Buffered Crossbar based Parallel Packet Switch

Zhuo Sun, Masoumeh Karimi, Deng Pan, Zhenyu Yang and Niki Pissinou
Florida International University
Email: {zsun003,mkari001, pand}@fiu.edu, yangz@cis.fiu.edu, pissinou@fiu.edu

***Abstract*—A parallel packet switch (PPS) provides huge aggregate bandwidth by combining the capacities of multiple switching fabrics. Most existing PPSs use output queued switches as the switching fabrics, which require speedup and result in high implementation cost. In this paper, we present a buffered crossbar based parallel packet switch (BCB-PPS), whose switching fabrics need no speedup. We propose the Batch-WF^{2}Q algorithm to dispatch packets to the parallel switching fabrics, and leverage the sMUX algorithm in [7] to schedule packet transmission for the switching fabrics. Such a design enables a simple round-robin algorithm to efficiently collect packets from the switching fabrics. In addition, our design requires no packet resequencing, and thus needs no buffers at either external or internal outputs. We show that BCB-PPS has tight delay guarantees and bounded buffer sizes. Finally, we present simulation data to verify the analytical results and to evaluate the performance of our design.**

I. Introduction

With rapid development of broadband based multimedia applications, there is an increasing demand for more bandwidth and better service [3]. It has been more and more difficult for traditional single switching fabric based switches to sustain such huge bandwidth. Parallel packet switches (PPSs) have thus emerged as a promising solution, which combine the capacities of multiple switching fabrics to provide large aggregate bandwidth. A typical PPS is shown in Figure 1, consisting of three parts: N external inputs, M internal switches, and N external outputs. External inputs dispatch incoming packets to internal switches, which switch the packets and send them to external outputs.

There are a number of PPSs [1] [6] [8] [10] [14] [18] in the literature. The PPS in [8] uses multiple output queued (OQ) switches as internal switches, and the external switch emulates an OQ switch with bounded delay and without resequencing. [8] also shows that if there is a coordination buffer of size NM cells at each external input and output, the external can emulate a wide variety of quality-of-service queueing disciplines. However, OQ switches require speedup of N, i.e. the switching fabric running N times faster than the input and output, and are not practical [5]. [1] presents a PPS that uses virtual input queues at external outputs to achieve packet-level load balancing and improve transmission delay. Similar to [8], it uses OQ switches as internal switches. [6] proposes a variable-length PPS to improve delay at external outputs, by introducing a waiting-length mechanism to achieve maximum utilization of the external output. However, the delay at the external input is worse than that of a simple round-robin algorithm. [10] proposes the multiple input-output-queued (MIOQ) switch consisting of two parallel internal switches. The MIOQ switch can exactly emulate an OQ switch running a variety of scheduling algorithms. However, the design considers only two internal switches and is not scalable. [18] presents the Flow-Mapping PPS (FM-PPS) with flow level load balancing. It ensures the packet order of each micro flow without costly resequencing. The internal switches are also OQ switches. In addition, FM-PPS needs buffers at external outputs, increasing hardware cost. [14] designs a PPS to eliminate speedup by using multiple input-output queued internal switches. However, the internal switches operate at the same line speed as the external switch, which makes the PPS unable to provide higher aggregate bandwidth. [15] designs a large scale ATM switch similar with PPS to employ multipath parallel distribution approach at the cell level. As can be seen, most existing PPSs use OQ switches or other switches that require speedup as internal switches, resulting in high implementation cost.

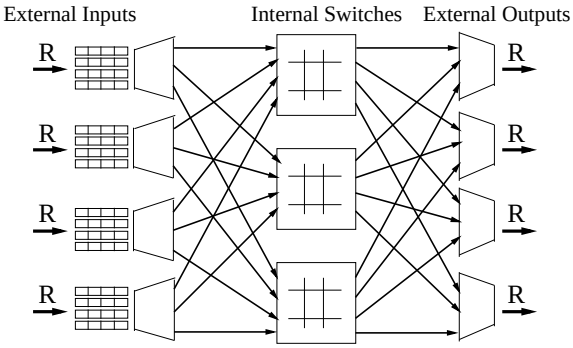


Figure 1. Architecture of Proposed Parallel Packet Switches.

In this paper, we present a buffered crossbar based parallel packet switch (BCB-PPS), which uses the sBUX buffered crossbars [7] without speedup as internal switches. It has been a common technique [4] [7] [8] [11] [12] [15] for high speed switches to operate on fixed length packets, also called cells. In this paper, we also consider a cell based design, and the external switch and internal switches all work in time slot modes. The length of an external (internal) time slot is the time for an external (internal) input to send or an external (internal) output to receive a cell. Denote the cell length as L, and the bandwidth of an internal switch as $\hat{R}$. Then the length of a internal time slot is $\hat{T} = L/\hat{R}$. With M internal switches, the external switch has aggregate bandwidth of $R = M\hat{R}$, and thus the length of an external time slot is $T = L/R = \hat{T}/M$.

In this paper, we propose the Batch-WF^{2}Q algorithm for cell dispatch at external inputs and leverage the sMUX algorithm in [7] for cell scheduling at internal switches. Such a design enables a simple round-robin algorithm for efficient packet collection at external outputs. We show that BCB-PPS has

bounded sizes for external input buffers, and needs no external or internal output buffers. We also analyze the delay guarantees achieved by BCB-PPS. Finally, we present simulation results to evaluate the design.

The rest of the paper is organized as follows. In Section II, we describe the switch architecture and present the switch scheduling algorithms. In Section III, we theoretically analyze the performance of our design. In Section IV, we present simulation data to verify the analytical results. In Section V, we conclude the paper.

II. Switch Scheduling Algorithms

In this section, we present the scheduling algorithms for BCB-PPS.

A. Switch Architecture

The architecture of BCB-PPS is illustrated in Figure 1. External inputs have buffers to store incoming cells, which are organized as virtual output queues (VOQs) [12] to avoid head of line blocking [9]. Similarly, internal inputs have buffers to store received cells, and they are organized as VOQs as well. External and internal outputs need no buffers. When a cell is transmitted to an internal output, it will be immediately collected by the corresponding external output and then immediately sent to the output line. Each internal switch is a sBUX buffered crossbar switch as in [7], with the same size as the external switch. In an internal switch, the internal inputs and outputs are connected by a buffered crossbar without speedup, which is a special crossbar with each crosspoint equipped with a small buffer storing up to two cells [7].

This paper considers a PPS with high aggregate bandwidth as well as tight performance guarantees. We assume that each flow is allocated a specific amount of bandwidth, which can be either explicitly requested by a guaranteed performance flow or obtained by estimating the traffic rate of a best effort flow as in [7]. The objective of the switch scheduling algorithms is then to guarantee the allocated bandwidth of each flow. For easy description of the algorithms, we first define some notations. Denote the i^{th} external input as In_i, the j^{th} external output as Out_j. Each In_i and Out_j have bandwidth of R. Define the traffic from In_i to Out_j as a flow F_{ij}, and denote the VOQ at In_i for F_{ij} as Q_{ij}. Use $R_{ij}(t)$ to represent the allocated bandwidth of F_{ij} at time t. To avoid over-subscription, we need $\sum_j R_{ij}(t) \leq R$ and $\sum_i R_{ij}(t) \leq R$. Denote the k^{th} cell of F_{ij} as C_{ijk}, which may be either a real cell or a dummy cell. Dummy cells are sent when Q_{ij} is empty, to ensure that each internal switch receives the same dispatched traffic.

B. Cell Dispatch for External Inputs

The main idea of cell dispatch at external inputs is to equally dispatch arriving traffic to each internal switch. Because all internal switches run the same scheduling algorithm, they will then have the same cell departure patterns, enabling easy cell collection at external outputs.

The cell dispatch problem can be modeled as a bandwidth sharing problem, in which N flows $F_{i1}, ..., F_{iN}$ at In_i share total bandwidth of R. The Generalized Processor Sharing (GPS) [16] model provides the ideal solution for bandwidth sharing. It divides available bandwidth into multiple logical channels based on the allocated bandwidth of different flows, so that each flow has its own channel and thus GPS achieves perfect fairness. However, GPS is based on a fluid system, which transmits traffic as continuous fluids of bits. Thus, GPS is not applicable to practical networks, because they transmit traffic by packets. There are a number of algorithms emulating GPS. In particular, several algorithms, such as WF^{2}Q [2], L-WF^{2}Q [19], and sMUX [7], provide the tightest $O(1)$ fairness guarantees.

Table I
Pseudo Code Algorithm Description

```
Cell Dispatch:
for each external input independently {
   identify eligible cells;
   select among eligible cells the one with the smallest
      GPS service finish time, say, a cell of F_ij;
   for p = 1 to M {
      if Q_ij is not empty {
         dispatch the head cell to the p^th internal switch;
      }
      else {
         dispatch a dummy cell to the p^th internal switch;
      }
   }
}

Cell Collection:
 for each external output independently {
   for p = 1 to M {
      collect a cell from the p^th internal switch;
      if the collected cell is a real cell {
         send it to the output line;
      }
      else {
         discard it;
      }
   }
}
```

We use an adapted version of WF^{2}Q, which we call Batch-WF^{2}Q, as the cell dispatch algorithm. Similar to WF^{2}Q, Batch-WF^{2}Q schedules cells based on their departure time in GPS. Use s_{ijk}^{GPS} and d_{ijk}^{GPS} to represent the service start and finish time of C_{ijk} in GPS, which can be calculated as $s_{ijk}^{GPS} = d_{ij(k-1)}^{GPS}$, and $\int_{s_{ijk}^{GPS}}^{d_{ijk}^{GPS}} R_{ij}(t)dt = L$.

The first step of Batch-WF^{2}Q is to identify eligible cells, and a cell is eligible if its GPS service start time is greater than or equal to the current system time. In other words, a cell that has started transmission in GPS is eligible in Batch-WF^{2}Q. The second step selects among the eligible cells the one with the smallest GPS service finish time. Ties are broken based on the input index. Until now, Batch-WF^{2}Q has been very similar to WF^{2}Q. Next, instead of transmitting one cell at a time, after Batch-WF^{2}Q finds the cell with the smallest GPS service finish time, it will transmit M cells from the same VOQ as a batch, one to each internal switch. The batch operation is to ensure that each internal switch receives the same dispatched traffic. If the VOQ becomes empty when dispatching the batch, dummy cells will be dispatched instead. Note that dummy cells will

not affect throughput, because they are only sent when there are no real cells, which will not happen if the traffic arrival rate of a flow is equal to its allocated bandwidth. For easy reading, the pseudo code description of the cell dispatch algorithm is given in Table I.

C. Cell Scheduling for Internal Switches

Internal switches use the sMUX algorithm in [7] to schedule cell transmission. sMUX schedules cells also based on allocated bandwidth of each flow. The external switch dispatches the traffic of F_{ij} at rate $R_{ij}(t)$, equally to the M internal switches. As a result, the flow from the i^{th} internal input to the j^{th} internal output has a traffic rate of $R_{ij}(t)/M$ at time t, which will be used by sMUX as the scheduling criteria. sMUX provides tight bandwidth and delay guarantees [7].

D. Cell Collection for External Outputs

Because we have taken cell collection into consideration when design the cell dispatch and scheduling algorithms, external outputs can use a simple round robin algorithm to efficiently collect cells from internal switches. Note that the cells dispatched in the same batch to different internal switches will arrive at their internal outputs at the same time. Thus, an external output just needs to collect cells one by one from the corresponding internal outputs in a circular manner. Since the cells are collected in the same order as they are dispatched to the internal switches, out-of-order delivery is avoided. Thus, each collected cell can be immediately sent to the output line, and the external output needs no buffer. For easy reading, the pseudo code description of the cell collection algorithm is given in Table I.

III. Performance Analysis

In this section, we analyze the performance of BCB-PPS.

A. External Input Delay

We first compare the external input delay. Use d_{ijk} and $d_{ijk}^{WF^2Q}$ to represent the departure time of C_{ijk} from the external input in Batch-WF^{2}Q and WF^{2}Q, respectively. Use $D_{ij}(t_1,t_2)$, $D_{ij}^{WF^2Q}(t_1,t_2)$, and $D_{ij}^{GPS}(t_1,t_2)$ to represent the numbers of bits of F_{ij} departing from the external input during interval $[t_1,t_2]$ in Batch-WF^{2}Q, WF^{2}Q, and GPS, respectively.

Lemma 1: The difference between the external input departure time of a cell C_{ijk} in Batch-WF^{2}Q and WF^{2}Q is bounded by $(N-1)(M-1)T$, i.e. $d_{ijk}-d_{ijk}^{WF^2Q} \le (N-1)(M-1)T$.

Proof: Because cells of the same flow always depart in the same order as they arrive, additional delay to C_{ijk} may only be caused by cells from another flow $F_{ij'}$. Consider the latest dispatched batch of $F_{ij'}$ before C_{ijk}, and use $C_{ij'k'}$ to represent the head cell of the batch. Use $C_{ijk''}$ to represent the head in the batch of C_{ijk}. Because both $C_{ij'k'}$ and $C_{ijk''}$ are batch heads and $C_{ij'k'}$ departs before $C_{ijk''}$ in Batch-WF^{2}Q, based on the Batch-WF^{2}Q policy we know $d_{ij'k'}^{WF^2Q} < d_{ijk''}^{WF^2Q} \le d_{ijk}^{WF^2Q}$. This indicates that $C_{ij'k'}$ does not introduce additional delay to C_{ijk}, and so do the cells of $F_{ij'}$ before $C_{ij'k'}$. Thus, only the remaining $M-1$ cells after $C_{ij'k'}$ in its batch may cause additional delay to C_{ijk}, or the additional delay from $F_{ij'}$ is at most $M-1$ external time slots. In the worst case, each of the the other $N-1$ flows introduces additional delay of $(M-1)T$, and thus the maximum difference between d_{ijk} and $d_{ijk}^{WF^2Q}$ is $(N-1)(M-1)T$. ■

Theorem 1: The difference between the external input departure time of a cell C_{ijk} in Batch-WF^{2}Q and GPS is bounded by $(N-1)(M-1)T+T$, i.e. $d_{ijk}-d_{ijk}^{GPS} \le (N-1)(M-1)T+T$.

Proof: Since $d_{ijk}^{WF^2Q} \le d_{ijk}^{GPS}+T$ by Theorem 1 in [2], we can prove the theorem by Lemma 1. ■

B. External Input Buffer Size

The external input buffer runs at high speed to accept new arriving cells, and we would like to calculate its size bound to avoid buffer overflow. Because F_{ij} is allocated a specific amount of bandwidth $R_{ij}(t)$, it is necessary to enforce admission control. Otherwise, if the flow can have an unrestricted burst arrival, it is impossible to avoid buffer overflow. The leaky bucket [9] is a widely used traffic shaping scheme, and we will use it for admission control. In the classical definition of a leaky bucket, the flow rate is a constant, which we extend in this paper to be a variable. The reason is that the allocated bandwidth of a flow may change at different time. Use $A_{ij}(t_1,t_2)$ to denote the number of arriving bits of F_{ij} during $[t_1,t_2]$. If F_{ij} is leaky bucket $(R_{ij}(t),\sigma_{ij})$ compliant, then during any interval $[t_1,t_2]$ $A_{ij}(t_1,t_2) \le \int_{t_1}^{t_2} R_{ij}(t)dt+\sigma_{ij}$. where σ_{ij} is called the burst size of F_{ij}. Intuitively, during any time interval, F_{ij} can have σ_{ij} more arriving traffic than what it can transmit.

Lemma 2: Supposing that a cell C_{ijk} is the head of a dispatched batch, its external input departure time in Batch-WF^{2}Q is greater than or equal to that in WF^{2}Q, i.e. $d_{ijk} \ge d_{ijk}^{WF^2Q}$.

Proof: Assume to the contrary that $d_{ijk} < d_{ijk}^{WF^2Q}$. There must be a cell $C_{ij'k'}$ that departs before C_{ijk} in WF^{2}Q, i.e.

$$d_{ij'k'}^{WF^2Q} < d_{ijk}^{WF^2Q} \tag{1}$$

but after C_{ijk} in Batch-WF^{2}Q. Using $C_{ij'k''}$ to represent the head cell in the batch of $C_{ij'k'}$. It is obvious that $C_{ij'k''}$ is before $C_{ij'k'}$ in Q_{ij} if $k'' \ne k'$, and thus $d_{ij'k''}^{WF^2Q} \le d_{ij'k'}^{WF^2Q}$. Furthermore, since both $C_{ij'k''}$ and C_{ijk} are batch heads and $C_{ij'k''}$ departs after C_{ijk} in Batch-WF^{2}Q, we know by the Batch-WF^{2}Q policy $d_{ij'k''}^{WF^2Q} > d_{ijk}^{WF^2Q}$. Combining the above two equations, we have $d_{ij'k'}^{WF^2Q} > d_{ijk}^{WF^2Q}$, which is a contradiction to (1). ■

Define $Q_{ij}(t)$ to be the number of bits of Q_{ij} in Batch-WF^{2}Q at time t.

Theorem 2: Supposing that the flow F_{ij} is leaky bucket $(R_{ij}(t),\sigma_{ij})$ compliant, the buffer size at the external input In_i is bounded by $NML+\sum_j \sigma_{ij}$, i.e. $\sum_j Q_{ij}(t) \le NML+\sum_j \sigma_{ij}$.

Proof: Define $DR_{ij}(t_1,t_2)$ to be the number of departed real bits of F_{ij} in Batch-WF^{2}Q during $[t_1,t_2]$. We have

$$A_{ij}(0,t) = DR_{ij}(0,t)+Q_{ij}(t) \tag{2}$$

Define $DD_{ij}(t_1,t_2)$ to be the number of dummy bits of F_{ij} in Batch-WF^{2}Q during $[t_1,t_2]$, and $D_{ij}(t_1,t_2) = DR_{ij}(t_1,t_2) + DD_{ij}(t_1,t_2)$. Thus

$$\begin{aligned} A_{ij}(0,t) + DD_{ij}(0,t) &= DR_{ij}(0,t) + DD_{ij}(0,t) + Q_{ij}(t) \\ &= D_{ij}(0,t) + Q_{ij}(t) \end{aligned} \quad (3)$$

We consider the following two cases based on whether F_{ij} has sent dummy cells by time t.

Case 1: F_{ij} has never sent a dummy cell, i.e. $DD_{ij}(0,t) = 0$. Since F_{ij} is leaky bucket $(R_{ij}(t), \sigma_{ij})$ compliant, we have $A_{ij}(0,t) + DD_{ij}(0,t) = A_{ij}(0,t) \le \int_0^t R_{ij}(t)dt + \sigma_{ij}$.

Case 2: F_{ij} has sent some dummy cells. We look at the latest batch that contains dummy cells, and denote the batch head as C_{ijk}. By Lemma 2, we have $d_{ijk}^{WF^2Q} \le d_{ijk}$, and thus $D_{ij}(d_{ijk}) = kL = D_{ij}^{WF^2Q}(d_{ijk}^{WF^2Q}) \le D_{ij}^{WF^2Q}(d_{ijk})$. By Theorem 1 in [2], we can obtain $D_{ij}^{WF^2Q}(d_{ijk}) \le D_{ij}^{GPS}(d_{ijk}) + L \le \int_0^{d_{ijk}} R_{ij}(t)dt + L$. Combining the above two equations, we have

$$D_{ij}(d_{ijk}) \le \int_0^{d_{ijk}} R_{ij}(t)dt + L \quad (4)$$

Denote the latest dummy cell as $C_{ij(k+x)}$. Note that $x \le M-1$, since C_{ijk} is the batch head and the number of cells in a batch is M. Thus

$$\begin{aligned} D_{ij}(d_{ij(k+x)}) &= D_{ij}(d_{ijk}) + xL \\ &\le \int_0^{d_{ijk}} R_{ij}(t)dt + L + (M-1)L \\ &= \int_0^{d_{ijk}} R_{ij}(t)dt + ML \end{aligned} \quad (5)$$

Since $C_{ij(k+x)}$ is a dummy cell, we know that Q_{ij} is empty at $d_{ij(k+x)}$, i.e. $Q_{ij}(d_{ij(k+x)}) = 0$. Therefore

$$\begin{aligned} & A_{ij}(0,d_{ij(k+x)}) + DD_{ij}(0,d_{ij(k+x)}) \\ =\ & DR_{ij}(0,d_{ij(k+x)}) + DD_{ij}(0,d_{ij(k+x)}) \\ =\ & D_{ij}(0,d_{ij(k+x)}) \end{aligned} \quad (6)$$

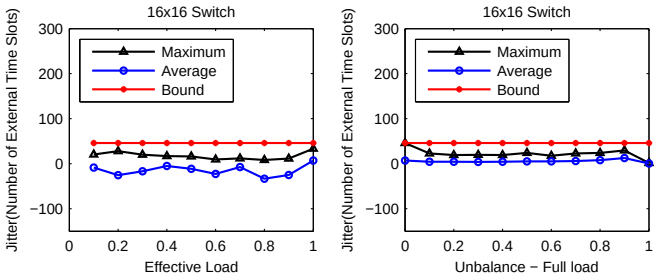
Because $C_{ij(k+x)}$ is the latest dummy cell, we know that $DD_{ij}(d_{ij(k+x)},t) = 0$ and

$$\begin{aligned} & A_{ij}(0,t) + DD_{ij}(0,t) \\ =\ & A_{ij}(0,d_{ij(k+x)}) + DD_{ij}(0,d_{ij(k+x)}) + A_{ij}(d_{ij(k+x)},t) \\ =\ & D_{ij}(0,d_{ij(k+x)}) + A_{ij}(d_{ij(k+x)},t) \\ \le\ & \int_0^{d_{ijk}} R_{ij}(t)dt + ML + \int_{d_{ijk}}^{t} R_{ij}(t)dt + \sigma_{ij} \\ \le\ & \int_0^{t} R_{ij}(t)dt + \sigma_{ij} + ML \end{aligned} \quad (7)$$

To sum up, in both case, we have

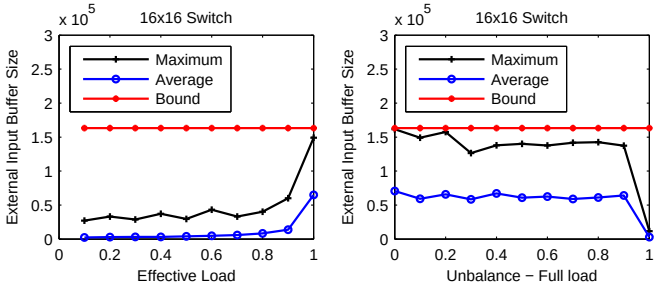
$$A_{ij}(0,t) + DD_{ij}(0,t) \le \int_0^t R_{ij}(t)dt + \sigma_{ij} + ML \quad (8)$$

Sum (8) for the N flows of In_i, and we obtain



(a) (b)

Figure 2. Jitter of Batch-WF^{2}Q (a) With different loads. (b) With different unbalanced probabilities.



(a) (b)

Figure 3. External Input buffer occupancy of Batch-WF^{2}Q (a) With different loads. (b) With different unbalanced probabilities.

$$\begin{aligned} & \sum_j A_{ij}(t) + \sum_j DD_{ij}(t) \\ \le\ & \sum_j \int_0^t R_{ij}(t)dt + \sum_j \sigma_{ij} + NML \\ =\ & \int_0^t \sum_j R_{ij}(t)dt + \sum_j \sigma_{ij} + NML \\ \le\ & \int_0^t Rdt + \sum_j \sigma_{ij} + NML \\ =\ & Rt + \sum_j \sigma_{ij} + NML \end{aligned} \quad (9)$$

Because in each external time slot, In_i sends either a real cell or a dummy cell, we know $D_{ij}(0,t) = Rt$ and therefore $\sum_j Q_{ij}(t) = \sum_j A_{ij}(t) + \sum_j DD_{ij}(t) - \sum_j D_{ij}(t) \le \sum_j \sigma_{ij} + NML$. ■

IV. Simulation Results

In this section, we conduct simulations to verify the analytical results in Section III. We consider a 16×16 BCB-PPS with 4 internal switches. The external switch has bandwidth $R = 1$ Gbps. We use fixed packet length which is 50 bytes for each cell. The internal switch has no speedup. For bandwidth allocation, we use the same model as that in [17] and [13]. The allocated bandwidth $R_{ij}(t)$ of flow F_{ij} at time t is defined by an unbalanced probability ω as follows.

$$R_{ij}(t) = \begin{cases} R(\omega + \frac{1-\omega}{N}), & if\ i = j \\ R\frac{1-\omega}{N}, & if\ i \neq j \end{cases} \quad (10)$$

When $\omega = 0$, In_i has the same amount of allocated bandwidth at each output port. Otherwise, In_i has more allocated bandwidth at Out_i, which is called the hotspot destination. Since in a real network, packet arrival is typically bursty and packets are highly correlated, arrival of a flow F_{ij} is constrained by a leaky bucket $(l * R_{ij}(t),\ \sigma_{ij})$, where l is the effective load and σ_{ij} is the burst size. We set the burst size of each flow to a fixed value of 10,000 bytes, and the burst may arrive at any time during a simulation run. We use two traffic patterns in the simulations. For the first traffic pattern, a flow has a variable allocated bandwidth. l is changed from 0.1 to 1 with an increment of 0.1, and for a specific l value, a random permutation of the 11 different ω values is used. For the second pattern, each flow has a fixed allocated bandwidth during a single simulation run. l is fixed to 1 and ω is varied from 0 to 1 with an increment of 0.1. Each simulation run lasts for 10 seconds, half of which are used as the warm-up period.

A. External Input Delay

In this subsection, we compare the external input departure time of a cell in Batch-WF^{2}Q and GPS, for which Theorem 1 gives the bound of the difference. We define jitter as the difference between the packet departure time of Batch-WF^{2}Q and GPS. Figure 2(a) illustrates the theoretical bound, maximum, and average delay difference for a representative flow F_{11} under traffic pattern one. As can be seen, the average delay difference is almost zero for different effective loads. The maximum is always less than or equal to the theoretical bound. Figure 2(b) shows the data under traffic pattern two. Similarly, the average delay difference is almost zero for different unbalanced probabilities and the maximum is always less than the theoretical bound. The maximum delay difference drops to zero when the unbalanced probability becomes 1, since at this point, all cells of In_i go to Out_i.

B. External Input Buffer Size Bound

Theorem 2 gives the upper bound of the external input buffer size to avoid overflow. In order to achieve 100% throughput for admissible traffic, input ports must have enough buffer space to avoid packet overflow. Figure 3(a) demonstrates the maximum and average external input buffer occupancy, and the theoretical bound under traffic pattern one. We can see that the maximum occupancy increases as the effective load grows, but does not exceed the theoretical bound. On the other hand, the average occupancy does not change much until the load increases to 1, i.e., the average occupancy is determined by the load. Figure 3(b) shows the data under traffic pattern two. We observe that the maximum occupancy does not change significantly with different unbalanced probabilities and is always smaller than the theoretical bound. Once the unbalanced probability becomes 1, the maximum and average occupancies fall to zero because all cells of In_i go to Out_i. The average occupancy is almost constant, and drops to zero for the same reason when the unbalanced probability becomes 1. It implies that the average occupancy is more affected by the load than the unbalanced probability.

V. Conclusions

In this paper, we have presented the buffered crossbar based parallel packet switch (BCB-PPS). It uses the sBUX buffered crossbars as internal switches, which require no speedup. We propose the Batch-WF^{2}Q algorithm for cell dispatch at external inputs, and leverage the sMUX algorithm in [7] for cell scheduling at internal switches. Such a combination enables a simple round-robin algorithm for efficient cell collection at external outputs. We show that BCB-PPS has tight delay guarantees and bounded buffer sizes. Finally, we present simulation data to verify the analytical results and to evaluate the performance of our design.

References

[1] A. Aslam and K. J. Christensen, "A parallel packet switch with multiplexors containing virtual input queues," *Computer Communications*, vol. 27, no. 13, pp. 1248–1263, August 2004.

[2] J. C. Bennett and H. Zhang, "WF2Q: worst-case fair weighted fair queueing," in *IEEE INFOCOM 1996*, San Francisco, CA, USA, 1996, pp. 120–128.

[3] W. K. Chen, *The Electrical Engineering Handbook*, 1st ed. Academic Press, November 2004.

[4] S. T. Chuang, S. Iyer, and N. McKeown, "Practical algorithms for performance guarantees in buffered crossbars," in *IEEE INFOCOM 2005*, vol. 2, Miami, FL, USA, 2005, pp. 981–991.

[5] S.-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *Selected Areas in Communications, IEEE Journal on*, vol. 17, no. 6, pp. 1030–1039, Jun 1999.

[6] Z. Haishan, X. Du, and Z. Zhenyu, "A parallel packet switch supporting variable-length packets," in *2005 International Conference on Communications, Circuits and Systems,*, vol. 1, 2005, pp. 613–617.

[7] S.-M. He, S.-T. Sun, H.-T. Guan, Q. Zheng, Y.-J. Zhao, and W. Gao, "On Guaranteed Smooth Switching for Buffered Crossbar Switches," *IEEE/ACM Trans. Netw.*, vol. 16, no. 3, pp. 718–731, 2008.

[8] S. Iyer and N. W. McKeown, "Analysis of the parallel packet switch architecture," *IEEE/ACM Trans. Netw.*, vol. 11, no. 2, pp. 314–324, 2003.

[9] J. F. Kurose and K. Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley Longman Publishing Co., 2002.

[10] H.-I. Lee and S.-W. Seo, "Matching output queueing with a multiple input/output-queued switch," *IEEE/ACM Trans. Netw.*, vol. 14, no. 1, pp. 121–132, 2006.

[11] R. B. Magill, C. E. Rohrs, and R. L. Stevenson, "Output-queued switch emulation by fabrics with limited memory," *Selected Areas in Communications, IEEE Journal on*, vol. 21, no. 4, pp. 606–615, 2003.

[12] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Netw.*, vol. 7, no. 2, pp. 188–201, 1999.

[13] L. Mhamdi and M. Hamdi, "MCBF: a high-performance scheduling algorithm for buffered crossbar switches," *IEEE Communications Letters*, vol. 7, no. 9, pp. 451–453, 2003.

[14] S. Mneimneh, V. Sharma, and K.-Y. Siu, "Switching using parallel input-output queued switches with no speedup," *IEEE/ACM Trans. Netw.*, vol. 10, no. 5, pp. 653–665, 2002.

[15] N. Moriwaki, A. Makimoto, Y. Oguri, M. Wada, and T. Kozaki, "Large scale ATM switch architecture for Tbit/s systems," in *IEEE GLOBECOM 1998*, vol. 1, 1998, pp. 334–338.

[16] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case," *IEEE/ACM Trans. Netw.*, vol. 1, no. 3, pp. 344–357, 1993.

[17] R. Rojas-Cessa, E. Oki, Z. Jing, and H. Chao, "CIXB-1: Combined input-once-cell-crosspoint buffered switch," in *IEEE HPSR*, Dallas, TX, 2001, pp. 324–329.

[18] L. Shi, G. Xia, and B. Liu, "Performance Guarantees for Flow-Mapping Parallel Packet Switch," in *Proc. IEEE Internationa Performance, Computing, and Communications Conference IPCCC*, 2007, pp. 109–116.

[19] P. Valente, "Exact GPS simulation with logarithmic complexity, and its application to an optimally fair scheduler," in *ACM SIGCOMM 2004* , New York, USA, 2004, pp. 269–280.