on CQ switches' performance and provides a theoretical guidance on designing the buffer size in pure CQ switch, which is an important step towards building ultra-high-speed switching fabrics.

1 Introduction

As content-rich Internet applications such as video streaming, audio streaming, file sharing, live video/voice call, become more and more popular, the demands for higher backbone bandwidth have grown extremely fast. For the increasingly growing link rate, the switching fabric in core routers only has a very short time (e.g 5.12ns for a 64 bytes long packets to be transmitted in a 100Gbps link) to schedule and send out a packet. Thus, how to reduce the scheduling time in switch fabrics becomes a huge challenge. Most of the previous switch fabrics, including input-queued (IQ) switch [1, 2], combined-input-and-crosspoint-queued (CICQ) switch [3, 4] and multi-stage switching fabrics such as [5, 6] allocate major buffers at linecards instead of switch fabrics. To avoid packets conflicting and damaged at the switch fabrics, every scheduling cycle in these approaches mandate a round-trip communication between the linecards and

* Corresponding author.

C.-H. Hsu et al. (Eds.): NPC 2014, LNCS 8707, pp. 35–48, 2014.

the switch module, which limits the switching speed. As [7] shows, in order to reduce power consumption, linecards and switch module in modern core routers are often placed in different racks with distance from a few meters to up to 60 meters. Assuming the length of inter-rack cable is 2 meters long and the propagation speed is 2×10^8 m/s [7], the back-of-envelope calculation shows that each scheduling cycle has at least a 20 ns delay caused by round-trip communication, which becomes a bottleneck for a high-speed switch.

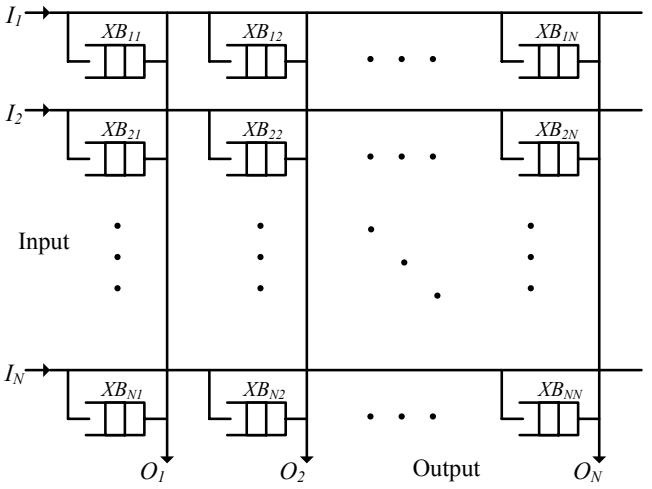

Fig. 1. The CQ switch model

Recently, to overcome above limitations, both academia [8–11] and industry [12] have a growing interest in crosspoint-queued (CQ) switch (illustrated in Fig. 1). Packets are buffered only at each crosspoint using on-chip memory thus switch decision can be made locally by each output scheduler independently, solely based on the conditions of the buffers in the same column as the output scheduler. Therefore, the scheduling algorithms can be made without communications between linecards and switch module, which greatly reduces the scheduling delay.

Although CQ switch was considered to be hard to implement due to the scarcity of on-chip memory many years ago, it has become feasible to implement CQ switch fabrics with large crosspoint buffers by modern technology. Recently, [8] revisited the CQ switch and proved the feasibility using semiconductor integration technology at that time by showing a crosspoint buffer could store over three mega bits packets for a switch with more than a hundred of ports.

Despite the great promise of CQ Switch, there lacks a clear understanding of how to design the crosspoint buffers to meet the switch fabric's overall performance requirement. In this paper, we take a first step towards this direction. We focus on understanding the impact of buffer size on CQ switches'

performance, because the on-chip memory resource used by CQ switch for crosspoint buffers is finite and very precious. Previously, [8] presents an accurate analytical model for pure CQ's throughput and delay, assuming a *buffer size of one* and independent and identically distributed (i.i.d.) uniform Bernoulli traffic. However, for larger buffer sizes, the authors introduce only approximate analytical models and simulation results for only throughput. No theoretical or simulating analysis on the switch's average delay has been presented for crosspoint buffer size larger than one. Later on, several papers [9–11] used *simulations* to study pure CQ switch's performance for buffers larger than one under traffic models, such as uniform Bernoulli and bursty.

Compared to these related works, this paper is the first one to provide an *exact* theoretical performance formula for pure CQ switch's *both throughput and delay performance* with buffer size *one and larger* under any independent Bernoulli (both uniform or non-uniform) traffic. The contributions of this paper are summarized as follows:

- To the best of our knowledge, this paper presents the first *exact* closed-form formula of the CQ switch's throughput with any given buffer size, and presents the first *exact* non-closed-form (but convergent) formula of the delay with any buffer size, both under independent Bernoulli traffic using static non-work-conserving random scheduling algorithm.
- Through mathematic proof as well as the comparison between theoretical value and simulation results, we show that theoretical value can serve as a conservative guidance (a loose performance lower bound) for designing buffer sizes of a CQ switch using work-conserving scheduling algorithms.
- Our real-trace simulation results show that with simple work-conserving algorithms, CQ switch is able to reach a very good performance with moderate memory resource consumption, which shows its feasibility in practical use.

Our work reveals the impact of buffer size on CQ switches' performance and provides a theoretical and conservative guidance on deciding the needed buffer size in pure CQ switches, which is an important step towards building ultra-high-speed switching fabrics. Having a better understanding on CQ switches is also an important step towards building multi-stage and multi-plane switching fabrics of large capacity. How to scale up CQ switches to a larger self-sufficient switching fabric is worthy of further studying, which is beyond the scope of this paper.

The rest of this paper is organized as follows. In Section 2, we introduce the CQ switch model and give some necessary definitions and notes. Next, in Section 3, we analyse the throughput and delay performance of CQ switch. We verify our analysis by simulations in Section 4. Finally, we conclude our paper and discuss the future work in Section 5.

2 The Crosspoint-Queued Switch

In this section, we briefly describe the CQ switch model and provide some fundamental definitions used in the rest of our paper.

2.1 The CQ Switch Model

Consider an $N \times N$ CQ switch shown in Fig. 1, the i-th input and i-th output are denoted by I_i and O_i respectively. XB_{ij} represents the crosspoint buffer between I_i and O_j, where $i, j = 1, \ldots, N$. We assume that time is slotted and all the packets are segmented into fixed cells before being sent into the switch, and all the internal and external links of the CQ switch have the same capacity of transferring one cell per time slot. We follow this assumption in the rest of the paper. XB_{ij} has the size of L_{ij} cells.

At the beginning of a time slot, there is one cell or none arriving at each input. If there is a cell arriving at input i heading to the output j at the start of a time slot, it is buffered in XB_{ij} in a first-in-first-out (FIFO) manner if the buffer is not full. The cell will be dropped in the case of that XB_{ij} is full. Within the same slot, the scheduler of each output independently selects one of the buffers in its column according to a certain scheduling algorithm, and sends the head of line (HOL) cells out of the switch through the output if the selected buffer is not empty. If an empty buffer is selected, there will be no cell scheduled out through this output in this time slot. Note that the departure steps at different output schedulers run in parallel.

2.2 Definitions

First we give some definitions that are related to the performance of a switch fabric.

Definition 1. The *throughput* of a switch fabric is the ratio of the amount of cells traversed the switch to the amount of cells arrived at the switch as time goes to infinity. We define TP as the throughput of the switch.

Definition 2. The *loss rate* of a switch fabric is the ratio of the amount of cells dropped by the switch to the amount of cells arrived to the switch as time goes to infinity. We define LR as the loss rate of the switch.

Proposition 1. *For a switch fabric which has finite buffers, the throughput of the switch equals* 1 *minus the loss rate of the switch, if the average cell arrival rate to the switch is greater than zero.*

Proof. Assume the total buffers of the switch can contain L cells. Let λ denotes the average arrival rate at all the inputs of the switch as time goes to infinity, $L^*(n)$ denotes the total amount of cells in the buffers of the switch at time slot n, hence $L^*(n) \leq L$. We define C_a, C_l, C_t as the total cells arrived, lost and traversed at the switch as time goes to infinity respectively. Obviously, $C_a = \lim_{n\to\infty} \lambda n$, $C_l = \lim_{n\to\infty} \lambda n \cdot LR$, then we have $C_t = \lim_{n\to\infty}(\lambda n - \lambda n \cdot LR - L^*(n))$. Thus, the throughput equals

$$TP = \frac{C_t}{C_a} = \lim_{n\to\infty}(1 - LR - \frac{L^*(n)}{\lambda \cdot n}) = 1 - LR$$

□

Definition 3. The *delay* of a switch fabric is the average delay of all the packets that traversed the switch as time goes to infinity. We define DL as the delay of the switch.

Then, we give some definitions related to the scheduling algorithms.

Definition 4. A scheduling algorithm is called *work-conserving* if, using this scheduling algorithm, any output of the switch will always be busy if at least one buffer destined to this output is not empty. Otherwise, the scheduling algorithm is called *non-work-conserving*.

Definition 5. A scheduling algorithm is called *static* if the rule of scheduling remains the same regardless of the system's state. Otherwise, it is called *dynamic*.

Definition 6. A static random scheduling algorithm is called *fair* if at each time slot, a column's output scheduler randomly (with the same probability) selects one of the crosspoint to send out its HOL cell.

At the last, we present a definition related to the arrival traffic.

Definition 7. The traffic at an input is said to be *uniform*, if each cell arriving at the input has the equal probability of going to any output of the switch.

3 Performance Analysis with Different Buffer Size

We focus on giving a theoretical throughput and delay calculation expression according to the buffer size in this section.

Consider the CQ switch model shown in Fig. 1. We assume the cell arrivals at each input are governed by independent Bernoulli process and with fixed probability heading to each output. Each output scheduler uses a static non-work-conserving random scheduling algorithm. We use the following notations:

$\rho_i \triangleq$ the Bernoulli parameter of the cell arrival process in input I_i.
$a_{ij}^k \triangleq$ the probability of k cells arrived at XB_{ij} in a given time slot. $k = 0, 1$.
$d_{ij} \triangleq$ the probability of any cell arrived at I_i heading to the output O_j. $\sum_{j=1}^{N} d_{ij} = 1$ and $0 \leq d_{ij} \leq 1$ for $i = 1, \ldots, N$.
$s_{ij} \triangleq$ the probability of crosspoint buffer XB_{ij} being selected by output O_j. $\sum_{i=1}^{N} s_{ij} = 1$ and $0 < s_{ij} < 1$ for $j = 1, \ldots, N$.

First, we present a formal description of a scheduling cycle in a time slot as follows:

- *Arrival Step:* At the beginning of a time slot, for input i, there exists a probability of ρ_i that one cell will arrive, and a probability of $1 - \rho_i$ that no cell will arrive. The cell arrived at input I_i has the probability of d_{ij} heading to the output O_j. Successive cells and cell arrivals at different inputs are independent.

– *Departure Step:* Within the same slot after the arrival step, each output scheduler picks a crosspoint buffer out of all the buffers in its column with a static non-work-conserving random scheduling algorithm. For output O_j, it selects crosspoint buffer XB_{ij} with the probability of s_{ij}, and schedules the HOL cell out of the switch if the selected buffer is not empty. Otherwise, no cells are transmitted through O_j in this time slot. Each output schedules cells independently and in parallel.

Let L_{ij} denotes the capacity of crosspoint buffer XB_{ij} in cells, we assume that $L_{ij} = L(i, j = 1, 2, \ldots, N)$ for the ease to present. It means that all the crosspoint buffers have the same capacity of L cells. We perform our analysis on a particular crosspoint buffer XB_{ij} without loosing generality.

We assume random variable A_{ij}, A_i, A to be the number of cells arrived to XB_{ij}, input I_i and the whole switch during a given time slot respectively. According to the conditions given above, the value of A_{ij} can only be 0 or 1. Recall that a_{ij}^k denotes the probability that k cells arrive at XB_{ij} in a time slot, then

$$\begin{aligned} a_{ij}^0 &= P\{A_{ij} = 0\} = 1 - \rho_i \cdot d_{ij} \\ a_{ij}^1 &= P\{A_{ij} = 1\} = \rho_i \cdot d_{ij} \\ a_{ij}^k &= P\{A_{ij} = k\} = 0 \quad k \neq 0, 1 \end{aligned} \tag{1}$$

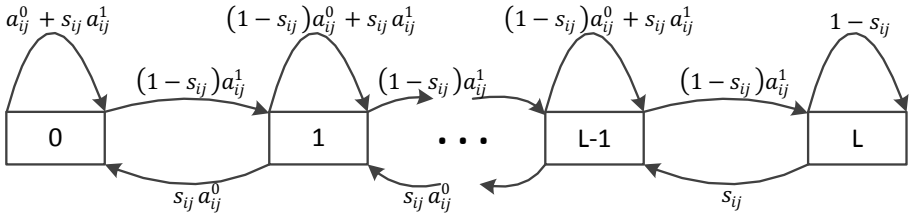

Fig. 2. The Quasi-birth-death state transition diagram for XB_{ij}

We define random variable $Q_{ij}(m)$ as the cells in XB_{ij} at the end of time slot m. According to the conditions stated before, we can find that $Q_{ij}(m)$ can be modeled as a discrete-time Quasi-birth-death process as Fig. 2 shows. The transition diagram can be interpreted as follows:

– The transitions from state l to $l + 1$ mean the probability that there is an arrival at the buffer and the buffer is not selected by the output scheduler.
– Transitions from state l to itself are calculated under 3 different conditions. 1) While $l = 0$, it equals the probability of one arrival and one departure plus with the probability of no arrival. 2) While $l = 1, \ldots, L - 1$, it equals the probability of one arrival and one departure plus with the probability of

no arrival and no departure. 3) While $l = L$, it equals the probability of one arrival and no departure (the cell will be dropped in the arrival step when the buffer is full) plus with the probability of no arrival and no departure.

– Transitions from state l to state $l-1$ are calculated under 2 different conditions. 1) While $l = 1, \ldots, L-1$, it equals the probability of no arrival and one departure. 2) While $l = L$, it equals the probability of the buffer being selected (the buffer length will still be L before the departure step begins as the cell will be dropped in the arrival step when the buffer is full).

Let Q_{ij} denotes the steady-state queue length of XB_{ij}, according to the formula of the steady-state probabilities of discrete-time Quasi-birth-death process [13], we can get the stead-state queue length distribution as follows:

$$\begin{aligned}
\eta_{ij}^0 &= \frac{1}{1 + \sum_{l=1}^{L-1} \left(\frac{(1-s_{ij})a_{ij}^1}{s_{ij}a_{ij}^0}\right)^l + a_{ij}^0 \left(\frac{(1-s_{ij})a_{ij}^1}{s_{ij}a_{ij}^0}\right)^L} \\
\eta_{ij}^l &= \eta_{ij}^0 \left(\frac{(1-s_{ij})a_{ij}^1}{s_{ij}a_{ij}^0}\right)^l, \quad l = 1, \ldots, L-1 \\
\eta_{ij}^L &= \eta_{ij}^0 a_{ij}^0 \left(\frac{(1-s_{ij})a_{ij}^1}{s_{ij}a_{ij}^0}\right)^L
\end{aligned} \tag{2}$$

where η_{ij}^l defines the steady-state probability of XB_{ij}'s length equals l, i.e $Q_{ij} = l$.

So far, we have derived the steady-state probability distribution of XB_{ij}'s length. Next, we will use these results to analyze the throughput and delay of the CQ switch.

3.1 Throughput Analysis

Obviously, the probability of a cell arrived at XB_{ij} being dropped equals the probability of XB_{ij} being full, i.e., η_{ij}^L for the steady-state.

We define the random variable D_{ij}, D_i and D as the number of cells dropped at XB_{ij}, input I_i and the whole switch during a given time slot at the steady-state. Obviously, D_{ij}, D_i could only be 0 or 1. Then, we can get the probability of a cell arrived at input I_i being dropped in a time slot as

$$P\{D_i = 1 | A_j = 1\} = \sum_{j=1}^{N} d_{ij}\eta_{ij}^L \tag{3}$$

The above equation comes from the fact that the probability of a cell arrived at input i being dropped in a given time slot equals the sum of probabilities that a cell arrived at I_i being dropped at any crosspoint buffer of this line.

Further, we have the expectation of the dropped cells at I_i during a time slot as following

$$E(D_i) = \rho_i \cdot P\{D_i = 1 | A_j = 1\} = \rho_i \left(\sum_{j=1}^{N} d_{ij} \eta_{ij}^{L} \right) \tag{4}$$

Thus, we get the expectation of dropped cells at the whole switch in a given time slot as

$$E(D) = E(\sum_{i=1}^{N} D_i) = \sum_{i=1}^{N} \rho_i \left(\sum_{j=1}^{N} d_{ij} \eta_{ij}^{L} \right) \tag{5}$$

Then, we get the loss rate of the CQ switch as follows

$$LR = \frac{E(D)}{E(A)} = \frac{\sum_{i=1}^{N} \rho_i \left(\sum_{j=1}^{N} d_{ij} \eta_{ij}^{L} \right)}{\sum_{i=1}^{N} \rho_i} \tag{6}$$

where random variable A denotes the number of cells arrived at the switch during a time slot and $E(A)$ means the expectation of A.

Therefore, from Proposition 1 we can acquire the closed-form formula of the throughput of the switch as

$$TP = 1 - LR = 1 - \frac{\sum_{i=1}^{N} \rho_i \left(\sum_{j=1}^{N} d_{ij} \eta_{ij}^{L} \right)}{\sum_{i=1}^{N} \rho_i} \tag{7}$$

3.2 Delay Analysis

Then, we analyze the average delay of CQ switch. Similarly, we begin with focusing on a certain crosspoint buffer XB_{ij}.

Let random variable W_{ij}, W_i denotes the time slots a cell spent in the steady-state (i.e., *delay*) in XB_{ij} and input I_i respectively. We assume that at the time a cell arriving at XB_{ij}, the buffer length $Q_{ij} = l (0 \leq l < L)$ and the cell has spent n time slots in XB_{ij}. We only consider the delay of a cell while it is not dropped by the switch because the delay of dropped cells is meaningless. Thus, we can have

$$P\{W_{ij} = n | Q_{ij} = l\} = C_n^{n-l} \left(1 - s_{ij}\right)^{n-l} \left(s_{ij}\right)^{l+1} \tag{8}$$

where $n = l, l+1, \ldots, \infty$. This equation denotes that the probability of a cell's delay $W_{ij} = n$ equals the probability of the buffer having been selected l times during n slots to move the cell to the HOL and the buffer being selected after n slots to schedule out the cell.

Thus, the steady-state probability of a cell's delay being n time slots in XB_{ij} equals

$$P\{W_{ij} = n\} = \begin{cases} \sum_{l=0}^{n} \left(\eta_{ij}^{l} P\{W_{ij} = n | Q_{ij} = l\} \right), & 0 \leq n \leq L-1 \\ \sum_{l=0}^{L-1} \left(\eta_{ij}^{l} P\{W_{ij} = n | Q_{ij} = l\} \right), & n > L-1 \end{cases} \tag{9}$$

Using the results of equation (2) and (1), we transform the above equation into

$$P\{W_{ij}=n\}=\begin{cases}\eta_{ij}^{0}s_{ij}\left(\frac{1-s_{ij}}{a_{ij}^{0}}\right)^{n}, & 0\le n\le L-1\\ \eta_{ij}^{0}s_{ij}(1-s_{ij})^{n}\sum_{l=0}^{L-1}\left[C_{n}^{n-l}\left(\frac{a_{ij}^{1}}{a_{ij}^{0}}\right)^{l}\right], & n>L-1\end{cases} \tag{10}$$

Then, using the equation above, we can derive the following formula of the mean delay of a cell in XB_{ij} which is not dropped as follows

$$E\{W_{ij}\}=\sum_{n=0}^{\infty}nP\{W_{ij}=n\} \tag{11}$$

Therefore, the mean delay of a cell coming into I_i which is not dropped equals that

$$E\{W_i\}=\sum_{j=1}^{N}d_{ij}E\{W_{ij}\} \tag{12}$$

Thus, we acquire the delay of the switch (i.e the average delay of all the packets that traversed the switch) from the following equation

$$DL=\frac{\sum_{i=1}^{N}\rho_i E\{W_i\}}{(\sum_{i=1}^{N}\rho_i)\cdot TP} \tag{13}$$

Although the above formula of the switch's delay is not closed-form, we have proven its convergency. The proof is omitted here due to the space limitation.

So far, we derive the precise expression of the CQ switch's throughput and delay using static non-work-conserving random scheduling algorithms. Naturally, an appropriate work-conserving scheduling algorithm will lead to a better performance compared to the non-work-conserving random scheduling algorithm that we use, to perform theoretical analysis. Next, we briefly prove that under independent Bernoulli traffic, work-conserving random scheduling (randomly selecting a crosspoint-buffer from all the non-empty ones) performs better than static non-work-conserving random scheduling algorithm both in throughput and average delay.

Theorem 1. *Under same independent Bernoulli traffic, a CQ switch using work-conserving random (WCRand) scheduling algorithm has a higher throughput and lower average delay than using non-work-conserving (nWCRand) fair random scheduling algorithm.*

Proof. Similarly, we could also build a discrete-time Quasi-birth-death diagram for WCRand as Fig. 2 shows. As stated before, for fair nWCRand, we have $s_{ij}=\frac{1}{N}$ in each steady state of queue length. Unlike nWCRand, s_{ij} of WCRand between different states in Fig. 2 are not the same. Let $s'_{ij}(m)$ denotes the probability of crosspoint buffer XB_{ij} being selected by output O_j using WCRand

in state m, and $s^*_{ij} = max\{s'_{ij}(m), 0 \leq m \leq L\}$. η'^{l}_{ij} defines the steady-state probability of XB_{ij}'s length equals l using WCRand. Because WCRand randomly selects a crosspoint-buffer from all the non-empty ones in each time slot, we can have

$$s^*_{ij} \geq \frac{1}{N} = s_{ij} \tag{14}$$

Thus, according to the formula of the steady-state probabilities of discrete-time Quasi-birth-death process, we can get $\eta'^{L}_{ij} \leq \eta^{L}_{ij}$. Therefore, it can be derived that WC-Rand has a higher throughput than nWCRand using equation (7). Also, from equation (11-13), we can easily get that WCRand has a lower average delay than nWCRand. □

Similarly, if we use *the frequency of a crosspoint queue being selected by work-conserving Round-Robin (WCRR) algorithm* to approximate *the probability of a crosspoint queue being selected by WCrand algorithm*, we can prove that WCRR also performs better than nWCRand. Furthermore, it is intuitive that longest-queue-first (LQF) scheduling has the highest throughput. A strict proof for these 2 work-conserving algorithms is beyond the scope of this paper. Later, we will show by our simulations that the above theoretical analysis provide an appropriate lower-bound for a CQ switch's performance using work-conserving algorithms.

4 Verification of Analysis and Real Trace Simulations

In this section, we first present simulation results under both uniform and non-uniform Bernoulli traffic to verify our former theoretical analysis in Section 4.1. We consider four scheduling algorithms in our simulations: nWCRand, WCRand, WCRR and LQF. We calculate the theoretical value (TV) of the loss rate and the delay of nWCRand scheduling algorithm, according to the former results we got under both uniform and non-uniform Bernoulli traffic. Various of simulations have been done under different loads and using CQ switches with different port numbers. All these results have verified our former analysis. Due to space limitation, we just present the results of 16×16 CQ under a heavy load of 0.95. Each simulation run was conducted for 10^9 time slots.

Secondly in Section 4.2, we present simulation results of a 16×16 switch fabric under real-trace traffic using the 4 work-conserving scheduling algorithms mentioned above. It's shown that with work-conserving algorithms, the CQ switch is able to reach a good performance with moderate memory resource consumption. Our data consists of two parts from CAIDA [14], 2 1-minute traces from 10Gbps links , one at San Jose and another from Chicago. All the packets are fragmented into 64 bytes long cells before sent into the switch fabric, and the time slot of the switch is set to be 51.2ns according to the transmission time of a cell on a 10Gbps link. We divide a 60 seconds trace into 16 equal size segments for 16 inputs. The destination port of each packet is set as the hash value of destination IP address. The traffic distribution under this situation is

not uniform but highly skewed and bursty. We believe this is similar to the real condition of the Internet. Approximately 1.7×10^7 packets with total length of 1.15×10^{10} bytes are sent into the switch fabric during each experimentation. We only present the simulation result of San Jose trace due to space limitation. The results for Chicago trace are similar.

4.1 Verification of Performance Analysis

From Fig. 3 and 4 we can see that, the results of non-work-conserving random scheduling algorithm are almost identical as the theoretical results we derived before, under both uniform and non-uniform traffic. Investigation into the slight difference at the right end of the curves shows that the difference is due to the computer random number generation are not 100% random. Under uniform Bernoulli traffic with heavy input load of 95% as we can see in Fig. 3(a), with crosspoint buffer size of 256 (such buffer size is easy to implement with modern semiconductor technology[8]), the loss rate of nWCRand can be as low as 10^{-7} using the theoretical results we got before. Such a loss rate is good enough for a lot of switch fabrics design and provide loose performance lower bound. Our results also show a simple work-conserving algorithm like Round-robin and Random can reach the same performance with only buffer size of 32 cells, and the theoretical results could serve as a loose performance lower bound for them. Using a more elaborated scheduling algorithm such as LQF, no packets are lost during the 10^{-9} time slots simulation with only buffer size of 16. As for the average delay, our theoretical analysis shows that with buffer size of 64, a CQ switch can have a stable average delay about 10^{-5} seconds using nWCRand, which is shown in Fig. 3(b). While using work-conserving algorithms, the average delay is much lower down to less than 10^{-6} seconds.

Similarly, under non-uniform traffic as shown in Fig. 4, our analytic results are also verified and effectively provide loose performance lower bound to work-conserving algorithms. ω in the picture defines the unbalanced probability (refer

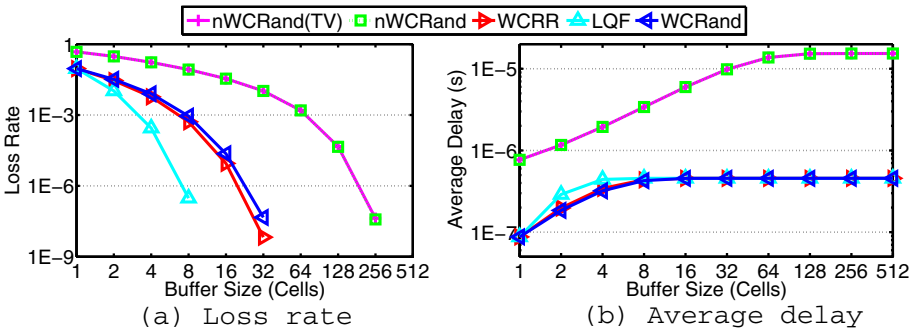

Fig. 3. Loss rate and average delay of a 16×16 CQ switch under uniform Bernoulli traffic with $\rho = 0.95$

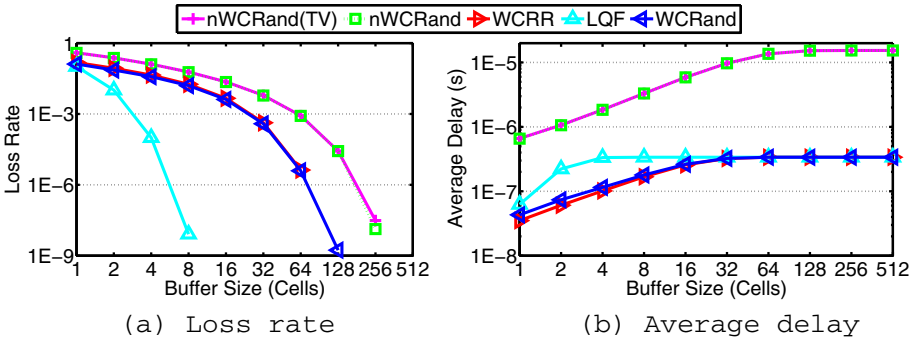

Fig. 4. Loss rate and average delay of a 16×16 CQ switch under non-uniform Bernoulli traffic with $\rho = 0.95$ and $\omega = 0.5$

to [3]) and $\omega = 0.5$ means the traffic is extremely non-uniform. Accordingly, we set the selecting probability parameters of the nWCRand used in this simulation as the same as the load unbalanced probability. The results are very similar to the uniform traffic, except nWCRand and WCRR having a much higher loss rate than uniform traffic because of the blindness to the traffic distribution of their scheduling manner. On the contrary, LQF has the ability to adjust to the imbalance.

4.2 Simulations under Real-Trace

Fig. 5 shows the result of a 16×16 CQ switch under real-trace traffic. The loss rate in Fig. 5(a) refers to the packet loss rate. Once a cell of a packet is dropped, the packet is counted as lost in the switch. We can see that a CQ switch only using crosspoint memory can reach a good performance with simple work-conserving scheduling algorithms. In Fig. 5(a) we can found that the switch can has a loss rate down to 10^{-6} with buffer size of 64 using LQF. A simple Round-robin or Random scheduling is able to reach the same performance with buffer size of

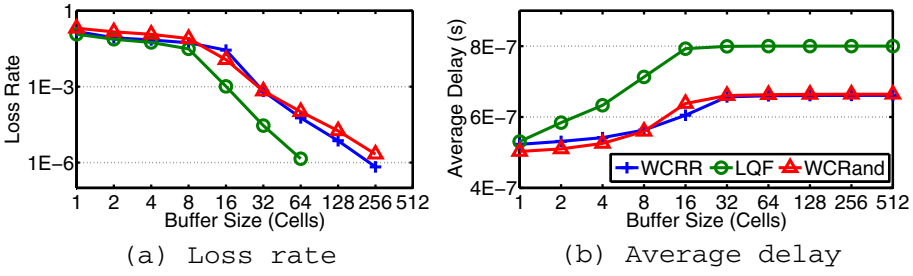

Fig. 5. Loss rate and average delay of a 16×16 CQ switch under real-trace traffic

256, which is totally within the capability of modern chip technology. Also, the delay performance shown in Fig. 5(b) is very good. The delay here refers to the average packet delay. We can see that WCRR and WCRand have a better delay performance than LQF. It's due to that, starvation, which greatly increases the delay, may happen using LQF algorithm.

This result under real-trace traffic demonstrates that a CQ switch with such scale can reach a very good performance with feasible crosspoint buffer size. Thus a self-sufficient CQ switch is exactly suitable for ultra-high-speed link in practical use.

5 Conclusion

This paper reveals the impact of buffer size on CQ switches performance and provides a theoretical guidance on designing the buffer size in pure CQ switch. Also, we show that CQ is a promising building block for high linerate switch fabrics. As a next step, we plan to actually design ultra-high-speed and large-port-number switch fabrics with multi-plane or mutli-stage structure, using CQ as building blocks to scale up. We also plan to design scheduling algorithms with performance better than round-robin algorithm, random or LQF presented in this paper.

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 Kuan Cheng for his useful discussion on the mathematical demonstration. Also, we thank Zhiyan Zheng for the packet trace of real operational network that he provided us. At last, we thank Juexing Liao for her proofreading on this paper.

References

[1] McKeown, N.: The islip scheduling algorithm for input-queued switches. IEEE/ACM Transactions on Networking 7(2), 188–201 (1999)
[2] Li, Y., Panwar, S., Chao, H.: On the performance of a dual round-robin switch. In: INFOCOM 2001, vol. 3, pp. 1688–1697 (2001)
[3] Rojas-Cessa, R., Oki, E., Jing, Z., Chao, H.: Cixb-1: combined input-one-cell-crosspoint buffered switch. In: HPSR 2001, pp. 324–329 (2001)
[4] He, S.M., Sun, S.T., Guan, H.T., Zheng, Q., Zhao, Y.J., Gao, W.: On guaranteed smooth switching for buffered crossbar switches. IEEE/ACM Transactions on Networking 16(3), 718–731 (2008)
[5] Oki, E., Jing, Z., Rojas-Cessa, R., Chao, H.J.: Concurrent round-robin-based dispatching schemes for clos-network switches. IEEE/ACM Trans. on Networking 10(6), 830–844 (2002)

[6] Iyer, S., Awadallah, A., McKeown, N.: Analysis of a packet switch with memories running slower than the line-rate. In: INFOCOM 2000, vol. 2, pp. 529–537. IEEE (2000)
[7] Abel, F., Minkenberg, C., Iliadis, I., Engbersen, T., Gusat, M., Gramsamer, F., Luijten, R.P.: Design issues in next-generation merchant switch fabrics. IEEE/ACM Trans. Netw. 15(6), 1603–1615 (2007)
[8] Kanizo, Y., Hay, D., Keslassy, I.: The crosspoint-queued switch. In: INFOCOM 2009, pp. 729–737. IEEE (2009)
[9] Radonjic, M., Radusinovic, I.: Impact of scheduling algorithms on performance of crosspoint-queued switch. Annals of Telecommunications - Annales Des Télécommunications 66, 363–376 (2011)
[10] Radusinovic, I., Radonjic, M., Simurina, A., Maljevic, I., Veljovic, Z.: A new analytical model for the cq switch throughput calculation under the bursty traffic. AEU - International Journal of Electronics and Communications 66(12), 1038–1041 (2012)
[11] Cao, Z., Panwar, S.S.: Efficient buffering and scheduling for a single-chip crosspoint-queued switch. In: Proceedings of the Eighth ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS 2012, pp. 111–122. ACM, New York (2012)
[12] Cisco crs carrier routing system 16-slot line card chassis system description. Cisco Systems, Inc. (2012)
[13] Trivedi, K.: Probability and statistics with reliability, queuing, and computer science applications. Wiley, New York (2002)
[14] Anonymized 2013 internet traces, `https://data.caida.org/datasets/passive-2013/`