LIST OF PUBLICATIONS


INTRODUCTION

Need for high speed internet is the major requirement for the internet society, as the usage has been widened in the last few years. High speed connectivity can be achieved with the adequate networking devices such as switches and routers. Among the various switching architectures, Buffered Crossbar Switches (BCS) are accepted for high switching efficiency through fixed length switching technologies. BCS can accept packets of variable length which are then segmented into fixed sized cells and are transmitted. On the other side, the cells are reassembled to form an original packet [1]. As input queued switch suffers from throughput limitation and output queued switch suffers from output port contention, BCS avoids these shortcomings. Also input queued switch suffers from Head of Line blocking [2] which can be overcome through the introduction of Virtual Output Queue (VOQ)[3], BCS uses VOQ to input the packets. A buffer is introduced at every crosspoint of BCS to hold the incoming cell. Size of the buffer can be assigned based on architecture requirement but normally it is one cell size. To improve the switching efficiency, buffer size can be enlarged but it introduces huge implementation cost. Switch performance is based on the effective utilization of the switch i.e. all the baselines should be used at every timeslot. For effective utilization, proper scheduling algorithm must be employed.

The primary objective of the scheduling algorithms is to achieve 100% throughput with no delay in all sorts of traffic. Usually the performance of the algorithms is not the same for both uniform and non-uniform traffics. Round Robin scheduler (RRS) [4]-[6] is used at input and output schedule to achieve 100% throughput under uniform traffic whereas for non-uniform traffic it lacks its performance. Moreover the average waiting time of the VOQ is more than 5ms for a 4 × 4 switch. Then the RRS at input schedule is replaced by Longest Queue First algorithm [7]-[8] which offers 100% and 75% throughput in Bernoulli uniform and non-uniform traffic respectively. Moreover it offers worse latency and fairness. To further improve the throughput at both the traffic formats, some algorithms uses a speedup of 2 in certain architectures [8]-[9] but speedup will provide only the half of the aggregate line throughput and also introduces a need for output queue leads to increase in implementation complexity. The authors in [10]-[11] achieved 100% throughput under uniform traffic through their proposed distributed algorithm but their performance gets dropped a maximum of 30% under non-uniform traffic. In [8]-[12], author(s) proposed an algorithm Oldest Cell First (OCF) which is simple to implement but offers poor performance in Bernoulli bursty traffic. Most Critical Buffer First (MCBF) [13] offers good stability and high performance but requires internal buffer state information for scheduling, thereby complexity gets increased. SQUISH and SQUID [14] achieves 100% throughput without speedup for any Bernoulli admissible traffic but its extended waiting time at the VOQ leads to starvation. Starvation will halt the movement of cells from a particular queue in the VOQ.

From our study, it is understood that most of the algorithm achieves 100% throughput under uniform traffic with or without speedup but their performance reduce upto 30% under non-uniform traffic. At a maximum, 70% throughput has been achieved by Longest Queue First with Round Robin scheduler (LQF-RR) under non-uniform traffic [4]-[6]. In this paper, we proposed a Prioritized Queue with Round-robin Scheduler (PQRS) for Buffered Crossbar Switches (BCS) with no speedup. Through simulation, average waiting time, throughput and average cell latency is measured for different load structure under Bernoulli non-uniform iid Traffic and Bernoulli non-uniform Bursty Traffic. The outcome is compared with the LQF-RR and is considerably very good. It is understood that designing a starvation free scheduling algorithm to achieve 100% throughput under non-uniform traffic is an open challenge and we made an attempt. The structure of the paper is as follows. In section 2, we defined the Buffered Crossbar Switch along with its properties. In section 3, Priority based BCS algorithm is proposed. Section 4 comprises of simulation results for Bernoulli non-uniform traffics and its comparative analysis. Finally section 5 concludes the paper.
2. BCS SCHEDULING

The performance of the Buffered Crossbar Switch is based on the scheduling algorithm which is employed. Scheduling algorithm decides when and where to switch the packets, therefore designing an appropriate scheduling is must for any BCS [15]. Here, every switch requires two schedules: an Arrival Schedule (AS) and a Departure Schedule (DS). At each timeslot, Arrival Schedule selects the cell which is transferred from VOQ to BCS buffer and Departure Schedule selects the cell transferred from BCS buffer to output Queue. An arrival schedule is possible only if any of the buffers in the crosspoint is empty. If the buffer size is unlimited, then there is no need for arrival schedule that is input can be directly sent to the buffer. Because of the implementation cost, unlimited buffer size is practically not possible and in this paper the buffer used is 1 cell size. Fig.1 shows the structure of Buffered Crossbar Switch.

![Fig.1. Buffered Crossbar Switch](image)

For a BCS, let ‘d’ denote arrival, ‘d’ denote departure, ‘B’ denote buffer, ‘t’ denote timeslot and ‘Q’ denote queue then \( L_{ad}(n) \) is the Queue length for any VOQ of size n. The crosspoint buffer \( B(n) \) where \( n = 0 \) or 1 for all the iterations. Let \( B_a \) is the Buffer occupancy through Arrival Schedule and \( B_d \) is the Buffer occupancy through Departure Schedule then the Buffer occupancy \( B_{ad}(n) \) for a particular timeslot is given in Eq.(1)

\[
B_a \leq B \text{ and } B_d \geq B
\]

Total number of cells available in VOQ during arrival schedule is denoted as \( C_{ad}(n) \) and for every schedule is at each time slot \( L_{ad} \leq 0 \). Cell arrival from input port through VOQ is a stochastic process \( A_d(t) \) and the arrival rate is denoted by \( \lambda_{ad} \). Therefore \( A_{ad}(t) \) denotes the arrival process from input port to output port. For an \( N \times N \) switch, the arrival schedule at time is represented as \( S^A(n) \). If the buffer is empty, then a switch of atleast one cell from input queue to buffer is possible as given in Eq.(2)

\[
\sum S^A_{ad}(n) \leq 1 \cdot S^A_{ad}(n) = 0
\]

\( S^D(n) \) is the Departure Schedule, where a switch of atleast one cell is possible from buffer to output queue, If the buffer is not empty then it is given in Eq.(3)

\[
\sum S^D_{ad}(n) \leq 1 \cdot S^D_{ad}(n) = 0
\]

Finally, for every timeslot \( t \), a switch is possible which includes both arrival and departure schedule as shown in Eq.(4)

\[
\forall S^D_{ad}(n) = [S^A : S^D]
\]

Every BCS has a set of properties which exhibits its character and is given as.

i. For each timeslot, a switch can use independent scheduling algorithms for arrival as well as departure schedule

ii. Every schedule should transfer atleast one cell from input port to buffer or/and from buffer to output port

iii. During every timeslot, input schedule is followed by output schedule

iv. For every timeslot, input schedule is possible only if any one of the buffer is empty and in parallel output schedule is possible only if the buffer is not empty

For any BCS, the switch is stable [14] if the algorithm used is Maximum Weight Matching (MWM) and the available queue size is bounded.

3. PRIORITIZED QUEUE WITH ROUND-ROBIN SCHEDULER

In this section, we propose the Prioritized Queue with Round-robin Scheduler (PQRS) for Buffered Crossbar Switches. It uses independent algorithms for arrival and departure schedule and PQRS works based on the principle of Maximum Weight Based (MWB) algorithms. The algorithm is as follows.

3.1 ARRIVAL SCHEDULE – PRIORITY QUEUE SCHEDULER (PQS)

i. For any non-uniform traffic, selection of queue for cell transfer from VOQ to a crosspoint is based on the Queue priority

ii. Queue priority is the number of cells occupied in the queue. For every switch, a bonus priority value of 1 will be distributed to all the queues in the VOQ and is summed with the actual priority. Bonus is not applicable to the queue which is used in the current timeslot.

iii. Queues with same priority have to follow the under said.

- A queue will not be selected for schedule for consecutive number of times unless all other queues are empty
- Queue which is not scheduled at least once, will be given the next opportunity
- Otherwise follow FIFO schedule

iv. If high prioritized queue is empty then opt for the next highest

3.2 DEPARTURE SCHEDULE

For each departure schedule \( S^D \) at time \( t \), if the buffer \( B^D \neq 0 \) then Round Robin (RR) schedule is used. If all the crosspoint buffers are empty then \( S^D = 0 \). Here both the scheduling algorithms are independent to each other. Since RR is a proven scheduler in output queued switches it is used in the departure schedule. Furthermore Extended RR is also a better option for departure schedule.
4. PERFORMANCE EVALUATION

4.1 SIMULATION ENVIRONMENT

We implemented a simulator in Java BCSSIM that models the buffered crossbar switch of size $N \times N$. In general for all the experiments, we used a $4 \times 4$ VOQ/BCS switch with a buffer size of 1 and no speedup is introduced at any stage. Input for BCSSIM is supplied through Bernoulli non-uniform iid traffic and Bernoulli non-uniform bursty traffic.

4.2 SIMULATION RESULTS

Under Bernoulli non-uniform iid traffic, the Average Waiting Time (AWT) of the entire queue in a VOQ is computed and difference between maximum and minimum AWT is less than 1ms which is shown in Fig.2. For Bernoulli non-uniform bursty traffic, the difference is slightly greater than 1ms. Therefore it is understood that all the queues are equally served for both the traffic patterns and hence it avoids starvation.

We implemented the PQRS to compute its throughput and delay performances for various non-uniform traffic patterns and compared the outcome with LQF-RR. During the simulation, the arrival rate considerably varies between $\lambda_a = 0.3$ to 0.7 to introduce Bernoulli non-uniform iid traffic to the switch. For such traffic, the switch behaves optimistically as shown in Fig.3 and Fig.4 until the arrival rate is $< 0.6$. That is, 90% throughput is achieved for arrival rate $\leq 0.6$ and it decreases to 84% for the load beyond that. Average Cell Latency (ACL) is less than 5% until 50% load offered and decreases upto 10% for maximum load. Comparing to LQF-RR, PQRS extend very good delay and throughput performance by 10%.

The Fig.5 shows the throughput analysis of PQRS and LQF-RR under Bernoulli non-uniform bursty traffic with respect to load. Above 85% throughput has been achieved by PQRS until 50% load offered and decreases upto 10% for maximum load. In all the cases PQRS outperforms LQF-RR by more than 10%. Fig.6 shows that the ACL of PQRS is 5% until
60% of the load is offered and drops up to 18% when maximum load is offered. Comparing to LQF-RR, PQRS offers minimum delay performance by more than 15%.

From the results it is understood that the throughput and delay performance get decreased when the load exceeds 70%. However the waiting time of the VOQ is stabilized for any load.

![Graph showing average cell latency as a function with respect to load for Bernoulli non-uniform bursty traffic](image)

Fig. 6. Average Cell Latency as a function with respect to load for Bernoulli non-uniform bursty traffic

5. CONCLUSION

This paper presents the Prioritized Queue with Round Robin scheduler for buffered crossbar switches. It uses a Prioritized Queue Scheduler on the arrival schedule and Round Robin algorithm on the departure schedule. These combined scheduling schemes got the essence of reducing the total waiting time involved in the VOQ. From the simulation results, it is proved that PQRS to be the better option for BCS scheduling in non-uniform traffic environments.

REFERENCES

Starvation Free Scheduler for Buffered Crossbar Switches

N. Narayanan Prasanth\textsuperscript{a*}, K. Balasubramanian\textsuperscript{b}, R. Chithra Devi\textsuperscript{c}

\textsuperscript{a}Department of Information Technology, National College of Engineering, Maruthakulam, India
\textsuperscript{b}Department of Computer Science & Engineering, Mepco Schlenk Engineering College, Sivakasi, India
\textsuperscript{c}Department of Information Technology, Dr. Sivanthi Aditanar College of Engineering, Tiruchendur, India

\begin{abstract}
Need for high speed internet connectivity has led to a substantial research in switching systems. Buffered crossbar switches have received a lot of attention from both research and industrial communities due to their flexibility and scalability. Designing a scheduling algorithm for buffered crossbar switches without starvation is a major challenge as of now. In this paper, we propose a Delay based prioritized queue with round-robin scheduler (DPQRS) which uses no speedup. Simulation result shows that DPQRS reduces the starvation considerably with maximum throughput and minimum delay comparable to PQRS and LQF-RR.
\end{abstract}

\begin{enumerate}
\item \textbf{ABSTRACT}

Need for high speed internet connectivity has led to a substantial research in switching systems. Buffered crossbar switches have received a lot of attention from both research and industrial communities due to their flexibility and scalability. Designing a scheduling algorithm for buffered crossbar switches without starvation is a major challenge as of now. In this paper, we propose a Delay based prioritized queue with round-robin scheduler (DPQRS) which uses no speedup. Simulation result shows that DPQRS reduces the starvation considerably with maximum throughput and minimum delay comparable to PQRS and LQF-RR.

\begin{enumerate}
\item \textbf{INTRODUCTION}

The annual growth of internet users is 8% compared to the population growth of 1.14%\textsuperscript{1}. This shows the impact of internet usage in the day-to-day life. Several techniques\textsuperscript{1} are used by the researchers to measure the internet traffic beyond its numerous limitations. It is the responsibility of the researchers to provide the necessary technology to attain high speed internet with enough bandwidth. High speed connectivity can be achieved with the adequate networking devices such as switches and routers. Among the various switching architectures, buffered crossbar switches (BCS) are considered due to the two advantages\textsuperscript{3}. First, the adoption of internal buffers makes the scheduling totally distributed, dramatically reducing the arbitration complexity. Second, and most importantly, these internal buffers reduce (or avoid) output contention as they allow the inputs to make cell transfers concurrently to a single output.

Many scheduling algorithms have been recently proposed for the BCS architecture. Round Robin Scheduling (RRS)\textsuperscript{4} is the simplest algorithm to implement on both the arrival and departure schedule. Another simplest algorithm, oldest cell first (OCF)\textsuperscript{5} on the arrival schedule and RRS in the departure schedule is also employed. Longest queue first (LQF) algorithm along with RRS is also proposed for BCS which offers good performance\textsuperscript{6} in uniform and non-uniform traffics compared to RRS and OCF-RR. Both LQF-RR and OCF-RR suffers from starvation, implementation complexity and are also time consuming during input schedule\textsuperscript{7}. With a speedup of 2, algorithm proposed in other works\textsuperscript{8} can achieve 100% throughput under any admissible traffic. The speed requirement can be further reduced to 2-1/N in the algorithm proposed elsewhere\textsuperscript{9}. In the literature\textsuperscript{10}, the proposal required an infinite-size buffer in the crosspoint to achieve maximum throughput. SQUID\textsuperscript{11} is a centralized algorithm which can achieve 100% throughput under any admissible traffic. Due to its communication complexity, starvation and latency i.e. delay, it is impossible to implement this algorithm in large scale high-speed switching systems. Author in another work\textsuperscript{12} proposed a distributed algorithm \textit{DISQUO} which results in high degree of starvation.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|}
\hline
\textbf{Parameter} & \textbf{Value} \\
\hline
\textit{DISQUO} & \textit{2-1/N} \\
\hline
\end{tabular}
\caption{Comparison of \textit{DISQUO} with Other Algorithms}
\end{table}

\begin{enumerate}
\item \textsuperscript{1}http://www.internetlivestats.com/internet-users
\end{enumerate
MCBF [13] considerably reduces the hardware complexity with optimal stability performance but suffers from starvation. From these studies, it is understood that no algorithm can achieve 100% throughput for any admissible traffic without starvation and speedup.

In this paper, we propose a delay based prioritized queue Round-robin scheduler (D-PQRS) for Buffered Crossbar Switches. During arrival schedule, selection of queue from Virtual Output Queue (VOQ) is based on Updated Queue Priority Value (UQPV). Every queue in a VOQ has a queue priority and at each timeslot a bonus value is added with queue priority which results in UQPV. Bonus value is computed based on the waiting time of a queue. We simulated a NxN BCS with VOQ where N=4, 16, 64, 256 and 512 and results show the starvation is considerably reduced in all VOQ’s of a switch and interestingly, the throughput and delay performance are also improved comparatively to PQRS and LQF-RR algorithms.

2. BUFFERED CROSSBAR SWITCH

As an alternative to Bufferless Crossbar Switch, Buffered Crossbar Switch is introduced to improve the switching throughput. Buffering is exclusively introduced at each crosspoint to avoid the cell loss. Initially, BCS was developed [14] with buffer as a separate unit but the implementation failed to match the expectations in terms of cell loss. Later, buffer was embedded on chip along with the switching circuits. Here, the numbers of ports were limited by the memory size that can be implemented in a module chip. Therefore the size of the memory determines the number of cells switched at a single timeslot. In order to overcome the excessive on-chip memory requirement, buffered crossbar switches with input queues were proposed [15, 16]. Virtual output queue scheme is introduced to ensure that there is no packet drop during the arrival process. Therefore Buffered crossbar switches with input VOQs (VOQ/BCS) have recently received high attention. VOQ/BCS also supports variable sized packets without the need of segmentation and reassembly (SAR) circuitry and speedup is not required [17].

![Buffered crossbar switch](image)

Figure 1. Buffered crossbar switch

Figure 1 shows the N x N VOQ/BCS with N input cards maintaining N VOQ’s, one for each output. The buffer available in the crosspoint is assumed to be of one cell size and if there is no packet drop during switching process then the switch (BCS) is said to be stable.

3. BCS SCHEDULING

The objective of the scheduling algorithms is to maintain synchronization between the cell switching processes. Flow of cells, in and out of the switch depends on the employed scheduling algorithm. BCS uses two independent scheduling algorithms named input or arrival scheduler and output or departure scheduler. Input scheduler selects a cell from the queue and passes to the vacant buffer of the switch whereas output scheduler transfers the cell from the buffer to output queue. A switching process is said to be complete if a cell passes the input as well as output schedule in order [16, 17].

4. DELAY BASED PRIORITIZED QUEUE With ROUND-Robin SCHEDULER (D-PQRS)

Proposed algorithm uses Delay based Priority Queue algorithm (DPQ) as arrival schedule and Round-Robin (RR) algorithm as Departure schedule.

4.1. Arrival Schedule

1. For any traffic, switching of cells from VOQ to crosspoint is based on Updated Queue Priority Value (UQPV).
2. Every queue in a VOQ has a priority value called Queue Priority (QP). QP is an integer value which is based on the number of cells occupied by the queue during the initial schedule.
3. During each switch, a Bonus value (\(B_i\)) is distributed to all the queues in the VOQ except the one which is used in the current time slot.
4. Bonus value is an integer value which is the number of timeslots the queue is waiting.
5. Queue priority is added with the bonus value to get the Updated Queue Priority Value (UQPV).
6. UQPV for the currently used queue will be reduced by one. i.e. UQPV = UQPV - 1
7. Queues with same UQPV have to follow the under said.
   • A queue will not be selected for schedule for consecutive number of times unless all other queues are empty.
Queue which is not scheduled at least once, will be given the next opportunity
- Otherwise follow FIFO schedule

8. If high prioritized queue is empty then opt for the next highest value.

4. 2. Departure Schedule  
For each departure schedule $S^D_t$ at time $t$, if the buffer $B^D_t \neq 0$ then Round Robin (RR) schedule is used. If all the crosspoint buffers are empty then $S^D_t = 0$. Here both the scheduling algorithms are independent to each other. Since RRS is a proven scheduler in output queued switches, it is used in the departure schedule.

4. 3. An Example  
Consider a 4x4 BCS which uses non-uniform iid traffic and it is assumed that all the input cells are arrived to the queue at the same time. At each timeslot, we evaluated the UQPV of every queue in a VOQ. Schedule for the next timeslot is based on the UQPV of a queue. Table 1 shows the computation of UQPV for timeslots $T_0$, $T_1$, $T_2$ and $T_3$. The process continues till the queue becomes empty.

5. PERFORMANCE EVALUATION

5. 1. Simulation Environment  
Proposed scheduler is simulated using BCSSIDM [18] with switch size $N=4$, 16, 64, 256 and 512. For any simulation, buffer size used is one cell and the switch is operated with speedup under non-uniform traffic patterns.

5. 2. Simulation Results

5. 2. 1. Average Waiting Time (AWT) Analysis  
Average Waiting Time is measured for the example stated in 4.3. It is assumed that all the cells arrived to the queue at the same time. Difference between the maximum and minimum AWT (D-AWT) of a queue is evaluated to identify the occurrence of starvation. Figure 2a shows the D-AWT of a queue under Bernoulli non-uniform iid traffic which is 0.5ms for D-PQRS comparable to 1ms and 3ms for PQRS and LQF-RR, respectively. Figure 2b shows the D-AWT of a queue under Bernoulli non-uniform bursty traffic which is 0.6ms for D-PQRS comparable to 1.3ms and 3.2ms for PQRS and LQF-RR, respectively. Therefore it is proved that D-PQRS considerably reduces the starvation comparable to PQRS and LQF-RR.

5. 2. 2. Throughput Analysis  
We examine the throughput of the scheduling algorithm by varying the switch sizes. Figure 3a depicts the relationship between the throughput and load under Bernoulli non-uniform iid traffic. There is no much difference in throughput for different switch sizes. Almost 95% throughput is achieved by all the switches when minimum load is offered. Throughput is almost maintained until 60% of load is supplied, then starts dropping to 80% at maximum load. Figure 3b depicts the relationship between the throughput and load under Bernoulli non-uniform bursty traffic. Similar to iid traffic, 95% of throughput is achieved by the algorithm for all the switch sizes, and drops to 75% when maximum load is offered.

5. 2. 3. Average Cell Latency Analysis  
Average cell latency is analyzed for the algorithm with different switch sizes. Figure 4a and 4b depict, when minimum load is offered, less than 5% of latency is occurred.
under non-uniform iid and bursty traffic. Over the time, latency constantly gets increased with respect to the load. A difference of 5% latency occurs between the switches 4x4 and 512x512 which confirms there is a little impact on switch size over latency.

5.2.4. Comparative Analysis  We examined the throughput and delay performance of DPQRS, PQRS and LQF-RR algorithms for switch size=256 under non-uniform traffic patterns. Figure 5a shows the throughput performance of all the algorithms under Bernoulli non-uniform iid traffic. Throughput of DPQRS is 90% at minimum load and 80% at maximum load which betters PQRS and LQF-RR by 5-10%. Figure 5b shows the throughput performance of all the algorithms under Bernoulli non-uniform bursty traffic. Throughput of DPQRS is 90% at minimum load and 78% at maximum load which betters PQRS and LQF-RR by 5-15%.
Figure 5c shows the delay performance of all the algorithms under Bernoulli non-uniform iid traffic. For DPQRS, average cell latency is nearly avoided at minimum load and 10% at maximum load which betters PQRS and LQF-RR by 7 and 17%, respectively. Figure 5d shows the average cell latency performance of all the algorithms under Bernoulli non-uniform bursty traffic. Average cell latency of DPQRS is 5% at minimum load and 15% at maximum load which betters PQRS and LQF-RR by 7 and 15%, respectively.

6. CONCLUSION

Designing an algorithm to achieve 100% throughput without starvation is a major challenge. D-PQRS reduces the starvation considerably and also achieves 80-95% throughput at different load structures, which is increased by 10% comparable to existing algorithms. The work can be extended to examine the algorithm with different buffer sizes and more research is also required to nullify the effect of starvation.

7. REFERENCES

Starvation Free Scheduler for Buffered Crossbar Switches

N. Narayanan Prasanth¹, K. Balasubramanian², R. Chithra Devi³

¹Department of Information Technology, National College of Engineering, Maruthakulam, India
²Department of Computer Science & Engineering, Mepco Schlenk Engineering College, Sivakasi, India
³Department of Information Technology, Dr. Sivanthi Aditanar College of Engineering, Tiruchendur, India

Paper history:
Received 17 November 2013
Received in revised form 28 November 2014
Accepted 29 January 2015

Keywords:
Buffered Crossbar Switch
Starvation
Router
Throughput
Delay

doi: 10.5829/idosi.ije.2015.28.04a.05
Performance analysis of buffered crossbar switch scheduling algorithms

Narayanan Prasanth*
Department of Information Technology,
National College of Engineering,
Maruthakulam, Tirunelveli – 27151,
Tamilnadu, India
Email: narayana.prasanth@gmail.com
*Corresponding author

Kannan Balasubramanian
Department of Computer Science and Engineering,
Mepco Schlenk Engineering College,
Sivakasi, Tamilnadu, India
Email: kannanbala@mepcoeng.ac.in

Abstract: The increasing demand for higher data rates on the internet requires routers that deliver high performance for high-speed connections. Nowadays high speed routers use the buffered crossbar switches, which have been the interest for research and commercialisation. In this paper, a study is made on the importance of buffered crossbar switches and their scheduling algorithms. Performance analysis is made for various buffered crossbar switch scheduling algorithms using non-uniform traffic patterns.

Keywords: buffered crossbar switches; BCS; routers; scheduling algorithms; throughput; delay; speedup.


Biographical notes: Narayanan Prasanth received his ME in Computer Science and Engineering from Anna University of Technology, Tirunelveli, India in 2009. He received his BTech in Information Technology from Pondicherry University, India in 2005. He is currently an Assistant Professor at National College of Engineering, Maruthakulam, Tirunelveli, Tamilnadu, India. His research area includes network switching and routing algorithms and its related issues.

Kannan Balasubramanian received his PhD in Computer Science and Engineering from University of California, Los Angeles, USA and MTech in Computer Science and Engineering from IIT Bombay, India and his MS (Tech.) in Computer Science from BITS, Pilani, India. He is working as a Professor at Mepco Schlenk Engineering College, Sivakasi, India. His research interest includes network architecture, protocols, security and performance issues.
1 Introduction

The exchange of information within the internet is made possible by the switches and routers interconnecting networks. These routers facilitate networks to communicate by using a shared language or protocol. Data’s are transmitted in the form of packets at a specified data rate. Switches and routers are required to process these packets. Packets are variable-length and therefore they are segmented before transmission at the input ports and switched from input to output in a cell-based fashion. Variable-length IP packets are reassembled at the output ports before they depart to other switches.

Packet switches identify the destination of packets at the input ports and forward them to the appropriate output ports, completing in this way the packet processing at layer 2 of the open system interconnection (OSI) model (Keslassy and McKeown, 2001; Yoshigoe, 2004). Those switches that find out information about the connectivity between networks and paths to reach different possible destinations are referred to as routers. This information is summarised in a forwarding table that is used to determine the output port of the switch according to the destination of the traversing packet. These routers are devices performing tasks at layers 2 and 3. Once output ports are defined in a forwarding table, a switch performs the scheduling and forwarding of packets.

Switching the packet from an input to an output port gets complex as there is the possibility that several packets need to go from different inputs to the same output. Therefore, this creates the necessity of interconnecting input and output ports, buffering packets and scheduling the packet switching time (Keslassy and McKeown, 2001; Yoshigoe, 2004). A scheduler selects a packet which is to be switched to the output at a time. Among the different types of switches, buffered crossbar switches (BCS) are considered viable and practical architectures for the design of high performance routers. It has good potential in overcoming the scheduling bottleneck experienced by alternative switching architectures. Crossbar switching fabrics are very popular for switch implementation because of their non-blocking capability, simplicity, and their market availability.

1.1 Buffered vs bufferless crossbar switches

As bufferless scheduling algorithms reach their practical limitations due to higher port numbers and data rates, internally BCS have started to attract researchers because of the great potential they have in solving the complexity and scalability issues faced by their bufferless predecessors (Keslassy and McKeown, 2001). The increasing demand for terabit switches and routers means that future commercial packet switches must be implemented with reduced scheduling complexity. The buffered crossbar architecture can inherently implement distributed scheduling schemes and has been considered a viable alternative to bufferless crossbar switches to improve performance. The presence of internal buffers drastically improves the overall performance of the switch as it offers two distinct advantages. First, the adoption of internal buffers makes the scheduling totally distributed, dramatically reducing the arbitration complexity. Second, and most importantly, these internal buffers reduce (or avoid) output contention as they allow the inputs to make cell transfers concurrently to a single output.

In most architecture buffered crossbars are preferred because they have simpler scheduling algorithms than a bufferless crossbar. In a bufferless crossbar, the scheduler must find a matching between inputs and outputs that leads to complex scheduling
algorithms. Also the buffered crossbar does not need a centralised scheduler since scheduling is distributed in BCS. Here, it can be pipelined to run at high speed, making buffered crossbars appealing for high performance switches and routers. Also it considerably reduces the hardware cost. Most researchers proved that buffered crossbars provide good throughput for admissible uniform traffic with simple algorithms. The paper is organised as follows. Section 2 shows the importance of BCS and their scheduling algorithms. Section 3 deals with the study of various scheduling algorithms. Section 4 produces the study report based on the comparisons of various scheduling algorithms.

2 Buffered crossbar switch

BCS have been considered a viable solution to improve the switching throughput as an alternative to bufferless crossbar switches (Yoshigoe, 2004; Mhamdi and Hamdi, 2003; Ye et al., 2010). In buffered crossbar architecture, buffering occurs exclusively at the crosspoints and is utilised to minimise cell loss. The number of ports is limited by the memory amount that can be implemented in a module chip. Figure 1 shows the buffered crossbar switch. Here small buffer memories are placed at the crosspoints and therefore it allows distributed scheduling decisions. Also, operation with variable-size packets now becomes feasible.

2.1 BCS scheduling

The process of selecting the cell to be switched at both input and output schedule is called scheduling. In BCS architecture, scheduling algorithms are used to select the cell/packet which is to be transmitted from input queue to crosspoint buffer and in-turn to output queue (Yoshigoe, 2004). The amount of buffering needed per crosspoint depends on the line rate multiplied by the backpressure round-trip time (RTT) product, which is affected by the packaging technology. In BCS, the switching operation actually consists of two scheduling phases named input and output schedule. During the input schedule, the scheduler will check the availability of crosspoint buffer for each port. If it is available, then the packets are switched. During the output schedule, the scheduler will look at all
crosspoint buffer and then randomly choose one cell uniformly from all occupied crosspoint buffer cells and remove the occupying packet and output it. This operation will be done in parallel during one time slot. The process of input schedule and output schedule is performed based on the scheduling algorithm used. Therefore performance of a switch is purely based on its scheduling algorithm.

2.2 Properties of scheduling algorithms

Some of the properties of scheduling algorithm are (Karimi et al., 2010, 2009):

- high throughput – an algorithm that keeps the backlog low in the VOQ’s, ideally the algorithm will sustain an offered load up to 100% on each input and output
- starvation free – the algorithm should not allow a nonempty VOQ to remain unserved indefinitely
- fast – to achieve the highest bandwidth switch, it is important that the scheduling algorithm does not become the performance bottleneck; the algorithm should therefore find a match as quickly as possible
- simple to implement – if the algorithm is to be fast in practice, it must be implemented in special-purpose hardware, preferably within a single chip.

2.3 Performance metrics

Some of the metrics which decides the performance of the switch scheduling algorithms are:

1. throughput
2. time factor (response time and time complexity)
3. delay performance (latency)
4. speedup (packet transmission time)
5. buffer size

The switch throughput is defined as the ratio between the output load and the input load of the switch. The maximum throughput is defined as the maximum input load after which the switch becomes unstable. Instability means that the input load is higher than the throughput of the switch; hence queues will keep growing indefinitely. The maximum throughput is also known as the saturation throughput of the switch and indicates the switch capacity. If the saturation throughput of a switch with a given scheduling algorithm equals to one, which is the maximum value due for a speedup of one, then the given scheduling algorithm is said to achieve 100% throughput. Given two scheduling algorithms both of which can achieve 100% throughput, the one with the shorter average cell delay is preferable. A scheduling algorithm is considered stable if it provides 100% throughput and it keeps the input buffer size bound in number of cells. Delay is measured as the period of time a cell is kept waiting in an input/internal/output buffer before being scheduled. Average delay is calculated from all the cells output during this period of time. Normalised load means the percentage of time slots, which have cells coming in, averaged over all inputs. Speedup – switches to have speedup of at least two, which
means that the crossbar needs to run twice faster than the input and output port. The speedup requirement increases the implementation cost and reduces the switch capacity.

3 Scheduling algorithms: a study

3.1 Maximum weight matching algorithm

In maximum weight matching (MWM) algorithms, a weight is attached with each request from inputs to outputs (Yoshigoe, 2004; Shen, 2010; Shen et al., 2007). The weight could be the number of cells in the VOQ, the waiting time of the HoL cell in the VOQ, etc. Some of the MWM algorithms are:

- **LQF**: The longest queue first (LQF) algorithm takes the number of cells in each VOQ as weight. The algorithm picks a match such that the sum of served queue lengths is maximised. LQF can lead to the starvation of some queues. Because it does not consider the waiting time of cells, queues with short length may be starved even though the wait time of their HoL cells surpasses the total weight time experienced by cells in a longer queue.

- **OCF**: The oldest cell first (OCF) algorithm uses the waiting times of HoL cells as weights. The OCF algorithm selects a match such that the sum of all served queues waiting time is maximised. Unlike the LQF algorithm, the OCF algorithm does not starve any queue and unserved HoL cells will eventually become old enough to be served.

3.1.1 Longest port first

Both longest port first (LQF) and OCF have high computation complexity of $O(N^3\log N)$ and it is impractical to implement them in hardware. The LPF algorithm (Shen, 2010; Shen et al., 2007) is designed to overcome the complexity problem of LQF and OCF and can be implemented in hardware at high speed. LPF uses the concept of both MWM algorithm and maximum size matching algorithm. LPF effectively finds the set of maximum size matches and from among this set, it chooses the match with the largest total weight. The weight of the LPF algorithm is the total number of cells queued at the input and output interfaces. This sum is called port occupancy, which represents the workload or congestion that a cell faces as it competes for transmission. Thus, a modified maximum size matching algorithm, which makes LPF less complex than LQF, is used to implement LPF. The process enables LPF to take advantage of both the high instantaneous throughput of a maximum size matching algorithm, and the ability of a MWM algorithm to achieve high throughput, and a small number of overflows even when the arriving traffic is non-uniform. LPF has a runtime complexity of $O(N^{2.5})$ which is lower than LQF. From the simulation results it is observed that the LPF can achieve 100% throughput for all traffic with independent arrivals. MWM scheduling algorithms (Yoshigoe, 2004; Shen, 2010; Shen et al., 2007) achieve 100% throughput under any admissible traffic. However, their good performance and stability come at the expense of high computation complexity. Also MWM has a significant complexity or require a large number of arithmetic operations when the number of inputs/outputs is large.
3.2 Approximating maximum size matching scheduling (AMSM) algorithms

Approximating maximum size matching algorithms (Yoshigoe, 2004) are fast and simple to implement in hardware with today’s technologies. They provide 100% throughput under uniform traffic and fairly good delay performance as well. However, they are not stable under non-uniform traffic. Most of the approximating maximum size matching algorithms are iterative algorithms.

3.2.1 Parallel iterative matching (PIM)

The PIM algorithm (Yoshigoe, 2004) attempts to approximate a maximum size matching algorithm by iteratively matching the inputs to the outputs until it finds a maximum size match. Each iteration consists of three steps:

1 request – each unmatched input sends a request to every output for which it has a queued cell
2 grant – if an unmatched output receives any requests, it grants one by randomly selecting from all requests
3 accept – if an input receives more than one grant, it accepts one by random selection.

The PIM algorithm faces some problems. First, randomness is difficult and expensive to implement at high speed. Each arbiter must make a random selection among the members of a time-varying set. Second, when the switch is oversubscribed, PIM can lead to unfairness between connections. Finally, PIM does not perform well for a single iteration: it limits the throughput to approximately 63%, only slightly higher than a FIFO switch.

3.2.2 Round robin matching algorithm (RRM)

RRM is the simplest algorithm (Yoshigoe, 2004; Karimi et al., 2010; Shah et al., 2007; Elhanany and Hamdi, 2007) for input queued switches. RRM work in three steps:

Step 1 Request. Each input sends a request to every output for which it has a queued cell.

Step 2 Grant. If an output receives any requests, it chooses the one that appears next in a fixed. Round-robin schedule starting from the highest priority element. The output notifies each input whether or not its request was granted. The pointer to the highest priority element of the round-robin schedule is incremented to one location beyond the granted input.

Step 3 Accept. If an input receives a grant, it accepts the one that appears next in a fixed round-robin schedule starting from the highest priority element. The pointer to the highest priority element of the round-robin schedule is incremented to one location beyond the accepted output. RRM delivers average delay performance under uniformly independent and identically distributed Bernoulli traffic. Updating the pointers at output arbiters is one of the reasons for poor performance. Synchronisation of the grant pointers also limits performance with random arrival patterns.
3.2.3 iSLIP scheduling algorithm

iSLIP is an iterative, round robin (RR) scheduling algorithm (McKeown, 1999) for input queued switches. It improves upon RRM by reducing the synchronisation of the output arbiters. It achieves this by not moving the grant pointers unless the grant is accepted. The Grant step of RRM is changed for iSLIP as follows: “If an output receives any requests, it chooses the one that appears next in a fixed round-robin schedule, starting from the highest priority element. The output notifies each input whether or not its request was granted. The pointer to the highest priority element of the RR schedule is incremented to one location beyond input if and only if the grant is accepted”. This leads to:

- lowest priority is given to the most recently made connection
- no connection is starved
- under heavy load, all queues with a common output have the same throughput.

From the simulation results (McKeown, 1999) it is found that iSLIP works similar to RRM and FIFO when the load is less. If the load is high, iSLIP gives better performance than the other two. The biggest drawback with iSLIP is complexity in implementing iSLIP in hardware.

3.3 Randomised algorithm (RA)

RA is a distributed switch scheduling algorithm (Mekkittikul and McKeown, 1998). The motivation for proposing randomised algorithms is to overcome the complexity of MWM algorithms and to achieve stability under any admissible traffic. It holds unit mean exponential random variables on the input side. By adjusting the intensity of the variables, the input is selected. Now, at some fixed time, for each input-output pair (i, j) set a timer that expires in \( V_{ij}(t) \) time units; these units do not correspond to time slots needed to switch a packet from an input to an output. Upon the expiration of the timer (i, j), input i is matched to output j unless either of them is already matched. This process continues until all timers expire. The timers are set based on local parameters only. However, the complexity of the implementation is not negligible due to the required sources of randomness. There exist at least two possible ways to obtain randomness:

1. **external** – a physical device (e.g., photon-based) auxiliary to the switching fabric can serve as a source of randomness (Poisson process); since these devices do not have to be implemented in silicon, they can potentially operate at very high speeds
2. **internal** – randomness can be extracted from packet payloads.

The algorithm (Mekkittikul and McKeown, 1998) with an appropriately chosen weight function has the following desired characteristics: High throughput, Starvation free and Simple to implement. Based on analysis report, it is found that the algorithm remains stable under high traffic loads and performs competitively in comparison with the MWM algorithm. Some of the examples for randomised algorithms are TASS, APSARA, LAURA, SERENA, etc.
3.4 **Independent scheduling algorithm (ISA)**

ISA is a simple packet scheduling algorithm (Balasubramanian and Sindhu, 2011) for the buffered crossbar switch. Input scheduling and switch scheduling of ISA are conducted in an independent and distributed manner. Both the input and the output ports make scheduling decisions based on the state information of the crosspoint buffers. Hamiltonian walk is used on the input side to select one of its VOQ’s which has the highest weight and whose corresponding crosspoint buffer is empty. The weight is calculated by the product of Hamiltonian weight and the queue length of the VOQ. The VOQ with the highest weight is sent to the crosspoint buffer. The crosspoint buffer is selected in RR fashion. Switch scheduling is similar to input port scheduling. When the transmission channel of an output port from the crosspoint buffers is idle, the output port selects a crosspoint buffered packet and saves it in its output queue. The analysis report (Balasubramanian and Sindhu, 2011) suggests that the time complexity in ISA is less because of independency in scheduling at input and output. ISA avoids packet drops therefore it achieves 100% throughput.

3.5 **Distributed scheduling algorithm (DISQUO)**

DISQUO is distributed scheduling algorithm (DISQUO) (Ye et al., 2010) for BCS. It achieves 100% throughput under any admissible arrival traffic with only a one-packet buffer at each crosspoint. In DISQUO, the input actions are performed at the beginning of each time slot and outputs transmit packets from the crosspoint buffers before the end of each time. The output ports have to learn the inputs decisions. The key point of DISQUO is that by observing crosspoint buffers, an input and an output can learn each other’s decisions implicitly. Simulation result shows that it can provide very good delay performance under different traffic arrivals. The simulation results also show that, by using DISQUO, packet delay is very weakly dependent on the switch size, which means that DISQUO can scale with the number of switch ports. The issue with DISQUO is its high implementation complexity and also it needs to keep state information.

3.6 **Counting method with a buffered crossbar (CM)**

Crossbar switches works in two phases such as input and output scheduling. The input scheduling policy gives preference to cells based on the input priority list. Similarly, the output scheduling policy gives preference to cells based on the output priority list. Therefore the cell which has high priority in both the scheduling policy will be switched first. Other cells will result in slackness. To overcome this issue, CM uses group by virtual output queue insertion policy (GBVOQ) (Pan et al., 2010):

1. When a cell arrives to a non-empty VOQ, the cell is inserted in the input priority list just behind the last cell belonging to the same VOQ. This ensures that cells destined to the same output are ordered based on departure order.

2. When a cell arrives to an empty VOQ, the cell is inserted at the head of the input priority list. The CM using GBVOQ provides rate and delay guarantees between each input/output pair. This method suffers when the input traffic is non-uniform and in most of the situations it fails to achieve 100% throughput (Chrysos and Katevenis, 2011). Further research is required to sort out the issues.
3.7 Critical buffer first (CBF)

In CBF, youngest internal buffer first (YBF) algorithm (Pan and Yung, 2006) is used at the input side whereas oldest internal buffer first (OBF) algorithm is used at the output side. It is known that the information on the internal buffers is sufficient for the schedulers to make effective decisions. The CBF scheme, especially its output scheduling OBF, is equivalent to the buffer-less oldest port first (OPF) algorithm (Pan and Yung, 2006). The output scheduling, OBF, performs its arbitration based on the input waiting time function used by OPF, defined as the waiting time sum of all HoL cells at the input and waiting for the same output. Input scheduling YBF, however, uses an opposite arbitration criterion to the output waiting time function used by OPF. Instead of giving priority to the request with the greatest output waiting function as in OPF, YBF gives priority to the request with the smallest output waiting time instead. In order to achieve fairness, both YBF and OBF maintain each a highest priority pointer to break ties in the presence of conflicts. Each input (respectively output) pointer is initialised to the index of the input (respectively output) it belongs to.

3.7.1 Current arrival first-priority removal (CAF_PRMV)

The input scheduling gives priority to the newly arriving packets while the output scheduling completes this task by serving the recently arrived packets to the internal buffer. The intuition behind this is to overcome the lack of performance under the non-uniform traffic without using any weight functions or state information. At each time slot, the current arrival first (CAF) algorithm (Pan and Yung, 2006) checks whether there is a new cell arriving at the input port. To accomplish this task in the output scheduling, CAF assigns a priority level to each cell leaving the input port. This level will decide the priority (urgency) of that cell in the output scheduling phase. The use of priority levels is an efficient choice. First, the output scheduling phase will be much simpler and faster than sorting for example. Second and most importantly, the adoption of priority levels makes the implementation easy and the hardware requirement simple. Simulation result shows it performs well at all traffics. It is totally stateless and requires simple hardware while working at very high speed.

3.8 Most critical buffer first (MCBF)

MCBF is a high performance novel scheduling algorithm (Mhamdi and Hamdi, 2003) for BCS. The main objective is to reduce the arbitration time. MCBF uses shortest internal buffer first (SBF) at the input side and longest internal buffer first (LBF) on the output side. Information used relies only on internal buffers which is sufficient for the schedulers to make effective decisions while being simple to implement in hardware. MCBF does not use any input state information, such as VOQs occupancies or VOQs head of-line (HoL) cells waiting time. MCBF is simple to implement and reduced hardware complexity compared to LQR, OCF, etc. MCBF’s scheduling decision is based on the number of cells in the internal buffers therefore it takes less time for arbitration. MCBF is a scheme which is almost stateless. It makes its arbitration without any type of state information about the input VOQs. The only feedback information that MCBF needs to have during its arbitration process is whether an input VOQ is empty or not.
Based on performance analysis it is understood that MCBF is stable under bursty uniform and Bernoulli non-uniform traffic.

3.9 Fair asynchronous segment scheduling (FASS)

FASS (Karimi et al., 2010) is designed for BCS without speedup to achieve constant performance guarantees with reduced crosspoint buffers. Here, the packets are segmented and transmitted, so as to reduce the crosspoint size. Two types of scheduling algorithm are used: input and output scheduling algorithm. Both requires the status of the crosspoint to send or receive segments. At input scheduling, timestamp is used to note down the input start and end time. A segment is eligible for input scheduling if its virtual input start time is smaller than or equal to the current system time. Similar approach is used at the output scheduling. From the simulation results it is understood that the FASS has strong switch stability by showing that the length of input virtual queues are finite. Therefore it achieves 100% throughput. Also, FASS requires no speedup for the crossbar, reducing implementation cost and improving switch capacity. The drawback is FASS practically achieves 100% throughput but the throughput slightly decreases when the segment size increases, because with the same simulation time, larger segments sizes have smaller probabilities to finish the transmission of the last segment.

3.10 LQF-RR Scheduling

LQF-RR scheduling approach (Banovic and Radusinovic, 2008) is proposed as a high throughput scheduling algorithm for BCS with one buffer per crosspoint. LQF and round robin (RR) schedulers are used as input and output scheduling algorithms respectively. Simulation results show that this approach is stable for any loading where the input/output load is $\leq 1/N$ and also it delivers 100% throughput by using fluid model techniques. This approach provides a consistent performance for both uniform and non-uniform traffics.

3.11 Stable queuing implementable design (SQUID)

Initially, a stable queue input-output scheduler with Hamiltonian walk (SQUISH) (Shen, 2010) for crosspoint buffered switches is proposed. Here, a crosspoint buffered switch with finite crosspoint buffers is used with no speedup. Each input first makes decisions on which crosspoint to send a cell based on buffer occupancy information. Each output simply chooses the longest available queue for service. Then, a comparison with a Hamiltonian walk schedule is applied to choose the schedule with a greater weight. The SQUISH algorithm can be shown to achieve 100% throughput for any admissible Bernoulli traffic. The complexity of SQUISH is only $O(\log N)$ low enough to make practical implementations feasible. To further reduce the complexity, SQUID (Shen, 2010) is proposed. With the availability of SQUISH and SQUID, the scheduler can serve additional cells in a time slot as compared to a bufferless crossbar because of the looser coupling between input and output scheduling facilitated by the availability of
crosspoint buffers. This reduces buffer occupancies, leading to significant delay performance improvement. SQUID can achieve 100% throughput for any admissible Bernoulli traffic, with the minimum required crosspoint buffer size being as small as a single cell buffer. Simulation shows that for most realistic traffic scenarios, the delay performance of SQUID is close to that of an ideal output-queued switch. Only under highly loaded (0.99) log-diagonal and diagonal traffic is the delay greater than an output-queued switch.

3.12 Prioritised queue with round-robin scheduler (PQRS)

It uses queue priority (Prasanth and Balasubramanian, 2014) at the arrival schedule and round-robin scheduler at the departure schedule. Selection of queue for cell transfer from VOQ to a crosspoint is based on the queue priority. Queue priority is the number of cells occupied in the queue. For every switch, a bonus priority value of 1 will be distributed to all the queues in the VOQ and is summed with the actual priority. Bonus is not applicable to the queue which is used in the current timeslot. Result (Prasanth and Balasubramanian, 2014) shows it outperforms the existing algorithms for non-uniform traffic patterns.

4 Performance analysis

4.1 Simulation environment

We used the simulator BCSSIM (Prasanth and Balasubramanian, 2014) that models the buffered crossbar switch of size $N \times N$. In general for all the experiments, we used a $4 \times 4$ VOQ/BCS switch with a buffer size of 1 and no speedup is introduced at any stage. Buffered Crossbar Switch Scheduling Algorithms such as DISQUO, LQF-RR, PQRS, MCBF, ISA and FASS are used for analysis. Input for BCSSIM is supplied through Bernoulli non-uniform iid traffic and Bernoulli non-uniform bursty traffic in-turn throughput and average cell latency are computed.

4.2 Simulation results and discussion

Table 1 shows the performance of various BCS scheduling algorithms for different load structures. Figure 2 shows the throughput of all the algorithms with load = 100,000 bits per schedule for Bernoulli non-uniform iid traffic. Result proves PQRS outperforms all other algorithms by more than 5% and it is understood when load exceeds 50% then the throughput starts dropping for all the algorithms. At the maximum, MCBF attains 55% of throughput when maximum load is offered. PQRS and LQF-RR provides a consistent performance throughout the simulation. Figure 3 shows the average cell latency of all the algorithms. PQRS completely outperforms all other algorithms by a minimum of 5% delay. Comparing other algorithms, average delay time varies with respect to load. In the worst case, FASS introduces more delay of 35% when maximum load is offered. There is no much difference between the DISQUO, ISA and MCBF.
Table 1  Performance of various scheduling algorithms

<table>
<thead>
<tr>
<th>Scheduling algorithm</th>
<th>Traffic pattern</th>
<th>Metric</th>
<th>Load offered</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Throughput (%)</td>
<td>25%</td>
</tr>
<tr>
<td>PQRS</td>
<td>Non-uniform iid</td>
<td>Throughput (%)</td>
<td>90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>2</td>
</tr>
<tr>
<td>LQF-RR</td>
<td>Non-uniform iid</td>
<td>Throughput (%)</td>
<td>82.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>8</td>
</tr>
<tr>
<td>MCBF</td>
<td>Non-uniform iid</td>
<td>Throughput (%)</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>10</td>
</tr>
<tr>
<td>ISA</td>
<td>Non-uniform iid</td>
<td>Throughput (%)</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>12</td>
</tr>
<tr>
<td>DISQUO</td>
<td>Non-uniform iid</td>
<td>Throughput (%)</td>
<td>72.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>15</td>
</tr>
<tr>
<td>FASS</td>
<td>Non-uniform iid</td>
<td>Throughput (%)</td>
<td>67</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Delay (%)</td>
<td>20</td>
</tr>
</tbody>
</table>

Figure 2  Throughput as a function with respect to load for Bernoulli non-uniform iid traffic (see online version for colours)
Figure 3  Average cell latency as a function with respect to load for Bernoulli non-uniform iid traffic (see online version for colours)

Figure 4 shows the throughput of all the algorithms with load = 100,000 bits per schedule for Bernoulli non-uniform bursty traffic. Result shows PQRS and LQF-RR outperforms all other algorithms by more than 5% and it is understood that when load exceeds 50% then the throughput starts dropping. At the maximum, FASS attains 53% of throughput when maximum load is offered. Throughput difference between the PGRS and FASS is around 25%. LQF-RR shows consistent performance until 70% load is offered and it drops to 77% at maximum load. Figure 5 shows the average cell latency between the compared algorithms. There is no difference between MCBF, LQF-RR and PQRS where all possess a delay between 5% and 30%. Delay time of other algorithms is not as expected with a worst scenario possessed by FASS, delay of 48% when maximum load is offered.

Figure 4  Throughput as a function with respect to load for Bernoulli non-uniform bursty traffic (see online version for colours)
Simulation results show there is no algorithm achieves 100% throughput without delay under non-uniform traffic patterns. It is understood that starvation has a huge impact over the throughput of these algorithms. Among all the algorithms, PQRS handled the starvation to some extent results in better throughput. But still considerable amount of research is needed to achieve maximum throughput in BCS.

5 Conclusions

BCS have become one of the unavoidable forces in the next generation networks. This is because of the need of high speed internet to the electronic community. High speed router uses BCS for the delivery of high speed internet. Therefore, BCS received a lot of interest from both research and industrial communities. Our study shows the importance of BCS and their scheduling algorithms along with their properties and performance metrics. Also studied about the various crossbar buffered scheduling schemes and performance analysis is made with them. From the analysis it is understood that PQRS and LQF-RR provides better throughput and delay performance compared to others but the fact is more research is required to further improve the results without starvation.

References


Performance analysis of buffered crossbar switch scheduling algorithms


IMPACT OF BUFFER SIZE ON PQRS AND D-PQRS SCHEDULING ALGORITHMS

N. Narayanan Prasanth¹, Kannan Balasubramanian² and R. Chithra Devi³

¹Department of Information Technology, National College of Engineering, India
E-mail: nmprcd@gmail.com
²Department of Computer Science and Engineering, Mepco Schlenk Engineering College, India
E-mail: kannanbala@mepcoeng.ac.in
³Department of Information Technology, Dr. Sivanthi Aditanar College of Engineering, India
E-mail: chitra_rajan2001@yahoo.co.in

Abstract
Most of the internet applications required high speed internet connectivity. Crosspoint Buffered Switches are widely used switching architectures and designing a scheduling algorithm is a major challenge. PQRS and D-PQRS are the two most successful schedulers used in Crosspoint Buffered Switches under uniform traffic. In this paper, we analysed the performance of PQRS and DPQRS algorithms by varying the crosspoint buffer size. Simulation result shows the delay performance of the switch increases if the size of the buffer increases.

Keywords:
Crosspoint Buffered Switch, Scheduling Algorithm, Unicast Traffic, Throughput and Delay Performance

1. INTRODUCTION

To attain high speed internet connectivity, crossbar fabric is used in switches and router implementations. Input and Output queued switches have their own limitations such as input-output port contention and speedup requirements. Combined Input Crosspoint Queued switch is complex to implement [3] which leads to automatic choice of Buffered Crossbar Switches (BCS). BCS is the commonly used switching architecture among different crossbar switches because of its simplicity and internal non-blocking capabilities [1], [2]. Buffer located in the crosspoint reduces scheduling overhead; Head of Line (HOL) cell blocking and input-output contention of a switch thereby improves throughput and delay performance [3], [4]. It is further improved by introducing virtual output queue (VOQ) in the BCS which completely eliminates HOL blocking. BCS permit their input and output ports to take scheduling decisions independently which avoids the need for centralized scheduler [3].

Designing a scheduling algorithm for BCS have received significant research attentions. Every BCS has a buffer at the crosspoint and its size is based on the employed scheduling algorithm. Many algorithms offer good performance with buffer of size one cell length but recent research focused on variable sized buffers. In [5], an ideal throughput is achieved for different buffer lengths under uniform Bernoulli i.i.d and non-uniform log-diagonal traffic patterns. In [6], a mathematical model for 2x2 BCS is proposed for larger crosspoint buffers, results in improved throughput and delay performance. In [7], simulation shows that RR-RR scheduler cannot be sufficient to provide 100% throughput with small buffers unless some speedup is introduced. A 32 x 32 switch with buffer capable of holding up to 1000 cells is implemented [3] to provide high performance. This encouraged us to analyse the Prioritized Queue with Round-robin Scheduler (PQRS) [8] and Delay based Prioritized Queue with Round-robin Scheduler (D-PQRS) [9] with broad range of buffer lengths. In this paper, we analysed the throughput and delay performance of PQRS and D-PQRS scheduling algorithms by varying the buffer size 1, 2, 4, 16, 64, 256 and 512. Section 2 discusses about PQRS and D-PQRS scheduling algorithms. Section 3 shows the performance of these algorithms with different buffer sizes and section 4 concludes the paper.

2. BUFFERED CROSSBAR SWITCH

The buffered crossbar switch with virtual output queue is shown in Fig.1. BCS holds buffer in the switch fabric rather than in the line cards which means the switch and buffer implemented in a single chip thereby reducing the implementation cost. At each timeslot, BCS requires two schedulers to switch a cell namely Arrival and Departure Schedule. Arrival Schedule selects a cell from the HOL of a queue and placed it in an empty crosspoint buffer and in parallel Departure Schedule selects a cell from a non-empty buffer transferred to output through respective port [8, 9]. At each timeslot, based on the employed scheduling algorithm, a cell is scheduled from VOQ to crosspoint buffer and from buffer to output port. Amount of cells stored in the crosspoint buffer is based on its size and it is practically viable to implement multi-sized buffer in the crosspoint of a switch.

2.1 SCHEDULING ALGORITHMS

PQRS uses Priority Queue Scheduler as input schedule and Round-robin algorithm as output schedule [8]. It is designed and simulated under Bernoulli non-uniform bursty and i.i.d. traffic with buffer size 1. Simulation result shows the difference between minimum and maximum average waiting time is less than 1ms thereby the algorithm considerable reduces starvation. D-PQRS uses Delay based Priority Queue Scheduler as input schedule and Round-robin algorithm as output schedule. Simulation result

Fig.1. Buffered Crossbar Switch with Virtual Output Queue

1235
shows that D-PQRS outperforms PQRS and LQF-RR algorithms in achieving high throughput with minimum delay and starvation effect [9].

2.2 BUFFER SIZING

The process of placing a buffer in the crosspoint of BCS/VOQ is termed as Buffering. It avoids input-output contention problem by distributing the scheduling schemes. Considering the hardware limitations and switch performance, it is difficult to finalize the buffer size. Buffers with ideal sizes can prevent the switch from packet overflowing. Normally buffer size of 1 cell size is used by the researchers for stable performance. In certain cases, researchers increased the buffer size to further increase the switch performance which resulted in high implementation cost [6].

Propagation Delay, Queueing Delay and Transmission Delay are the three components involved in end to end latency of a moving packet [10]. Among the three components, queueing delay is the only variable component which is controlled by buffer sizing. A correct sized buffer will considerably reduce the implementation and operational complexity of a switch. Therefore to identify the appropriate buffer size, the performance of PQRS and D-PQRS algorithms are analysed by varying its buffer size.

3. PERFORMANCE ANALYSIS

Throughput, average cell latency and packet loss are the three parameters which decide the switch performance. Three different traffic patterns such as Uniform Bernoulli traffic, Non-uniform Bernoulli i.i.d. traffic and Non-uniform Bernoulli Bursty traffic are used to analyze the 4×4 switch performance. Each simulation is conducted with one million timeslots of load ranging from probabilities \( p = 0.1 \) to 1. At each traffic patterns, performance of the switch is noted for different buffer sizes such as 1, 2, 4, 16, 64, 256 and 512.

3.1 THROUGHPUT ANALYSIS

In each simulation, throughput of a switch is defined as the ratio of the cumulative number of cells entering it successfully by the cumulative number of cell arrived. Throughput is observed for both the algorithms with different buffer size under uniform and non-uniform traffic patterns. In Fig.2, throughput as a function of buffer length under Uniform Bernoulli traffic BCS provides a minimum throughput of 98% for buffer sizes 1 to 4 and maximum throughput of 100% for buffer size greater than 64. Variation in throughput between PQRS and DPQRS is less than 2% for all buffer sizes. Throughout the simulation, switch uses the same load therefore practically it is understood that the larger buffer offers more throughput than the smaller one. The Fig.3 depicts the throughput performance of switch under Bernoulli non-uniform i.i.d. traffic. PQRS and DPQRS achieves a minimum of 90% and 94% respectively for buffer size 1 and achieves a maximum of 96% and 92% for buffer size 2 and 4. There is no further improvement in the throughput performance even if the buffer size is increased from 16 to 512. This is because the load used to analyze the performance of the switch is same irrespective to buffer sizes. As a result, an average 3-4% increase in throughput is achieved by DPQRS and PQRS when buffer size is increased as 2 and 4. It will be very much interesting to analyse the algorithms with constant increase in load structure.
Fig.4. Throughput as a function of buffer length under Bernoulli Non-uniform Bursty Traffic

Behaviour of the switch is depends on the employed scheduling algorithm and its traffic patterns. From the simulation results, it is understood that the buffer size has a little influence over the scheduling algorithms under both uniform and non-uniform i.i.d. traffic. But for bursty traffic, there is no impact on the schedulers.

3.2 AVERAGE CELL LATENCY

In each simulation, average cell latency is defined as the average waiting time of cells traversing through the input and output port of the switch. In all the simulation experiments, average cell latency is expressed in terms of milliseconds (ms). The Fig.5 shows the average cell latency of a switch for both the algorithms under Bernoulli uniform traffic. Less than 5ms of delay is noted for buffer size less than 16 and for buffer size greater than 64, switch operates without delay. This is because the load structure used for different buffer size is same and therefore switch with greater buffer size can operate without delay under uniform traffic.

Average Cell Latency of the switch under Bernoulli non-uniform I.I.D. traffic and Bernoulli non-uniform traffic are depicted in Fig.6 and Fig.7 respectively. For I.I.D. traffic, an average delay of 20-25ms is measured for both the algorithms with buffer size less than 4 and an average delay of 10-15ms is measured with buffer size greater than 4. An average difference of 5% delay is noted between DPRS and PQRS algorithms. Also it is clear that an increase in buffer size will reduce the cell delay by 10ms. For bursty traffic, average delay ranges from 22 to 32ms for PQRS and from 12 to 22ms for DPQRS and a 10ms difference is noted between them. Throughout the simulation, larger buffers hold less delay than shorter buffers because we use the same load structures. Switch with small buffer has long waiting time for the cells in the VOQ and therefore it offers more delay.
4. CONCLUSION

The paper analysed the influence of buffer size in PQRS and DPQRS algorithms. Throughput performance of the switch is increased by 2% when large sized buffers are used. Under uniform traffic, average cell latency is null for buffer size greater than 16. Under non-uniform traffic patterns, an increase in 10ms is noted for larger buffers compared to smaller ones. For different load structures, delay analysis with different buffer sizes would be interesting. Larger sized buffers have a significant impact in case of average cell latency rather than throughput performance.

REFERENCES


### LIST OF CORRECTIONS

<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>List of corrections as suggested by the Examiner</th>
<th>As existing in the thesis</th>
<th>As corrected in the thesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In page no. 18, “two types of Uniform Traffic Models are Uniform traffic and Non-Uniform Traffic”</td>
<td>Two types of Uniform Traffic Models are Uniform traffic and Non-Uniform Traffic</td>
<td>Two types of traffic models are Uniform traffic and Non-Uniform Traffic</td>
</tr>
<tr>
<td>2</td>
<td>In page no. 20, “Average Cell Latency is computed through the averages of all cell delays” – all the cell delays of during some period</td>
<td>Average Cell Latency is computed through the averages of all the cell delays.</td>
<td>Average Cell Latency is computed through the averages of all the cell delays measured during the respective simulation period or for the amount of cells used for switching.</td>
</tr>
<tr>
<td>3</td>
<td>In page 45, RRS is used but it is not included in “List of abbreviations”</td>
<td>RRS is not included in “List of abbreviations”</td>
<td>RRS included in “List of abbreviations”</td>
</tr>
<tr>
<td>4</td>
<td>Eq. (1) is not clear</td>
<td>$B_a \leq B$ &amp; $B_d \geq B$</td>
<td>$B_a \leq B(n)$ &amp; $B_d \geq B(n)$</td>
</tr>
<tr>
<td>5</td>
<td>Timeslot &lt;=0, how is it?</td>
<td>Timeslot $t_{sl} \leq 0$</td>
<td>Timeslot $t_{sl} = 1$</td>
</tr>
<tr>
<td>6</td>
<td>In figure, unit of Average Waiting Time is not mentioned.</td>
<td>Unit of Average Waiting Time unit is not mentioned in the figure 3.1</td>
<td>Average Waiting Time unit is mentioned in Fig. 3.1</td>
</tr>
<tr>
<td>7</td>
<td>In page no. 73, Fig.1 buffered crossbar switch is absent.</td>
<td>The buffered crossbar switch with virtual output queue is shown in Fig.1.</td>
<td>The buffered crossbar switch with virtual output queue is shown in Fig.1.11</td>
</tr>
<tr>
<td>8</td>
<td>Page no.75, last sentence is not clear</td>
<td>PQRS and DPQRS achieves a minimum of 90% and 94% respectively for buffer size 1 and achieves a maximum of 96% and 92% for buffer size 2 and 4.</td>
<td>PQRS achieves minimum throughput of 90% when buffer size is 1 and maximum throughput of 92% when buffer size is 4 or above. DPQRS achieves minimum throughput of 93% when buffer size is 1 and maximum throughput of 96% when buffer size is 4 or above.</td>
</tr>
<tr>
<td>9</td>
<td>In page no. 96, cell delay gets reduce by buffer size increases. How is it?</td>
<td>Simulation shows there is no significant increase in throughput for different buffer sizes but cell delay gets reduced with respect to increase in buffer size.</td>
<td>Simulation shows there is no significant increase in throughput for different buffer sizes but cell delay gets reduced with respect to increase in buffer size. This is achieved by switching more number of cells in a single timeslot based on the availability of buffer. Also, the size of the input is not changed throughout the simulation.</td>
</tr>
</tbody>
</table>

**SUPERVISOR**
<table>
<thead>
<tr>
<th>Sl. No.</th>
<th>List of corrections as suggested by the Examiner2</th>
<th>As existing in the thesis</th>
<th>As corrected in the thesis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>In page no.2, fig.1.1 shows data up to the year 2014, now it is appropriate to update the figure to include data up to 2016</td>
<td>Fig.1.1 shows data up to the year 2014</td>
<td>Fig.1.1 data is updated up to the year July 2016</td>
</tr>
<tr>
<td>2</td>
<td>In page no.17, properties (desirable) of scheduler be discussed in brief, like what do you mean by less implementation cost? Is it in terms of space and time?</td>
<td>In certain cases, researchers increased the buffer size to further increase the switch performance which resulted in high implementation cost</td>
<td>In certain cases, researchers increased the buffer size to further increase the switch performance which resulted in high implementation cost which refer both space and time requirement.</td>
</tr>
<tr>
<td>3</td>
<td>In page no.17, what do you mean by “Execute the scheduler without speedup”?</td>
<td>Execute the scheduler without speedup</td>
<td>Execute the schedulers without speedup i.e. use of speedup in the switch fabric should be avoided</td>
</tr>
<tr>
<td>4</td>
<td>Section 1.5 Simulation Environment on page no.18 be moved to later chapter or appendix</td>
<td>1.5 Simulation Environment is available at page no.18 of Chapter1</td>
<td>As the simulator is used from Chapter 2 onwards, it is necessary to provide simulator information in Chapter 1 itself.</td>
</tr>
<tr>
<td>5</td>
<td>HOL full form is mentioned in page no.29, but it is used in earlier pages. Mention the full form along with the short form in the first usage. Subsequent instances you may use the short form.</td>
<td>HOL full form should be used in the first usage and for subsequent instance use short form.</td>
<td>HOL full form is available at page no.10 of Chapter 1. Therefore HOL short form is used in page no.29.</td>
</tr>
<tr>
<td>6</td>
<td>HOL or HoL? Be consistent throughout the thesis</td>
<td>HOL and HoL are used in different scenarios.</td>
<td>HOL is used. Corrections are carried out in page no. 24 and 28</td>
</tr>
<tr>
<td>7</td>
<td>Objective 4 on page no.43 can be reframed properly.</td>
<td>Design, implement and test the proposed scheduling algorithm for buffered crossbar switches</td>
<td>Propose new scheduling algorithms for buffered crossbar switches</td>
</tr>
<tr>
<td>8</td>
<td>Best, average and worst case performance of the proposed algorithm be discussed in detail in section 3.4 and 4.4</td>
<td>In section 3.4 and 4.4, best, average and worst case performance of the proposed algorithms are not available</td>
<td>In section 3.4 and 4.4, best, average and worst case performance of the proposed algorithms are included</td>
</tr>
<tr>
<td>9</td>
<td>Fig.5.1 graph be plotted for some more values of buffer size, 8 and 10 to see the effect of buffer size correctly and infer the conclusion appropriately</td>
<td>In fig.5.1, buffer size used in the graph for computing the throughput under uniform traffic are from 1 to 7</td>
<td>In fig.5.1, buffer size used in the graph for computing the throughput under uniform traffic are 1, 2,4,16, 64, 256 and 512</td>
</tr>
<tr>
<td>10</td>
<td>Section 7.1 contributions, on page no.95 be devoted to highlight only the research contribution to this thesis, but not the detailed study</td>
<td>In section 7.1 Contributions, detailed study made on the existing algorithms is included</td>
<td>In section 7.1 Contributions, detailed study made on the existing algorithms is removed</td>
</tr>
</tbody>
</table>