Resequencing Cells in an ATM Switch

Jonathan S. Turner

WUCS-91-21

July 18, 1996

Department of Computer Science
Campus Box 1045
Washington University
One Brookings Drive
St. Louis, MO 63130-4899

Abstract

ATM switching systems are required to maintain ordering for cells within a given virtual circuit. This is most commonly achieved using a switching system in which cells in a given virtual circuit are routed along a common path through the switching system. This solution has the drawback that it can lead to significant virtual circuit blocking unless the switching network is engineered with a large speed advantage, relative to the external links. Switching systems in which cells are routed independently, on the other hand, can balance the load dynamically and operate with a minimal speed advantage. This report describes a resequencer that can be used with such a switching system to put cells back in the proper order, gives an economical implementation of such a resequencer and reports on a simulation study of its effectiveness.

Resequencing Cells in an ATM Switch

Jonathan S. Turner

Figure 1 illustrates an ATM switching system, based on a three stage Beneš network topology with fixed path routing. Switching systems using this architecture are being developed by several manufacturers [1, 4]. The left hand diagram shows several virtual circuits passing through the network, each consuming a specified fraction of the capacity of the network's internal data paths (for example, the virtual circuit that enters at the top left consumes 60% of the capacity of the data paths it is routed through). In these networks, virtual circuits may share a common data path so long as the sum of their bandwidth requirements is less than the system's internal data path capacity (normalized to 1 here). Notice that if one attempts to establish a connection with a traffic requirement of 0.7 between the input marked a and the output b, it will be blocked, as there is no path joining these two points with enough available bandwidth to support the connection. To ensure that blocking never occurs, the Beneš network can be engineered so that it's internal data paths are three times faster than the external links. For a $2k-1$ stage Beneš network, the speed advantage required to achieve nonblocking operation for point-to-point traffic is $(2k-1) - 2(k-2)/n^{1/k}$ where n is the number of inputs and outputs to the network (see [2, 3] for details).

In the case of broadcast or multicast virtual circuits, an even larger speed advantage is required to achieve nonblocking operation. The right hand diagram in Figure 1 shows a multicast virtual circuit with a bandwidth requirement of 0.6. Note that any connection request from c that has a bandwidth requirement exceeding 0.4 will be blocked. Virtual circuits such as this one, in which branching occurs in the first stage, can lead to large amounts of blocking, but cannot be avoided in general. Nonblocking operation in a three stage Beneš network requires a speed advantage of $\sqrt{n}$ in this case.

ATM switching systems such as the one described in [5] allow cells passing through the system to follow different paths. This allows even load distribution and eliminates the need for any speed advantage beyond that required to absorb short-term contention. On the other hand, this approach does allow cells to exit the system in a different order from that in which they entered. This can cause problems for applications, which often require that cells be received in the same order as they were sent. In this paper we describe a mechanism for reordering cells after they exit an ATM switch that effectively eliminates this problem, detail an implementation of this mechanism and report on simulations performed to evaluate its effectiveness.

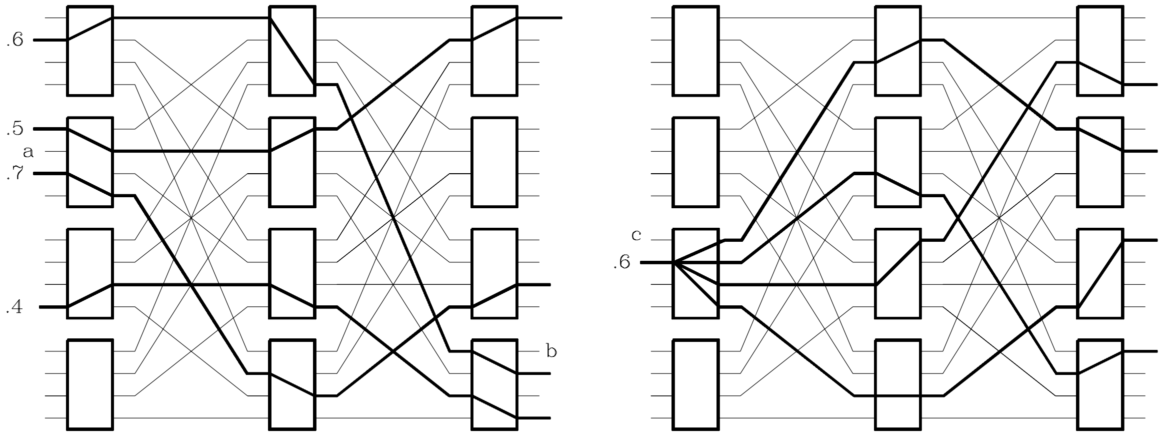

Figure 1: Fixed Path Routing in ATM Switches

1. Overview of Resequencing Mechanism

Figure 2 shows the overall organization of an ATM switching system that includes a resequencing mechanism. As cells are read from the input buffers on the left side of the switch, a time stamp is added by a *Time Stamp Circuit* (TSC), which records the time at which the cell enters the switch (the TSCs are all driven from a common clock). When cells leave the switch they enter a resequencing buffer which is managed by a *Resequencing Buffer Controller* (RBC). As the cell leaves the switch, the RBC computes its age from the time of entry and current time, then stores the age and the cell's location in the output buffer in an internal table.

When a cell is to be read from the buffer, the buffer controller selects the oldest cell in the buffer and provides its address to the output buffer which then sends it out. If the oldest cell is "not old enough," no cell is output. The purpose of this is to allow cells stored within the switching network to catch up with cells that have already reached the output buffer. This mechanism can ensure that all but a vanishingly small fraction of misordered cells are put back in order. Simulation results described below, indicate that a 64 port resequencing buffer provides ample protection against out-of-sequence cells for a network such as that described in [5].

The key element in this mechanism is the buffer controller. In our judgement, a reasonable buffer controller should have a complexity of no more than 25% of the buffer itself. The design we present below can, with careful implementation, meet that target. The buffer controller has two main functions. First, during a buffer output cycle, it must select the cell with the oldest time stamp, check that the cell is old enough and if so, supply the cell's address to the buffer. Second, during a buffer input cycle, it must select an empty slot that the arriving cell can be placed in and supply its address to the buffer controller.

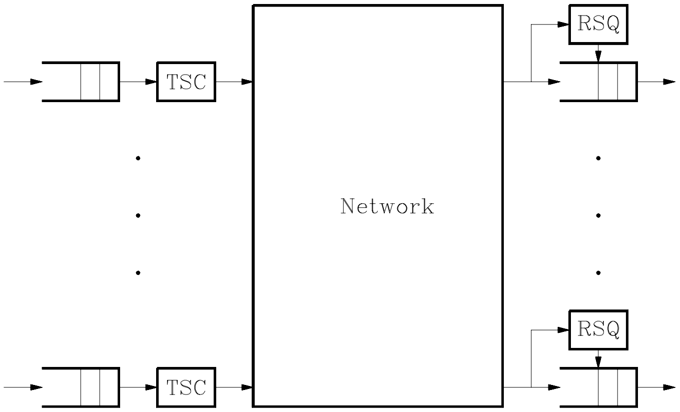

Figure 2: ATM Switch with Resequencing Mechanism

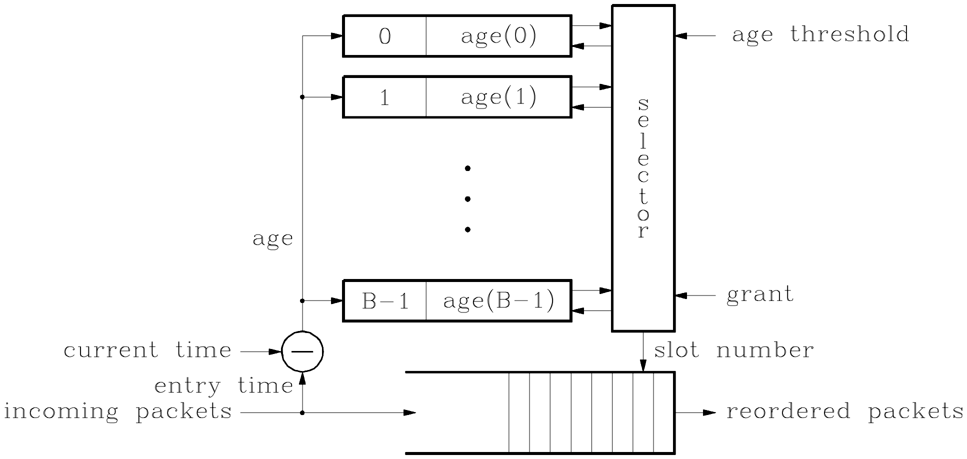

Figure 3: Resequencing Buffer Organization

2. Controller Organization and Operation

Figure 3 shows the organization of the resequencing buffer and buffer controller. The buffer is organized as a set of *slots* with each slot being large enough to hold a single ATM cell. The controller is organized similarly, with a *control slot* for each buffer slot. Each control slot contains two pieces of information, a *slot number* which specifies the buffer slot it is associated with and the age of the cell (if any) stored in that slot. The *selector*, at the right, selects the oldest cell during output operations, compares that cell's age to a given age threshold, and if appropriate, forwards the cell's slot number to the buffer which then forwards the cell to the downstream circuitry. The *grant* signal is asserted by the downstream circuit if it is prepared to receive a cell; this provides a simple form of flow control. During input operations, the selector selects any idle control slot inserts the age of the arriving cell into that slot, and passes the slot number to the buffer, which places the arriving cell in the specified slot.

The selector uses a distributed contention resolution mechanism illustrated in Figure 4.

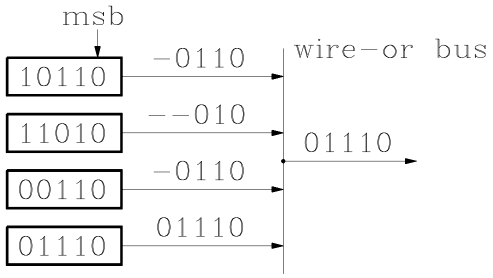

Figure 4: Distributed Contention Resolution

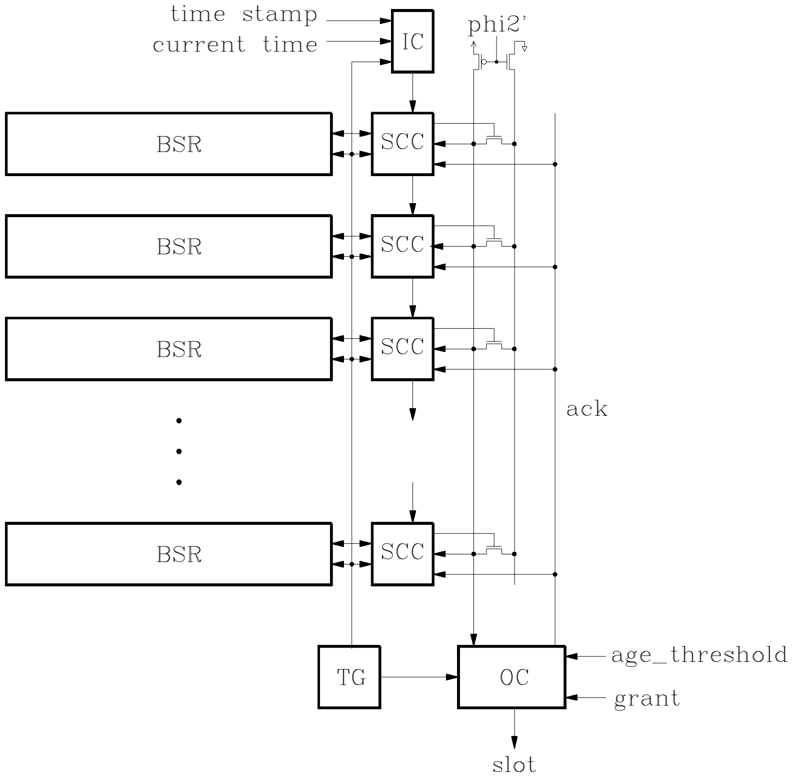

Figure 5: Buffer Control Circuit

The registers at left each contain a unique binary value which is placed sequentially on a wire-or bus, with the most significant bit (MSB) first. As the bits are placed on the bus, each register simultaneously examines the state of the bus and if the bus value differs from the value it put on the bus, it stops contending. Since the stored values are unique, only one register will remain in contention at the end of the cycle, and its value is the one placed on the bus.

Figure 5 gives a more detailed view of the buffer controller, as it might be implemented in a CMOS integrated circuit with a two phase, non-overlapping clock. Each control slot consists of two parts, a *Bidirectional Shift Register* (BSR), and a *Slot Control Circuit* (SCC). The BSR stores the slot number and age information and the SCC keeps track of whether the slot contains a cell or not and participates in the distributed contention resolution. The selector consists of a pre-charged bus that implements the wire-or contention mechanism (more precisely, it's a wire-nor) and an Output Circuit (OC), which on output compares the age of the oldest cell to the age threshold, checks the grant line, then provides an acknowledgment signal if the given cell can be output. Other components include an Input Circuit (IC) which computes the age of an arriving cell and the Timing Generator (TG) which generates the various control signals needed to coordinate the activities of the other components.

Figure 6 illustrates an output operation. The left side of the figure shows the initial state of the BSRs, with the slot number on the left and age on the right, with the most significant bit of each at the right hand end. The busy/idle status of each control slot is held in a flip flop indicated in the figure. During the write operation, the BSRs of the busy slots are rotated to the right, with successive bits placed on the bus. The contention is performed over both the age and the slot number, ensuring that there is a unique winner in the contention. Assuming the winner is acknowledged by the OC, its busy/idle flip flop is cleared, giving the final status shown at the right.

Figure 7 illustrates an input operation. In this case, contention is done using only the slot numbers of the idle slots. However, during the write operation, the age fields of the busy slots are also incremented. Incrementing is most efficiently implemented using a bit-serial incrementer, which requires that the bits be presented least-significant bit first. Consequently, the BSR is rotated to the left rather than to the right. This doesn't affect the selection of an idle slot significantly, since any idle slot will do as well as another. Notice in the figure, that the incoming cell is inserted into the idle slot that wins the contention and that the age fields of the busy slots are incremented appropriately.

3. Control Slot Implementation

Figure 8 shows details of the Slot Control Circuit. The lower flip flop (called the data present flip flop) is set if there is a cell stored in the buffer position associated with this slot control circuit. The upper flip flop (called the contention flip flop) is set when the the slot control circuit is contending for the bus. The *Incrementer* (INC) contains a serial incrementer plus some signal steering circuitry to pass signals between different pairs of its serial data ports `A, B, C` and `D`. For example, when the control signal `A->B` is asserted, data on the `A` port is passed to the `B` port with a one clock tick delay. When the control signal `B+1,D->A` is asserted and `selB+1` is asserted, the data on port `B` is incremented and passed to `A` with a one tick delay (the data is assumed to come in least significant bit first); if `selB+1` is not asserted, the data on the `D` port is passed to the `A` port with a one tick delay.

At the start of an operation, the `init` signal sets the contention flip flop to enable the contention resolution mechanism. During an output operation, the `right` and `A->B` signals are asserted causing the BSR's contents to be rotated to the right and the `read` signal is asserted allowing the slots whose data present flop flops are set to contend. Note that the contention flip flop is cleared when the bus state differs from the bit placed on the bus. At

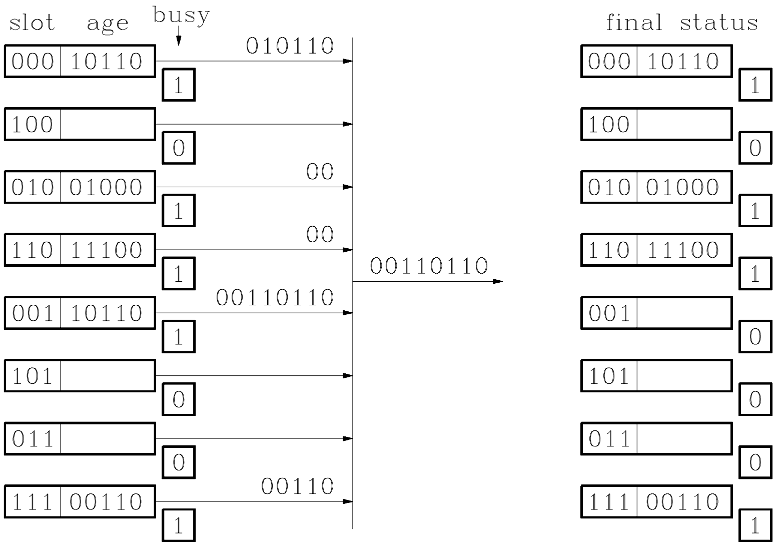

Figure 6: Slot Operation — Output

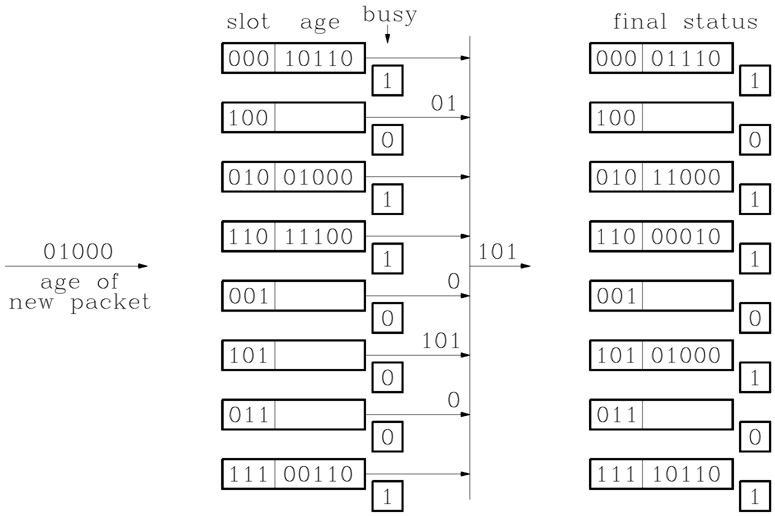

Figure 7: Slot Operation — Input

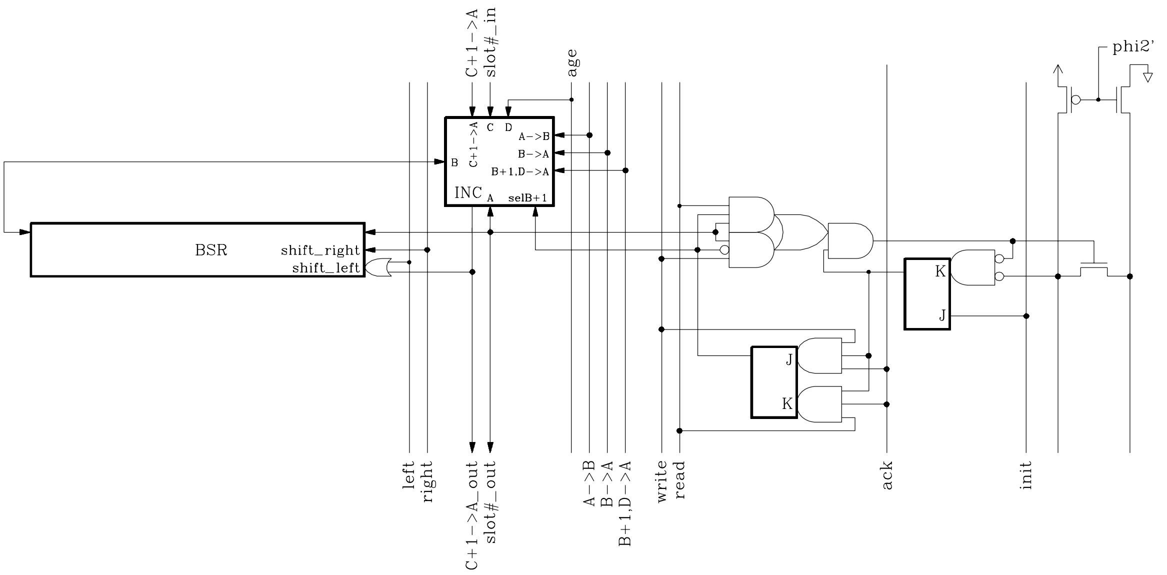

Figure 8: Slot Details

the end of the operation, only the winner's contention flip flop remains set and so the `ack` signal received at the end of the cycle will clear only the winner's data present flip flop.

During an input operation, the `left` and `B->A` signals are first asserted causing the BSR to rotate to the left and the slot number to be placed on the contention bus, if the slot is idle. At the time the age field passes through the incrementer, the `B->A` signal is replaced with the `B+1,D->A` signal, incrementing the age fields of the busy slots and inserting the age field of the arriving cell (which is present on the `age` line) into the idle slots. Figure 9 gives the timing signals for an output and an input operation; in the diagram, a three bit slot number and five bit age field are assumed.

Some additional signals are required for initialization of the buffer control circuit. What's required here is to load each BSR with its slot number. This is done using the `slot#_in` and `C+1->A` signals. The incoming slot number is incremented and passed to the BSR and the next slot control circuit. The `C+1->A` signal is used to time the incrementing and is then delayed by one clock tick to control the shifting of the slot number into the BSR; it is then passed to the next slot control circuit.

Two other details should be mentioned. If a busy slot's age reaches the maximum value that can be stored, incrementing should be disabled. This can be implemented with an additional flip flop that is set when the age field reaches the maximum and disables incrementing in this case. Alternatively, busy slot's for which the age field exceeds the maximum representable value can be discarded by clearing the data present flip flop. We also need a mechanism to detect when there are no idle slots. In the circuit described, the absence of an idle slot is indistinguishable from the condition where slot 0 is the only idle slot. This ambiguity is most easily handled by simply disabling slot 0, by forcing its data present flip flop to always be cleared. While this solution sacrifices one control slot (and the corresponding buffer slot) it simplifies the slot control circuit. Alternatively, one could

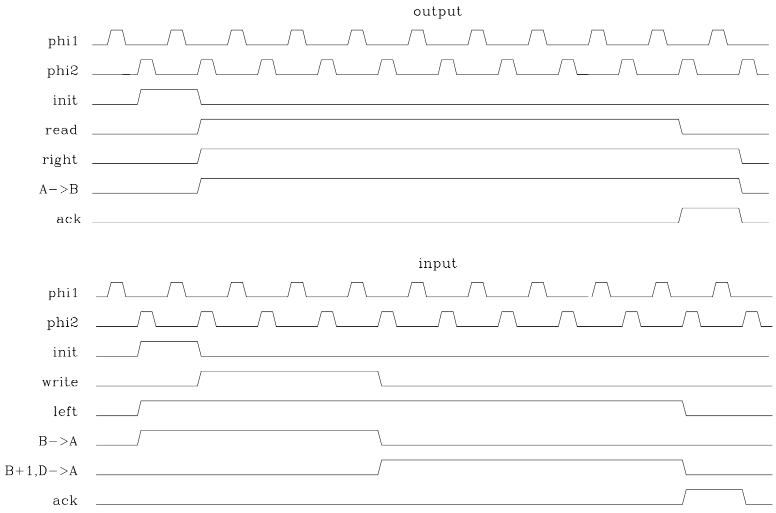

Figure 9: Timing of Control Slot Operations

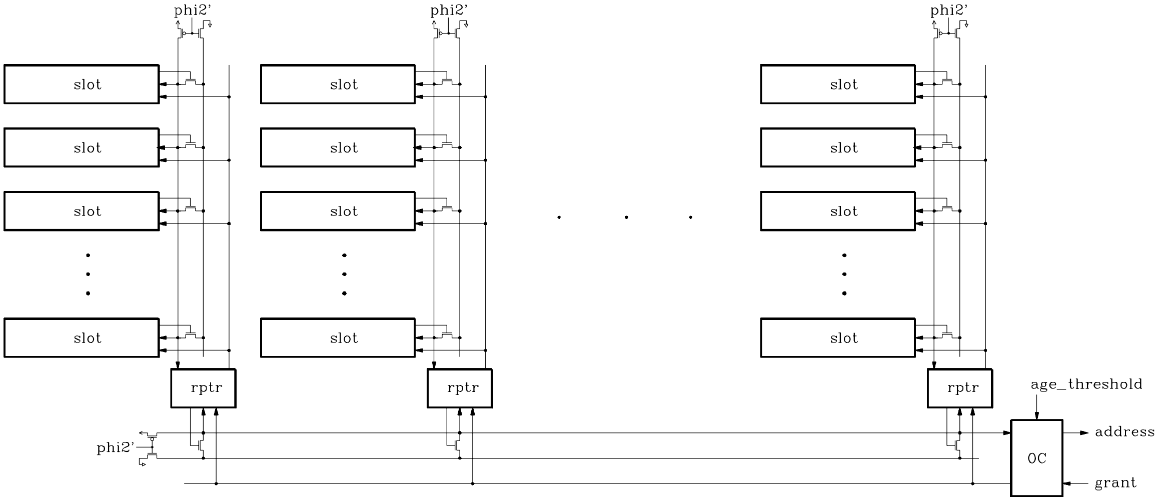

Figure 10: Two Level Contention Circuit

alter the input operation, by appending the complement of the busy/idle flip flop's state to the slot number during the bus contention.

In a large resequencer, a single level contention bus as described above would operate only at low speed due to the capacitive loading placed on the bus by all the slot control circuits. We can speed up the design by having several contention buses, each shared by a smaller number of slot control circuits. The winning contenders from each of these buses would then be passed on to a global contention bus through a set of *repeaters*, as illustrated in Figure 10. With this division, each bus can operate at a higher clock speed. The only drawback is a one bit time delay in reaching the final decision, which in most situations is not a serious problem. Of course, the same idea can be extended to additional levels if needed.

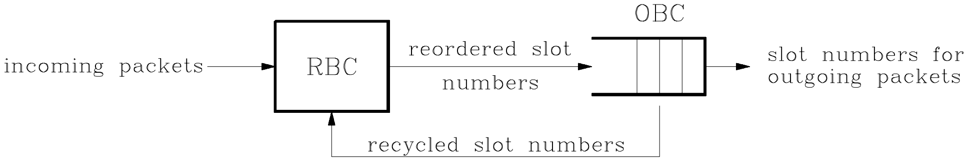

Figure 11: Combined Resequencing and Output Buffer Control

4. Combined Resequencing and Output Buffer Control

In a typical application in an ATM switching system, the resequencing buffer precedes a larger output buffer. The storage for the cells is best provided by a conventional random access memory chip, controlled by a buffer controller. We can avoid using separate memory chips for the resequencing buffer and the output buffer using the buffer controller organization shown in Figure 11. This shows incoming cells being received at the left by the RBC. When a cell is old enough to be transmitted, the RBC passes its slot number to an *Output Buffer Controller* (OBC). The OBC queues the received slot numbers for later transmission. Like the RBC, the OBC holds the slot numbers of unused slots in addition to the slot numbers of those slots that are to be output.

The combined buffer controller can be viewed as operating in two phases. During the first phase, the RBC outputs the slot number of its oldest cell if it exceeds the age threshold and if the OBC is prepared to receive it. At the same time, the OBC uses the number of the first busy slot in its queue to read a cell from the external RAM. If a cell was output by the RBC during the first phase, then during the second phase, an idle slot number is passed from the OBC to the RBC to fill the position vacated by the slot read from the RBC. Then the slot number read from the RBC in the first phase is inserted in the OBC. The "recycled slot number" can be used for a cell entering the RBC if there is one.

The OBC can be implemented using a small RAM containing slot numbers, a read and write pointer into this RAM and a state machine. It operates much like a RAM-based FIFO with the exception that when an item is being written to the FIFO, we first read out and recycle the slot number pointed to by the write pointer and only then do we write the new slot number into that position and increment the pointer. The state machine must also, of course, include circuitry to initialize the RAM with the appropriate slot numbers.

If we let B be the total number of buffer slots, $b = \log_2 B$, R be the number of resequencer slots and m the number of bits used to represent the age, then the cost of the RBC and OBC together is approximately,

$$R((b+m)x_1 + x_2) + (B-R)bx_3$$

where x_1 is the cost of one bit in the BSR, x_2 is the cost of one slot control circuit and x_3 is the cost per bit in the OBC's RAM. If we equate cost with transistor count, then reasonable values for the x_i's are $x_1 = 12$, $x_2 = 150$ and $x_3 = 8$. Using these values and $B = 256$, $R = 64$ and $m = 8$ gives a cost of approximately 34K transistors. For comparison, 108K bits of external memory are needed to store 256 ATM cells (where each cell is 53 bytes long), so the cost of the buffer controller is an acceptably small fraction of the intrinsic cost of

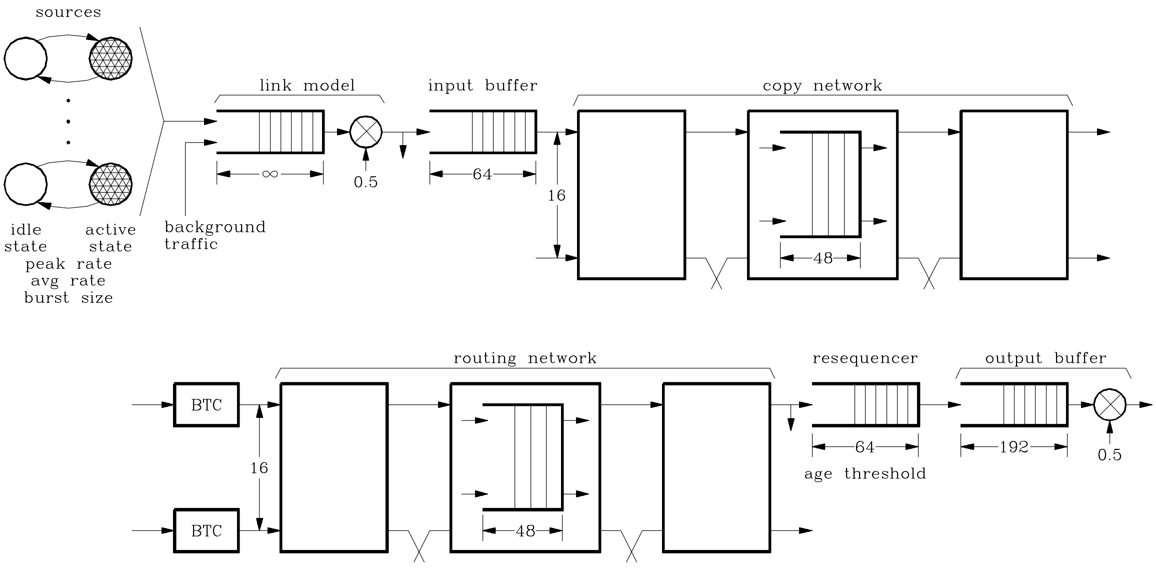

Figure 12: Broadcast Packet Switch Simulation Model

storing the cells. If we increase B to 1024, without changing R and m, the cost of the controller is approximately 100K transistors as compared to 434K bits of external memory needed.

5. Simulation of an ATM Switch with Resequencing

We have evaluated the effectiveness of the resequencer on a variant of the broadcast packet switching system described in [5]. The system comprises a copy network followed by a set of *Broadcast Translation Circuits* (BTC) and a routing network (see [5] for an explanation of the function of each of these components). The simulation model for the variant considered here is illustrated in Figure 12. Note that both networks are implemented using a three stage Beneš topology with 16 port switches, that each include a single shared buffer of 48 slots. This gives a total of 256 inputs and outputs to the system. Each switch in the networks can regulate the flow of cells into it using a simple flow control mechanism, so that cells cannot be lost within the network. The copy network is preceded by an input buffer of 64 slots and the routing network is followed by the resequencing buffer and output buffer. Cells can be lost due to overflow at either the input buffer or the resequencing buffer.

In the simulated system, we assume that the data rate of the internal data paths is 300 Mb/s, while the data rate of the external links is 150 Mb/s. This means that the output buffer can transmit only one packet for every two of the system's internal operational cycles. This is modeled by allowing the output buffer to send a packet with probability 0.5 on each cycle. We model a previous switch and the connecting link in a similar way by inserting an unbounded buffer between the sources and the switch's input buffer. This buffer is also permitted to send a packet with probability 0.5 on each cycle.

Figure 13: Simulation Results

The input traffic to the system is generated using a bursty source model. Each source alternates between idle and active states with geometrically distributed holding times in each state. When active, a source generates packets with geometrically distributed inter-packet times. The packets from each bursty source are sent to a fixed set of outputs, so that the effects of bursts on output links and switch data paths are accurately modeled. In addition to the bursty traffic, a specified background traffic can also be inserted. The background traffic follows a Bernoulli arrival process and each background packet is independently and randomly addressed.

In the results discussed here, half the total traffic was generated by bursty sources with a peak rate of 30 Mb/s, an average rate of 1 Mb/s and an average burst size of 50 cells. The average fanout for the bursty sources was 4, and the number of bursty sources per input varied to create the desired offered load. In each simulation run, a total of approximately 4 million cells were passed through the network. The left hand plot in Figure 13 shows the average queueing delay through the network with bursty traffic and an age threshold of 40. The bottom curve in this set, shows the average delay in the input buffer alone, the second curve gives the delay for the input buffer, plus the copy network, the third curve from the bottom gives the delay for the input buffer, copy and routing networks, and so forth. Notice that at small offered loads, most of the delay is in the resequencer; that is, most cells pass through the copy network and routing network with little delay, but then wait in the resequencer until their age threshold is reached. We note that for ATM switching systems, the actual magnitude of the delay penalty imposed by the resequencer in this case is just under 120 μs, which is significantly smaller than the circuit delays imposed by a digital time-division telephone switch.

Notice that queueing delay in the output buffer starts increasing sharply when the offered load is around 100 Mb/s. On the other hand, the delays in the switching network and input buffer stay very flat until the offered load reaches about 120 Mb/s. This indicates that the switching system is essentially "transparent" to the arriving traffic so long as the offered load is 120 Mb/s or less. Since the link overloads at a smaller offered load, we observe that the link is the primary bottleneck to the traffic rather than the switch.

The right hand plot in Figure 13 shows the fraction of cells that are lost due to overflows in the resequencer/output buffer combination or the input buffer. When cells are lost at the resequencer, they are almost always lost because the output buffer is full; while it is possible for the resequencer to overflow when the output buffer is not full, this happens very rarely. Hence, the cell loss rate at the resequencer/output buffer combination is virtually identical to the cell loss experienced in a system with no resequencer and an output buffer that can hold the same number of cells as the resequencer/output buffer combination. The plot also includes two curves which show the fraction of cells that are delayed so much by the switch that they arrive at the resequencer later than the time that they would normally have been passed on to the output buffer. Such late cells are potentially out of sequence. The two different curves give the lateness results when operating with an age threshold of 40 in one case and 56 in the other.

If we assume that the acceptable cell loss rate and the acceptable out-of-order rate are equal and no greater than say 10^{-4}, then the cell loss experienced at the resequencer/output buffer combination constrains the system to a smaller offered load than does the probability of late arrivals at the resequencer. In other words, for traffic loads that the links are capable of supporting, the resequencer has no measureable impact on the cell loss rate.

6. Closing Remarks

The primary conclusion to be drawn from this work is that ATM switching systems with per packet routing and cell resequencing offer a viable, effective and economical alternative to systems with fixed path routing. We have shown that the incremental cost of adding a resequencer to an output buffer of an ATM switch is quite modest and have demonstrated that the probability of mis-sequencing cells can be made vanishingly small, even in the presence of bursty traffic.

An interesting question arises from this work. As more buffering is added to the copy and routing networks, their data carrying capacity increases, but at the same time the delay variation goes up, increasing the size of the resequencer needed and the delay penalty imposed by it. It may be possible to operate with a much smaller resequencer if an adaptive age threshold is used. The idea here is for the resequencer to observe the average age of the incoming cells and set the age threshold at some level higher than the observed average. During periods of small average delay, a small threshold would be used, allowing cells to pass through the resequencer with minimal delay. During periods of high average delay, a large threshold would be used, but the delay added by the resequencer could remain about the same.

References

[1] Coudreuse, J. P. and M. Servel. "Prelude: An Asynchronous Time-Division Switched Network," *International Communications Conference*, 1987.

[2] Melen, Riccardo and Jonathan S. Turner. "Nonblocking Multirate Networks," *SIAM Journal on Computing*, 4/89.

[3] Melen, Riccardo and Jonathan Turner. "Nonblocking Multirate Distribution Networks," *Proceedings of Infocom 90*, 6/90.

[4] Suzuki, Hiroshi, Hiroshi Nagano, Toshio Suzuki, Takao Takeuchi and Susumu Iwasaki. "Output-buffer Switch Architecture for Asynchronous Transfer Mode," *Proceedings of the International Communications Conference*, 6/89.

[5] Turner, Jonathan S. "Design of a Broadcast Packet Network," *IEEE Transactions on Communications*, June 1988.