Making Parallel Packet Switches Practical*

Sundar Iyer, Nick McKeown
Computer Systems Laboratory, Stanford University
Stanford, CA 94305-9030
{sundaes, nickm}@stanford.edu

Abstract -- **A parallel packet switch (PPS) is a switch in which the memories run slower than the line rate. Arriving packets are spread (or load-balanced) packet-by-packet over multiple slower-speed packet switches. It is already known that with a speedup of $S \geq 2$, a PPS can theoretically mimic a FCFS output-queued (OQ) switch. However, the theory relies on a centralized packet scheduling algorithm that is essentially impractical because of high communication complexity. In this paper, we attempt to make a high performance PPS practical by introducing two results. First, we show that small co-ordination buffers can eliminate the need for a centralized packet scheduling algorithm, allowing a full distributed implementation with low computational and communication complexity. Second, we show that without speedup, the resulting PPS can mimic an FCFS OQ switch within a delay bound.**

Keywords--**packet-switch; output-queueing; inverse-multiplexing; load-balancing; Clos' network.**

I. INTRODUCTION

The capacity of high performance packet switches (e.g. Internet routers and ATM switches) is limited by the random access time of commercially available memories. While switching capacity requirements have grown, random access times in commercial DRAMs have remained essentially unchanged.[1] This has lead to an evolution of packet switch architectures from output-queued (OQ) and shared memory designs (in which the memory bandwidth must equal the capacity of the switch), to input-queued (IQ) or combined input and output-queued (CIOQ) designs where the memory bandwidth need equal approximately the data rate of a single line.

As line rates increase beyond OC192 (10Gb/s) to say OC768 (40Gb/s) — and even OC3072 (160Gb/s) — it becomes difficult, perhaps impossible, to buffer packets as fast as they arrive. For example, a 40-byte TCP segment arriving on a 160Gb/s line must be written to and read from a buffer in less than 1ns. This should be compared to the 50ns random access time of today's DRAMs.

The purpose of this paper is not to argue that line rates will continue to increase — on the contrary, it could be argued that DWDM will lead to a larger number of logical channels each operating no faster than, say, 10Gb/s. We simply make the observation that if line rates do increase, then memory bandwidth limitations may make packet buffers difficult or impossible to implement.

II. BACKGROUND

In a previous paper [1] we proposed the parallel packet switch (PPS) as a way to overcome the memory bandwidth limitation. A key attribute of the PPS is that its packet buffers can run slower than the line rate; by increasing parallelism they can be made to operate arbitrarily slowly. The PPS architecture is illustrated in Figure 1, and is based on the 3-stage Clos Network [2]. The main difference is that a Clos network is an unbuffered switch fabric, whereas the PPS contains buffered packet switches in its center stage. Figure 1 shows an example of a 4×4 PPS. Each port is connected to all three *output-queued* center stage switches which operate independently and in parallel. Packets arriving at an input port are examined by the demultiplexor, then sent to one of the slower speed center stage switches (or "layers"). Packets are processed individually; i.e. there is no guarantee that packets belonging to the same flow or to the same output will pass through the same layer. In fact, the demultiplexor will ideally spread packets equally over all layers. Packets are stored in the output-queues of the center stage switches and are delivered to the multiplexor at their time of departure. The architecture as such is not novel and previous work [4][5][6][7][8] has described load-balancing or "inverse-multiplexing" [9][10][11][12] systems. However, we are not aware of other published work that studies the performance of a PPS.

Figure 1 shows that packets[2] from each input operating at line rate R are sent over k links each operating at a rate of at least R/k. In general, the internal links in the center stage switches operate at a rate $S(R/k)$, where S is the *speedup*.

We previously explored whether a PPS can be made to mimic[3] an output queued switch. In particular, we proved the following theorem in [1].

1. The random access time (the time to retrieve data at random from any memory location) should not be confused with the memory I/O time (the time to send retrieved data off-chip to the requester). While new memory technologies, such as RAMBUS [3], SDRAMs and DDRAMs have fast I/O times, they use memory cores with random access times of approximately 50ns.

*This research was supported by the National Science Foundation, under NGI contract ANI-9872761, the Industrial Technology Research Institute (Taiwan) and the Alfred P. Sloan Foundation.

2. The terms packet and cell are used inter-changeably throughout the rest of this paper.

3. Two switches are said to mimic [13][14][15][16] each other, if under identical inputs, identical packets depart from each switch at the same time.

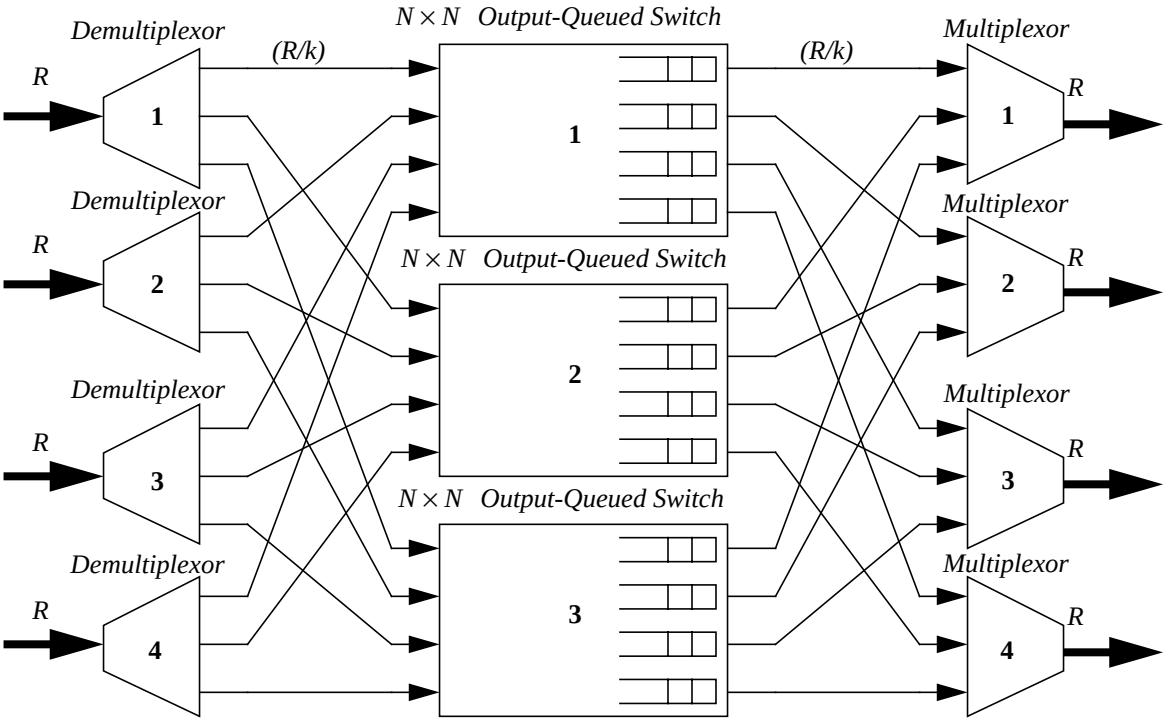


Figure 1: The architecture of a Parallel Packet Switch based on output-queued switches.

Theorem 1:(Sufficiency) With a speedup of $S \geq 2$, a PPS can mimic a FCFS OQ unicast switch.

A. Limitations of Previous Work

Unfortunately, Theorem 1 can only really be viewed as a theoretical, rather than a practical result — the demultiplexor must run a packet scheduling algorithm (called CPA [1]) that limits the capacity of the PPS for the following two reasons.

1. **Communication complexity.**
   CPA requires each input to contact a centralized scheduler every arbitration cycle. With N ports, N requests must be communicated to and processed by the arbiter each cycle. This requires a high speed control path running at the line rate between every input and the CPA scheduler. Furthermore, CPA requires that the departure order (i.e. the order in which packets are sent from each layer to a multiplexor), be conveyed to each multiplexor and stored.

2. **Speedup**.
   CPA requires a speedup of two in the center stage switches. The PPS therefore over provisions the required capacity by a factor of two and the links are on average only 50% utilized.

In addition to the difficulty of implementation, CPA does not distribute traffic equally among the center stage switches, making it possible for buffers in a center stage switch to overflow even though buffers in other switches are not full. This leads to inefficient memory usage.[4]

Another problem with CPA is that it requires each multiplexor to explicitly read, or fetch, each packet from the correct layer in the correct sequence. This feedback mechanism makes it impossible to construct each layer from a pre-existing unaltered switch or router.

In summary, our previous results lead to *large communication complexity, high speedup requirement, inefficient utilization of buffer memory,* and *special-purpose hardware for each layer.* In this paper, we overcome these problems via the introduction of small memories (presumably on-chip) in the multiplexors and demultiplexors, which:

1. Enable the demultiplexors and multiplexors to operate independently, eliminating the communication complexity,

2. Remove the speedup requirement for the internal layers,

3. Allow the buffers in the center stage switches to be utilized equally, and

4. Allow a feed-forward data path in which each layer may be constructed from pre-existing, “standard” output-queued switches.

In Section III, we introduce some terminology and definitions

[4] It is possible to create a traffic pattern that does not utilize up to 50% of the buffer memory for a given output port.

which will be used in the rest of this paper. In Section IV-A, we introduce a distributed layer selection algorithm for a PPS. In Section IV-B we see how the distributed layer selection algorithm for a PPS combined with a small co-ordination buffer in the multiplexors can mimic an FCFS OQ switch. In Section IV-C, we show how the distributed layer selection algorithm combined with a buffer in both the multiplexors and the demultiplexors can eliminate both the speedup and communication complexity in a PPS. Finally in Section V, we discuss how a PPS can be implemented.

III. Definitions

Definition 1: ***Cell*** *— A fixed-length packet (though not necessarily equal in length to a 53-byte ATM cell). Variable length packets arriving to a PPS are assumed to be segmented into cells, carried across the switch, then reassembled prior to departure. This is common in high performance packet switches, and is not discussed further here.*

Definition 2: ***Time slot*** *— The time taken to transmit or receive a cell at a link rate of R.*

Definition 3: ***Internal time slot*** *— This is the time taken to transmit or receive a fixed length cell at a link rate of R/k, where k is the number of center stage switches in the PPS.*

Definition 4: ***Shadow OQ switch*** *— In this paper, we will assume that there exists an OQ switch, called the "shadow OQ switch", with the same number of input and output ports as the PPS. The line interfaces of the shadow OQ switch also operate at line rate R and receive identical input traffic patterns as the PPS.*

Definition 5: ***Relative queueing delay*** *— Consider a PPS switch and an OQ switch that both receive the same stream of cells. A cell's relative queueing delay is the increased queueing delay (if any) that it receives in the PPS switch relative to the delay it receives in the OQ switch. Note that relative queueing delay only includes differences attributed to queueing. Differences in fixed delay (e.g. because of differences in propagation delay) are not included in this measure.*

Definition 6: ***Input link constraint —*** *Because each internal link runs at rate $S(R/k)$, an input demultiplexor can send a cell to a specific layer at most once every $\lceil k/S \rceil$ time slots.*

Definition 7: ***Available input link set —*** *AIL(i,n) is the set of layers to which input demultiplexor i can start sending a cell in time slot n. From [1], $|AIL(i, n)| \geq k - \lceil k/S \rceil + 1$.*

Definition 8: ***Output Link Constraint*** *— Because each internal link runs at rate $S(R/k)$, a layer can send a cell to an output multiplexor at most once every $\lceil k/S \rceil$ time slots.*

Definition 9: ***Departure Time —*** *When a cell arrives, the demultiplexor selects a departure time for the cell. A cell arriving to input i at time slot n and destined to output j is assigned the departure time $DT(n, i, j)$.*

Definition 10: ***Available Output Link Set —*** *$AOL(j, DT(n, i, j))$, is the set of layers that can send a cell to external output j at time slot $DT(n, i, j)$ in the future. From [1], $|AOL(j, DT(n, i, j))| \geq k - \lceil k/S \rceil + 1$.*

IV. A Distributed Approach

In the centralized algorithm (CPA) the most complex decisions are made by the demultiplexors. If, at time n, a cell arrives at input i destined for output j, then port i's demultiplexor must choose the layer to send the cell to based on $AIL(i, n)$ and $AOL(j, DT(n, i, j))$. While knowledge of $AIL(i, n)$ is local to input i, $AOL(j, DT(n, i, j))$ requires knowledge of all cells arriving at other input ports destined to output j.

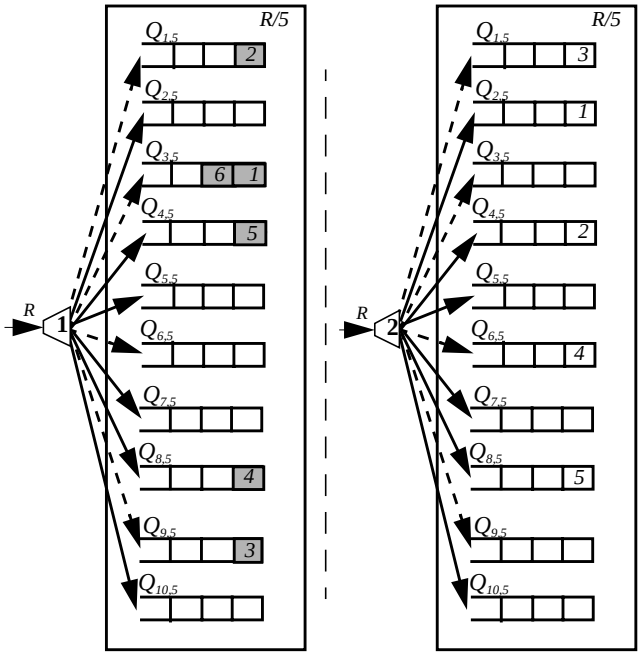
Since our goal is to enable each of the demultiplexors and multiplexors to operate independently, we will consider a distributed algorithm in which the demultiplexors only have local information. In other words, a demultiplexor decides which layer to send a cell to based only on the knowledge of cells that have arrived at its input. When a cell arrives, a demultiplexor determines its departure time, then sends the cell to a layer that can deliver the cell to the correct output without violating the output link constraints.

We will work towards the main result of the paper in three steps. First, we will explore what happens in a PPS with a speedup of $S = 2$ when the demultiplexors use only local information. We find that the PPS comes close to mimicking a FCFS switch, but cells can become mis-sequenced by the PPS, preventing precise mimicking. The mis-sequencing can be bounded which motivates — in the second step — the addition of a small co-ordination buffer in each multiplexor to resequence the cells before transmitting them on the external line. The co-ordination buffer operates at the line rate, R, and so compromises our original goal of having no memories running at the line rate. However, under certain conditions, the buffer is small enough to be placed on-chip, and so may be acceptable. In the third and most important step, we introduce another co-ordination buffer of the same size in the demultiplexor. We find that this allows the PPS to mimic an FCFS OQ switch without speedup.

A. Step 1: Can a PPS, with a speedup of 2, mimic a FCFS OQ switch using only local information?

Our distributed algorithm will use independent demultiplexors that have only local information. The following three definitions will help us describe the algorithm:

Definition 11: ***Local departure time*** *— When a cell arrives the demultiplexor selects the cell's departure time. If the departure*



(a) FCFS order of cells from input 1

(b) FCFS order of cells from input 2

Figure 2: Insertion of cells in a FCFS order for output 5 in a PPS with ten layers and speedup two. $Q_{k,5}$ refers to output queue number 5 in the internal switch k. The numbers in the cells denote the FCFS sequence of the arrivals on the demultiplexors. (a) The FIFO order of cells inserted from input one. Cells sent by input one are shown shaded. (b) The FIFO order of cells inserted from input two. These cells are shown unshaded.

time is determined locally by an independent demultiplexor (without knowledge of cells arriving at different inputs), then we call it a local departure time. A cell arriving to input i at time slot n and destined to output j is assigned the local departure time $LDT(n, i, j)$.

Definition 12: ***Conflict free order —*** *An ordering of cells destined to output j is said to be conflict-free if output j can transmit these cells in order without violating the output link constraints.*

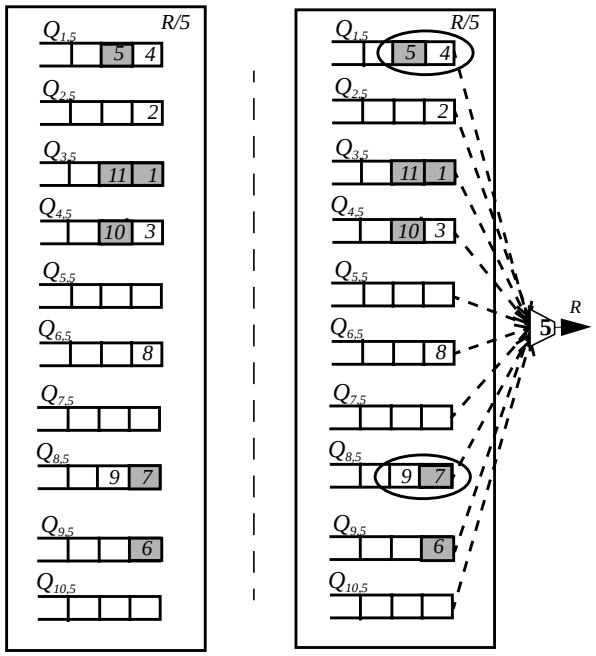
Definition 13: ***Local AOL set —*** *The local available output link set $LAOL(j, LDT(n, i, j))$ is the set of layers that have not been sent any of the previous $\lceil k/S \rceil - 1$ cells from input i to output j. The LAOL set, unlike the AOL set, is oblivious to cells arriving at different inputs. By definition, the LAOL set leads to a conflict-free order of cells.*

The algorithm. Each demultiplexor i maintains an available input link set $AIL(i, n)$ and, for each output j, a local available output link set $LAOL(j, LDT(n, i, j))$. Demultiplexor i selects local departure time $LDT(n, i, j)$ so as to maintain FCFS order for cells between input i and output j, then selects a layer which is common to both $AIL(i, n)$ and $LAOL(j, LDT(n, i, j))$.

An example. An example of decisions made by the independent demultiplexors is illustrated in Figure 2a. It shows a PPS with $k = 10$ center stage switches and a speedup of $S = 2$. The demultiplexors operate at a line rate of R and send cells to the center stage switches over links which operate at a line rate $SR/k = R/5$. The six cells shown arrive at input port 1 and are destined to output port 5. These cells are shown shaded and are distributed by input port 1 in a FCFS conflict free order; i.e. any two cells in the same center stage switch will depart at least $k/S = 5$ time slots apart. Figure 2b illustrates another FCFS conflict-free order of cells (which are shown unshaded) sent by input port 2 to the center stage switches of the same PPS.

Lemma 1: (Sufficiency) A speedup of $S \geq 2$ is sufficient for a PPS with independent demultiplexors to send cells from each input to each output in a conflict-free order.

By definition, the *LAOL* set maintained by input i for output j forms a conflict-free ordering on output j. It suffices to show that there will always exist a layer $l \in \{AIL(i, n) \cap LAOL(j, LDT(n, i, j))\}$, i.e. $AIL(i, n) \cap (LAOL(j, LDT(n, i, j))) \neq \varnothing$, which must be satisfied if $|AIL(i, n)| + |LAOL(j, LDT(n, i, j))| > k$. But we know from Definition 13 that $|LAOL(j, LDT(n, i, j))| \geq k - \lceil k/S \rceil + 1$. So from Defini-



(a) Combined FCFS order

(b) Blocking for cells 5 and 9

Figure 3: Departure of cells in a FCFS order for output 5 in a PPS with ten layers and speedup two. $Q_{k,5}$ refers to output queue 5 in the internal switch k. The numbers in the cells denote the FCFS sequence of the departure on multiplexor 5. (a) The actual correct order of departure of cells. (b) Blocking of requests for reading cell 5 and cell 9.

tions 7 and 13, we can conclude that $|AIL(i, n)| + |LAOL(j, LDT(n, i, j))| > k$ if $S \geq 2$. ❐

The problem. The distributed approach allows a PPS with a speedup of $S \geq 2$ to send cells from each independent demultiplexor in a FCFS conflict-free order to each output j. The problem is that cells destined to output j arrive independently from all inputs, and so the algorithm does not guarantee that cells destined to output j are available to depart in a conflict-free order. To illustrate the point, consider Figure 3 — the same PPS as in Figure 2 with $k = 10$ center stage switches and with a speedup of $S = 2$. In the example, all cells are once again destined to output port 5. Cells arriving to input port 1 (2) are shown shaded (unshaded), respectively. Assuming that the only cells sent to output port 5 are from input ports 1 and 2, a combined FCFS order of departure is shown in Figure 3a (there can be multiple such FCFS departure orders depending on the exact arrival time of cells). In Figure 3b, the multiplexor at output port 5 receives these cells in FCFS departure order. Note that since cells 4 and 5 are consecutive and are queued in layer 1, cell 5 cannot reach the output multiplexor until at least $\lceil k/S \rceil = 5$ time slots after cell 4. Similarly, cell 9 cannot reach the multiplexor until at least 5 time slots after cell 8. Hence, the shadow FCFS OQ switch is not mimicked, cells may reach the multiplexor in non-FCFS order.

B. Step 2: The addition of co-ordination buffers at each multiplexor to enable a PPS to mimic an FCFS OQ switch

If we can bound the time by which a cell can miss its FCFS departure time, we can place a small co-ordination buffer in the multiplexor to re-sequence cells and then transmit them in their correct order. Cells could be arranged to depart at the same time as in the shadow FCFS OQ switch, but delayed by a constant relative queueing delay bound.

To this end, we now modify the PPS slightly, placing a small co-ordination buffer in the multiplexor at each output. (We will determine later how large the co-ordination buffer need be). The PPS still operates with speedup $S = 2$, but the operation of each layer is modified slightly. When a layer has a cell that has reached its departure time, it immediately attempts to send the cell to the multiplexor. If the link to the multiplexor is busy, the layer holds the cell until the link is free. If a cell reaches its output multiplexor ahead of a cell that should depart before it, the cell is buffered until it can depart in its correct FCFS order.

Theorem 2:(Sufficiency) A PPS with a speedup $S \geq 2$, with independent demultiplexors and multiplexors, and with a co-ordination buffer in each multiplexor can mimic a FCFS OQ switch with a relative queueing delay bound of $\lceil N/S \rceil$ internal time slots.

We omit the proof for brevity and because the result is subsumed by Step 3 below.

Intuitively, how can the small co-ordination buffer at each multiplexor allow the demultiplexors to use only local information? It is because the independent demultiplexor can send a cell to a layer that instantaneously violates the output link constraint. The co-ordination buffer affords the layer some short-term flexibility, allowing it to wait for the link to the multiplexor to become free before sending a cell.

C. Step 3: Eliminating speedup by adding a co-ordination buffer at each demultiplexor

We can take this idea one step further by introducing an additional, identical co-ordination buffer at the demultiplexor. The aim is to give the demultiplexor some short-term flexibility as to when it need send a cell to the chosen layer. Rather than delivering the cell to its layer immediately, the demultiplexor holds the cell until the link is free. The key observation is that even with no speedup, the link will become free periodically. Intuitively, if the demultiplexor has enough space to hold the cell until the link is free, then speedup can be eliminated.

The co-ordination buffer in the demultiplexor. Figure 4 shows how buffers are arranged in each demultiplexor as multiple equal size FIFOs, one per layer. FIFO $Q(i, l)$ holds cells at demultiplexor i destined for layer l. When a cell arrives, the demultiplexor makes a local decision (described below) to choose which layer the cell will be sent to. If the cell is to be sent to layer l the cell is queued first in $Q(i, l)$ until the link becomes free. When the link from input i to layer l is free, the head of line cell (if any) is removed from $Q(i, l)$ and sent to layer l.

The buffers in each multiplexor are arranged the same way. And so FIFO $Q'(j, l)$ holds cells at multiplexor j from layer l.

We will refer to the maximum length of a FIFO ($Q(i, l)$ or $Q'(j, l)$) as the FIFO length.[5] Note that if each FIFO is of length d, then the co-ordination buffer can hold a total of kd cells.

Before describing the modified algorithm, we need one more definition.

Definition 16: ***Buffered Available Input Link Set —*** *The buffered available input link set, $BAIL(i, n)$, is the set of layers to which external input port i can start sending a cell between time slot n and $n + dk$, where d is the FIFO length. This is the set of layers for which the number of cells queued for that layer at time slot n is less than the FIFO length.*

Note that $|BAIL(i, n)| \leq k, \forall (i, n)$.

The algorithm. The PPS has no speedup. Each demultiplexor i maintains a buffered available input link set $BAIL(i, n)$ and, for each output j, a local available output link set $LAOL(j, LDT(n, i, j))$. Demultiplexor i selects local departure time $LDT(n, i, j)$ so as to maintain FCFS order for cells between input i and output j, then selects layer l which is common to both $BAIL(i, n)$ and $LAOL(j, LDT(n, i, j))$. The cell is

[5] It will be convenient for the FIFO length to include any cells in transmission.

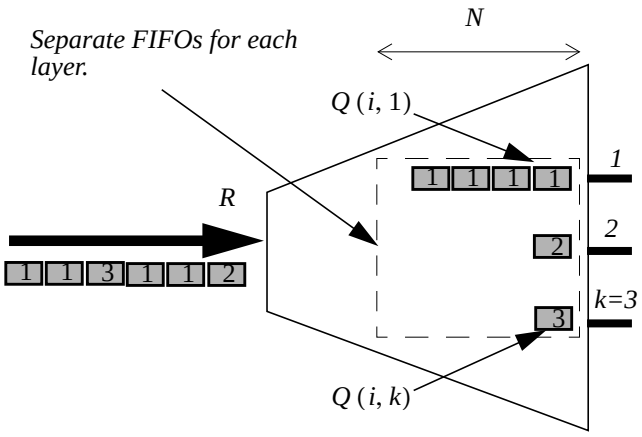


Figure 4: The demultiplexor, showing k FIFOs, one for each layer, and each FIFO of length d cells. The example PPS has $k = 3$ layers.

stamped with $LDT(n, i, j)$ and written to the tail of FIFO $Q(i, l)$ in the demultiplexor. When the link from demultiplexor i to layer l is free, the head-of-line cell (if any) is sent from $Q(i, l)$. Next, the cell is queued in the output queue of the center stage switch. When the departure time of the cell has been reached, the layer sends the cell to the output multiplexor when the link is next free. As before, the co-ordination buffer in the multiplexor holds the cell until its correct FCFS departure time.

Key to the operation of the algorithm is the way that the $LAOL(.)$ set evolves.

Lemma 2: In a PPS with $S = 1$, $LAOL(j, LDT(n, i, j))$ evolves in a round robin manner.

Proof: When $S = 1$, the *LAOL* set is the set of layers that have not sent any of the last $k-1$ cells to external output j. Once an output receives $k-1$ cells from a given input, there can be only one layer available in the *LAOL* set. After that the *LAOL* set changes in a fixed round robin order. ❐

We can now determine the rate at which cells are written into, and read from, the demultiplexor's buffer, and hence determine its size.

Lemma 3: The number of cells $D(i, l, T)$ that demultiplexor i writes to FIFO $Q(i, l)$ in time interval T time slots, is bounded by

$$D(i, l, T) \le T \qquad \text{if } T \le N,$$
$$D(i, l, T) < \frac{T}{k} + N \qquad \text{if } T > N.$$

Proof: The *LAOL* decides which FIFO a cell is written into. From Lemma 2, the *LAOL* set changes in a round robin manner, which means that for every k cells received by a demultiplexor for a specific output, exactly one cell is sent to each layer. We can write $S(i, T) = \sum_{j=1}^{N} \bar{S}(i, j, T)$, where $\bar{S}(i, j, T)$ is the sum of the number cells sent by the demultiplexor i to output j in any time interval of T time slots and $S(i, T)$ is the sum of the number of cells sent by the demultiplexor to all outputs in that time interval T. Let $T > N$. Then, we have,

$$D(i, l, T) \le$$
$$\sum_{j=1}^{N} \left\lceil \frac{\bar{S}(i, j, T)}{k} \right\rceil \le \left\lceil \sum_{j=1}^{N} \frac{\bar{S}(i, j, T)}{k} \right\rceil + N - 1 =$$
$$\left\lceil \frac{S(i, T)}{k} \right\rceil + N - 1 \le \left\lceil \frac{T}{k} \right\rceil + N - 1 < \frac{T}{k} + N$$

since $S(i, T)$ is bounded by T. The proof for $T \le N$ is obvious. ❐

We are now ready to determine the size of the co-ordination buffer in the demultiplexor.

Theorem 3:(Sufficiency) A PPS with independent demultiplexors and no speedup can send cells from each input to each output in a conflict free order with a co-ordination buffer at the demultiplexor of size Nk cells.

Proof: A cell of size C corresponds to C/R units of time, allowing us to re-write Lemma 3 as $D(i, l, T) \le R/Ck + N$ (where T is in units of time). Thus the number of cells written into each demultiplexor FIFO is bounded by $R/Ck + N$ cells per unit time over all time intervals. This can be represented as a leaky bucket source with an average rate $\rho = R/Ck$ cells per unit time and a bucket size $\sigma = N$ cells for each FIFO. Each FIFO is serviced deterministically ar rate $\mu = R/Ck$ cells per unit time. Hence from [17], a FIFO of length N will not overflow. ❐

It now remains for us to determine the size of the co-ordination buffers in the multiplexor. This proceeds in an identical fashion.

Lemma 4: The number of cells $D'(j, l, T)$ that multiplexor j reads from FIFO $Q'(i, l)$ in time interval T time slots, is bounded by

$$D'(j, l, T) \le T \qquad \text{if } T \le N,$$
$$D'(j, l, T) < \frac{T}{k} + N \qquad \text{if } T > N.$$

Proof: Cells destined to multiplexor j from a demultiplexor i are arranged in a round robin manner, which means that

for every k cells received by a multiplexor from a specific input, exactly one cell is read from each layer. We write $S'(j, T) = \sum_{i=1}^{N} \overline{S'}(i, j, T)$, where $\overline{S'}(i, j, T)$ is the sum of the number of cells from demultiplexor i which were delivered to the external line by multiplexor j in time interval T, and $S'(i, T)$ is the sum of the number of cells from all the demultiplexors that were delivered to the external line by the multiplexor in time interval T. Let $T > N$. Then we have,

$$D'(i, l, T) \leq$$

$$\sum_{i=1}^{N} \left\lceil \frac{\overline{S'}(i, j, T)}{k} \right\rceil \leq \left\lceil \sum_{i=1}^{N} \frac{\overline{S'}(i, j, T)}{k} \right\rceil + N - 1 =$$

$$\left\lceil \frac{S'(i, T)}{k} \right\rceil + N - 1 \leq \left\lceil \frac{T}{k} \right\rceil + N - 1 < \frac{T}{k} + N$$

since $S'(i, T)$ is bounded by T. The proof for $T \leq N$ is obvious. ❐

Finally, we can determine the size of the co-ordination buffers at the multiplexor.

Theorem 4:(Sufficiency) A PPS with independent multiplexors and no speedup can receive cells for each output in a conflict-free ordering with a co-ordination buffer of size Nk cells.

Proof: The proof is almost identical to Theorem 3. From Lemma 4, we can bound the rate at which cells in a multiplexor FIFO need to be delivered to the external line by $R/Ck + N$ per unit time over any time interval. Cells are sent from each layer to the multiplexor FIFO at fixed rate $\mu = R/Ck$ cells per unit time. Again from [17], if each FIFO is of length N cells, the FIFO will not overflow. ❐

Now that we know the size of the buffers at the input demultiplexor and the output multiplexor — both of which are serviced at a deterministic rate — we can bound the relative queueing delay with respect to a FCFS OQ switch.

Theorem 5:(Sufficiency) A PPS with independent demultiplexors and multiplexors and no speedup, with each multiplexor and demultiplexor containing a co-ordination buffer of size Nk cells, can mimic a FCFS OQ switch with a relative queueing delay bound of $2N$ internal time slots.

Proof: We consider the path of a cell in the PPS where the cell may potentially face a queueing delay. These are as follows:-

1. The cell may be queued at the FIFO of the demultiplexor before it is sent to its center stage switch. From, Theorem 3, we know that this delay is bounded by N internal time slots.
2. The cell then awaits service in the output queue of a center stage switch for D_{OQ} time slots (the number of time slots it would have been queued in the shadow OQ switch).
3. The cell may then be queued in the FIFO at the multiplexors. From Theorem 4, this is bounded by N internal time slots.

The queuing delay faced by a cell in the PPS is given by $N + D_{OQ} + N$ internal time slots. Thus, the relative queuing delay is bounded by $(N + D_{OQ} + N) - D_{OQ} = 2N$ internal time slots. ❐

V. Implementation Issues

Given that our main goal is to find ways to make a FCFS PPS (more) practical, we now re-examine its complexity in light of the techniques described:

1. Demultiplexor:
- Each demultiplexor maintains a buffer of size Nk cells running at the line rate R, arranged as k FIFOs. Given that our original goal of having no buffers run at the line rate, it is worth determining how large the buffers need be, and whether they can be placed on-chip. For example, if $N = 1024$ ports, cells are 64-bytes long, and $k = 10$, the co-ordination buffer is about 5Mbits per multiplexor and demultiplexor. This can be (just) placed on-chip using today's SRAM technology, and so can be made both fast and wide. However, for much larger *N*, *k* or *C* this approach may not be practicable.
- The demultiplexor must add a tag to each cell indicating the arrival time of the cell to the demultiplexor. Apart from that, no sequence numbers need to be maintained at the inputs or added to cells.

2. **Center stage OQ Switches**:
- The input delay, D_i, (the number of internal time slots for which a cell had to wait in the demultiplexor's buffer) can be calculated by the center stage switch using the arrival timestamp. If a cell arrives to a layer at internal time slot t, it is first delayed until internal time slot $\hat{t} = t + N - D_i$, where $1 \leq D_i \leq N$, to compensate for its variable delay in the demultiplexor. After the cell has been delayed, it can be placed directly into the center stage switch's output queue.

3. Multiplexors:
- Each multiplexor maintains a co-ordination buffer of size Nk running at the line rate R.
- The multiplexor re-orders cells based upon the arrival timestamp. Note that if FCFS order only needs to be maintained between an input and an output then the timestamps can be eliminated. A layer simply tags a cell with the input port number on which it arrived. This would then be a generalization of the methods described in [4].

VI. Conclusions

Typically, the very fastest packet switches have used multiple stages of buffering to avoid the bottleneck of centralized arbitration decisions, and have used speedup to mitigate the effects of blocking. While this approach appears to work well in simulation, analyzing precisely how multi-stage packet switches perform with multiple stages of buffering is difficult.

A PPS, on the other hand, allows switching capacity to be increased (almost) arbitrarily, yet still allows us to analyze the performance, and to mimic the behavior of the well-known and well-studied FCFS OQ switch.

A PPS also enables the construction of switches in which each line operates at faster than the bandwidth of a single buffer memory.

Our results take us a little closer to a general way to continue the growth in capacity of electronic packet switches beyond the memory bandwidth limit. Our work continues with the extension of these results to packet switches that guarantee packet delay for both unicast and multicast traffic.

Acknowledgments

The authors would like to thank Pankaj Gupta for useful discussions.

References

[1] S. Iyer, A. Awadallah, N. McKeown, "Analysis of a packet switch with memories running slower than the line rate," in *Proc. IEEE INFOCOM '00*, pp.529-537.

[2] C. Clos, "A study of non-blocking switching networks," *Bell Systems Technical Journal* 32, 1953.

[3] http://www.rambus.com.

[4] Hari Adiseshu, Guru Parulkar, and George Varghese, "A reliable and scalable striping protocol,". in *Proc. ACM Sigcomm 1996.*

[5] J. Turner, "Design of a broadcast packet switching network," *IEEE Trans. on Communications*, pp. 734-743, June 1988.

[6] I. Widjaja, A. Leon-Garcia, "The helical switch: A multipath ATM switch which preserves cell sequence," *IEEE Trans. on Communications*, Vol. 42, no. 8, pp. 2618-2629, Aug 1994.

[7] F. Chiussi, D. Khotimsky, S. Krishnan, "Generalized inverse multiplexing of switched ATM connections," in *Proc. IEEE Globecom '98 Conference. The Bridge to Global Integration*, Sydney, Australia, Nov. 1998.

[8] F. Chiussi, D. Khotimsky, S. Krishnan, "Advanced Frame Recovery in Switched Connection Inverse Multiplexing for ATM," in *Proc. ICATM '99 Conference*, Colmar, France, Jun. 1999.

[9] P. Fredette, "The past, present, and future of inverse multiplexing," *IEEE Communications*, pp. 42-46, April 1994.

[10] J. Duncanson, "Inverse multiplexing", *IEEE Communications*, pp. 34-41, April 1994.

[11] J. Frimmel, "Inverse multiplexing: Tailor made for ATM," *Telephony*, 231(3), pp 28-34, July 1996.

[12] Bandwidth ON Demand INteroperability Group. Interoperability Requirements for Nx56/64 kbit/sCalls, September 1992.

[13] S. Chuang, A. Goel, N. McKeown, B. Prabhakar, "Matching output queueing with a combined input/output-queued switch," *IEEE J.Sel. Areas in Communications*, Vol. 17, no. 6, pp. 1030-1039, June 1999.

[14] B. Prabhakar and N. McKeown, "On the speedup required for combined input and output queued switching," *Automatica*, Vol. 35, pp. 1909-1920, December 1999.

[15] P. Krishna, N. Patel, A. Charny, R. Simcoe, "On the speedup required for work-conserving crossbar switches," *IEEE J.Sel. Areas in Communications*, Vol. 17, no. 6, pp. 1057-1066, June 1999.

[16] I. Stoica and H. Zhang, "Exact emulation of an output queuing switch by a combined input and output queuing switch," in *Proc. IEEE IWQoS '98*, Napa, CA, May 1998.

[17] R. L. Cruz, "A Calculus for Network Delay," Part I. IEEE Transactions on Information Theory, vol 37, January, 1991, 114-131.