Analysis of a packet switch with memories running slower than the line-rate*

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

***Abstract* -- Our work is motivated by the desire to build a very high speed packet-switch with extremely high line-rates. In this paper, we consider building a packet-switch from multiple, lower speed packet-switches operating independently and in parallel. In particular, we consider a (perhaps obvious) parallel packet switch (PPS) architecture in which arriving traffic is demultiplexed over k identical, lower speed packet-switches, switched to the correct output port, then recombined (multiplexed) before departing from the system. Essentially, the packet-switch performs packet-by-packet load-balancing, or "inverse-multiplexing" over multiple independent packet-switches. Each lower-speed packet switch, operates at a fraction of the line-rate, R ; for example, if each packet-switch operates at rate R/k no memory buffers are required to operate at the full line-rate of the system. Ideally, a PPS would share the benefits of an output-queued switch; i.e. the delay of individual packets could be precisely controlled, allowing the provision of guaranteed qualities of service. In this paper, we ask the question: Is it possible for a PPS to precisely emulate the behavior of an output-queued packet-switch with the same capacity and with the same number of ports? The main result of this paper is that it is theoretically possible for a PPS to emulate a FCFS output-queued packet-switch if each layer operates at a rate of approximately $2R/k$. This simple result is analogous to Clos' theorem for a three-stage circuit switch to be strictly non-blocking. We further show that the PPS can emulate any QoS queueing discipline if each layer operates at a rate of approximately $3R/k$.**

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

I. INTRODUCTION

Wavelength division multiplexing (WDM) is making available long-haul fiber-optic links with very high capacity. WDM makes this possible by allowing a single fiber to contain multiple separate channels; today each channel typically operates at OC48c (2.5Gb/s), OC192c (10Gb/s) and in some systems at OC768c (40Gb/s). The packets or cells carried on each WDM channel are switched, or routed, by packet-switches (e.g. ATM switches, Frame Relay switches and IP routers) that process and then switch packets between different channels. It would be desirable to process packets in the optical domain, without conversion to electronic form. However, all packet-switches need buffering (by definition), and it is not economically feasible today to store packets optically. And so packet-switches will continue to use electronic buffer memories for some time to come.

But at the data rates anticipated for individual WDM channels, we may not be able to buffer packets at the speed at which they arrive and depart. For example, if a memory is 512-bits wide,[1] and buffers packets for a 160Gb/s WDM channel, the memory needs to perform a read or write operation every 1.6ns. This is impractical today; and unfortunately the time to perform a random access into an affordable memory is decreasing only slowly with time.

It is the overall goal of our work to design a high capacity packet-switch (e.g. multiple terabits/second) that: (1) Supports individual line-rates in excess of the speeds of available electronic memory, and (2) Is capable of supporting the same qualities of service as an output-queued switch. These two goals cannot be realized alone by a conventional output-queued (OQ) switch; this is because OQ switches require buffer memory that operates at N times the line-rate, where N is the number of ports of the switch. This certainly doesn't meet our goal of memory running *slower* than any individual line-rate.

Likewise, we can't use the other widely used techniques for reducing memory bandwidth; namely, input-queued (IQ) and combined input and output queued (CIOQ) switches. In an input-queued switch each memory operates at the same speed as the external line-rate. While an improvement over OQ switches neither of our goals are met: (1) An IQ switch does not meet our requirement to use memories slower than the external line-rate, and (2) IQ switches are unable to provide the same QoS guarantees as an OQ switch. Because of the limited QoS capabilities of IQ switches, CIOQ switches were studied and recently, it was shown that a variety of qualities of service are possible in a CIOQ switch in which the memory operates at *twice* the line-rate [1]. Obviously, this doesn't meet our goal for memory speed.

We would like an architecture that overcomes these limitations, yet is practical. To this end, we consider here a parallel packet-switch (PPS) architecture comprised of multiple identical lower-speed packet-switches operating independently and in parallel. An incoming stream of packets is spread, packet-by-packet, by a demultiplexor across the slower packet-switches, then recombined by a multiplexor at the output. As seen by an arriving packet, a PPS is a single-stage packet-switch; all of the buffering is contained in the slower packet-switches, and hence our first goal is met because no buffers in a PPS need run as fast as the external line-rate. The demultiplexor selects an internal packet-switch (or "*layer*") and sends the arriving packet to that layer, where it is queued awaiting its departure time. When the packet's departure time arrives, the multiplexor requests that the packet be removed from its queue, and places the packet on the outgoing

*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.

1. 512-bits, or 64-bytes is about the maximum that is practical or efficient because of ATM cell size and average packet sizes.

line.

It is an important characteristic of a PPS that the multiplexor and demultiplexor functions contain no buffering. However, they must make intelligent decisions; and as we shall see, the precise nature of the demultiplexing ("spreading") and multiplexing functions are key to the operation of the PPS.

We are interested in the question: Can we select the demultiplexing and multiplexing functions so that a PPS can precisely emulate, or mimic, the behavior of an output-queued switch? Two different switches are said to *mimic* each other, if under identical inputs, identical packets depart from each switch at the same time [2]. If it is possible for a PPS to mimic an output-queued switch, it will be possible to control delay of individual packets and therefore provide QoS. In this paper we show that a PPS can precisely emulate an output-queued switch which provides delay guarantees. Furthermore, each layer of the PPS may consist of a single CIOQ switch with memories operating slower than the rate of the external line.

We are not aware of any literature on the PPS architecture, but the architecture itself is not novel; "load-balancing" and "inverse-multiplexing" systems [3][4] have been around for some time, and this architecture is a simple extension of these ideas. There is similar work, which studied inverse ATM multiplexing and how to use sequence numbers to re-synchronize cells sent through parallel switches or links [5][6][7][8]. However, we are not aware of any analytical studies of the PPS architecture. As we shall see, there is an interesting analogy between the (buffered) PPS architecture and the (unbuffered) Clos Network [9].

II. Definitions

Before proceeding, it will be useful to define some terms used throughout this paper:

Cell: A fixed-length packet; not necessarily equal in length to a 53-byte ATM cell. For the purposes of this paper, although packets arriving to the switch may have variable length, we will assume that they are processed and buffered internally as fixed length "cells". This is common practice in high performance routers; variable length packets are segmented into cells as they arrive, carried across the switch as cells, and reassembled back into packets before they depart.

Time slot: Refers to the time taken to transmit or receive a fixed length cell at a link rate of R.

Output-Queued (OQ) Switch: A switch in which arriving packets are placed immediately in queues at the output, where they contend with packets destined to the same output waiting their turn to depart. The departure order may simply be first-come first-served (FCFS) in which case we call it a FCFS-OQ switch. Other service disciplines, such as WFQ [10], GPS [11], Virtual Clock [12], and DRR [13] are widely used to provide QoS guarantees. A characteristic of an OQ switch is that the buffer memory must be able to accept (write) N new cells per time-slot where N is the number of ports, and read one cell per cell time. Hence, the memory must operate at $N + 1$ times the line-rate.

Input-Queued (IQ) Switch: A switch in which arriving cells are queued at the input, where they contend with packets waiting to go to any output. The service discipline is determined by a switch scheduler, or arbiter, which removes at most one cell per time-slot from each input, delivers it across a (usually non-blocking) switch-fabric and onto the outgoing line. A characteristic of an IQ switch is that the buffer memory need only write and read one new cell per time-slot. Hence, the memory must operate at twice the line-rate.

Work-conserving: A system is said to be work-conserving if its outputs never idle unnecessarily. In the context of a packet-switch, this means that an output never idles when there is a cell in the buffers of the packet-switch destined to that output. A consequence of a system being work-conserving is that the throughput is maximized, and the average latency of cells is minimized. An input-queued switch is not, in general, work-conserving because a cell can be held at an input queue while its output idles. An output-queued switch is work-conserving, and so a necessary, but not sufficient, condition for a switch to mimic output-queueing is that it be work-conserving.

PIFO Queues: A "Push-In First-Out" queue ordering is defined according to the following rules:

1. Arriving cells are placed at (or, "push-in" to) an arbitrary location in the queue,
2. The relative ordering of cells in the queue does not change once cells are in the queue, i.e. cells in the queue cannot switch places, and
3. Cells may be selected to depart from the queue only from the head of line.

PIFO queues are quite general and can be used to implement QoS scheduling disciplines such as WFQ, GPS and strict priorities.

III. The Parallel Packet Switch Architecture

In this paper we focus on the specific type of PPS illustrated in Figure 1 in which the center-stage switches are OQ. The figure shows a 4×4 PPS, with each port operating at rate R. Each port is connected to all three output-queued switches (we will refer to the center-stage switches as "layers"). When a cell arrives at an input port, the demultiplexor selects a layer to send the cell to; the demultiplexor makes its

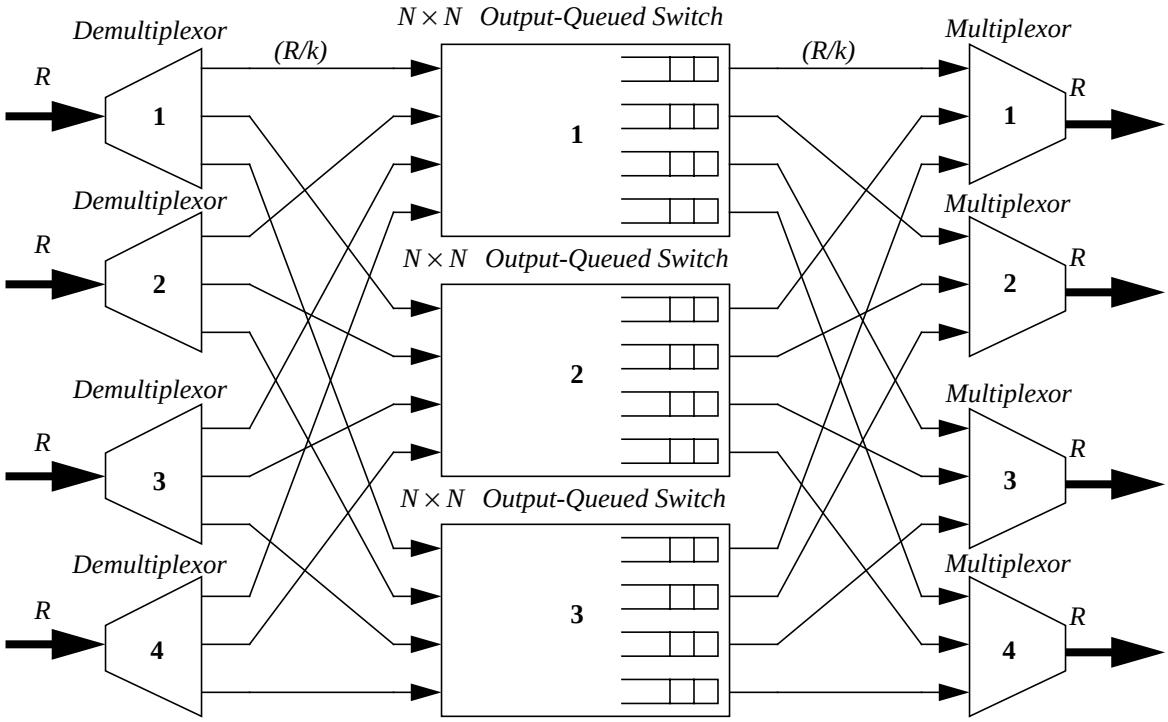


Figure 1: The architecture of a Parallel Packet Switch based on output-queued switches.

choice of layer using a policy that we will describe later. Since the cells from each external input, of line rate R, are spread ("demultiplexed") over k links, each input link must run at a speed of at least R/k, otherwise buffering (operating at the line-rate) would be required in the demultiplexor. Each of the layers receive cells from the N input ports, then switches each cell to its output port. During times of congestion, cells are stored in the output-queues of the center-stage, waiting for the line to the multiplexor to become available. When the line is available, the multiplexor selects a cell from among the corresponding k output queues in each layer. Since each multiplexor receives cells from k output queues, the queues must operate at a speed of at least R/k to keep the external line busy.

Externally the switch appears as an $N \times N$ switch with each port operating at rate R. Note that neither the muliplexor nor the demultiplexor contain any memory, and that they are the only components running at rate R.

We can compare the memory-bandwidth requirements of an $N \times N$ parallel packet-switch with those for an OQ switch with the same aggregate bandwidth. In an OQ switch, the memory bandwidth must be at least $(N + 1)R$, and in a PPS at least $(N + 1)R/k$. But we can reduce the memory bandwidth further using a CIOQ switch. From [1], we know that an OQ switch can be mimicked precisely by a CIOQ switch operating at a speedup of two. So, we can replace each of the output-queued switches in the PPS with a CIOQ switch, without any change in operation. The memory bandwidth in the PPS is reduced to $3R/k$, (one read operation and two write operations per cell time) which is independent of N, and may be reduced arbitrarily by increasing k, the number of layers.

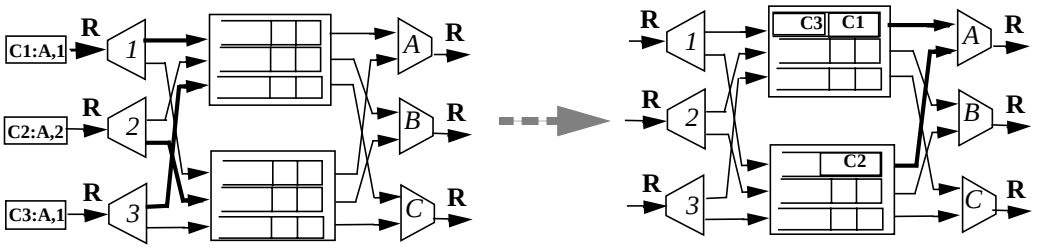
A. The need for speedup

It is tempting to assume that, because each layer is output queued, it is possible for the PPS described above to perform identically to an OQ switch. This is actually not the case unless we use speedup. To see why, we demonstrate that without speedup a PPS is not work-conserving, and hence cannot mimic an OQ switch.

Consider the PPS in Figure 2 with three ports and two layers ($N = 3$ and $k = 2$). The external lines operate at rate R, and the internal lines at rate $R/2$.

Assume that the switch is empty at time $t = 0$, and that three cells arrive, one to each input port, and all destined to output port A. At least two of these inputs will choose the same layer. Let inputs 1 and 3 both choose layer 1 and send cells $C1$ and $C3$ to layer 1 in the first time slot. This is shown in Figure 2a. Input port 2 sends cell $C2$ to layer 2. These cells are shown in the output queues of the internal switches and await departure. In the second time slot, both the input ports which sent cells to the same layer in the first time slot, receive cells destined to output port B. As shown in the figure, cells $C4$ and $C5$ arrive at input ports 1 and 3 and they both must be sent to layer 2; this is because the internal line rate between the demultiplexor and each layer is only $R/2$, limiting a cell to be sent over this link only once every other time-slot. Now the problem becomes apparent: cells $C4$ and $C5$ are in the same layer, and they are the only

(a) Cells arriving at time slot 0 and being sent to the middle stage switches



(b) Cells arriving at time slot 1 and being sent to the middle stage switches

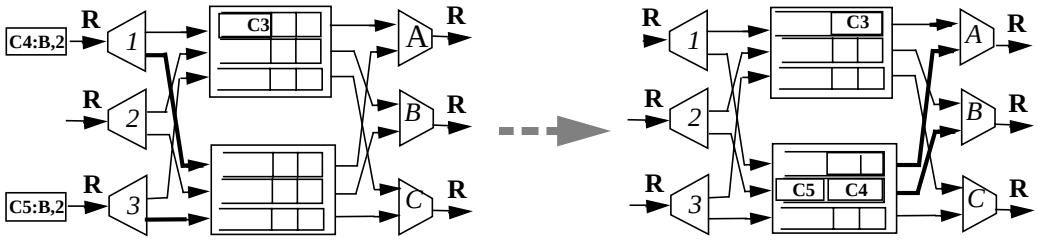


Figure 2: A $3x3$ PPS with an arrival pattern that makes it non work-conserving. The notation $Ci:A, m$ denotes a cell numbered i, destined to output port A, and sent to layer m.

cells in the system destined for output port B at time slot 1. These two cells cannot be sent back-to-back in consecutive time-slots, because the link between the layer and the multiplexor operates only at rate $R/2$. So, cell $C4$ will be sent, followed by an idle time-slot at output port B, and the system is no longer work-conserving. Hence, the system cannot mimic an OQ switch.

Definition 1: ***Concentration:*** *Concentration is a term we will use to describe the situation when a disproportionately large number of cells destined to the same output are concentrated on a small number of the internal layers.*

Concentration is undesirable as it leads to unnecessary idling because of the limited line-rate between each layer and the multiplexor. One way to alleviate the effect of concentration is to use faster internal links. In general, we will use internal links that operate at a rate $S(R/k)$, where S is the *speedup* of the internal link.

For example, in our counter-example above, the problem could be eliminated by running the internal links at a rate of R instead of $R/2$ (i.e. a speedup of two). This solves the problem because the external output port can now read the cells back-to-back from layer two. But this appears to defeat the purpose of operating the internal layers slower than the external line rate. Fortunately, we will see in the next section that the speedup required to eliminate the problem of concentration is independent of the arriving traffic, R, N and is almost independent of k. In particular, we find that with a speedup of $2k/(k+2) \approx 2$, for large k, the PPS is work-conserving and can precisely mimic a FCFS-OQ switch.

B. Link Constraints

The operation of a PPS is limited by two constraints. We call these the Input Link Constraint and the Output Link Constraint as defined below.

Definition 2: ***Input Link Constraint-*** *An external input port is constrained to send a cell to a specific layer at most once every $\lceil k/S \rceil$ time slots. This is because the internal input links operate S/k times slower than the external input links. We call this constraint the input link constraint, or ILC.*

Definition 3: ***Allowable Input Link Set-*** *The ILC gives rise to the allowable input link set, AIL(i,n), which is the set of layers to which external input port i can start sending a cell in time slot n. This is the set of layers that external input i has not started sending any cells within the last $\lceil k/S \rceil - 1$ time slots. Note that $|AIL(i,n)| \le k, \forall (i,n)$.*

$AIL(i,n)$ evolves over time, with at most one new layer being added to, and at most one layer being deleted from the set in each time slot. If external input i starts sending a cell to layer l at time slot n, then layer l is removed from $AIL(i,n)$. The layer is added back to the set when it becomes free at time $n + \lceil k/S \rceil$.

Definition 4: ***Output Link Constraint-*** *In a similar manner to the ILC, a layer is constrained to send a cell to an external output port at most once every $\lceil k/S \rceil$ time slots. This is because the internal output links operate S/k times slower than the external output links. Hence, in every time slot an*

external output port may not be able to receive cells from certain layers. This constraint is called the output link constraint, or OLC.

Definition 5: ***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 $DT(n, i, j)$. The departure time could, for example, be the first time that output j is free and able to send the cell. As we shall see later, other definitions are possible.*

Definition 6: ***Available Output Link Set****- The OLC gives rise to the available output link set $AOL(j, DT(n, i, j))$, which is the set of layers that can send a cell to external output j at time slot $DT(n, i, j)$ in the future. $AOL(j, DT(n, i, j))$ is the set of layers that have not started sending any cells to external output j in the last $\lceil k/S \rceil - 1$ time slots before time slot $DT(n, i, j)$. Note that, since there are a total of k layers, $|AOL(j, DT(n, i, j))| \le k, \forall (j, DT(n, i, j))$.*

Like $AIL(i, n)$, $AOL(j, DT(n, i, j))$ can increase or decrease by at most one layer per time slot; i.e. if a layer l starts to send a cell to output j at time slot $DT(n, i, j)$, the layer is deleted from $AOL(j, DT(n, i, j))$ and then will be added to the set again when the layer becomes free at time $DT(n, i, j) + \lceil k/S \rceil$. However, whenever a layer is deleted from the set, the index $DT(n, i, j)$ is incremented. Because in a single time slot up to N cells may arrive at the PPS for the same external output, $AOL(j, DT(n, i, j))$ may change up to N times per time slot. This is because $AOL(j, DT(n, i, j))$ represents the layers available for use at some time $DT(n, i, j)$ in the future. As each arriving cell is sent to a layer, a link to its external output is reserved for some time in the future. So effectively, $AOL(j, DT(n, i, j))$ indicates the schedule of future departures for output j, and at any instant, $Max(DT(n, i, j)) + 1, \forall (n, i)$ indicates the first time in the future that output j will be free.

C. Lower Bounds on the size of the link constraint sets

The following two lemmas will be used shortly to demonstrate the conditions under which a PPS can mimic a FCFS-OQ switch.

Lemma 1: The size of the available input link set, $|AIL(i, n)| \ge k - \lceil k/S \rceil + 1$, for all $i, n \ge 0$; where S is the speedup on the internal input links.

Consider external input port i. The only layers that i cannot send a cell to are those which were used in the last $\lceil k/S \rceil - 1$ time slots. (The layer which was used $\lceil k/S \rceil$ time slots ago is now free to be used again). $|AIL(i, n)|$ is minimized when a cell arrives to the external input port in each of the previous $\lceil k/S \rceil - 1$ time slots, hence $|AIL(i, n)| \ge k - (\lceil k/S \rceil - 1) = k - \lceil k/S \rceil + 1$. ❐

Lemma 2: The size of the available output link set, $|AOL(j, DT(n, i, j))| \ge k - \lceil k/S \rceil + 1$, for all $i, j, n \ge 0$.

Proof: The proof is similar to Lemma 1. We consider an external output port which reads cells from the internal switches instead of an external input port which writes cells to the internal switches. ❐

IV. MIMICKING A FCFS-OQ SWITCH

Theorem 1: (Sufficiency) If a PPS guarantees that each arriving cell is allocated to a layer l, such that $l \in AIL(i, n)$ and $l \in AOL(j, DT(n, i, j))$, (i.e. if it meets both the ILC and the OLC) then the switch is work-conserving.

Proof: Consider a cell C that arrives to external input port i at time slot n and destined for output port j. The demultiplexor chooses a layer l that meets both the ILC and the OLC; i.e. $l \in \{AIL(i, n) \cap AOL(j, DT(n, i, j))\}$, where $DT(n, i, j)$ is the index of $AOL(.)$ and represents the first time that external output j is free in the future at time slot n. Since the ILC is met, C is sent to layer l immediately without buffering. C is placed in output queue j of layer l where it awaits its turn to depart. In fact, the departure time of the cell $DT(n, i, j)$ has already been picked when it arrived at time n. C is removed from its queue at $DT(n, i, j)$ and sent to external output port j. The reason that C departs at $DT(n, i, j)$ is because, by definition of $AOL(j, DT(n, i, j))$, external port j was busy receiving cells from other layers up until the time C departs, and hence the output was never idle when there was a cell in the system destined to it. ❐

Theorem 2: (Sufficiency) A speedup of $2k/(k+2)$ is sufficient for a PPS to meet both the input and output link constraints for every cell.

For the ILC and OLC to be met, it suffices to show that there will always exist a layer l such that $l \in \{AIL(i, n) \cap AOL(j, DT(n, i, j))\}$, i.e. that $AIL(i, n) \cap (AOL(j, DT(n, i, j))) \ne \varnothing$, which must be satisfied if $|AIL(i, n)| + |AOL(j, DT(n, i, j))| > k$. From Lemma 1 and Lemma 2 we know that $|AIL(i, n)| + |AOL(j, DT(n, i, j))| > k$ if $S > 2k/(k+2)$. ❐

Corollary 1: A PPS can be work conserving, if $S > 2k/(k+2)$.

Having shown that a PPS can be work-conserving, we now show that with the same speedup, the switch can mimic a FCFS-OQ switch.

Theorem 3: (Sufficiency) A PPS can exactly mimic a FCFS-OQ switch with a speedup of $S > 2k/(k+2)$.

Proof: Consider a PPS with a speedup of $S > 2k/(k+2)$ which, for each arriving cell, selects a layer that meets both the ILC and the OLC. A cell destined to output j and arriv-

ing at time slot n is scheduled to depart at time slot $DT(n, i, j)$, which is the index of $AOL(j, DT(n, i, j))$. By definition of $AOL(j, DT(n, i, j))$, $DT(n, i, j)$ is the first time in the future that output j is idle, and so it is also the time that the cell would depart in a FCFS-OQ switch. Therefore, each cell departs at the same time that it would in a FCFS-OQ switch, and hence the PPS mimics its behavior. ❐

A detailed description of the algorithm suggested by this proof appears in Appendix A. It is interesting to note that this theorem is in some ways analogous to the requirements for a 3-stage Clos network to be strictly non-blocking. In fact, the proof is quite similar. Yet the two systems are clearly different — a PPS buffers cells as they traverse the switch, while a Clos network is a bufferless fabric used mostly in circuit-switches.

V. Providing QoS guarantees

We now extend our results to find the speedup requirement for a PPS to provide the same QoS guarantees as an OQ switch. To do this, we find the speedup required for a PPS to implement any PIFO scheduling discipline.

Theorem 4: (Sufficiency) A PPS can exactly mimic any OQ switch with a PIFO queueing discipline with a speedup of $S > 3k/(k+3)$.

Proof: As defined in Section II a PIFO queueing policy can insert a cell anywhere in its queue but it can not change the relative ordering of cells once they are in the queue. Consider a cell C that arrives to external input port i at time slot n and destined for output port j. The demultiplexor will determine the time that each arriving cell must depart, $DT(n, i, j)$ to meet its delay guarantee. The decision made by the demultiplexor at input i amounts to selecting a layer so that the cell may depart on time. Notice that this is very similar to the previous section in which cells departed in FCFS order requiring only that a cell depart the first time that its output is free after the cell arrives. The difference here is that $DT(n, i, j)$ may be selected to be ahead of cells already scheduled to depart from output j. The demultiplexor's choice of layer l to send an arriving cell C must meet three constraints:

1. The link connecting layer l to output j must be free at $DT(n, i, j)$. i.e. $l \in \{AOL(j, DT(n, i, j))\}$.
2. All the other cells destined to output j after C must also find a link available. In other words, if the demultiplexor picks layer l for cell C, it needs to ensure that no other cell requires the link from l to output j within the next $(\lceil k/S \rceil - 1)$ time slots. Since the cells following cell C have already been sent to specific layers, it is necessary that the layer l being chosen be distinct from the layers which the next $\lceil k/S \rceil - 1$ cells use. Formally we can write this constraint as
   $l \in \{AOL(j, DT(n, i, j) + \lceil k/S \rceil - 1)\}$
3. The link connecting the demultiplexor at input i to layer l must be free at time slot n. Hence $l \in \{AIL(i, n)\}$.

Thus layer l must meet all the above constraints i.e.

$$l \in \{AIL(i, n) \cap AOL(j, DT(n, i, j)) \cap AOL(j, DT(n, i, j) + \lceil k/S \rceil - 1)\}$$

For a layer l to exist,

$$AIL(i, n) \cap AOL(j, DT(n, i, j)) \cap AOL(j, DT(n, i, j) + \lceil k/S \rceil - 1) \neq \varnothing$$

This will be satisfied when,

$$|AIL(i, n)| + |AOL(j, DT(n, i, j))| + |AOL(j, DT(n, i, j) + \lceil k/S \rceil - 1)| > 2k$$

From Lemma [1] and [2] we know that,

$$|AIL(i, n)| + |AOL(j, DT(n, i, j))| + |AOL(j, DT(n, i, j) + \lceil k/S \rceil - 1)| > 2k$$

if,

$$S > 3k/(k+3)$$. ❐

Figure 3 shows an example of a PPS with $k = 10$ layers and $S = 3$. A new cell C arrives destined to output 1 and has to be inserted in the priority queue for output 1 which is maintained in a PIFO manner. Figure 3a shows that the cell is constrained by the AIL to be sent to layers $\{2, 4, 5, 7, 8, 10\}$. These layers are shown darkened in Figure 3a. It is decided that cell C must be inserted between $C6$ and $C7$. Figure 3b shows the intersection of the two AOL sets for this insertion. Cell C is constrained by the AOL to use layers $\{1, 6, 7, 10\}$. Figure 3c shows the candidate layers for insertion i.e. layers 7 and 10. Cell C is then inserted in layer 10.

VI. Conclusions

While it is difficult to predict the growth of the Internet over the coming years, it seems certain that packet-switches will be required with: (1) increased switching capacity, (2) support for higher line-rates, and (3) support for differentiated qualities of service. All three of these requirements present challenges of their own. For example, higher capacity switches may require new architectures; higher line-rates may grow to exceed the capabilities of commercially available memories, making it impractical to buffer packets as they arrive; and the need for differentiated qualities of service may require performance comparable to output-queued switches. The difficulty of overcoming these challenges could be seen as an argument for circuit switching where switching is simple, buffers are not needed, and qualities of service are achieved through peak-provision of bandwidth.

While the use of circuit-switching may be appealing, we consider here an alternative — a parallel packet switch (PPS) which achieves high capacity by placing multiple packet switches in parallel, rather than in series with each other as is common in multistage switch designs. Hence, each packet that passes through the system encounters only a single stage of buffering; furthermore, and most interestingly, the system

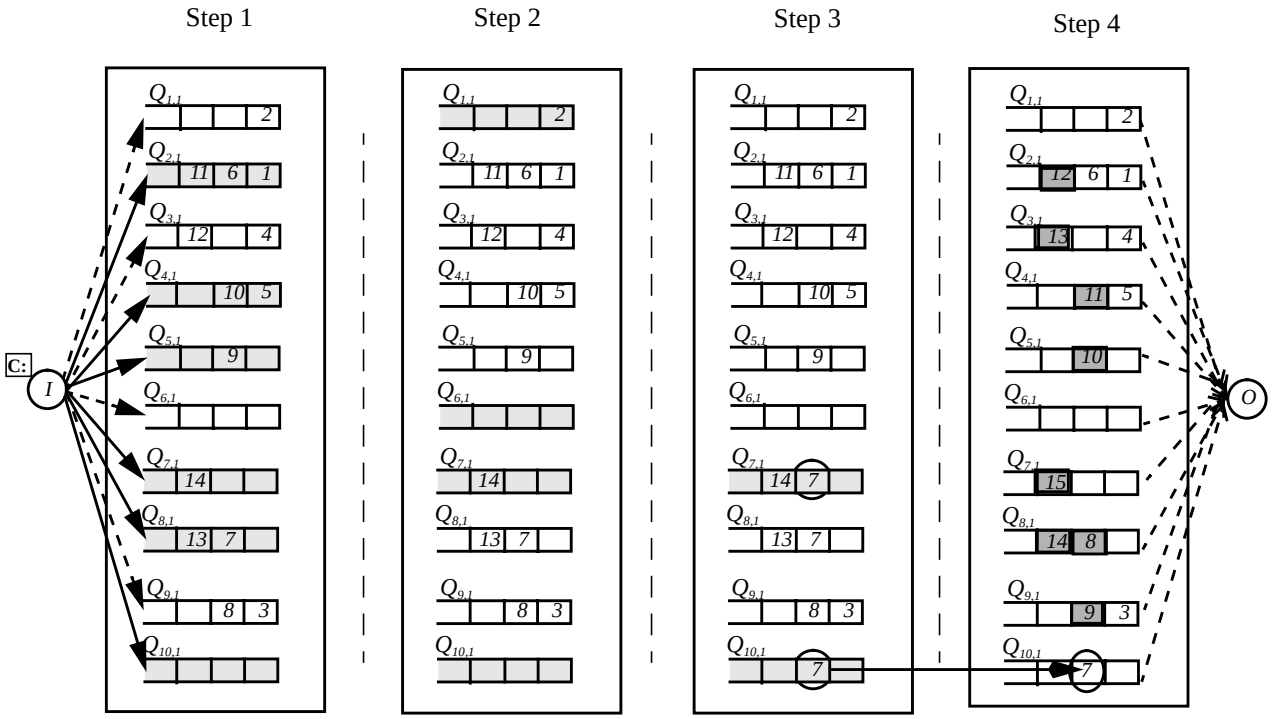


Figure 3: Insertion of cells in a PIFO order in a PPS with ten layers. $Q_{k,1}$ refers to output queue number one in the internal switch k. (a) The AIL constrains the use of layers $\{2, 4, 5, 7, 8, 10\}$. (b) The cell is to be inserted before cell number 7. The two AOLs constrain the use of layers $\{1, 6, 7, 10\}$ (c) The intersection constrains the use of layers {7,10}. (d) Layer 10 is chosen. The cell number 7 is inserted.

line-rates may operate in excess of the speed of the buffer memory. The main result of this paper is that it theoretically possible to build such a PPS that exactly mimic an output-queued packet switch regardless of the nature of the arriving traffic. The mimicking can be maintained even when the system is providing guaranteed qualities of service. In short, and at first glance, it appears that all three of the challenges outlined above are overcome by the PPS system.

Unfortunately, as described above, the result is only a theoretical one. There are two hindrances to making the PPS practical. First, each layer of the PPS is an output-queued (OQ) switch itself, which implies that the speed of its buffer memory must grow linearly with the number of ports of the PPS. We *can* overcome this by replacing each output-queued switch with a Combined Input-Output Queued (CIOQ) switch that mimics its behavior using an internal speedup of two. Each memory in the system could now run at a rate independent of the number of ports of the PPS, and can be made arbitrarily small by increasing the number of packet-switches used. This solution, however, requires that the CIOQ switch be made practical — something for which work is in progress, but no solution is yet available.

The second hindrance is that the theoretical result assumes a centralized algorithm to determine which layer each arriving cell is sent to. We believe that the algorithm may take too long to run, as its running time grows linearly with the number of ports of the PPS.

So, in summary, we think of this work as a first step towards building high-capacity switches that support guaranteed qualities of service in which memory bandwidth is not the bottleneck. In our future work, we will strive to make these results more practical.

Acknowledgments

We would like to thank Pankaj Gupta of Stanford University for his many useful comments along the way.

References

[1] 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.

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

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

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

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

[6] H. Kim, A. Leon-Garcia, "A self-routing multistage switching network for broadband ISDN," *IEEE J. Sel. Areas in Communications*, pp. 459-466, April 1990.

[7] 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.

[8] F. Chiussi, D. Khotimsky, S. Krishnan, "Generalized inverse multi-

plexing of switched ATM connections," in *Proc. IEEE Globecom '98 Conference. The Bridge to Global Integration*, Sydney, Australia, Nov. 1998.

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

[10] A. Demers, S. Keshav; S. Shenker, "Analysis and simulation of a fair queueing algorithm," *J. of Internetworking: Research and Experience*, pp.3-26, 1990.

[11] A. Parekh and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single node case," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 344-357, June 1993.

[12] L. Zhang, "Virtual Clock: A new traffic control algorithm for packet switching networks," *ACM Transactions on Computer Systems*, vol.9 no.2, pp.101-124, 1990.

[13] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," in *Proc. ACM Sigcomm*, Sep 1995, pp. 231-242.

APPENDIX A: A Centralized Algorithm for a PPS

We now present an insert and dispatch scheme, called CPA (Centralized Parallel Packet Switch Algorithm) in order to emulate a FCFS-OQ switch based on the results obtained in Section IV.

A. Notation

1. $HOL(j, l)$, denotes the head of line of cell at output port j of internal switch l.
2. $T(c, j)$, denotes the sequence number tagged to a cell c which is destined to output port j The sequence $T(c, j)$ denotes the FIFO order of all cells destined to output port j. These tags are unique for each cell destined to output port j.

B. Steps in CPA

CPA consists of two parts which operate independently at each of the external input and the output ports. There is a centralized scheduler which maintains the allowable output link set for each output.

1. **De-multiplexor:** Each external input port maintains its allowable input link set. When a cell c arrives to external input port i destined to output port j at time slot n, the input port requests the centralized scheduler for a layer. A copy of $AIL(i, n)$, and as well as the destination port number j is sent. The centralized scheduler then computes a layer l, such that $l \in \{AIL(i, n) \cap AOL(j, DT(n, i, j))\}$, as described in Theorem 1. The centralized scheduler also returns a tagged sequence number $T(c, j)$ associated with the arriving cell c. The external input i tags the cell with the output port number j and sequence number $T(c, j)$ and transmits the cell to layer l. The values of $AIL(i, n)$, $AOL(j, DT(n, i, j))$, n and $DT(n, i, j)$ are updated.
2. **Multiplexor:** We assume that the cells at the output queues of the internal switches are maintained in sorted order in the ascending order of their sequence numbers. Each external output port j computes a layer l such that $T(HOL(j, l), j) = Min\,(\forall l)\, T(HOL(j, l), j)$. The output port chooses this layer to dispatch the next cell.

In the example shown in Figure 4a at the input cells $C1$, $C2$, $C3$ arrive at a 4×4 PPS with $k = 3$ layers and $S = 2$ at time $t = 0$. These cells are sent to layers $1, 2, 2$ respectively. The AIL and the AOL is updated at the input after each cell is sent. Cells $C4, C5$ arrive at time slot $t = 1$ and are sent to layers $3, 1$ respectively.

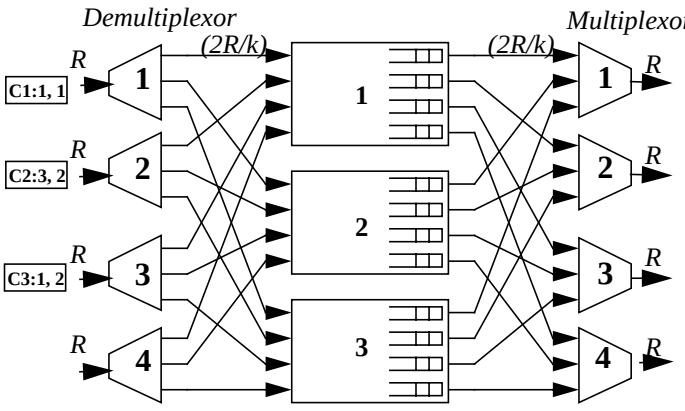
At the output as shown in Figure 4b, cells $C1, C3, C4, C5$ are tagged sequence numbers $1, 2, 3, 4$. This is the FIFO order in which they are sent to output 1 respectively. Cell $C2$ is tagged sequence number 1 as it is the first cell destined to output 3. The output ports choose the order of departure by calculating the minimum amongst all cells at the head of line.

C. Practical Considerations

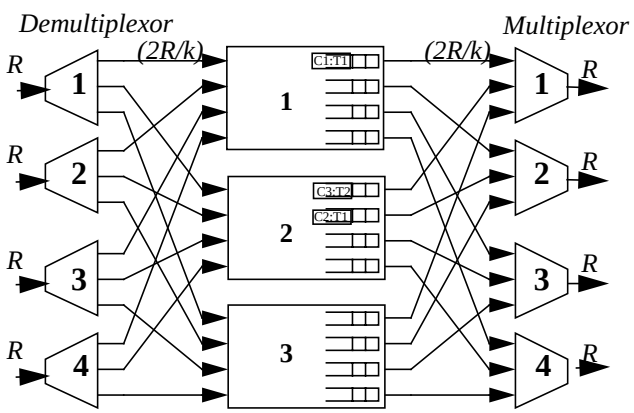
1. **De-multiplexor**: In CPA the maintenance of AIL is relatively straightforward since by definition the AIL is maintained locally by each input and it changes at most once every external time slot. However the maintenance of AOL requires significant computation as it might get updated N times in a single time slot. This also necessitates a large amount of communication between each input and the central scheduler which maintains the AOL for all outputs. Clearly, this is impractical for large N.
2. **Multiplexor**: Each external output chooses the correct order of cells by computing the minimum tagged sequence number among all cells at the head of line of its corresponding output queue. A maximum of k such computations will have to be performed in each time slot. We can reduce this time by performing a one time sort operation on the head of line cells and then maintaining these values in sorted order for each additional head of line cell which appears at the output.

There is also the problem of sequence numbers being exhausted and other issues related to the re-use of sequence numbers [8]. We do not address these issues in this paper.

(a) Cells arriving at time slot 0 and being sent to the middle stage switches

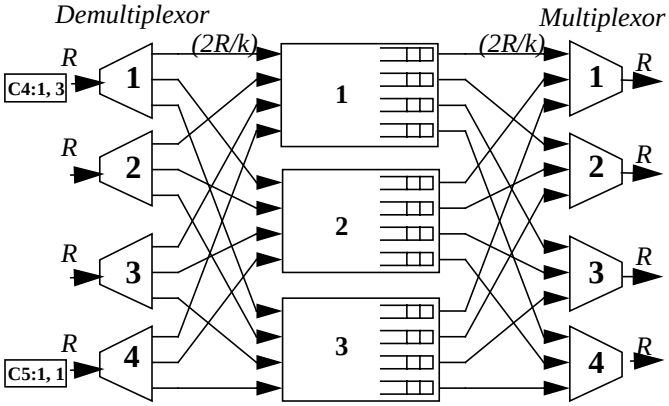


Cell C1 chooses layer 1 arbitrarily from {1,2,3} ^ {1,2,3}
AOL(1,1) is updated to {1,2,3} - {1} = {2,3}
AIL(1,1) is updated to {1,2,3} - {1} = {2,3}
Cell C2 chooses layer 2 arbitrarily from {1,2,3} ^ {1,2,3}
AOL(3,1) is updated to {1,2,3} - {2} = {1,3}
AIL(2,1) is updated to {1,2,3} - {2} = {1,3}

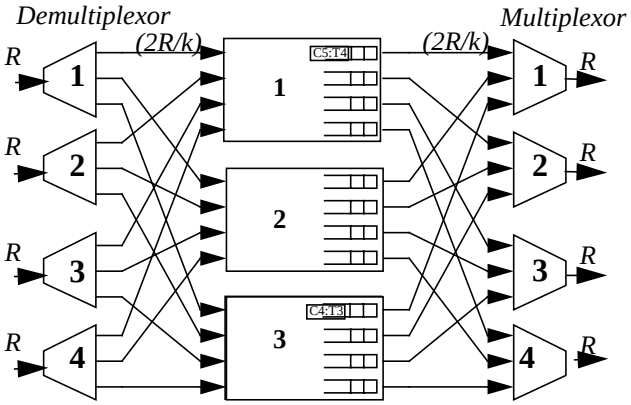


Cell C3 has a departure time d(0,3,1)=1
Cell C3 has to choose from AIL(3,0) ^ AOL(1,1)
Cell C3 chooses layer 2 from {1,2,3} ^ {2,3}
AOL(1,2) is updated to {2,3} - {2} + {1} = {1,3}
AIL(3,1) is updated to {1,2,3} - {2} = {1,3}

(b) Cells arriving at time slot 1 and being sent to the middle stage switches



Cell C4 has an expected departure time d(1,1,1) = 2
Cell C4 has to choose from AIL(1,1) ^ AOL(1,2)
Cell C4 chooses layer 3 from {2,3} ^ {1,3}
AOL(1,3) is updated to{1,3} - {3} + {2} = {1,2}
AIL(1,2) is updated to {2,3} - {3} + {1} = {1,2}



Cell C5 has an expected departure time d(4,1,1)=3
Cell C5 has to choose from AIL(4,1) ^ AOL(1,3)
Cell C5 chooses layer 1 from {1,2,3} ^ {1,2}
AOL(1,4) is updated to {1,2} - {1} + {3} = {2,3}
AIL(4,2) is updated to {1,2,3} - {1} = {2,3}

Figure 4: The CPA algorithm for a 4×4 PPS. The notation $Cq:j,l$ denotes a cell numbered q, destined to output port j, and sent to layer 1. The notation $Cq:Tp$ denotes the same cell numbered q, being tagged with a sequence number p.