IEICE TRANS. INF. & SYST., VOL.E98–D, NO.12 DECEMBER 2015

PAPER *Special Section on Parallel and Distributed Computing and Networking*

Novel High Performance Scheduling Algorithms for Crosspoint Buffered Crossbar Switches

Xiaoting WANG[†], ***Nonmember***, **Yiwen WANG**[†a)], ***Member***, **Shichao LI**[†], ***and*** **Ping LI**[†], ***Nonmembers***

SUMMARY The crossbar-based switch fabric is widely used in today's high performance switches, due to its internally nonblocking and simply implementation properties. Usually there are two main switching architectures for crossbar-based switch fabric: internally bufferless crossbar switch and crosspoint buffered crossbar switch. As internally bufferless crossbar switch requires a complex centralized scheduler which limits its scalability to high speeds, crosspoint buffered crossbar switch has gained more attention because of its simpler distributed scheduling algorithm and better switching performance. However, almost all the scheduling algorithms proposed previously for crosspoint buffered crossbar switch either have unsatisfactory scheduling performance under non-uniform traffic patterns or show poor service fairness between input traffic flows. In order to overcome the disadvantages of existing algorithms, in this paper we propose two novel high performance scheduling algorithms named MCQF_RR and IMCQF_RR for crosspoint buffered crossbar switches. Both algorithms have a time complexity of $O(\log N)$, where N is the number of input/output ports of the switch. MCQF_RR takes advantage of the combined weight information about queue length and service waiting time of input queues to perform scheduling. In order to further reduce the scheduling complexity and make it feasible for high speed switches, IMCQF_RR uses the compressed queue length information instead of original queue length information to schedule cells in input VOQs. Simulation results show that our novel scheduling algorithms MCQF_RR and IMCQF_RR can demonstrate excellent delay performance comparable to existing high performance scheduling algorithms under both uniform and non-uniform traffic patterns, while maintain good service fairness performance under severe non-uniform traffic patterns.

key words: *crosspoint buffered crossbar switches, scheduling, delay, fairness*

1. Introduction

With the new network applications and services increasing continually, Internet traffic tends to grow rapidly. To keep pace with the growth demand of Internet traffic, it is necessary to explore high-performance switches and routers. Many switch fabrics for switches and routers have been studied and implemented in the past academic researches [1]–[3]. One of the most important switch fabrics is crossbar-based architecture, because of its internally non-blocking property and simplicity. Since the output contention that traffic arrivals from different inputs ports may be simultaneously destined for the same output port, it's necessary to add buffering in crossbar-based switch fabric to avoid packets loss. Based on the location relationship between the buffers and the switch fabric, crossbar-based switch fabric can be classified into output queued (OQ) switch, input queued (IQ) switch, virtual output queued (VOQ) switch, combined input and output queued (CIOQ) switch and combined input-crosspoint queued (CICQ) switch.

For OQ switches, packets can only be buffered at the output ports, as shown in Fig. 1 (a). Once a packet arrives at the input, it can be immediately transferred to the output queues. OQ switches can provide excellent switching performance and Quality of Service (QoS) guarantees applying sophisticated scheduling algorithms [4], [5]. However, OQ switches become impractical due to the high switch fabric bandwidth and memory bandwidth requirements. For a N × N OQ switch, the switch fabric needs to operate at the speed of N times the line rate, in order to meet the situation when N input ports transfer packets to the same output simultaneously. Besides, the output buffer access speed requires N times the line rate. Thus, OQ switches are infeasible for high-speed and large-scale applications.

On the other hand, IQ switches have attracted more attention because of the low-bandwidth requirement and high scalability. The IQ structure is shown in Fig. 1 (b), where the packets can only be buffered at the input ports. The switch fabric just needs to operate at the line rate without internal speedup. However, IQ switches suffer from the head-of-line (HoL) Blocking problem which makes the switch throughput degrade to just 58.6% [6]. In order to overcome HoL blocking, it is well-known to use VOQ architecture [7]. Figure 1 (c) shows a VOQ switch, where each input port maintains a separate FIFO queue for each output port, and the packet destined for an output will be buffered in the corresponding VOQ. Because VOQ switches significantly improve the throughput performance upon IQ switches, they become more attractive. However, VOQ switches need centralized schedulers to resolve input contention and output contention [8]. For this purpose, many scheduling algorithms for VOQ switches have been proposed, which can be modeled as the matching problem in a bipartite graph [9]. These centralized scheduler collects all the input requests to select a match without contention from all possible matches [10], [13], [14]. Unfortunately, almost all these scheduling algorithms either have a high time complexity which is impractical for high-speed implementation [11] or couldn't achieve good switching performance [12].

The common solution to improve performance inefficiency of VOQ switches is to provide internal speedup

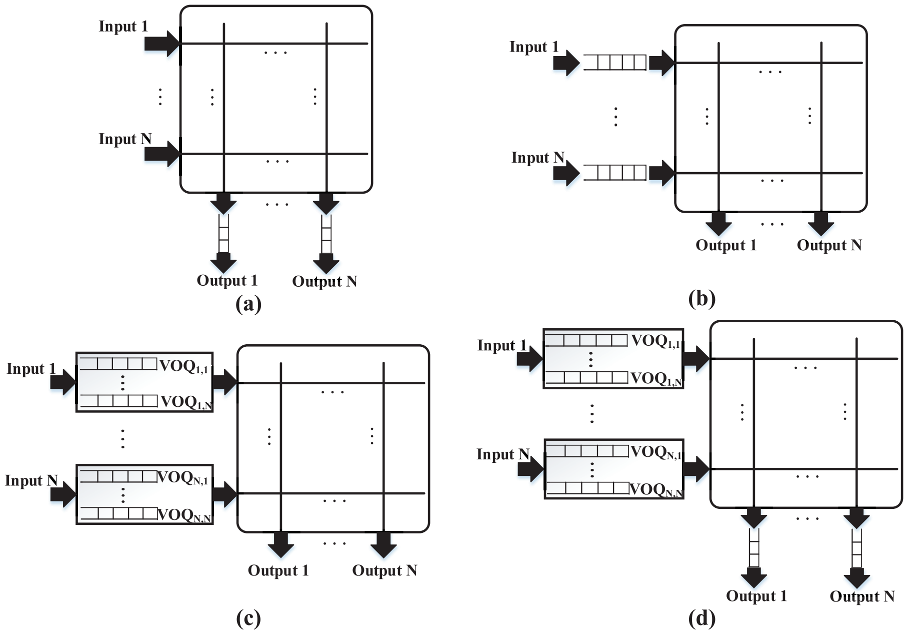
---

Manuscript received December 31, 2014.
Manuscript revised May 18, 2015.
Manuscript publicized September 15, 2015.

[†] The authors are with University of Electronic & Science Technology of China (UESTC), China.
a) E-mail: yiwen@uestc.edu.cn

DOI: 10.1587/transinf.2015PAP0010

Copyright © 2015 The Institute of Electronics, Information and Communication Engineers



(a) (b) (c) (d)

Fig. 1 Architectures of four crossbar-based switch fabrics. (a) OQ switches. (b) IQ switches. (c) VOQ switches. (d) CIOQ switches.

$f > 1$. As is shown in Fig. 1 (d), CIOQ switches require buffers at both the input ports and output ports to store packets, to make packets arrive at an output at a higher speed than the line rate. It has been proved that CIOQ switches with two times speedup can provide 100 percent throughput working with maximal matching algorithms [15]. However, there are several drawbacks for the CIOQ switch: (1) it also needs a complex centralized scheduler to resolve input contention and output contention, hence the time complexity is still high; (2) internal speedup reduces the time available for the scheduler and results in the increasement of implementation complexity; (3) internal speedup seriously affects the switch's power consumption. All these drawbacks limit the CIOQ switch's scalability at high-speed.

Crosspoint buffered crossbar switches, also called CICQ switches, have been considered as an alternative solution to VOQ switches to overcome the high scheduling complexity and improve the switching performance [16]. Figure 2 depicts a CICQ architecture, where a small buffer is added at each crosspoint in traditional crossbar fabric. Due to the introduction of crosspoint buffers, input contention and output contention of VOQ switches can be relaxed, which significantly simplifies the scheduling task. There are two independent and distributed scheduling phases involved in a CICQ switch: input scheduling and output scheduling. In input scheduling, each input selects a HoL packet from the input VOQs and delivers it to the corresponding crosspoint buffer. In output scheduling, each output selects a HoL packet from crosspoint buffers for the same output port and transfers it to the output port. The distributed scheduling algorithms for CICQ switches are different from the centralized scheduling algorithms for VOQ/CIOQ switches. Compared with VOQ switches, time complexity of distributed scheduling algorithm is dramatically reduced and the switching performance can be significantly improved. It was shown that CICQ switches can achieve better throughput and delay performance than internally bufferless crossbar switches [17]. Therefore, CICQ architecture is considered to be a good choice for high performance switches.

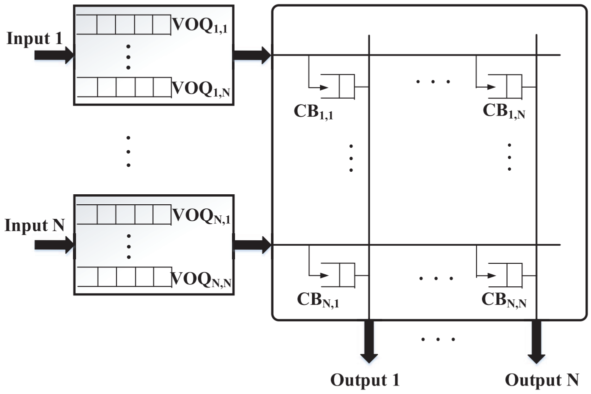


Fig. 2 Crosspoint buffered crossbar (CICQ) switches.

So far there have been much research on scheduling algorithms for the CICQ architecture [18]–[20]. Compared to conventional internally bufferless crossbar switches, these algorithms exhibit better throughput and delay performance. However, they either cannot achieve satisfactory performance under non-uniform traffic patterns [18], [20] or have poor service fairness performance [19]. Tracking Fair Quota Allocation (TFQA) [30] aims for achieving both good scheduling performance and fairnes by employing a Fair Quota Allocation (FQA) mechanism and Prioritized Dual Round Robin (PDRR) input scheduling scheme. However, for some non-uniform traffic TFQA is still unable to maintain stability at heavy input load, and the fairness is poor under severe non-uniform traffic patterns.

In this paper, we propose a novel high performance scheduling algorithm for CICQ architecture, named Most Critical Queue First-Round Robin (MCQF_RR), and evaluate its performance under various traffic patterns. It takes advantage of the combined weight information about queue occupancy and service waiting time of input VOQs to make efficient scheduling decisions. MCQF_RR can provide good fairness performance superior to existing high performance scheduling algorithm in [19] while maintain excellent delay performance under uniform and non-uniform traffic patterns. The time complexity of MCQF_RR is $O(\log N)$. In order to further reduce the scheduling complexity of MCQF_RR, we then propose the Improved Most Critical Queue First-Round Robin (IMCQF_RR) algorithm, which uses compressed weight information instead of original weight information in input scheduling. The implementability complexity of IMCQF_RR is greatly reduced comparing with MCQF_RR. Simulation results show that IMCQF_RR could also achieve low delay comparable to the existing efficient algorithms under uniform and non-uniform traffic patterns. In addition, IMCQF_RR can maintain great fairness performance under severe non-uniform traffic patterns.

The remainder of this paper is organized as follows: Section 2 presents an overview of the existing scheduling algorithms for CICQ switches and points out their advantages and limitations. In Sect. 3 and Sect. 4, we introduce our scheduling algorithms and make a detailed description and analysis for them. In Sect. 5, we present an experimental study about delay and fairness performance of different existing algorithms, and discuss the implementability briefly. In Sect. 6, we conclude the paper.

2. Related Work

Several important requirements must be taken into account to design an efficient scheduling algorithm for high performance switches [3]: (1) Scheduling efficiency. A good scheduling algorithm should provide high throughput and low delay; (2) Fairness. The algorithm should maintain good service fairness among different queues without queue starvation; (3) Scheduling complexity. The scheduling algorithm should have a low time complexity, otherwise the scalability in high-speed switches will be limited. There have been many scheduling algorithms proposed for CICQ architecture, which can be classified into three main categories.

One category is *Round Robin (RR)* based scheduling algorithm. In [18], RR_RR uses a simple pointer updating mechanism in input scheduling as well as output scheduling, which contributes good fairness performance. The time complexity of RR_RR is only *O(1)*, and it can achieve 100% throughput and good delay performance under uniform traffic. However, it performs unstably under non-uniform traffic. Many improvements for the performance inefficiency have been made, such as Round Robin with Adaptable-size Frame (RR-AF) [21], Differential Round Robin (DRR) [22] and Round Robin-Longest Queue Detecting (RR-LQD) [23]. The basic idea of them is based on different pointer updating mechanisms to ensure efficient service of long queues and service fairness of short queues. Although they show satisfactory performance with *O(1)* time complexity under uniform traffic, the performance under some non-uniform traffic patterns are still not good enough.

The other category is weight-based scheduling algorithm, such as Longest Queue First-Round Robin (LQF_RR) [19] and Oldest Cell First-Oldest Cell First (OCF_OCF) [16]. They take advantage of weight information of input VOQs such as queue length or waiting time of HoL cells in scheduling processes, to achieve high performance under both uniform and non-uniform traffic. It has been proved that through fluid model techniques, LQF_RR can provide 100 percent throughput for uniform input traffic and exhibit good stability for unbalanced traffic [19]. However, since LQF_RR always favors the most occupied VOQ in input scheduling, some queues with low occupancy will appear poor service fairness and even permanent queue starvation. In [16], OCF_OCF performs scheduling based on comparison of waiting time of HoL cells, which could improve the fairness performance. However it needs a complex time stamping mechanism and the implementation complexity will be largely increased. Both LQF_RR and OCF_OCF have a time complexity of $O(\log N)$ with hardware parallel comparison circuits. However, it is still time consuming to perform input scheduling due to complex comparison for weight information of input VOQs, which are represented as a large number of input values. For LQF_RR, it consumes at least $O(\log N \cdot \log B)$ time units to complete the comparison of queue lengths [24] in input scheduling, where N is the port number and B is the number of bits needed to represent the maximum queue length. For today's switches, the buffer size for an input port can be dozens or even hundreds of MB, that means the maximum queue length is much large, thus the scheduling time $O(\log N \cdot \log B)$ will be much longer.

In order to shorten the scheduling time, a class of *crosspoint buffer state* based scheduling algorithms were proposed, such as Most Critical Buffer First (MCBF) [20] and Shortest Crosspoint Buffer First (SCBF) [25]. The main idea of these algorithms is to perform scheduling based on comparison of crosspoint buffer occupancies instead of input VOQs' information. The time complexity of MCBF and SCBF are respectively $O(\log N)$ and $O(N \cdot \log N)$. Because the capacity of crosspoint buffer is smaller than input VOQ, input values of comparison circuit of the schedulers can be decreased compared with LQF_RR [19], hence the scheduling time will be shorter. However, they could not keep good performance for some non-uniform traffic patterns since they don't consider the influence of VOQ' length on stability of VOQs. CAF_PRMV was proposed to overcome the lack of performance without using input VOQs' information under non-uniform traffic [26]. It has a time complexity of $O(\log N)$ and short scheduling time like MCBF. However, CAF_PRMV has poor scalability for large-scale switches. In addition to crosspoint buffer occupancies returned from the buffered crossbar chip, CAF_PRMV needs more information about served cells in input VOQs exchanged between input ports and buffered crossbar chip. With the increasement of the number of switch ports and capacity of crosspoint buffer, the buffered crossbar chip must provide more pins resource for information exchange, which is usually unacceptable for realization.

3. The Most Critical Queue First-Round Robin Scheduling Algorithm (MCQF_RR)

In this section, we propose a Most Critical Queue First-Round Robin scheduling algorithm, named MCQF_RR for the CICQ architecture. The purpose is to improve the performance inefficiency of existing outstanding scheduling algorithms. As mentioned in Sect. 2, the well-known LQF_RR [19] can achieve relative higher throughput and delay performance under any traffic patterns among the existing three categories. However, as LQF_RR always favors the VOQ with the highest occupancy in input scheduling, the VOQ with low occupancy will suffer poor service fairness and even permanent queue starvation. In order to overcome the disadvantage of poor service fairness performance LQF_RR has, meanwhile to maintain excellent delay performance under various admissible traffic patterns, MCQF_RR takes advantage of combined weight information about queue occupancy and service waiting time of input VOQs to make efficient scheduling decisions. MCQF_RR favors the VOQ with the largest combined weight in input scheduling, namely the most critical queue will obtain preferential service. More specifically, the low occupied VOQ with much longer service waiting time will get efficient service, thus the good service fairness will be ensured. In the next section we first introduce some basic notations which will be used in this article, then describe MCQF_RR in details.

3.1 Notation

The architecture of an N × N CICQ switch model illustrated in Fig. 2 is considered in this paper. The CICQ switch operates at the line rate, and consists of N input ports, N output ports and a buffered crossbar fabric. The input ports and output ports are connected by the buffered crossbar fabric. Assume that packets with variable length arriving to the input ports are first segmented into fixed length cells for switching, and then reassembled into original variable length packets before leaving the output ports. Each input port maintains N separated VOQs to store cells destined for N output ports, where $VOQ_{i,j}$ holds cells arrived to input port i and destined for output j. There are N^2 crosspoint buffers placed at the crosspoints inside the buffered crossbar fabric. The crosspoint buffer $CB_{i,j}$ holds cells coming from input port i and destined for output port j. We give some definitions which will be used throughout the paper as follows:

Time slot: A time slot is the fixed time required to transmit a cell at the input line rate.

Eligible $VOQ_{i,j}$ ($EVOQ_{i,j}$): $VOQ_{i,j}$ is considered to be eligible if it is nonempty and its corresponding crosspoint buffer $CB_{i,j}$ is not full.

Eligible crosspoint buffer ($ECB_{i,j}$): $CB_{i,j}$ is considered to be eligible if it is nonempty.

$L_{i,j}(n)$ denotes the queue length of $VOQ_{i,j}$ at the beginning of time slot n, represented as the number of cells hold in $VOQ_{i,j}$.

$T_{i,j}(n)$ denotes the service waiting time of $VOQ_{i,j}$ which continuously loses service opportunities by time slot n since the last service.

3.2 Algorithm Description

MCQF_RR is based on the Most Critical Queue First (MCQF) policy at the input scheduling, and the output scheduling is based on the Round Robin (RR) policy. The input scheduling MCQF performs arbitration based on the combined weight information about the queue length $L_{i,j}(n)$ and service waiting time $T_{i,j}(n)$ of input VOQs. It gives the priority to the EVOQ with the greatest combined weight, to keep good delay performance and improve fairness performance upon LQF_RR. In order to achieve simplicity and fairness among crosspoint buffers, the output scheduling uses the fair round robin arbitration scheme similar to LQF_RR [19]. The process of the MCQF_RR is as follows:

Input Scheduling Phase:

For each input i ($0 \leqslant j \leqslant$ N-1), in each time slot n:

- **Step 1.** Starting from the highest priority pointer's location, select the first $EVOQ_{i,a}$ corresponding to $min_j L_{i,j}(n)$, and the first $EVOQ_{i,b}$ corresponding to $max_j L_{i,j}(n)$ respectively.
- **Step 2.** Compare the service waiting time $T_{i,a}(n)$ of $EVOQ_{i,a}$ with $L_{i,b}(n) + T_{i,b}(n)$ of $EVOQ_{i,b}$.
  – If $T_{i,a}(n) > L_{i,b}(n) + T_{i,b}(n)$, then give the priority to $EVOQ_{i,a}$ and send its HoL cell to crosspoint buffer $CB_{i,a}$.
  – Otherwise, serve $EVOQ_{i,b}$ and send its HoL cell to crosspoint buffer $CB_{i,b}$.
- **Step 3.** At the end of the time slot n, update the service waiting time $T_{i,j}(n)$ of each $VOQ_{i,j}$ ($0 \leqslant$ j $\leqslant$ N-1) and the highest priority pointer's location.
  – If a $VOQ_{i,j}$ has obtained the cell service, set $T_{i,j}(n) = 0$ and move the highest priority pointer to the location $(j + 1)(modN)$.
  – Else, set $T_{i,j}(n) = T_{i,j}(n) + 1$.

Output Scheduling Phase:

For each output j, in each time slot n:

Starting from the highest priority pointer's current location, select the first Eligible crosspoint buffer $ECB_{i,j}$.

– If any $ECB_{i,j}$ is found, send its HoL cell to the output j and move the highest priority pointer to the location $(i + 1)(modN)$ at the end of the time slot.

– Else, the highest priority pointer's location remains unchanged.

Figure 3 shows the service unfairness problem for a 2

× 2 CICQ switch MCQF_RR considered. Traffic flows from different inputs to different outputs with $\lambda_{1,1} = 1$, $\lambda_{1,2} = 0$ arrive at the switch, where $\lambda_{i,j}$ is the cell arrival rate of $VOQ_{i,j}$. We assume that for all i and j, the queue length of each input VOQ is one cell at the beginning. In time slot 1, there is a single cell arriving at input $VOQ_{1,1}$, but no cells arrive at $VOQ_{1,2}$. If Longest Queue First (LQF) scheme [19] is used in input scheduling phase, it will give preference to $VOQ_{1,1}$ with longest queue length, thus $VOQ_{1,2}$ will lose service. As there are continuous incoming cells in $VOQ_{1,1}$ in the following times slots, $VOQ_{1,2}$ will remain unserved all the time. Therefore, LQF will lead to service unfairness among different VOQs and even starve the queue with low occupancy. By contrast, our input scheduling phase MCQF improves this unfairness problem and maintains high delay performance of the switch taking advantage of combined weight information of each VOQ to schedule.

- **Good fairness performance.** If a $VOQ_{i,j}$ is not served, its service waiting time $T_{i,j}(n)$ will increase, that is the $VOQ_{i,j}$ that hasn't receive effective service for a long time will be assigned a larger $T_{i,j}(n)$. In Fig. 3, if the service waiting time $T_{1,2}(n)$ of $VOQ_{1,2}$ with lowest occupancy increases to large enough being greater than the combined weight $T_{1,1}(n) + L_{1,1}(n)$ of $VOQ_{1,1}$ with longest occupancy, which means that $VOQ_{1,2}$ may appear service unfairness and need urgent service, then MCQF guarantees service to the $VOQ_{1,2}$. Hence, no queues will be starved of service permanently.
- **High delay performance.** If a $VOQ_{i,j}$ has continuous new cells arriving, its queue length $T_{i,j}(n)$ will increase, thus $VOQ_{i,j}$ with larger occupancy that hasn't receive effective service for a long time will be assigned a larger combined weight $T_{i,j}(n) + L_{i,j}(n)$. As is shown in Fig. 3, if $VOQ_{1,1}$ with highest occupancy has a combined weight value of $T_{1,1}(n) + L_{1,1}(n)$ greater than or equal to the service waiting time $T_{1,2}(n)$ of $VOQ_{1,2}$ with lowest occupancy, it means that the cells in $VOQ_{1,1}$ may have much longer queue delay which will affect the overall delay performance of CICQ switch, then $VOQ_{1,1}$ will obtain priority service. In this way, the queue occupancy of each input VOQ will be stable and high delay performance can be ensured.

Similar to LQF_RR, we select round-robin as the output scheduling scheme in MCQF_RR because of its fairness and simple implementation.

3.3 Complexity of MCQF_RR

In term of time complexity, for input scheduling, the selection of $EVOQ_{i,a}$ with highest occupancy and $EVOQ_{i,b}$ with lowest occupancy in Step 1 both are $O(N \cdot \log N)$ at most, and can be reduced to $O(\log N)$ implemented with hardware parallel comparison circuits. The comparison in Step 2 has a time complexity of *O(1)*. The time complexity of output scheduling is also *O(1)*. Therefore, the overall time complexity of MCQF_RR is $O(\log N)$.

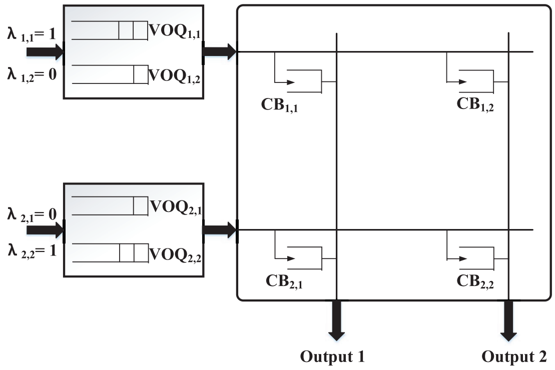


Fig. 3 Service unfairness problem for a 2 × 2 CICQ switch.

We set the variation range of the queue length $L_{i,j}(n)$ and service waiting time $T_{i,j}(n)$ of input $VOQ_{i,j}$ respectively as [0, L_{max}] and [0, T_{max}], where L_{max} is the maximum number of cells $VOQ_{i,j}$ can hold and T_{max} is the maximum service waiting time. Since variation range of the queue length $L_{i,j}(n)$ is [0, L_{max}], the number of bits needed to represent L_{max} is $\log L_{max}$. Therefore, it will consume $O(\log N \cdot \log L_{max})$ time to complete the parallel comparison of queue lengths in Step 1, and the input scheduling time will be $O(\log N \cdot \log L_{max})$, which is the same as LQF_RR.

4. The Improved Most Critical Queue First-Round Robin Scheduling Algorithm (IMCQF_RR)

To further reduce the input scheduling complexity of MCQF_RR, in this section we present an improved Most Critical Queue First-Round Robin scheduling algorithm (IMCQF_RR), which uses compressed weight information instead of original weight information in input scheduling while maintains high delay performance and fairness performance for various traffic patterns.

4.1 Algorithm Description

In order to reduce the parallel comparison time of queue lengths, we first use a transformation function to compress the range [0, L_{max}] of original queue length $L_{i,j}(n)$ into a short range before input scheduling, the range [0, T_{max}] of original service waiting time $T_{i,j}(n)$ is compressed as well. Compressed queue length, denoted as $CL_{i,j}(n)$, reflects the queue length of $VOQ_{i,j}$ compressed, and compressed service waiting time, denoted as $CT_{i,j}(n)$, reflects the service waiting time of $VOQ_{i,j}$ compressed. An efficient transformation function for compression should satisfy the requirements as follows:

(1) For input $VOQ_{i,j}$, $CL_{i,j}(n)$ should increase monotonically with $L_{i,j}(n)$;

(2) For input $VOQ_{i,j}$, $CL_{i,j}(n)$ should be less than or equal to $L_{i,j}(n)$;

(3) Transformation function for compression should be easily implemented in hardware.

There are two common classes of transformation functions: linear function and nonlinear function. Different functions have different influences on the scheduling performance of the scheduling algorithm. In this paper, we use a nonlinear function f(x) given in (1) to map original queue length and service waiting time information to a short range:

$$f(x) = \lfloor \log_2(1 + x) \rfloor \qquad (x > 0) \tag{1}$$

For queue length $L_{i,j}(n)$, its variation range can be compressed from $[0, L_{max}]$ to $[0, \lfloor \log_2(1 + L_{max}) \rfloor]$ of $CL_{i,j}(n)$ using f(x), and the number of bits needed to represent the maximum queue length reduced from $\log L_{max}$ to just $\log \lfloor \log_2(1 + L_{max}) \rfloor$. For hardware implementation of f(x), two simple ways can be chosen: (1) The correspondence between original queue length $L_{i,j}(n)$ and compressed queue length $CL_{i,j}(n)$ can be defined as a lookup table; (2) The compression of original queue length $L_{i,j}(n)$ can be implemented with a simple encoding circuit.

According to the compressed queue length $CL_{i,j}(n)$ and compressed waiting time $CT_{i,j}(n)$ of $VOQ_{i,j}$, we present IMCQF_RR as follows:

Input Scheduling Phase:

For each input i ($0 \leqslant j \leqslant N-1$), in each time slot n:

- **Step 1.** Starting from the highest priority pointer's location, select the first $EVOQ_{i,a}$ corresponding to $min_j CL_{i,j}(n)$, and the first $EVOQ_{i,b}$ corresponding to $max_j CL_{i,j}(n)$ respectively.
- **Step 2.** Compare the compressed service waiting time $CT_{i,a}(n)$ of $EVOQ_{i,a}$ with $CL_{i,b}(n) + CT_{i,b}(n)$ of $EVOQ_{i,b}$.
  – If $CT_{i,a}(n) > CL_{i,b}(n) + CT_{i,b}(n)$, then give the priority to $EVOQ_{i,a}$ and send its HoL cell to crosspoint buffer $CB_{i,a}$.
  – Otherwise, serve $EVOQ_{i,b}$ and send its HoL cell to crosspoint buffer $CB_{i,b}$.
- **Step 3.** At the end of the time slot n, update the service waiting time $T_{i,j}(n)$ of each $VOQ_{i,j}$ ($0 \leqslant j \leqslant N-1$) and the highest priority pointer's location.
  – If a $VOQ_{i,j}$ has obtained the cell service, set $T_{i,j}(n) = 0$ and move the highest priority pointer to the location $(j + 1)(modN)$.
  – Else, set $T_{i,j}(n) = T_{i,j}(n) + 1$.

The Output Scheduling Phase is based on Round Robin scheme, the same as MCQF_RR.

4.2 Properties of IMCQF_RR

In this section, we compare IMCQF_RR with existing scheduling algorithms and summarize its main characteristics as follows:

First of all, with the transformation function f(x), IMCQF_RR based on compressed weight information can maintain good delay performance similar to MCQF_RR based on original weight information, which means that the compressed weight information can reflect actual queue situation very well.

Second, input IMCQF scheduling essentially favors the input VOQ with the maximum combined compressed weight, so that each VOQ can receive efficient service without queue starvation. Similar to MCQF_RR, IMCQF_RR can achieve good service fairness performance.

Finally, the overall time complexity of IMCQF_RR is also $O(\log N)$. However, since the value of compressed queue length $CL_{i,j}(n)$ is much smaller than that of original queue length $L_{i,j}(n)$, the number of bits needed to represent $CL_{i,j}(n)$ is just $\log \lfloor \log_2(1 + L_{max}) \rfloor$. The comparison time of queue length can be largely decreased, and the input scheduling complexity is greatly reduced to $O(\log N \cdot \log \lfloor \log_2(1 + L_{max}) \rfloor)$, compared with $O(\log N \cdot \log L_{max})$ for MCQF_RR and LQF_RR.

5. Performance Study

We have conducted simulations to evaluate the delay performance and fairness performance of MCQF_RR and IMCQF_RR, aiming to compare with the existing algorithms.

5.1 Simulation Settings

We simulate a 16 × 16 CICQ switch operating with a per-port line rate of 1 Gbps, each VOQ has a buffer size of 1,000,000 cells. The buffer size of each crosspoint buffer is 1 cell. In our simulations, the principal item for evaluating switching performance is the cell delay, which is measured as the time (in time slot) taken for a cell to travel from an input port to its target output port. The cell delay includes queuing delays both in the input buffer and the crosspoint buffer. Average cell delay is the mean value of delay considering all the cells gathered over a time interval of 1,000,000 time slots. Normalized input load $\rho \in [0, 1]$ is denoted as the percentage of time slots which have cells incoming over all simulation time slots. In current simulation phase synthetic traffic patterns which have been widely used in previous papers [17] are adopted, later real life traffic will be employed to evaluate the performance in the next hardware implementation phase. Based on the 16 × 16 CICQ switch, eight scheduling algorithms are selected as below:

- RR_RR [18], Round Robin-based, for its low time complexity of *O(1)* and good fairness.
- RR-LQD [23], another Round Robin-based scheduling algorithm, where a different pointer updating policy is used in input scheduling, for its delay performance approximate to LQF_RR under some non-uniform traffic and low time complexity of *O(1)*.
- LQF_RR [19], a weight-based scheduling algorithm, where the weight is the number of cells queued in the input VOQ, for its stability and superior performance over other algorithms under various traffic patterns.
- OCF_OCF [16], another weight-based scheduling algorithm, where the weight is the queuing delay of the HoL cell in corresponding buffer, also for its stability.
- MCBF [20], a *crosspoint buffer state* based scheduling algorithm, for its better performance than other algorithms under uniform traffic patterns.

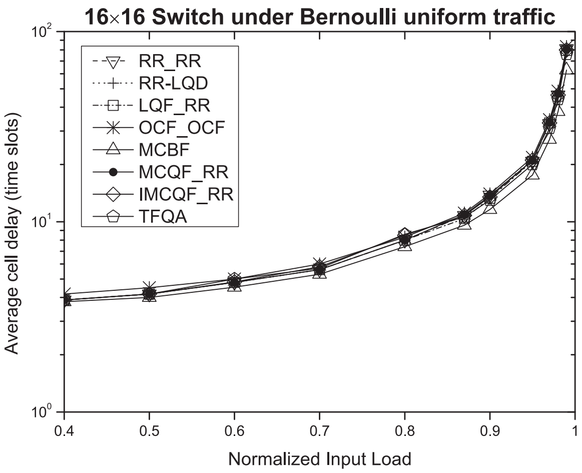


Fig. 4 Average cell delay under Bernoulli uniform traffic.

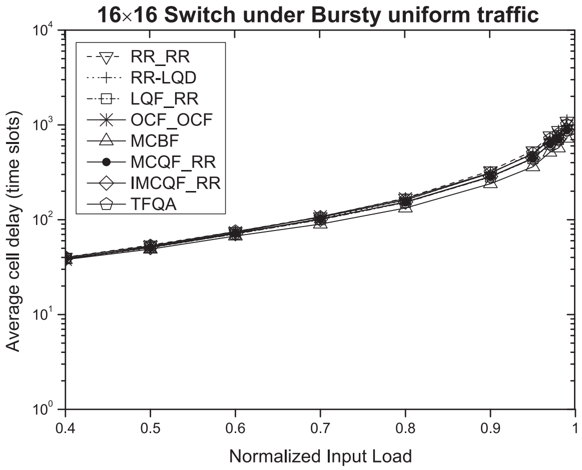


Fig. 5 Average cell delay under Bursty uniform traffic.

- MCQF_RR and IMCQF_RR, the algorithms proposed in this paper.
- TFQA [30], a scheduling algorithm aiming for good fairness performance.

5.2 Delay Performance

5.2.1 Uniform Traffic

We use the Bernoulli and Bursty uniform traffic patterns [18] to analyze the delay performance.

Figure 4 shows the average delay performance of different algorithms under Bernoulli uniform traffic. We see that all the algorithms behave similarly with low average cell delay, which increases gradually as the input load grows.

Figure 5 depicts the average delay performance of the algorithms under Bursty uniform traffic with burst lengths of 32 cells ($l = 32$). It is shown that, because of the effect of burst property, the average cell delays achieved by various algorithms significantly increase as the burst length increases and are almost identical. Similar to results in Fig. 4, MCBF has a slightly lower delay than other algorithms under Bursty uniform traffic.

5.2.2 Non-Uniform Traffic

The performance evaluation of our novel scheduling algorithms is performed under three non-uniform traffic patterns: Diagonal Traffic [27], Log-diagonal [28] and Unbalanced Traffic [18]. The non-uniform traffic is more skew and more difficult to schedule than uniform traffic.

- Diagonal Traffic
  For diagonal traffic, two-thirds of input load of each input i is destined for output i, and the remaining one-third is destined for output (i + 1) mod N, where N is the number of ports. The diagonal traffic for a 3 × 3 switch is defined as:

$$\lambda_{i,j} = \begin{pmatrix} 2\rho/3 & \rho/3 & 0 \\ 0 & 2\rho/3 & \rho/3 \\ \rho/3 & 0 & 2\rho/3 \end{pmatrix} \tag{2}$$

  where ρ is the input load of each input.
- Log-diagonal Traffic
  In log-diagonal traffic, each input i has input load for all outputs, but sends twice as much traffic to output j than to (j+1) mod N, that is $\lambda_{i,j} = 2\lambda_{i,(j+1)modN}$. For example, the log-diagonal traffic can be represented by a traffic matrix as:

$$\lambda_{i,j} = \begin{pmatrix} \frac{\lambda_0}{2^1} & \cdots & \frac{\lambda_0}{2^N} \\ \vdots & \ddots & \vdots \\ \frac{\lambda_{N-1}}{2^N} & \cdots & \frac{\lambda_{N-1}}{2^{N-1}} \end{pmatrix} \tag{3}$$

  where λ_i is the load at input i.
- Unbalanced Traffic
  For unbalanced traffic, there is an unbalanced probability ω as the fraction of the input load of each input port destined for a output, and the remaining load is uniformly distributed among other outputs. The arrival rate of each $VOQ_{i,j}$ is defined as:

$$\lambda_{i,j} = \begin{pmatrix} \rho(\omega + \frac{1-\omega}{N}) & \cdots & \rho \cdot \frac{1-\omega}{N} \\ \vdots & \ddots & \vdots \\ \rho \cdot \frac{1-\omega}{N} & \cdots & \rho(\omega + \frac{1-\omega}{N}) \end{pmatrix} \tag{4}$$

Figure 6 shows the average delay results of different algorithms for diagonal traffic. As the figure illustrates, scheduling algorithms such as RR_RR and MCBF which perform well under uniform traffic show instability even for load $\rho < 0.9$ under diagonal traffic. The reason is that they don't consider the information about input VOQs during input scheduling. Among the other algorithms considering input VOQ state, TFQA cannot maintain stability at $\rho > 0.9$ because the FQA scheme allocating quota for input VOQs based on RR manner may lead to large discrepancy between assigned quotas and actual length of heavily occupied queues and severely affect their stability in input scheduling; OCF_OCF and RR-LQD perform slightly worse

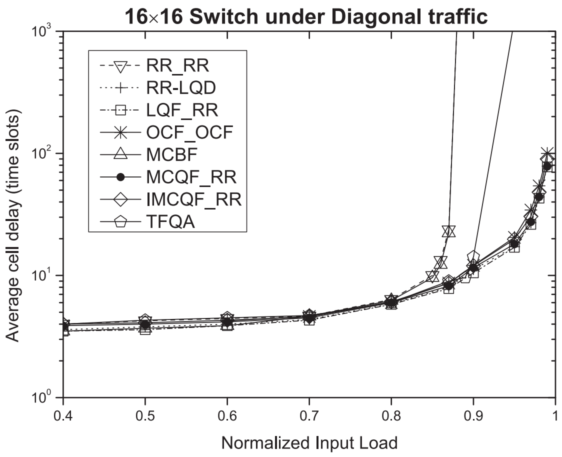


Fig. 6 Average cell delay under diagonal traffic.

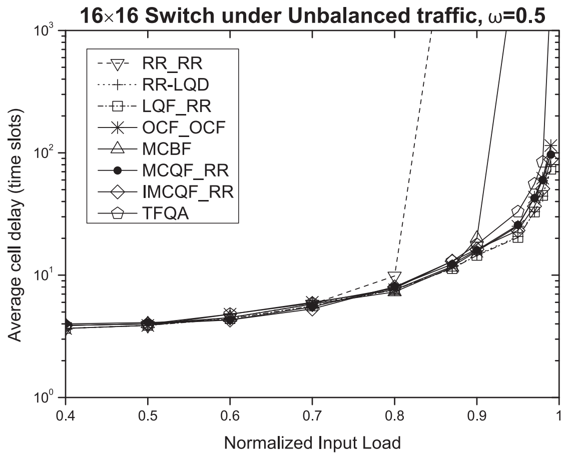


Fig. 8 Average cell delay under unbalanced traffic.

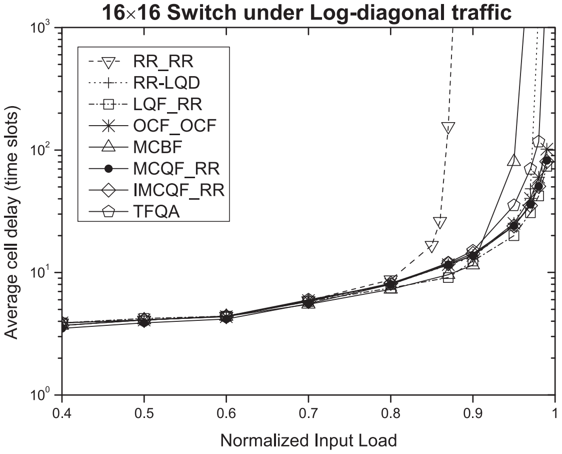


Fig. 7 Average cell delay under log-diagonal traffic.

than our proposed algorithms for $\rho > 0.95$, while the average delays of MCQF_RR and IMCQF_RR for different input loads are always close to the best LQF_RR.

Figure 7 plots the average delay results under log-diagonal traffic. We can see that LQF_RR has the lowest average delay, and our proposed algorithms deliver low average delays almost the same as LQF_RR. OCF_OCF is slightly inferior to MCQF_RR and IMCQF_RR at heavy load $\rho > 0.95$, and especially when ρ increases towards 100%, delay differences between OCF_OCF and our algorithms tend to increase much more dramatically. TFQA saturates for load $\rho > 0.98$ due to the reason also for diagonal traffic. RR-LQD saturates for $\rho > 0.95$ because the input scheduling based on detecting the longest queue may not guarantee the actual VOQ with the longest occupancy always get efficient service, then the queue length grows quickly and becomes unstable. RR_RR and MCBF show instability for load ρ larger than 0.85 and 0.9 respectively, because of the lack of information about queue length of input VOQ in input scheduling.

For unbalanced traffic (ω=0.5), the average delay results are depicted in Fig. 8. We see that RR_RR saturates for load $\rho < 0.85$, MCBF saturates for load around 0.9 and TFQA saturates for $\rho > 0.98$. Among the other scheduling algorithms, the average delay of OCF_OCF increases most dramatically for heavy load. MCQF_RR, IMCQF_RR and RR-LQD always achieve excellent delay performance comparable to LQF_RR.

From the above results, the proposed MCQF_RR and IMCQF_RR exhibit good delay performance comparable to LQF_RR and are superior to other algorithms for heavy load under non-uniform traffic patterns. This attributes to that they take advantage of queue length and service waiting time of input VOQs in input scheduling, making the queues with heavy load efficiently served and guaranteeing the stability of the queues with different occupancy.

5.3 Fairness Performance

In addition to achieving good delay performance, an efficient scheduling algorithm must guarantee good service fairness among the input traffic. In this section, the fairness performance of MCQF_RR and IMCQF_RR will be examined. According to the delay results of all the algorithms displayed in Sect. 5.2, we mainly compare the fairness performance of our proposed algorithms with LQF_RR, OCF_OCF and TFQA. As the output scheduling phase has little effect on the switching performance, hence we just consider different input scheduling phases in fairness performance comparison. In this paper, we make use of the fairness index defined in [29] to evaluate the fairness performance, let

$$FI(D_1, D_2, \cdots, D_N) = \frac{(\sum_{j=1}^{N} D_j)^2}{N \sum_{j=1}^{N} D_j^2} \tag{5}$$

where D_j is the average cell delay of $VOQ_{i,j}$, $j \in [1, 2, \cdots, N]$, for input i. The fairness index $FI(D_1, D_2, \cdots, D_N)$ has a range of [0, 1]. When FI=1, it means that all the VOQs of input i have the same average cell delay and the fairness performance is best. And the smaller is FI, the worse is the fairness performance. Then, we define an av-

Table 1 FI and AFI results for (A) IMCQF_RR, (B) MCQF_RR, (C) LQF_RR, (D) OCF_OCF and (E) TFQA under log-diagonal traffic.

Input load ρ	(A)	(B)	(C)	(D)	(E)
90%	0.9343	0.9498	0.8708	0.9143	0.8245
95%	0.8874	0.9404	0.7227	0.8654	0.6765
97%	0.8649	0.9097	0.6621	0.8547	0.4300
98%	0.8946	0.8381	0.6252	0.8401	—
99%	0.8709	0.6568	0.5692	0.8136	—
100%	0.4942	0.4433	0.3740	0.7627	—
AFI	0.8244	0.7897	0.6373	0.8418	0.6981

Table 2 FI and AFI results for (A) IMCQF_RR, (B) MCQF_RR, (C) LQF_RR, (D) OCF_OCF and (E) TFQA under asymmetric traffic.

Input load ρ	(A)	(B)	(C)	(D)	(E)
90%	0.8486	0.8741	0.8344	0.8991	0.7942
95%	0.8246	0.8087	0.6694	0.7789	0.6502
97%	0.7633	0.7535	0.5803	0.7055	0.5814
98%	0.7288	0.6705	0.5376	0.6253	0.4565
99%	0.7598	0.6569	0.4973	0.6549	0.2881
100%	0.5484	0.5437	0.4332	0.7666	—
AFI	0.7456	0.7179	0.5920	0.7384	0.5541

Table 3 FI and AFI results for (A) IMCQF_RR, (B) MCQF_RR, (C) LQF_RR, (D) OCF_OCF and (E) TFQA under unbalanced traffic.

Input load ρ	(A)	(B)	(C)	(D)	(E)
90%	0.8193	0.8853	0.7967	0.9421	0.9005
95%	0.7187	0.8313	0.7774	0.8922	0.8614
97%	0.7318	0.8310	0.8416	0.8727	0.8205
98%	0.7906	0.8547	0.9027	0.8547	0.6972
99%	0.8537	0.9151	0.9617	0.8419	—
100%	0.9640	0.9954	0.9948	0.9255	—
AFI	0.8130	0.8855	0.8791	0.8847	0.8199

erage value of FI as average fairness index (AFI), which is calculated as the fairness index averaged over all the considered loads of input traffic.

The fairness index results under log-diagonal traffic and asymmetric traffic [31] for load ρ (90%–100%) are presented in Table 1 and 2. It is shown that MCQF_RR and IMCQF_RR achieve much better fairness than LQF_RR for these two non-uniform traffic patterns, and AFI of IMCQF_RR is significantly higher than LQF_RR by almost 20% for log-diagonal traffic. Because log-diagonal and asymmetric traffic are much severe [30], the length of high occupied queue is much longer than low occupied queue. LQF always favors the longest queue, thus the average delay of the low occupied VOQ will be much higher than the high occupied VOQ and this results in serious fairness degradation. On the contrary, our algorithms ensure that all the VOQs have relatively similar average delay and further improve the fairness performance greatly. OCF_OCF provides high fairness indexes because of the Oldest Cell First scheme which gives priority to the HoL cell with the greatest waiting time among all the queues to avoid service unfairness. The fairness of TFQA are still poor for severe log-diagonal and asymmetric traffic.

Table 3 shows the fairness index results under unbalanced traffic ($\omega = 0.5$) for load ρ (90%–100%). We see that the fairness of MCQF_RR is higher than LQF_RR, and IMCQF_RR performs slightly worse than LQF_RR. MCQF_RR achieves better fairness due to the same reason as the severe traffic patterns. Because the unbalanced traffic is much easier to handle than severe log-diagonal and asymmetric traffic, LQF based on the queue length enables the average delay of each VOQ to become close and good fairness could be obtained.The main reason for slightly reduced fairness of IMCQF_RR may be the influence of transformation function f(x). For example, if the difference in queue length between two VOQs is not too large, their corresponding compressed queue lengths may become equal due to the transformation function $f(x) = \lfloor \log_2(1 + x) \rfloor$. As IMCQF performs input scheduling according to the compressed queue length, the average delay of heavy occupied VOQ may become larger and the average delay of low occupied VOQ may become smaller, thus the fairness of IMCQF_RR may be affected to some extent. OCF_OCF and TFQA achieve good fairness for the unbalanced traffic.

From the above results, MCQF_RR and IMCQF_RR have significantly improved fairness comparing with LQF_RR under severe log-diagonal and asymmetric traffic patterns because of the adaptable scheduling scheme based on the combined information about queue length and waiting time of input VOQs. Although there are some decline in the fairness of IMCQF_RR for unbalanced traffic, the percentage of drop is very small and it still has good fairness.

5.4 Comprehensive Comparison

To compare the algorithms more comprehensively, we analyze the implementability briefly in this section. We mainly realize basic structure for MCQF_RR, IMCQF_RR and LQF_RR based on FPGA and compare their resource overheads, while OCF_OCF and TFQA are discussed briefly because of poor implementability.

Table 4 shows the resource overheads of MCQF_RR, IMCQF_RR and LQF_RR for a 4×4 CICQ switch based on Xilinx Virtex-5 XC5VFX70T FPGA, and we assume maximum VOQ length is represented by 16 bits. The main macros of LQF-RR are 12 two-input(16-bit) comparators. For MCQF_RR, the macros are 12 two-input(16-bit) comparators, a two-input(16-bit) adder and a two-input(17-bit) comparator. IMCQF_RR have three types of macros including 12 two-input(4-bit) comparators, a two-input(4-bit) adder and a two-input(5-bit) comparator. For overall utilization in FPGA, MCQF_RR utilizes more LUTs than LQF_RR when N=4, but the ratio of increase will be reduced with the increase of N. As IMCQF_RR needs parallel comparison circuits which are much simpler than MCQF_RR, the number of LUTs is greatly decreased and similar to LQF_RR for N=4. When N becomes larger, IMCQF_RR utilizes less LUTs than LQF_RR; and the larger is N, the greater is the proportion of decrease. Therefore, the realizability of IMCQF_RR is better than LQF_RR.

OCF_OCF and TFQA have poor implementability. For OCF_OCF, the OCF scheme performs a complicate time stamping mechanism which is too difficult and unrealistic

Table 4 The resource overheads based on Xilinx Virtex-5 XC5VFX70T FPGA.

Algorithm	Main macros (4 Ports)	Main macros (N Ports)	LUTs (N=4)
`LQF_RR`	12 2-input(16bit) comparators	N(N-1) 2-input(16bit) comparators	137
`MCQF_RR`	12 2-input(16bit) comparators A 2-input(16bit) adder A 2-input(17bit) comparator	N(N-1) 2-input(16bit) comparators A 2-input(16bit) adder A 2-input(17bit) comparator	225
`IMCQF_RR`	12 2-input(4bit) comparators A 2-input(4bit) adder A 2-input(5bit) comparator	N(N-1) 2-input(4bit) comparators A 2-input(4bit) adder A 2-input(5bit) comparator	139

Table 5 Performance characteristics of IMCQF_RR, MCQF_RR, LQF_RR, OCF_OCF and TFQA.

Algorithm	Delay	Fairness	Implementability
`IMCQF_RR`	Very Good	Very Good	Best
`MCQF_RR`	Very Good	Very Good	Medium
`LQF_RR`	Best	Worst	Medium
`OCF_OCF`	Medium	Best	Worst
`TFQA`	Worst	Worst	Worst

to implement. Each cell in the switch should be assigned a time stamp which may become too large to calculate the cell's waiting time in the queue. Besides, communication overhead between input VOQs and buffered crossbar is increased greatly because input scheduling phase needs comparing the queue length of crosspoint buffers in addition to the waiting time of HoL cells. For TFQA, calculation of the adaptive threshold in input scheduling needs time-consuming multiplication and division operations which are difficult to implement. The FQA scheme also significantly increases the communication overhead such as quota and VOQ length information interaction between input VOQs and buffered crossbar.

According to the experimental and discussion results, we obtain the performance characteristics include delay, fairness and implementability of various algorithms as shown in Table 5. The performance characteristics are classified into four grades: Best, Very Good, Medium and Worst. We see that the proposed IMCQF_RR behaves the best, MCQF_RR performs less well due to the medium implementability. LQF_RR and OCF_OCF are weaker than our algorithms respectively because of the worst fairness and the worst implementability. TFQA is the worst-performing algorithm.

6. Conclusion

This paper first introduces a Most Critical Queue First-Round Robin (MCQF_RR) scheduling algorithm for CICQ switches, which is based on the combined weight information about queue length and service waiting time of input VOQs in input scheduling and round robin in output scheduling. The time complexity of MCQF_RR is $O(\log N)$. Simulation results show that MCQF_RR can provide good delay performance comparable to high performance scheduling algorithm LQF_RR under any admissible input traffic patterns, and more importance is that it achieves fairness performance superior to LQF_RR under non-uniform traffic patterns. To further reduce the scheduling complexity of MCQF_RR, we then propose an Improved Most Critical Queue First-Round Robin (IMCQF_RR) scheduling algorithm. The implementability complexity of IMCQF_RR is much lower than LQF_RR and MCQF_RR as the number of ports N increases. Our simulation results indicate that IMCQF_RR also exhibits good delay performance comparable to LQF_RR, and shows significantly better fairness performance than LQF_RR under severe non-uniform traffic patterns.

References

[1] N. McKeown, M. Izzard, A. Mekkittikul, B. Ellersick, and M. Horowitz, The Tiny Tera: A Packet Switch Core, IEEE Micro, vol.17, no.1, pp.26–33, Jan./Feb. 1997.

[2] F. Abel, C. Minkenberg, P. Luijten, M. Gusat, and I. Iliadis, A Four-Terabit Packet Switch Supporting Long Round-Trip Times, IEEE Micro, vol.23, no.1, pp.10–24, Jan./Feb. 2003.

[3] H.J. Chao and B. Liu, High Performance Switches and Routers, Wiley-IEEE Press, 2007.

[4] S.T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, Matching output queueing with a combined input output queued switch, IEEE J. Sel. Areas Commun., vol.17, no.6, pp.1030–1039, June 1999.

[5] H. Zhang, Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks, Proc. IEEE, vol.83, no.10, pp.1374–1396, Oct. 1995.

[6] M. Karol, M. Hluchyj, and S. Morgan, Input versus Output Queuing on a Space Division Packet Switch, IEEE Transaction on Communication, vol.35, no.12, pp.1347–1356, Dec. 1987.

[7] S. Keshav and R. Sharma, Issues and trends in router design, IEEE Commun. Mag., vol.36, no.9, pp.144–151, Dec. 1998.

[8] N. McKeown, Scheduling algorithms for input-queued cell switches, Ph.D. thesis, University of California at Berkeley, May. 1995.

[9] G. Chartrand, Introductory graph theory. Dover Publications, Dec. 1984.

[10] A. Mekkittikul, and N. McKeown, A practical scheduling algorithm to achieve 100% throughput in input-queued switches, Proc. IEEE INFOCOM'98, pp.792–799, March 1998.

[11] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, Achieving 100% throughput in an input queued switch, IEEE Trans. Commun., vol.47, no.8, pp.1260–1267, Aug. 1999.

[12] N. McKeown, The iSLIP Scheduling Algorithm for Input-Queued Switches, IEEE/ACM Trans. Netw., vol.7, no.2, pp.188–201, April 1999.

[13] D.N. Serpanos and P.I. Antoniadis, FIRM: A Class of Distributed Scheduling Algorithms for High-Speed ATM Switches with Input Queues, Proc. IEEE INFOCOM'00, pp.548–554, March 2000.

[14] H.J. Chao, Saturn: A Terabit Packet Switch Using Dual Round-Robin, IEEE Commun. Mag., vol.38, no.12, pp.78–84, Dec. 2000.

[15] J.G. Dai and B. Prabhakar, The Throughput of Data Switches with and without Speedup, Proc. IEEE INFOCOM'00, pp.556–564, March 2000.

[16] M. Nabeshima, "Performance evaluation of combined input- and crosspoint-queued switch," IEICE Trans. Commun., vol.E83-B, no.3, pp.737–741, March 2000.

[17] K. Yoshigoe and K.J. Christensen, An Evolution to Crossbar Switches with Virtual Output Queuing and Buffered Cross Points,

IEEE Netw., vol.17, no.5, pp.48–56, Sept./Oct. 2003.

[18] R. Rojas-Cessa, E. Oki, Z. Jing, and H.J. Chao, CIXB-1: Combined Input One-Cell-Crosspoint Buffered Switch, Proc. IEEE Workshop High Performance Switching and Routing (HPSR), pp.324–329, May 2001.

[19] T. Javadi, R. Magill, and T. Hrabik, A. High-Throughput Scheduling Algorithm for a Buffered Crossbar Switch Fabric, Proc. IEEE International Conference on Communications, pp.1581–1591, June 2001.

[20] L. Mhamdi and M. Hamdi, MCBF: A High-Performance Scheduling Algorithm for Buffered Crossbar Switches, IEEE Communication Letters, vol.7, no.9, pp.451–453, Sept. 2003.

[21] R. Rojas-Cessa, and E. Oki, Round-robin selection with adaptable-size frame in a combined input-crosspoint buffered switch, IEEE Commun. Lett., vol.7, no.11, pp.555–557, Nov. 2003.

[22] J. Luo, Y. Lee, and J. Wu, A non-uniform traffic oriented scheduling algorithm in combined input-crosspoint-queued (CICQ) switch, International Journal of Parallel, Emergent and Distributed System, vol.21, no.4, pp.279–292, Aug. 2006.

[23] Z. Yun, L. Peng, W. Zhao, and C. Tian, RR-LQD: A novel scheduling algorithm for CICQ switching fabrics, Proc. 15th Asia-Pacific Conference on Communications (APCC), pp.846–849, Oct. 2009.

[24] A. Mekkittikul, Scheduling non-uniform traffic in high speed packet switches and routers, Ph.D. dissertation, Stanford University, Nov. 1998.

[25] X. Zhang, and L.N. Bhuyan, An efficient scheduling algorithm for combined input-crosspoint-queued (CICQ) switches, GLOBECOM'04. IEEE, pp.1168–1173, Nov./Dec. 2004.

[26] L. Mhamdi, M. Hamdi, C. Kachris, S. Wong, and S. Vassiliadis, High-performance switching based on buffered crossbar fabrics, Computer Networks, vol.50, no.13, pp.2271–2285, Sept. 2006.

[27] P. Giaccone, D. Shah, and B. Prabhakar, "An implementable parallel scheduler for input-queued switches," IEEE Micro, vol.19, no.1, pp.1090–1096, Jan./Feb. 1999.

[28] R. Rojas-Cessa, and C.-B. Lin, VOQ Captured-frame matching algorithms for scalable input-queued packet switches, COMPUTER COMMUNICATIONS, vol.30, no.10, pp.2149–2161, July 2007.

[29] R. Jain, A. Durresi, and G. Babic, Throughput fairness index: An explanation, ATM Forum Document: ATM-Forum/99-0045, Feb. 1999.

[30] N. Hua, P. Wang, D. Jin, L. Zeng, B. Liu, and G. Feng, Simple and Fair Scheduling Algorithm for Combined Input-Crosspoint-Queued Switch, IEEE ICC, pp.6305–6310, 2007.

[31] A. Bianco, M. Franceschinis, S. Ghisolfi, A.M. Hill, E. Leonardi, F. Neri, and R. Webb, Frame-based matching algorithms for input-queued switches, IEEE HPSR, pp.69–76, 2002.

Xiaoting Wang was born in Shan Xi Province, China, in 1987. She received the B.S. degree from North China University of Water Resources & Electric in 2009. And now she is working towards the Ph.D. degree at the Institute of Microelectronics and Solid-State Electronics of University of Electronic Science and Technology of China. Her research work is in the field of scheduling and architecture of high-performance switches, and SOC implementation.

Yiwen Wang was born in Liaoning Province, China, in 1966. He received his B.S. degree and Ph. D. degree in electrical engineering from Wuhan University and Tokyo Institute of Technology, respectively. From 1988 to 2000, he worked as a senior VLSI design engineer in Northeast Microelectronics Institute of China. From 2005 to 2007, he worked as a special associate professor in Tokyo Institute of Technology. He is currently a professor in University of Electronics Sciences and Technology of China. His research interests include scheduling and management of high-performance switches,reconfigurable SoC architecture, Fingerprint authentication. Etc.

Shichao Li was born in 1993 in Chong Qing Province. Now he is an undergraduate students. He will received the B.S.degree from University of Electronic Science and Technology of China in 2015. His research work is in the field of SIP and SOC implementation.

Ping Li was born in Sichuan Province, China, in 1957. He received the B.S. and the M.S. degrees from UESTC in 1982 and 1987, respectively, both in microelectronic engineering. He is now full Professor and Deputy Director of State key Laboratory of Electronic Thin Films and Integrated Devices UESTC. His research interests include FPGA design and test, SOC design and verification.