Mathematical Analysis of the Parallel Packet Switch with a Sliding Window Scheme

Chia-Lung Liu, Chin-Chi Wu, and Woei Lin

Abstract: **This work analyzes the performance of the parallel packet switch (PPS) with a sliding window (SW) method. The PPS involves numerous packet switches that operate independently and in parallel. The conventional PPS dispatch algorithm adopts a round robin (RR) method. The class of PPS is characterized by deployment of parallel low-speed switches whose all memory buffers run more slowly than the external line rate. In this work, a novel SW packet switching method for PPS, called SW-PPS, is proposed. The SW-PPS employs memory space more effectively than the existing PPS using RR algorithm. Under identical Bernoulli and bursty data traffic, the SW-PPS provided significantly improved performance when compared to PPS with RR method. Moreover, this investigation presents a novel mathematical analytical model to evaluate the performance of the PPS using RR and SW method. Under various operating conditions, our proposed model and analysis successfully exhibit these performance characteristics including throughput, cell delay, and cell drop rate.**

Index Terms: **Bernoulli data traffic, bursty data traffic, parallel packet switch (PPS), round robin (RR), sliding window (SW).**

I. INTRODUCTION

In a conventional packet switching system, the speed of the memory must be at least equal to the external line rates. For examples, an output queued (OQ) switch requires buffer memories that operate at N times the link speed [1], [2], where N is the number of ports of the switch; in an input queued (IQ) switch, each memory operates at the same speed as the external line rate [1], [2]; in a combined input-and-output queued (CIOQ) switch, the memory operates at twice the line rate [3], [4]. Therefore, when external line rates increase from OC192 (10 Gb/s) to OC768 (40 Gb/s) or even OC3072 (160 Gb/s), traditional packet switches, whose buffers cannot process the packets at the same line rate, will not be usable.

In a high-speed network, the parallel packet switch (PPS) using a round robin (RR) method is a good choice for transferring data [5], [6]. A PPS, which overcomes memory bandwidth limitations, comprises multiple low-speed packet switches operating independently and in parallel. Each lower speed packet switch operates at a fraction of the input line rate R. For instance, each slower packet switch can operate at an internal line rate of R/K, where K is the number of slower speed switches. PPS with a RR algorithm is named RR-PPS, which provides better performance than conventional packet switch under the high-speed network [7]. The RR-PPS is reviewed in Section II-B of this work.

The key problem in a RR-PPS is that it cannot effectively use the memory space in low-speed switches, so it requires a lot of memory to buffer packets [5], [7]. The total buffering requirement for a RR-PPS is $2KN^2$ cells $+KNL$ [5], where L is queue size. For $N = 64,\ K = 128$, and $L = 256$ kbytes, the total buffer needed in the entire RR-PPS is 2112 Mbytes (for 64 byte cells). It is expensive to implement using current SRAM and SDRAM technology. This work describes a novel sliding window (SW) packet switching method for PPS, called SW-PPS. The SW concept comes from [8], [9]. This novel switching scheme overcomes the shortcomings of RR-PPS, and uses memory space in the low-speed switches more effectively than RR scheme. According to experimental results, even with less available memory in low-speed switches, SW-PPS performed well. It also maintains PPS's advantage that all memory buffers and internal lines run slower than the external line rate. Therefore, SW-PPS can be implemented in the high-speed network. Besides, the experimental results indicate that SW-PPS outperformed RR-PPS under identical Bernoulli and bursty data traffic.

II. BACKGROUND

A. Definitions

Some terms, used throughout this paper, are defined.

- **Cell:** A fixed-length packet, though not necessarily equal in length to a 53-byte ATM cell. Although packets that arrive at the switch may have various lengths, for the purposes of this work, we will assume that they are segmented and processed internally as fixed-length cells. This is common practice in high-performance routers; variable-length packets are segmented into cells as they arrive, are carried across the switch as cells, and are reassembled back into packets before they depart.
- **External time slot:** The time taken to transmit or receive a fixed-length cell at an external link rate of R.
- **Internal time slot:** The time taken to transmit or receive a fixed-length cell at an internal link rate of (R/K), where K is the number of low-speed switches in the PPS.
- **OQ switch:** A switch in which arriving packets are placed immediately in queues at the output, where they contend with other packets with the same output destination. The departure order is first-come first-serve (FCFS). One characteristic of an OQ switch is that the buffer memory must be able

Manuscript received June 13, 2006; approved for publication by Saewoong Bahk, Division III Editor, April 13, 2007.

C.-L. Liu is with the Computer Science Institute of the National Chung-Hsing University, Information & Communications Research Lab., Industrial Technology Research Institute, Hsinchu 310, Taichung, 402 Taiwan, email: s9056005@cs.nchu.edu.tw.

C.-C. Wu is with Computer Science Institute of the National Chung-Hsing University and Nan Kai Institute of Technology, Tsaotun, 54243, Nantou, Taiwan, email: wcc007@nkc.edu.tw.

W. Lin is with the Computer Science Institute of the National Chung-Hsing University, email: wlin@cs.nchu.edu.tw.

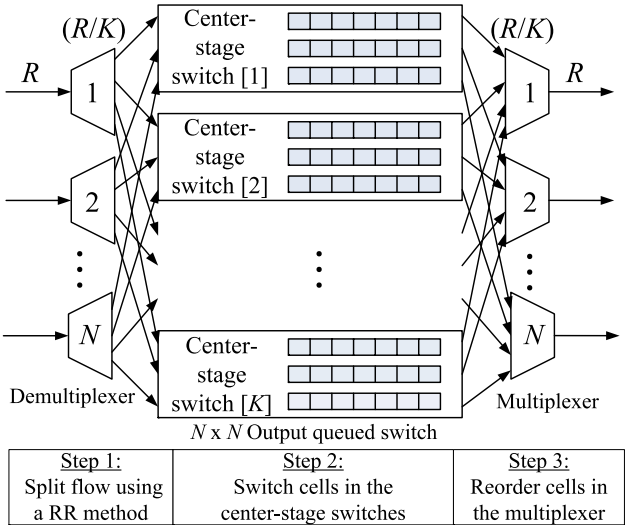


Fig. 1. Architecture of the PPS.

to accept (write) N new cells per time slot, where N is the number of ports. Accordingly, the memory must operate at N times the line rate.

- **PPS restriction:** An input link constraint of PPS [5], [6]. As shown in Fig. 1, each external input of line rate R is spread (demultiplexed) over K lines; each internal input link must run at a minimum speed of R/K. In other words, the external line rate runs K times faster than the internal line rate. If the transmission of a cell in the external line spends one external time slot, transmission of a cell in internal line of the PPS spends K external time slots (i.e., one internal time slot). Hence, in Fig. 1 when a particular demultiplexer sends a cell to a particular center-stage switch at external time slot t, this demultiplexer sends another cell to this center-stage switch after external time slot $t + K - 1$ due to the internal line rate of R/K. Consequently, for a demultiplexer at an external time slot, not total center-stage switches conform to the PPS restrictions.

B. RR-PPS Architecture and Algorithm

The architecture of a PPS resembles that of a Clos network [10], as displayed in Fig. 1. The demultiplexers, center-stage switches (slower speed switches), and multiplexers correspond to the three stages of a Clos network. Fig. 1 shows an $N \times N$ PPS, with each port operating at rate R. Each port is connected to all K OQ switches (the center-stage switches are referred to as layers). Since cells form each external input of line rate R are spread (demultiplexed) over K links, each input link must run at a minimum speed of R/K. Each layer of PPS consists of a single $N \times N$ OQ or CIOQ switch with memory operating slower than the external line rate. Since each multiplexer receives cells from K output queues, each line to the multiplexer must operate at a minimum speed of R/K to keep the external line busy. All the memories of PPS run slower than the external line rate, so the number of center-stage switches is limited. Assume that the external line rate is R and that the number of center-stage switches is K. If the center-stage switches are CIOQ switches, then $2(R/K) < R$ and; as a result, we have $K > 2$. Similarly,

```
//Algorithm for RR-PPS demultiplexer
//
//Define the following functions:
//   receive()              Receives a cell
//   classify(cell)         Returns destination port
//   append(cell, sn[i])    Appends sn
//   send(cell, pn)         Sends cell
//
//Define the following variables:
//   i        Destination port number index
//   sn[]     Sequence number
//   pn       Plane number

While true do
   cell = receive()          //Step 100
   i = classify(cell)        //Step 102
   sn[i] + +                 //Step 104
   append(cell, sn[i])       //Step 106
   pn = (pn%K) + 1           //Step 108
   send(cell, pn)            //Step 110
end
```

Fig. 2. Architecture of the PPS.

at center-stage OQ switches, it is required that $N(R/K) < R$. Therefore, we have $K > N$.

The algorithm of RR-PPS proceeds in three distinct steps (shown in Fig. 2).

Step 1) Split every flow in the demultiplexer using a round robin procedure: When a cell arrives at an input port, the demultiplexer selects a low-speed switch (or "layer") to which it will send the cell. Using a RR scheme, cells from a flow are distributed to the center-stage switches. Fig. 2 shows the RR algorithm within each demultiplexer. Cells are received (step 100), classified (step 102), appended with their sequence numbers (step 106), and then sent to the center-stage switches (step 110). The layer number is indexed by an output port of the demultiplexer (variable pn). Each demultiplexer maintains a round robin pointer pn. By RR method (step 108), this pointer contains a value in the range $[1, K]$. When pointer $pn = x$, the arriving cell is sent to layer x.

Step 3) Deliver cells in the center-stage OQ switches: Each center-stage OQ switch receives cells from its N input ports, and then switches each cell to its output port according to the destination address of the cell [1], [2]. Due to output link congestion and internal line speed of R/K, cells are stored in the output queues of the center-stage OQ switch until the line to the multiplexer becomes available. When the line to the multiplexer becomes available, the head-of-line (HOL) cell in the output queue is delivered to the output link of the center-stage OQ switch.

Step 3) Reorder cells in the multiplexer: The link to the multiplexer operates at a rate of R/K. While the line to the multiplexer is free, the multiplexer selects a cell from the corresponding K output queues in each center-stage OQ switch. The goal for the multiplexer is that its buffer stores, reorders,

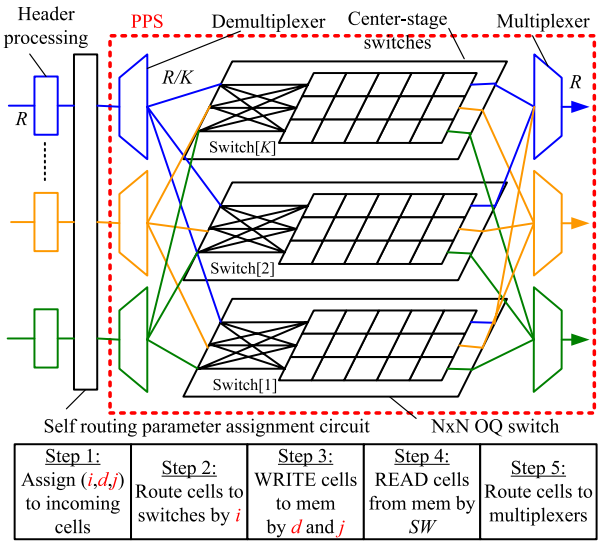


Fig. 3. Architecture of the SW-PPS.

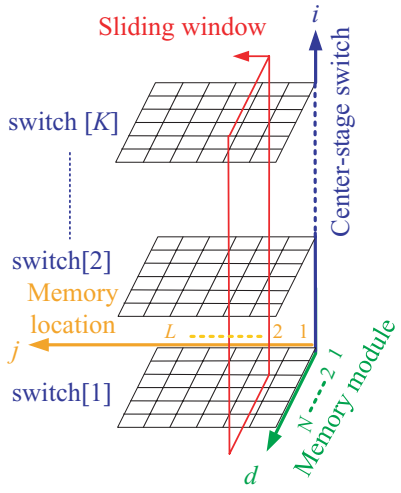


Fig. 4. Architecture of the SW-PPS.

and then transmits cells in the correct order based on the cell's sequence number [5], [7].

The primary weakness of RR-PPS is its inefficient use of memory, a product of RR algorithm. In step 108 in Fig. 2, RR scheme at the demultiplexer chooses a layer to store the arriving cell by alternate method. However, it can't send the cell to the particular layer, whose output queue has the least cells. This work proposes a novel SW scheme for PPS to improve its use of memory. The rest of the paper is organized as follows. In Section III, the SW-PPS architecture and algorithm are introduced. In Section IV, a mathematical analytical model for RR-PPS and SW-PPS is presented. In Section V, we present the analytical results, and compare them with simulation results. We also present some simulation results for bursty data traffic. Finally, Section VI concludes the paper.

III. SW-PPS ARCHITECTURE AND ALGORITHM

A. SW-PPS Architecture

Fig. 3 shows the overall architecture of the SW-PPS. The SW-PPS is constructed by the self-routing parameter assignment circuit and PPS. The SW-PPS is divided into the following independent stages: (1) The self-routing parameter assignment circuit; (2) the demultiplexers; (3) the slower speed center-stage packet switches; and (4) the multiplexers. Incoming cells use the parameter assignment circuit to navigate through the entire PPS. In Fig. 3, the destined output port of the incoming cell, extracted by applying header processing circuits, is indicated by d. The incoming cell's destination address d is delivered to a self-routing parameter assignment circuit. In processing incoming cells, the assignment circuit employs the output destination d and a parameter assignment algorithm to create an additional group of self-routing parameters (i, d, and j). These self-routing parameters (i, d, and j) are attached to arriving cells as a self-routing tag (step 1 in Fig. 3). Incoming cells then use the attached tags to navigate through the demultiplexers and center-stage switches. Parameters (i, d, and j) are defined as follows: The variable i in parameters informs the center-stage switch where the cell will be stored; variable d indicates which memory module in the ith center-stage switch the cell will be stored in; and variable j designates the memory location in the dth memory module where the cell will be stored. The demultiplexer uses the parameter i of the routing tag of an incoming cell to send the cell to its ith output port which is connected to the corresponding ith layer (step 2 in Fig. 3). Input ports of the demultiplexers connect itself to the parameter assignment circuit, and the output ports of the demultiplexers connect with each center-stage switch of the switching system. The demultiplexers retain the parameters d and j information from the cell's self-routing tag to center-stage switches. The center-stage switch employs the parameter d to send the received cell to the dth memory module, and uses the parameter j to write the received cell in the jth memory location of the dth memory module. During the cell WRITE cycle for a cell arriving at the ith center-stage switch, the cell is written to jth memory location in a given dth memory module based on the self-routing parameter (i, d, and j) (step 3 in Fig. 3). During the cell READ cycle, cells in the memory are output from memory module according to the present location of the SW (step 4 in Fig. 3) and are finally delivered to respective multiplexers (step 5 in Fig. 3).

B. Switching Operations in the SW-PPS

According to the SW-PPS switching schedule, the overall memory space, including all cell memory locations in all of the center-stage switches, is represented as a three-dimensional (3-D) memory space (shown in Fig. 4). The memory locations in the global memory space are represented by a 3-D coordinate system (i, d, and j), where the first coordinate i represents the ith center-stage switch; $i \in [1, K]$, where K is the number of center-stage switches. The second coordinate d indicates the dth memory module; $d \in [1, N]$, where N is the number of ports of the center-stage switch. The third coordinate j designates the jth memory location in the memory module; $j \in [1, L]$, where L represents the number of memory locations in each memory module, which denotes the queue length. As shown in Fig. 4, the ith layer is the memory space of the center-stage switch and is

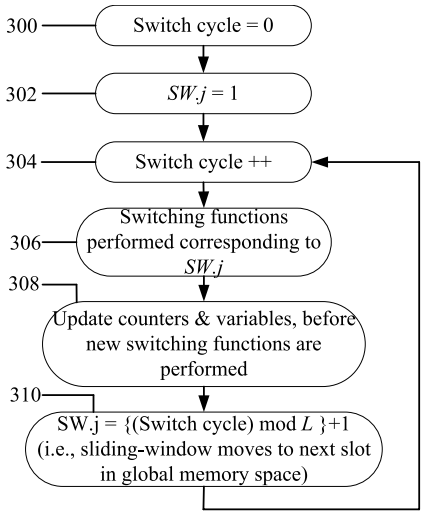


Fig. 5. Flowchart depicting traversal of SW.

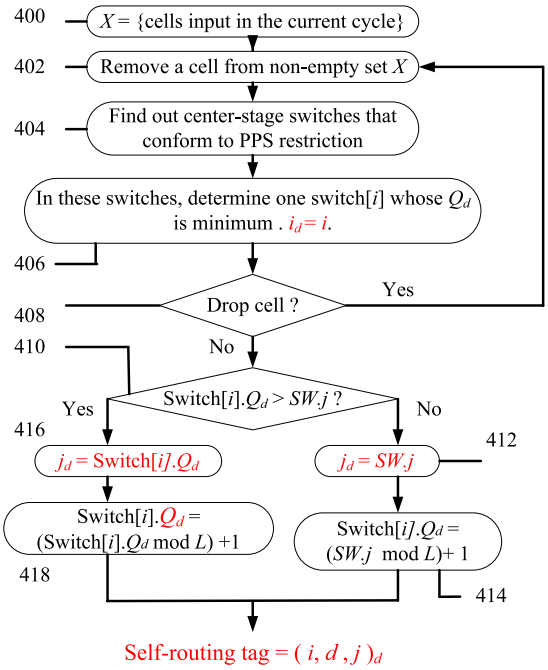


Fig. 6. Assignment process for self-routing tag.

designated by switch[1], switch[2], $\cdots$, switch[K]. Each layer is divided into N memory modules. Each memory module consists of the L consecutive memory slots. The SW can be regarded as a pointer to a group of cells in the 3-D memory space (shown in Fig. 4). The SW advances one memory location during each internal cycle (switch cycle). The location of the SW in the global memory space is recorded by one variable, $SW.j$. For example, in Fig. 4, the SW is a pointer to the memory location $= j = 2$. For such a state of the SW in the above example $SW.j = 2$. The variable $SW.j$ holds an integer value which is incremented by one on the completion of SW traversal of a given memory location. Furthermore, cells in the 3-D memory space are sent to multiplexers according to the current location of the SW.

The flowchart in Fig. 5 shows SW traversal across the entire 3-D memory space and its relation to the switch cycle and switching operation. The SW pointer is updated along with the switching functions performed every switch cycle. In the flowchart of Fig. 5, step 300 indicates the beginning of the switch operation. Step 302 shows the initial value of the variable $SW.j$ indicating initial location of the SW in the global memory space. The onset of the switch operation is shown in step 304 and various switching functions are performed on the incoming cells in step 306. The switching functions in step 306 include some of the following operations: Read destination addresses from headers of the incoming cells, update counters and tables, attach new self-routing tags to cells, write cells to memory, read cells from the memory, etc. After switching functions in step 306, counters and variables are updated in step 308 to account for changes, due to the switching. After switching and updating the variables in steps 306 and 308, the SW is then advanced to the next location in step 310. After step 310, the flow loops back to step 304 to start a new switching cycle corresponding to a new updated position of the SW pointer. With the updated location of the SW denoted by the variable $SW.j$ (in step 310), the new switching functions are performed after the flow-control loops back to step 304. In the new switch cycle in step 304, the system again performs new switching functions in step 306. The underlying switching function of the SW scheme at step 306 is that during the input phase of each switch cycle, incoming cells are assigned memory locations within the global memory space with the help of self-routing parameter assignment circuit and during the output phase of each switch cycle, all the cells belonging to the memory location pointed by the SW are sent out. The SW, as shown in Fig. 4, cyclically scans the entire 3-D memory space. Switching functions are performed corresponding to new value of $SW.j$ pointer during its traversal of the 3-D global memory space.

C. Determination of Parameters (i, d, and j)

Assigning of self-routing parameters (i, d, and j) to incoming cells is computed by the parameter assignment circuit. An additional routing tag carrying the self-routing parameters (i, d, and j) is attached to the incoming cell. The self-routing parameters help cells to self-navigate through the switching system. Determination of self-routing parameters (i, d, and j) by the assignment circuit to an incoming cell is shown by the flowchart in Fig. 6. The symbols used therein are described as follows.

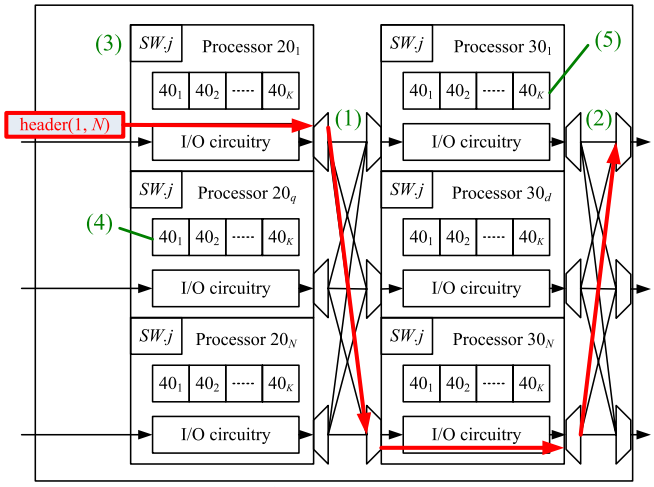
- d is the output-port destination of the cell.
- (i_d and j_d) are the parameters (i, and j) of the incoming cell destined to output port d.
- $Switch[i].Q_d$ is a pointer inside the dth memory module of $switch[i]$ (i.e., ith center-stage switch) for cells destined to output port d. $Switch[i].Q_d$ points the memory location that is the next position of the last cell in the queue. Furthermore, $Switch[i].Q_d$ increased with the number of cells in the dth memory module of switch[i].
- X is a set of arriving cells during a given external cycle. $0 \leq |X| \leq N$, where N is the number of input ports.

The flowchart in Fig. 6 shows the assignment process for the self-routing parameter (i, d, and j) to the incoming cells. Step 400 is the initial state, and X cells are input in a given external

cycle through the N input ports of SW-PPS. Step 402 shows removal of a cell from the nonempty set of input cells. When the incoming cell selected in step 402 is destined to output port d, the assignment process must determine which layers conform to the PPS restriction (step 404) for this cell, and then finds out the minimum value of $Switch[i].Q_d$ in these layers' dth memory module (step 406). After it finds out the minimum $Switch[i].Q_d$ which is in the ith layer, the value of i_d is assigned as i for the incoming cell, i.e., $i_d = i$. According to step 408, if the number of cells in the dth queue of the switch[i] (ith layer) is greater than queue size L, the cell destined to d is dropped, and the assignment process loops back to step 402 to process another cell. In step 410, $Switch[i].Q_d$ and $SW.j$ are compared. If $Switch[i].Q_d \leq SW.j$, then it means it is the only cell in this memory module and it doesn't need to wait because there are no other cells waiting for that destination port in this memory module. In such a case, j_d is assigned as the current location of the SW for the incoming cell (step 412), i.e., $j_d = SW.j$. Consequently, cells are quickly sent to multiplexers according to the current location of the SW. In step 414, $Switch[i].Q_d$ is moved to the next position of the $SW.j$. If $Switch[i].Q_d > SW.j$, then it means that the queue has been building up for the cells destined to output port d. For the incoming cell, j_d is assigned to the next position of the last cell in the queue ($j_d = Switch[i].Q_d$) in step 416. According to step 418, $Switch[i].Q_d$ is then advanced to the next position. Finally, self-routing tag is attached to the arriving cell.

The self-routing parameter assignment circuit in Fig. 3 and flowchart in Fig. 6 use counters and tables (shown in Fig. 7) to help the determination process of self-routing parameters. Fig. 7 provides one method for implementing the distributed self-routing parameter assignment circuit of SW-PPS. For each internal cycle, the $SW.j$ in each processor (shown in Fig. 7) updates independently according to the SW traversal flow diagram in Fig. 5. In Fig. 7, the tables $40_1, 40_2, \cdots, 40_K$ of the processor 20_q, $q \in [1, N]$, record which center-stage switches conform to the PPS restriction for the demultiplexer q (for step 404 in Fig. 6). If the demultiplexer q sends a cell to switch[i] (i.e., ith layer), $i \in [1, K]$, at time t, the demultiplexer q sends another cell to switch[i] after time $t + K - 1$ (PPS restriction); in other words, the table 40_i of the processor 20_q records that the demultiplexer q sends a cell to switch[i] at time t and sends next cell to switch[i] after time $t + K - 1$. According to tables of the processor 20_q, the parameter assignment circuit can identify which center-stage switches conform to the PPS restriction, and does not require feedback from the internal switch layers. Hence, the tables in the processors 20_q are updated according to the PPS restriction. The assumption is that the incoming cell, whose header is header(s, d), comes from PPS's input port s and is destined to PPS's output port d. After the header of the incoming cell passes through the processor 20_q in the assignment circuit, the self-routing of header(s, d) is according to d. For example, in Fig. 7, self-routing of header$(1, N)$ is according to the destination address N. The tables $40_1, 40_2, \cdots, 40_K$ of the processor 30_d, $d \in [1, N]$, record the value of $Switch[i].Q_d$. In other words, the value of $Switch[i].Q_d$ in the dth memory module of switch[i] is recorded in the table 40_i of the processor 30_d. This information is used to identify step 406 in Fig. 6. Because

(1) Self routing of header(s, d) is according to d in the circuit.
(2) Self routing of header(s, d) is according to s in the circuit.
(3) SW.j updates according to the flowchart in Fig. 4.
(4) Tables record which switches conform to the PPS restriction (for step 404), and are updated according to the PPS restriction.
(5) The tables record the value of Q_d (for step 406), and are updated according to steps 410 to 418.



Assuming that header(s, d) comes from sth input port of PPS, and goes to dth output port of PPS.

Fig. 7. Assignment process for self-routing tag.

$Switch[i].Q_d$ value in the table 40_i of the processors 30_d is updated according to steps 410 to 418, the tables in the processors 30_d also do not require feedback from the center-stage switches. After the cell header passes through the processor 30_d in the assignment circuit, it must return to its original source address. Hence, the self-routing of header(s, d) is according to s. For instance, self-routing of header$(1, N)$ is according to 1 in Fig. 7, after header$(1, N)$ passes through the processor 30_N. When cell headers arrive at the output ports of the parameter assignment circuit, they have become to the self-routing parameters (i, d, and j) by a parameter assignment algorithm (Fig. 6), and then are attached to cells as a self-routing tag. Afterward, the cells with their self-routing tag are forwarded to respective demultiplexers.

The SW scheme, which has O(N) complexity (due to steps 404 and 406 in Fig. 6) and does not need feedback from the center-stage switches, is not too complex to be practically implemented. The memory size of the self-routing parameter assignment circuit is $2KN$ tables, which may be sufficiently small to allow the memory to be placed on chip. For instance, if N is 256 ports, table is 256 bytes, K is 10 layers, and center-stage switches are CIOQ switches, then the memory size in the assignment circuit is 1.25 Mbytes. This can be placed on a chip using today's SRAM technology. Moreover, a small queuing delay in the parameter assignment circuit can be neglected. Additionally, the SW-PPS employs a self-routing parameter i for each incoming cell to be self-routed through demultiplexers (step 2 in Fig. 3), so it doesn't need buffer in the demultiplexers.

Although the idea of SW-PPS comes from SW-switch [8], SW-PPS and SW-switch are of completely different architec-

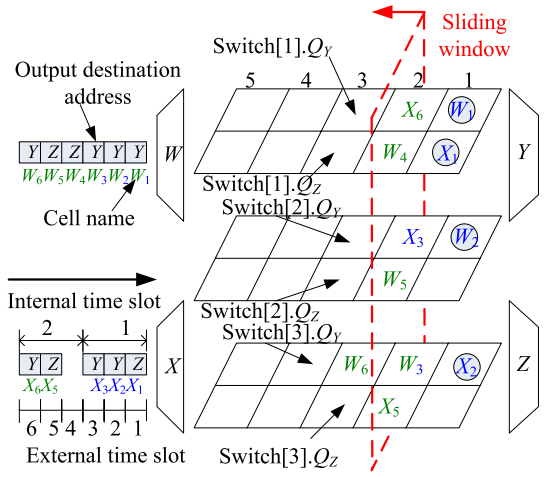


Fig. 8. Occupancy of 3-D memory space of the SW-PPS for two internal cycles of input cell arrivals. (Input cells are denoted by their output destination address.)

Table 1. Variation of symbols

Internal time slot	0	1			2		
External time slot	0	1	2	3	4	5	6
$Switch[1].Q_Y =$	1	2	2	2	2	2	3
$Switch[1].Q_Z =$	1	2	2	2	3	3	3
$Switch[2].Q_Y =$	1	1	2	3	3	3	3
$Switch[2].Q_Z =$	1	1	1	1	1	3	3
$Switch[3].Q_Y =$	1	1	2	3	3	3	4
$Switch[3].Q_Z =$	1	1	1	1	1	3	3
$SW.j =$	0	1			2		

Table 2. Time chart for output of cells

Internal time slot	2	3	4
Output Y	$W_1W_2X_2$	$X_6X_3W_3$	W_6
Output Z	X_1	$W_4W_5X_5$	
$SW.j =$	2	3	4

tures, algorithms, and contributions. The SW-switch is a class of share memory switch architecture with plural memory modules, but the SW-PPS in this investigation is a novel kind of PPS [5], [6] architecture whose all center-stage switches operate lower than network speed. Hence, the center-stage of SW-switch is plural memory modules, and that of SW-PPS is a group of low-speed switches. The algorithm of SW-switch fits the multiple memory modules that are physically one layer but logically multiple layers. However, SW-PPS algorithm is employed on the PPS architecture, physically multiple layers and logically multiple layers, and SW-PPS scheme must consider the PPS restriction to dispatch cells. The SW-switch features high scalability when compared with other shared-memory switches. Our work shows that SW-PPS uses memory space more effectively than RR-PPS, while retaining RR-PPS's merit that internal line speed can be lower than external line speed.

D. Illustration of the SW-PPS

Fig. 8 presents a stream of cells that arrive at the demultiplexers of 2×2 SW-PPS. W and X represent the two demultiplexers of the switch. Y and Z represent the two multiplexers. Each incoming cell is referred to by its destination address. For instance, the cell that arrives in the second external cycle on the demultiplexer W is destined to output port "Y". For illustration, Fig. 8 depicts a configuration of the 2×2 switch, for which the queue length $L = 5$ cells and the number of center-stage switches deployed, $K = 3$. Based on the parameter settings, $N = 2$, the required minimum number of center-stage switches $K = 3$ (Section II-B). The incoming streams of cells in six external cycles are W_i and X_i; $i \in [1, 6]$. For illustration, cell W_1 in Fig. 8 comes from demultiplexer W in external time slot 1 and is destined to multiplexer Y. For two internal cycles (six external cycles), Table 1 presents the variation of $Switch[i].Q_d$ and $SW.j$ with time in the example in Fig. 8. Table 2 presents a detailed time chart, which shows that cells arrive at the multiplexers in different internal cycles along with the traversal of SW. In the multiplexers, cells are stored, reordered, and then transmitted in the correct order [5], [7]. Fig. 8 also displays the occupancy of the 3-D memory space after two internal time slots. The SW counter in the second internal cycle demonstrates that it is currently processing the cells that belong to the $SW.j = 2$. The circled cells represent earlier occupancy of the cells in the global memory space before being output in earlier cycles. In the example in Fig. 8, the parameter assignment circuit controls the growth of the number of cells in the queue (step 408 in Fig. 6). When the number of cells in the queue exceeds queue length (in this example, queue length $L = 5$ cells), all other incoming cells, destined to the congested output port, are dropped. In Fig. 8, W_3 must conform to the restriction of PPS (for demultiplexer W, $switch[1]$ and $switch[2]$ has been used in external time slot 1 and 2) and only the $switch[3]$ is adopted. For the same reason, W_4 selects only the $switch[1]$. When the demultiplexer X sends X_2 to $switch[3]$ at external time $t = 2$, demultiplexer X sends another cell to $switch[3]$ after external time $t+K-1 = 4$ (PPS restriction). Therefore, X_5 has two choices, $switch[1]$ and $switch[3]$. X_5 chooses the $switch[3]$, because $Switch[3].Q_Z = 1$ is smaller than $Switch[1].Q_Z = 3$ (step 406 in Fig. 6). Similarly, X_6 also has two choices, $switch[1]$ and $switch[2]$, and chooses $switch[1]$. In Table 1, because of $(Switch[2].Q_Y = 1) \leq (SW.j = 1)$ in external time slot 1 (step 410 in Fig. 6), $Switch[2].Q_Y$ equals two in external time slot 2 $(Switch[2].Q_Y = (SW.j \bmod L)+1 = (1 \bmod 5)+1 = 2$, step 414 in Fig. 6). Because of $(Switch[2].Q_Y = 2) > (SW.j = 1)$ in external time slot 2 (step 410 in Fig. 6), $Switch[2].Q_Y$ is three in external time slot 3 $(Switch[2].Q_Y = (Switch[2].Q_Y \bmod L) + 1 = (2 \bmod 5) + 1 = 3$, step 418 in Fig. 6).

IV. ANALYSIS OF RR-PPS AND SW-PPS

This section presents an analytical performance model for PPS using RR and SW scheme under uniform traffic. This work is an approximation of the performance analysis of PPS. The PPS is further simplified to an output queue represented by a Markov chain [11], [12]. Finally, three equations are derived for

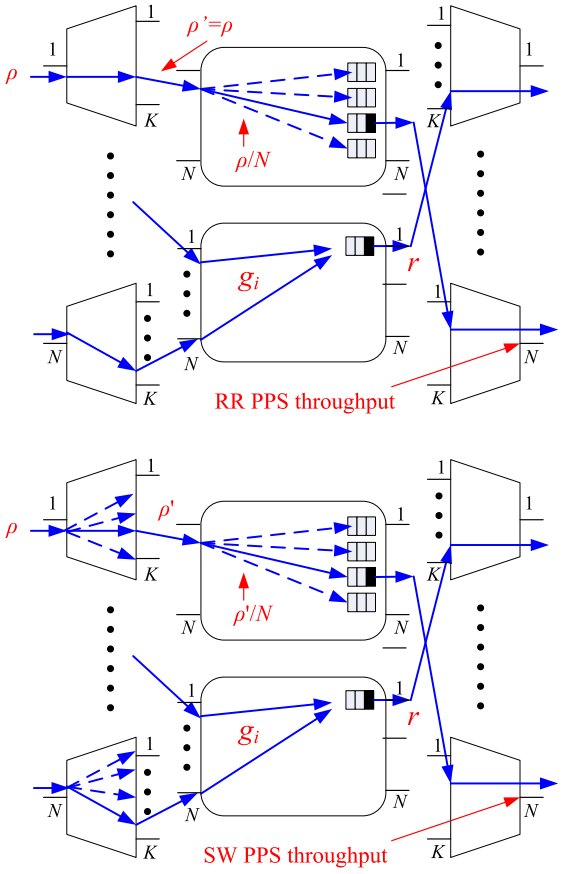


Fig. 9. Relationships among probabilities for RR-PPS and SW-PPS.

performance measures. The following assumptions are made:

- Cells arrive according to a Bernoulli process, and the size of the cell is fixed.
- Packet destinations are uniformly distributed.
- Center-stage switches are OQ switches, and each queue length has a finite capacity.

For clarity, the following is a list of notations used in the development of the performance model and its subsequent analysis. Fig. 9 is given to illustrate the relations among these notations.

- N: Number of ports of the PPS
- K: Number of center-stage switches
- ρ: Input load
- ρ': Internal load of the PPS
- L: Size of buffer(queue length)
- $P_j(t)$: Probability that j cells are stored in a buffer at internal cycle t
- $P_{drop}(t)$: Probability that a buffer overflows at internal cycle t
- g_i: Probability that i cells arrive at the same output buffer
- r: Probability that a cell in a buffer successfully moves to the multiplexer
- $\bar{r}$: Probability that a cell in a buffer unsuccessfully moves to the multiplexer ($\bar{r} = 1 - r$)

A. *Analysis for the RR-PPS*

Because there is no HOL blocking problem in center-stage OQ switches [1], [2], a HOL cell in a buffer can successfully move to the multiplexer. Therefore, we must assume the probability $r = 1$. Due to the RR-PPS input load $= \rho$, the internal load (ρ') is ρ/K in an external cycle and equals ρ in an internal cycle. Hence, when the internal load $(\rho') = \rho$ in an internal cycle, we can obtain g_i and following equations.

$$g_i = \binom{N}{i}(\rho/N)^i(1-\rho/N)^{N-i}, \quad 0 \le i \le N. \tag{1}$$

In the equation above, the probability that the arriving cell chooses one from N buffers is ρ/N, and g_i is the probability that a total of i cells arrive at the same output buffer (shown in Fig. 9).

Now we develop the state transition diagram of the Markov chain. We divide the Markov chain model into two cases, $N \ge L+1$ and $N < L+1$, as illustrated in Figs. 10 and 11. In this model, the PPS of size N is further simplified to an output queue whose length equals L. If $N \ge L+1$, it is possible that the output queue overflows at internal cycle t even if there are no cells in the queue at internal cycle $t-1$. In other words, P_0 can go straight to P_{drop} in Fig. 10. The reason is that when N cells from N demultiplexers arrive at the output queue in which there are no other cells, only L cells can successfully store in the buffer and only one cell departing from the queue moves to the multiplexer simultaneously. Other incoming cells, destined to this queue, are dropped. If $N < L+1$, the above situation won't overflow, and P_0 can't go straight to P_{drop}, as shown in Fig. 11.

In the diagram in Fig. 10 ($N \ge L+1$), the buffer has $L+2$ states, each corresponding to a specific number of cells in the buffer. The rates of cell flows are used in deriving the following equilibrium probabilities. If $N \ge L+1$, we derived following equations.

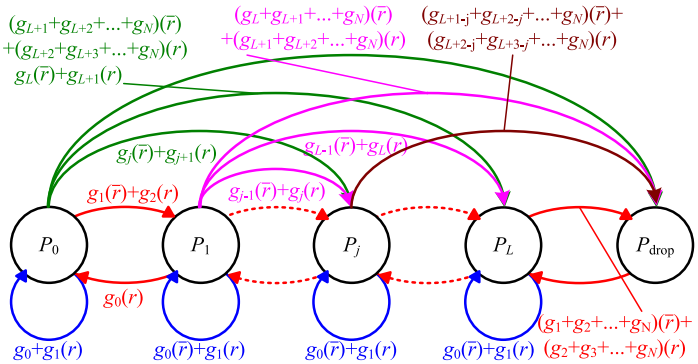


Fig. 10. State transition diagram for output buffer of depth L when $N \ge L+1$.

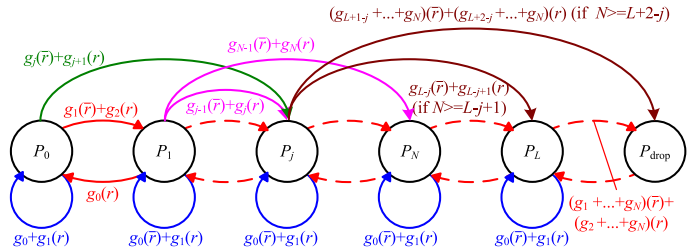


Fig. 11. State transition diagram for output buffer of depth L when $N < L+1$.

$$P_0(t+1) = P_0(t)(g_0 + g_1 r) + P_1(t) g_0 r \tag{2}$$

$$P_j(t+1) = \sum_{n=0}^{j} P_n(t) g_{j-n} \bar{r} + \sum_{n=0}^{j+1} P_n(t) g_{j+1-n} r, 1 \le j \le L-1 \tag{3}$$

$$P_L(t+1) = \sum_{n=1}^{L} P_n(t) g_{L-n} \bar{r} + \sum_{n=0}^{L} P_n(t) g_{L+1-n} r + P_{drop}(t) \tag{4}$$

$$P_{drop}(t+1) = \sum_{n=0}^{L} \sum_{i=L+1-n}^{N} P_n(t) g_i \bar{r} + \sum_{n=0}^{L} \sum_{i=L+2-n}^{N} P_n(t) g_i r. \tag{5}$$

In Fig. 11, we now present a state-transition diagram for output buffer of depth L when $N < L+1$. For simplicity, in the figure we show only transitions into a reference state j and a few relevant states. If $N < L+1$, the following equations are obtained.

$$P_0(t+1) = P_0(t)(g_0 + g_1 r) + P_1(t) g_0 r \tag{6}$$

$$P_j(t+1) = \sum_{n=0}^{j} P_n(t) g_{j-n} \bar{r} + \sum_{n=0}^{j+1} P_n(t) g_{j+1-n} r, \quad 1 \le j \le N-1 \tag{7}$$

$$P_j(t+1) = \sum_{n=j-N}^{j} P_n(t) g_{j-n} \bar{r} + \sum_{n=j-N+1}^{j+1} P_n(t) g_{j+1-n} r, \ N \le j \le L-1 \tag{8}$$

$$P_L(t+1) = \sum_{n=L-N}^{L} P_n(t) g_{L-n} \bar{r} + \sum_{n=L-N+1}^{L} P_n(t) g_{L+1-n} r + P_{drop}(t) \tag{9}$$

$$P_{drop}(t+1) = \sum_{n=L-N+1}^{L} \sum_{i=L+1-n}^{N} P_n(t) g_i \bar{r} + \sum_{n=L-N+2}^{L} \sum_{i=L+2-n}^{N} P_n(t) g_i r. \tag{10}$$

B. Analysis for the SW-PPS

Because SW scheme will choose one switch whose Q_d is minimum (shown in steps 404 and 406 in the Fig. 6), we obtained internal load (ρ') of the SW-PPS and g_i.

$$\rho' = \rho / \binom{1^{K(1-\rho)+1}}{1}. \tag{11}$$

For the above equation, an arriving cell will pick one of the switches that conform to the PPS restriction, and Q_d of the selected switch is the minimum. $K(1-\rho)+1$ is the number of switches that conform to the PPS restriction. When the external load (ρ) increased, the number of switch conforming to the restriction of PPS decreased. For example, when $K = 10$ and $\rho = 0.8$, the average number of switches complying with the restriction is three. Therefore, the arriving cell can select the minimum Q_d from these three switches.

$$g_i = \binom{N}{i} (\rho'/N)^i (1 - \rho'/N)^{N-i}, \quad 0 \le i \le N. \tag{12}$$

In (12), the possibility of selecting one from N output queues is ρ'/N for the coming cell, and g_i is the probability that a total of i cells come to the identical output queue (see Fig. 9). The part of the P_0, P_j, P_L, and P_{drop} for SW-PPS is the same as (2) to (10), and $r = 1$ (due to the HOL blocking problem free). The Markov chain for the SW-PPS is similar to that of RR-PPS. However, the difference is that SW-PPS uses the (12) but RR-PPS adopts the (1) to derive g_i. Now that all the steady-state probabilities are available, the three primary performance measures are given as follows:

$$Drop\_Rate(\rho, t, N, K, L) = P_{drop}(t). \tag{13}$$

Because a cell arriving at the switch either passes through or drops out of the system, the throughput and internal cell delay of a PPS buffer are obtained as follows:

$$\begin{aligned} Throughput(\rho, t, N, K, L) &= (Input\_Load) - (Drop\_Rate) \\ &= \rho - P_{drop}(t) \end{aligned} \tag{14}$$

$$\begin{aligned} &Internal\_Delay(\rho, t, N, K, L) \\ &\quad = \frac{[\sum_{i=1}^{L} i P_{drop}(t)] + L P_{drop}(t)}{1 - P_0(t)}. \end{aligned} \tag{15}$$

V. COMPARISON AND SIMULATION

A. Comparison of Analytical Results and Simulation Results

A time-progression method is used to calculate the analytical results for RR-PPS and SW-PPS. This approach works with (1) to (15) as follows. First, initial values are entered into the performance probability equations: $P_0(0) = 1$ and $P_j(0) = P_L(0) = P_{drop}(0) = 0$. The values of the following probabilities in the next time step are then calculated: $P_0(1)$, $P_j(1)$, $P_L(1)$, and $P_drop(1)$. The computation continues until $P_0(t)$, $P_j(t)$, $P_L(t)$, and $P_{drop}(t)$ reach steady-state values, which are then substituted into the closed forms of the three performance measures and the numerical results are computed.

The measures of interest used to evaluate performance are mean throughput, cell loss ratio, and average internal cell delay. Suppose a 64×64 PPS. And suppose the number of center-stage OQ switches (K) = 128 and 64, and the queue size (L) = 16 and 4 cells. At each input port, cells are generated by a Bernoulli process with the same rate [9], [11]. Figs. 12–17 show the mathematical analysis and simulation results for SW (K, L, x) and RR (K, L, x). The parameter $x = S$ represents the simulation result (dotted curves in Figs. 12–17), and $x = Q$ represents the result of queuing analysis. We can find that analytical results are similar to the simulated results. Hence, the experimental results indicate accuracy of the analytical results. Our proposed model and analysis successfully exhibit these performance characteristics. In (11), $(K(1-\rho)+1)$ has to be an integer value. Hence, input load (ρ) of the analysis model must plug one but that of

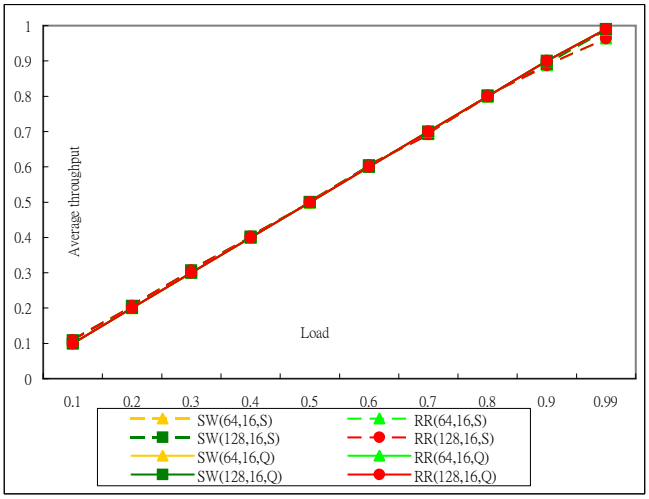


Fig. 12. Comparison of throughput ($L = 16$).

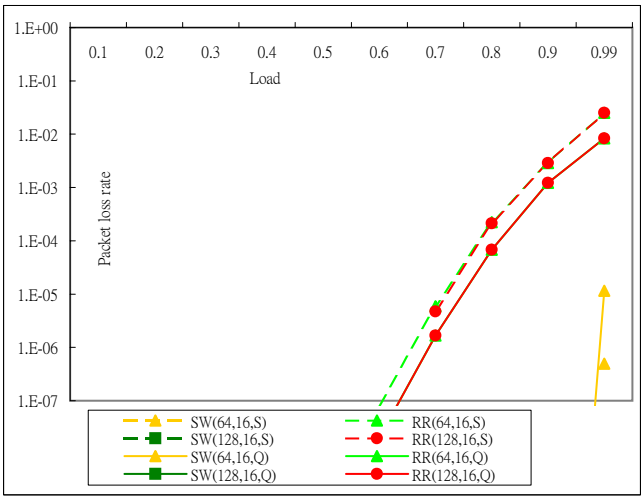


Fig. 14. Comparison of cell drop rate ($L = 16$).

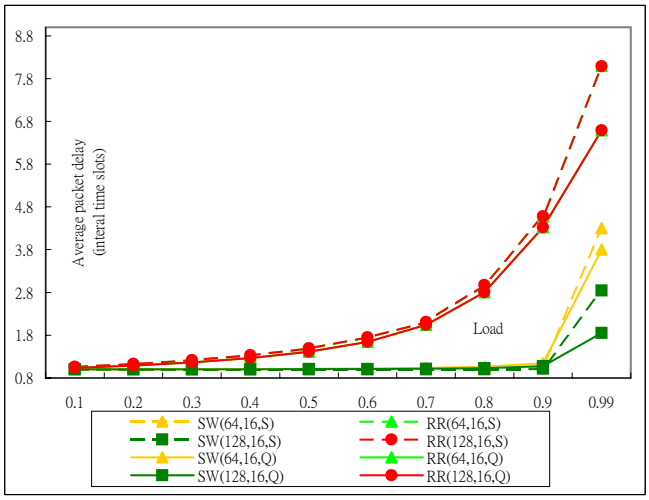


Fig. 13. Comparison of cell delay ($L = 16$).

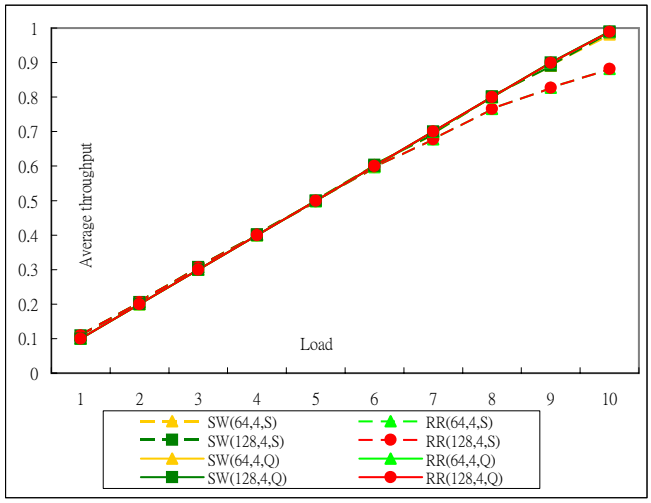


Fig. 15. Comparison of throughput ($L = 4$).

the simulation can plug 0.99, when traffic load is 0.99. Consequently, the SW-PPS result of the simulation and the analysis are little different when traffic load is 0.99 in Figs. 12–17.

Figs. 12–17 compare the performance for SW (K, L, S) and RR (K, L, S). When $L = 16$ cells, the SW-PPS provides slightly higher throughput (Fig. 12). It has a lower cell delay and cell drop rate than the RR-PPS (Figs. 13 and 14). Under the same conditions with queue size $(L) = 4$ cells, Figs. 15–17 compare the performance of SW (K, L, S) with that of RR (K, L, S). The RR-PPS's throughput falls to 88%, but the SW-PPS's throughput remains at around 99% (Fig. 15). In Figs. 16 and 17, when the queue size decreases, SW-PPS still outperforms RR-PPS. This reveals that even when less memory is available, SW-PPS performs well; in other words, SW-PPS employs memory more effectively than RR-PPS. The SW scheme selects one of the switches that conform to the PPS restriction, and Q_d of the selected switch is the minimum. Moreover, $Switch[i].Q_d$ increased with the number of cells in the dth memory module of the ith switch. In other words, to store arriving cells, SW scheme will choose the particular memory module having the minimum number of cells. Therefore, SW-PPS employs memory more effectively than RR-PPS, and supplies better cell drop rate, cell delay, and throughput. In the simulation results, a small queuing delay in the parameter assignment circuit is neglected, because in Section III-C, we explain that the complexity the SW scheme is O(N), and the memory size of the assignment circuit can be sufficiently small to allow the memory to be placed on chip using today's SRAM technology.

B. Simulation Result for Bursty Traffic Model

A bursty traffic [7], [8], [11] is produced employing a two state ON-OFF model to investigate performance of the PPS architecture. The bursty traffic model was generated using a two-state ON-OFF model in which an arrival process to an input port alternates between ON (active) and OFF (idle) periods (Fig. 18). During the ON period, cells arrive at an input port continuously in consecutive cell time slots, and no cells are generated during the OFF period. In other words, cells arrive in consecutive slots in the ON period and no cells arrive in the OFF period. If an input is in the OFF state, it will switch to the ON state with

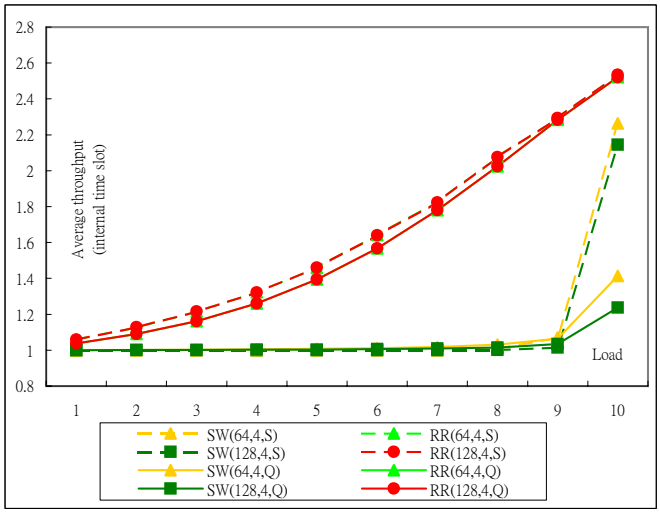


Fig. 16. Comparison of cell delay ($L = 4$).

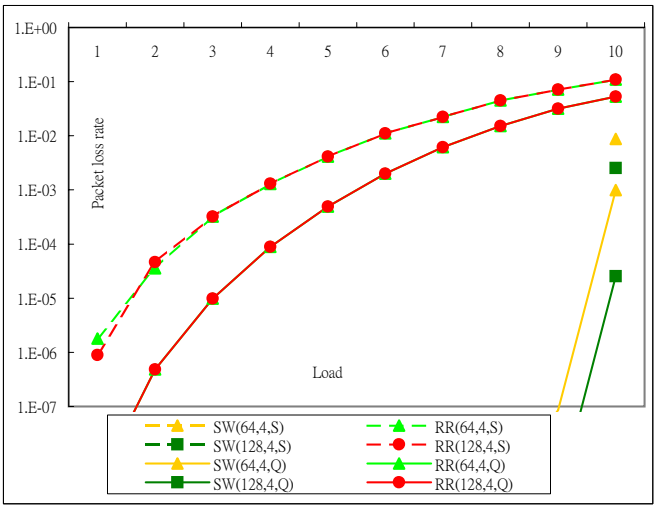


Fig. 17. Comparison of cell drop rate ($L = 4$).

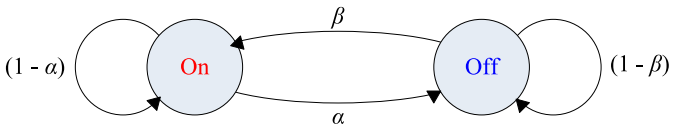


Fig. 18. Two-state on-off model.

probability β, while it will stay in the same state with probability $1-\beta$. If an input is in the ON state, it will switch to the OFF state with probability α, while it will stay in the same state with probability $1-\alpha$. Hence, $1-\alpha = \Pr$ [it is in ON state at $t+1$ | it is in ON state at t], $1-\beta = \Pr$ [it is in OFF state at $t+1$ | it is in OFF state at t]. The average burst length (ABL), average idle length (AIL), and average load (AL) are derived as follows.

$$ABL = \sum_{K=1}^{\infty} K\alpha(1-\alpha)^{K-1} = 1/\alpha \tag{16}$$

$$AIL = \sum_{K=1}^{\infty} K\beta(1-\beta)^{K-1} = 1/\beta \tag{17}$$

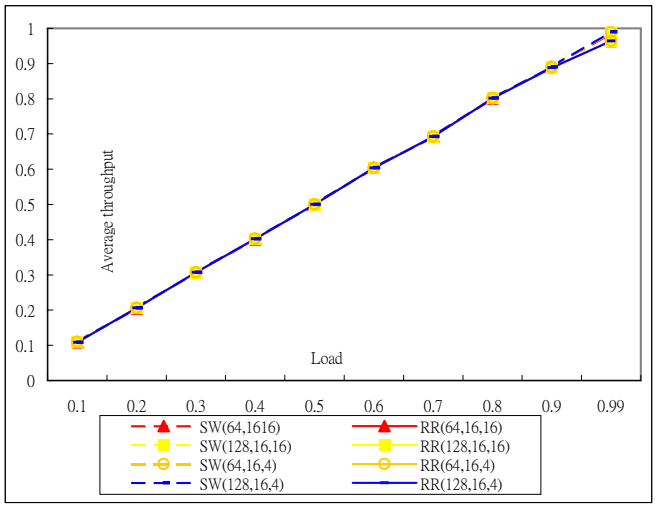


Fig. 19. Comparison of through ($L = 16$).

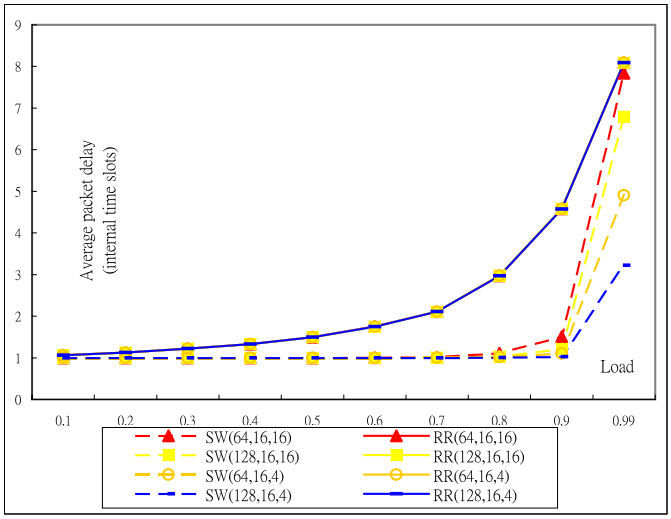


Fig. 20. Comparison of cell delay ($L = 16$).

$$AL = \frac{ABL}{ABL + AIL} = \frac{\beta}{\alpha+\beta}. \tag{18}$$

In this section, we employ the bursty traffic model and assume that $ABL = 16$ and 4 cells. Suppose a 64×64 PPS and that the number of center-stage OQ switches was $K = 128$ and 64 with queue length $(L) = 16$ and 4. The input load ρ varies from 0.1 to 0.99. Figs. 19–24 compare the performance of SW (K, L, ABL) with that of RR (K, L, ABL). SW-PPS provided a higher throughput than RR-PPS (Fig. 19) while $L = 16$ cells. The cell delay and cell drop rate for SW-PPS are apparently lower than those of RR-PPS (Figs. 20 and 21). When queue size is $L = 4$ cells, Figs. 22–24 show the comparison of the performance of SW (K, L, ABL) and RR (K, L, ABL). It is observed that SW-PPS still provided a higher throughput (of about 98%, Fig. 22) than RR-PPS (about 87%, Fig. 22) under bursty traffic. In Figs. 23 and 24, under the queue size$= 4$, the internal cell delay and cell drop rate for SW-PPS are obviously lower than those of RR-PPS. Therefore, even under the bursty traffic, the SW-PPS outperforms regardless of reduction in memory

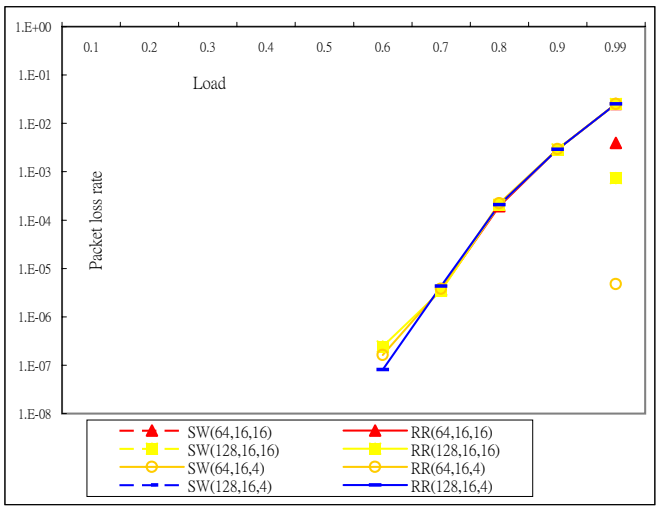


Fig. 21. Comparison of cell drop rate ($L = 16$).

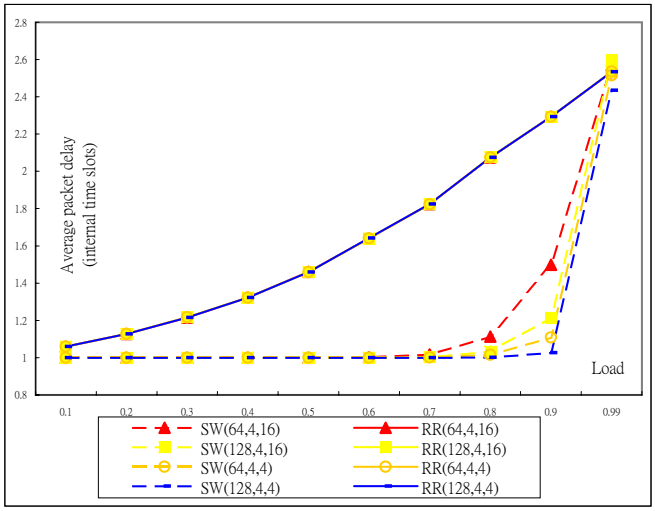


Fig. 23. Comparison of cell delay ($L = 4$).

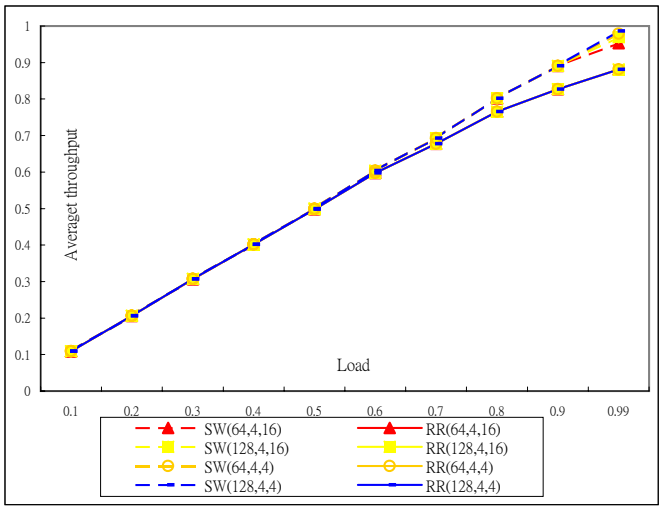


Fig. 22. Comparison of throughput ($L = 4$).

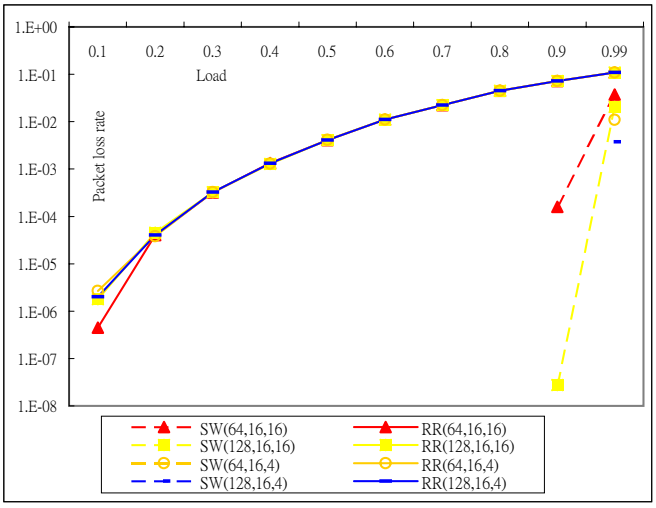


Fig. 24. Comparison of cell drop rate ($L = 4$).

size; the experimental results show that the SW-PPS offers better memory space efficiency.

VI. CONCLUSION

Higher line rates may soon exceed the speed of commercially available memories. It becomes difficult to buffer packets as fast as they arrive when using traditional packet switches. Accordingly, this investigation proposes a new SW packet switching scheme for the PPS, called the SW-PPS. It has the RR-PPSs advantages that all memory buffers and internal line rate run slower than the external line rate. However, it uses memory more effectively. Clearly, even if the memory space is reduced, the SW-PPS still performs well. Moreover, in the SW-PPS each incoming cell has a self-routing tag, enabling it to be self-routed through the switching system. The SW-PPS provides a higher throughput, lower cell delay, and lower cell drop rate than the RR-PPS under identical Bernoulli traffic and bursty traffic. Another significant contribution of this work is that we propose a new analytical model for evaluating RR-PPS and SW-PPS. The proposed analytical model is a general one in the sense that it assumes arbitrary switch size, buffer size, and number of center-stage switches.

ACKNOWLEDGMENTS

The authors are grateful to Prof. Lin of the National Chung-Hsing University. He offered the author very useful advice. The authors are also thankful to Dr. Wu of the Nan Kai Institute of Technology for their help and encouragement, and wish to thank them for useful discussions.

REFERENCES

[1] M. Karol, M. Hluchyj, and S. Morgan, "Input versus output queueing on a space-division packet switch," *IEEE Trans. Commun.*, vol. 35, no. 12, pp. 1347–1356, Dec. 1987.

[2] J. Hui and E. Arthurs, "A broadband packet switch for integrated transport," *IEEE J. Sel. Areas Commun.*, vol. 5, no. 8, pp. 1264–1273, Aug. 1987.

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

[4] M. Yang and S. Zheng, "An efficient scheduling algorithm for cioq switches with space-division multiplexing expansion," in *Proc. IEEE INFOCOM*, vol. 3, 2003, pp. 1643–1650.
[5] S. Iyer and N. McKeown, "Analysis of the parallel packet switch architecture," *IEEE/ACM Trans. Networking*, vol. 11, no. 2, pp. 314–324, Apr. 2003.
[6] W. Wang, L. Dong, and W. Wolf, "A distributed switch architecture with dynamic load-balancing and parallel input-queued crossbars for terabit switch fabrics," in *Proc. INFOCOM*, vol. 1, 2002, pp. 352–361.
[7] A. Aslam and K. Christensen, "Parallel packet switching using multiplexors with virtual input queues," in *Proc. LCN*, 2002, pp. 270–277.
[8] S. Kumar, "The sliding-window packet switch: A new class of packet switch architecture with plural memory modules and decentralized control," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 4, pp. 656–673, May 2003.
[9] S. Kumar and D. Agrawal, "The sliding-window approach to high performance atm switching for broadband networks," in *Proc. IEEE GLOBECOM*, vol. 1, 1996, pp. 772–777.
[10] C. Clos, "A study of nonblocking switch networks," *Bell System Technical J.*, vol. 32, no. 2, pp. 406–424, 1953.
[11] C. L. Liao and W. Lin, "Performance analysis of general cut-through switching on buffered min switches," *J. Inf. Science Eng.*, vol. 18, no. 5, pp. 745–762, 2002.
[12] T. Szymanski and S. Shaikh, "Markov chain analysis of packet-switched banyans with arbitrary switch sizes, queue sizes, link multiplicities and speedups," in *Proc. IEEE INFOCOM*, vol.3, 1989, pp. 960–971.

Chia-Lung Liu received the B.S. degree from the Department of Computer Science at Tamkang University, Taiwan, in 2001, and the M.S. degree from the Institute of Computer Science at National Chung-Hsing University, Taiwan, in 2002.

He is currently working toward the Ph.D. degree in the computer science area from the Institute of Computer Science at National Chung-Hsing University, Taiwan. His research areas include high-speed switching, computer networks, parallel processing, and performance evaluation.

Chin-Chi Wu received the B.S. and M.S. degrees in Department of Computer Science from National Chiao-Tung University in 1990 and 1992, respectively. He is working toward the Ph.D. degree in the Institute of Computer Science, National Chung- Hsing University.

He is currently a lecturer with the Department of Information Management, Nan-Kai Institute of Technology, Nantou, Taiwan. His research interests include computer networks, parallel processing, and high-speed switching.

Woei Lin received the B.S. degree from National Chiao-Tung University, Taiwan, in 1978, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Texas at Austin, USA in 1982 and 1985, respectively.

He is currently a professor with the Institute of Computer Science, National Chung-Hsing University, Taiwan, R.O.C. He was an associate professor of Department of Electrical Engineering at University of Hawaii, and an assistant professor of Department of Electrical Engineering at Penn State University, USA. His research areas include high-speed switching, computer networks, parallel processing, and performance evaluation.