

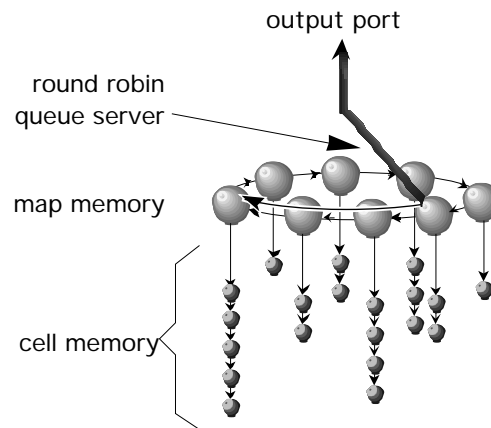
# **Bandwidth reservation in an ATM switch using the Associative Chandelier**

P.J. Lyons

## **Abstract**

An ATM switch based on the simple Associative Chandelier design can buffer data and prioritise channels' access to link bandwidth. High-priority channels can send more cells along a link each time they are serviced than low-priority channel can. However when a high-priority channel shares the link with a large number of low-priority channels, the sheer number of other channels can reduce its bitrate to an unacceptably low value, even though it is achieving a higher bitrate than any other individual channel. This report describes a modification to the priority system which supports true bandwidth reservation by virtual channels. It achieves this by dynamically controlling the service given to the buffer queue of each reserved-bandwidth channel. At each service, the channel is allocated the full output bandwidth of the communications link until the net bitrate it has received since its last service is equal to the proportion of the communications link's bandwidth which has been reserved for the channel.





1. Block diagram of the chandelier data structure

### Background to the chandelier concept

In an earlier paper, Lyons, McGregor and Moretti (1996) described the associative chandelier. Figure 1 shows this hardware implementation of a dynamic data structure, an adaptation of a software buffer originally designed for a slow (RS232) network called MasseyNet (Lyons and McGregor, 1987), (Lyons, McGregor and van Oeveren, 1987) and (Lyons and McGregor, 1990), which incorporated Bleazard's (1979) packet-switching technique.

MasseyNet was designed to support file transfers and interactive computing. The former require high throughput; the latter require short response times. The network did not provide local echo of typed characters, but relied in character echo from the mainframe at the other end of the network, so it was necessary for characters to be transferred rapidly through the network. Woodson (1982) had demonstrated that interactive computer users felt that character echo was instantaneous if it occurred within 0.1s of the keystroke. In order to satisfy this requirement with a single class of service, MasseyNet incorporated a buffering mechanism called the chandelier. This prevented characters typed by a user from having to queue in a node behind a large chunk of data, and prevented round-trip delays exceeding 0.1s.

The chandelier was designed to provide fair access to the channel bandwidth for all the virtual channels. Fair access was interpreted to mean an equal allocation of the bandwidth to all virtual channels which had data buffered in the switch. Thus if only one virtual channel had data buffered, it would be allocated all the available bandwidth, whereas if there were 4, 5, or 9 such channels, each would receive 1/4, 1/5 or 1/9 of the bandwidth. The allocation was inherently dynamic, so that as soon as any channel's buffered data was flushed from the switch, or as soon as data arrived for a virtual channel which did not currently have any data buffered, the other channels' allocation was updated.

The chandelier provides all virtual channels using a particular output port with equal access to the data link, by buffering their cells in separate queues. A round robin server moves around the queues, removing a cell from each and outputting it whenever the output port is free to transmit another cell. Thus even if a particular virtual channel has dumped a large number of cells into cell memory (the node's buffer) other virtual channels' cells will not be delayed by having to queue behind this large amount of data. The round robin server obtains access to the buffered data *via* map memory, a separate memory architecture which supports a circular list of pointers to the cell queues; it removes an entry from this list when a virtual channel's queue empties, and reinstates when new cells for that virtual channel are buffered. Thus the round robin server does not waste time dealing with virtual channels which currently have no data buffered in the node. Figure 1 shows the circular list and the associated cell queues referred to as the *chandelier data structure* (often abbreviated simply to "chandelier")

## **Incorporating the chandelier into an ATM switch**

Lyons, McGregor and Moretti (1996) incorporated the chandelier concept into a design for a high-speed ATM switch in which the switching function is achieved by passing all cells through a buffer. The buffer incorporates two parts; the *map memory* associatively matches a cell's incoming routing information and points to the cell's virtual channel's buffer queue (in *buffer memory*, the second part of the buffer). Then when the switch is ready to remove an item from that queue, the map memory also supplies the ongoing routing information. Thus the switching function turns up as a by-product of the switch's buffering function.

During the design of this high-speed hardware version of the original MasseyNet network, it became apparent that it would be trivial to modify the fair bandwidth allocation system so that channels' access to the communications link could be prioritised. The basis of the prioritisation scheme was simple; instead of moving on to the next virtual channel as soon as it had taken one cell from the current channel, the round robin server would continue to serve a virtual channel until a number of cells equal to the virtual channel's priority had been transferred to the output port, or the queue became empty. Thus a priority-4 channel would output up to four cells each time the round-robin server reached it, whereas a priority-1 channel would only be able to output 1 cell at each visit.

Unfortunately, although this system is simple, it cannot be relied upon to provide a fixed bandwidth to a particular channel. This is because each channel's service rate is dependent on the size of the circular list in map memory. The round robin server services each channel with buffered data in turn. Thus, as more virtual channels establish links through, and buffer data in, a node, the throughput of all channels will be reduced, even though the high-priority channel will still be receiving a higher allocation than the others. For some types of data transfer, most notably video, this is not sufficiently precise. Live video is generated at a constant rate, and, perhaps more importantly, is consumed at a constant rate. The data rate in the network must not decrease below this rate, or the receiver will have no picture to display. For this reason, ATM systems are required to be able to provide guaranteed data rates.

ATM switching techniques are capable of allocating bandwidth in a highly dynamic way. The switch's "customers" (virtual channels) only consume bandwidth when they actually have data to send, and when the number of "customers" generating data for the switch is low, those customers receive high rates of throughput. Conversely, when the number of virtual channels is high, and many of them are generating data, they all receive low rates of throughput. The conventional approach to providing a reserved-bandwidth service is to implement some type of fixed multiplexing such as time-division or frequency-division multiplexing, in which a constant amount of bandwidth is always available to the customer, whether it is used or not. This is wasteful of bandwidth, in contrast to the "conservation-minded" dynamic approach of ATM, and at first glance, seems difficult to integrate into the dynamic ATM environment.

However, the associative chandelier's prioritisation can easily be modified to provide reserved bandwidth, *without* compromising its efficient dynamic allocation of bandwidth when it is not being used by the reserving channel.

The technique maintains a history of recent use of the link, and at each service, removes sufficient cells from the reserved-bandwidth virtual channels to ensure that they receive their allocated amount of bandwidth since the last service. It bases this determination on the number of cells which other channels have received since the reserved-bandwidth channel's last output phase. This replaces the passive bandwidth-allocation of the earlier priority-based scheme with a dynamic allocation scheme based on an observed pattern of usage of the switch.

The remainder of this paper describes the chandelier and the dynamic bandwidth allocation scheme in more detail, and discusses how to incorporate the latter into the hardware of an associative-chandelier ATM switch.

### **The Associative Chandelier**

The output ports in the ATM switch share a buffering system. It is based on two separate memories; map memory, stores administrative information for the virtual channels, such as routing tables; cell memory stores buffered data. Any location in these memories may be allocated to any output port, but for our present purposes, it is simpler to consider that there is a separate data structure associated with each port. All virtual channels which have a connection established through the switch have an entry in the administrative memory, which operates associatively.

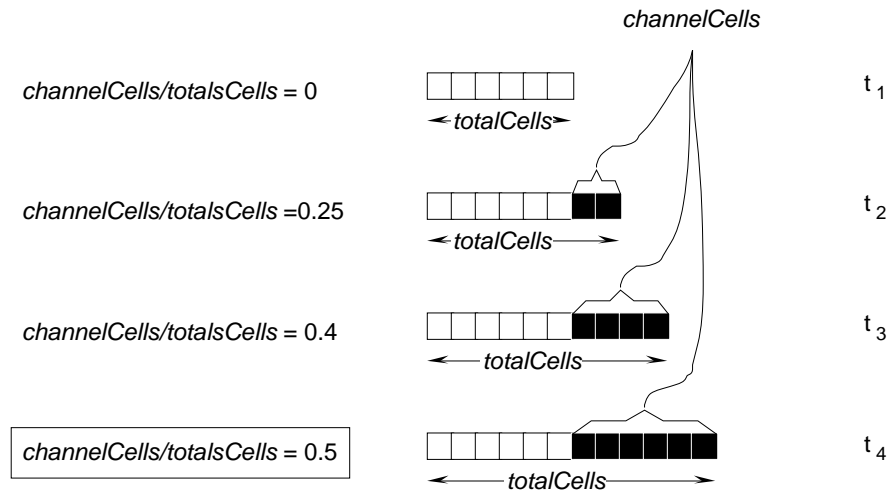
Thus, when a new cell arrives, its administrative information (routing information, pointers to its virtual channel's buffer, etc) can be found in the associative memory very rapidly, no matter where it is located. If the virtual channel already has data buffered in cell memory, the cell is stored in a queue in cell memory; the virtual channel's entry in map memory has pointers to both ends of this queue, and the cell can therefore rapidly be added into (and, later, removed from) the buffer. If the virtual channel does not already have data buffered when the cell arrives, a new buffer queue is created for it in cell memory, and it is entered into the queue.

In addition, the virtual channel's entry in the administrative memory is linked into a circular list of virtual channel headers belonging to the output port for which the cell is destined. It is this circular list which the server moves around, using the pointer information contained in the list entries to access cells from the queues stored in the buffer memory. As each entry in the circular list references a different virtual channel, each successive cell output through the port will generally come from a different virtual channel. Exceptions to this generalisation occur

- when the list has only one entry (then that entry will be visited repeatedly without interruption)
- when the priority system is in operation (then the server will pause at the virtual channel until the number of cells output from that virtual channel's queue is equal to the channel's priority)
- when the queue is empty.

### **An approach to guaranteeing bandwidth**

The prioritisation system described at the end of the preceding paragraph maintains the relative priority between different channels, but it cannot guarantee a constant bandwidth to a particular channel, as, if many channels have data buffered in the switch, then the server will have to output a large number of cells before returning to the channel in question, and its net throughput will be reduced accordingly. If this channel is to obtain its required bandwidth, the number of cells which will then have to be removed from its queue will be a set fraction of the number of cells transferred from the other channels. If the channel in question has been allocated too low a priority, the number of cells transferred will be too small; if the channel has been allocated too high a priority, the channel will receive an unfair slice of the bandwidth.



## 2. The server discharges its service debt to the reserved-bandwidth channel C

Clearly, the system needs to be more responsive to the current behaviour of the switch. We can accomplish this by making the number of cells the server removes from the queue a variable, dependent on the amount by which the channel has fallen behind its reserved allocation.

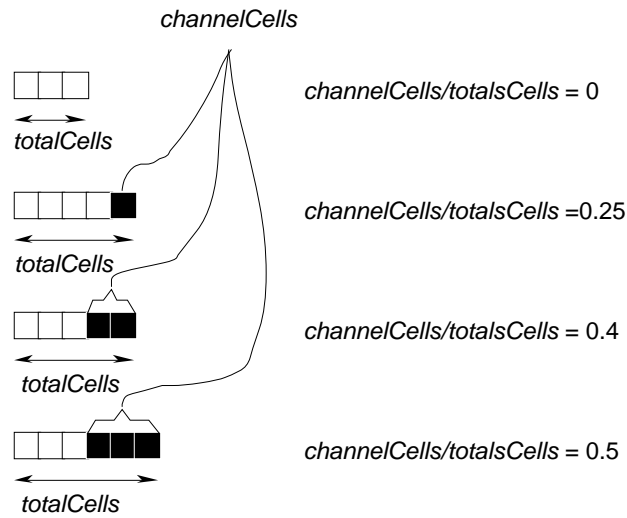
This more dynamic behaviour can be achieved if

- the system can count *totalCells*, the total number of cells transferred since the last visit to the channel.
- the system can count *channelCells*, the number of outgoing packets taken from the reserved bandwidth channel at this service
- the channel's reserved bandwidth can be expressed as a fraction of the total data rate of the communications link, *allocation*
- the server can stop removing the current channel's cells when  $channelCells/totalCells = allocation$

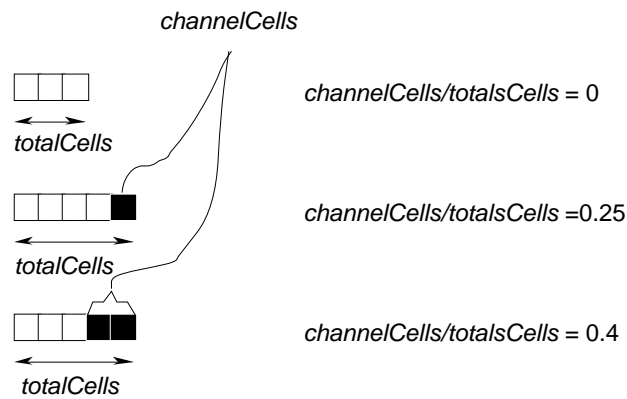
Consider C, a channel which is entitled to half of the bandwidth of the link. Figure 2 shows a series of snapshots of the cells output from its output port. The first shows (in white) all the cells which have been transferred to the output port since C was last visited by the server. The server is about to visit C again. The second and subsequent diagrams show the situation at successively later times, while the server is attending to C. In these diagrams, cells (black) which have been output by C are visible. The server continues to remove cells from C until the value of  $channelCells/totalCells$  becomes equal to 0.5. At that stage, the "service debt" which was accumulated while the server was attending to other channels has been paid, the server is free to move on to another channel, and the cycle begins again.

Figure 3a illustrates the dynamic, adaptive behaviour of the approach. If, on the server's next cycle round the buffer, the server has removed fewer cells from other channel's queues (because they were emptied on the previous visit), then the total number of cells which will have to be removed from C, before the desired 0.5  $channelCells/totalCells$  ratio is achieved, is proportionately fewer. That is, the server has spent less time on other channels' business, so it needs to spend less time on C's in order for C to receive the requisite proportion of the link's capacity.

(a)



(b)



3. (a) Fewer cells need to be removed from C when the other channels are less active  
(b) Fewer cells need to be removed from C when it requires less bandwidth

Figure 3b also illustrates the system's adaptivity. It shows a server cycle for a different channel with a lower amount of reserved bandwidth (40% of the link's capacity). Such an approach has several benefits. It will

- ensure that the channel with reserved bandwidth receives it when required
- dynamically allocate that bandwidth to other channels when the reserving channel does not use it (for example when the channel is being used to transfer an MPEG-compressed picture of a scene with very little movement, such as a snooker game)
- allow the reserved channel to receive more than the reserved bandwidth when a small number of other channels is using the node, as then a single cell will be removed from the channel's queue at each visit by the server, and, if the circular list of channels with buffered data is small enough, this can be enough to keep the channel's throughput above its allocation.

## Hardware implementation

If there were no speed constraints, this could all be achieved easily, in software. However, the system is intended to achieve the greatest possible speed, and is designed for implementation in hardware, with the greatest possible efficiency. The data structures and control sequences for the associative chandelier have been designed so that each cell can be output without loops, particularly loops involving memory accesses, because of the need to transfer data through the switch as the greatest possible rate, and bandwidth reservation needs to operate under the same conditions.

An efficient hardware algorithm for implementing the above technique is shown below. Consider a reserved-bandwidth channel *C*. When the server is about to move on to another channel after dealing with *C* (or when *C*'s queue of cells is initialised), the value of `totalCells` is stored in the channel's map memory entry. All of the map memory's 1024 locations can potentially be allocated to the virtual channels connected through a single output port, so a 10-bit counter is used. It is incremented as each cell is transferred to the output port. As the round robin output server moves from channel to channel round the chandelier, it continues incrementing `totalCells` whenever a cell is transferred to the output port. The difference between the two values of `totalCells` (one in the counter, the other retrieved from map memory) represents a service debt which the server must discharge before *C* can be said to have received its full allocation of bandwidth. If the bandwidth reserved by *C* is high, then transferring a single cell to the output will only discharge a small proportion of the debt (see Figures 3 and 4). If the bandwidth reserved is low, then a single cell from *C* will discharge a proportionally higher amount of the debt. These two observations give us the basis for a technique for ensuring that the switch can provide sufficient bandwidth to *C* without performing the time-consuming division (`channelCells/totalCells`) implied in the earlier discussion.

- Every time a cell from any channel is transferred to the output port, the counter `totalCells` is incremented.
- When the server reaches channel *C*, the switch sets a hardware variable `repayment` to the old value of `totalCells`, which was stored in the channel's map memory entry at the end of the server's last visit to *C*. It also retrieves from map memory the value of a `repaymentMultiplier` ( = total link capacity/reserved bandwidth) for the channel.
- The server transfers a number of cells from channel *C* to the output port, incrementing `repayment` by `repaymentMultiplier` at each transfer.
- When `totalCells` falls between the previous and current values of `repayment`, or when the channel's buffer becomes empty, the server moves to the next virtual channel, but first, it saves the current value of `totalCells` in *C*'s map memory entry

This algorithm uses only addition in dynamically calculating the number of cells which must be output from channel *C* at each service in order to achieve the reserved output bandwidth (The division necessary to calculate `repaymentMultiplier` only needs to be performed once, when the channel is established through the switch, by the switch's administrative software; it is not a load on the hardware).

This calculation (that is, the minimum number of cells to output from *C*) establishes the minimum service rate which *C* will receive, and it will occur when the switch is buffering a large number of cell queues. When other channels are not using the switch - even momentarily - or when the load imposed by other channels is so light that transferring a single cell at each service will allow it to exceed its allowance, then there is nothing to stop *C* from receiving a larger allocation of bandwidth than it has been allocated. This may not have much practical consequence; customers transferring video will have to reserve enough bandwidth to ensure

lossless transfer of their signal, but will neither generate nor consume data faster than this rate under normal circumstances, as the whole purpose of reserving bandwidth is to allow at least one end of the link to transfer real time data.

Note also that if channel C does not make use of its reserved bandwidth, the bandwidth is available to other channels. For example, if the link is transferring MPEG compressed video, a certain minimum bandwidth will need to have been reserved, to allow for highly mobile sequences of images with a low correlation, and (which amounts to the same thing) cuts between scenes. However, the transfer rate will often fall below this when the difference between successive images is small. In such circumstances, C's buffer will often become empty before the service debt has been paid (the service debt is paid in full each time the server visits the reserved bandwidth channel; it is not a running average). Allocating the bandwidth to other channels in this circumstance is not unfairly depriving C of bandwidth it has reserved; as soon as any more cells are buffered for channel C, the value of *totalCells* is stored in C's map memory entry, and the server starts accumulating service debt to C again. Thus C will receive its reserved allocation of bandwidth so long as it has data buffered in the switch. It is C's responsibility to ensure that it makes use of the bandwidth it has been allocated, not the switch's.

## Conclusion

The Associative Chandelier is a hardware design for a buffering system for ATM switches. This paper has presented a minor modification of the design originally presented by Lyons, McGregor and Moretti, (1996) which makes it capable of supporting bandwidth reservation by virtual channels. It achieves this by dynamically controlling the service given to the buffer queue of each reserved-bandwidth channel. At each service, the channel is allocated the full output bandwidth of the communications link until the net bitrate it has received since its last service is equal to the proportion of the communications link's bandwidth which the channel has reserved.

The algorithm described for accomplishing this is simple to implement in hardware, and it will be fast; the only calculations are additions, and there are no loops at the single-cell-output level (although the approach involves a loop outputting a number of cells, there is no penalty associated with this; the cells were going to be output anyway; the algorithm merely determines the order of their going). It requires two fields in the switch's chandelier (map) memory; the field previously used for storing channel priority can now be used for storing the repayment multiplier, the other (new) field will be used to store the *totalCells* between services.

Note that a switch could provide both high-priority and reserved bandwidth services at the cost of a single bit of extra storage per channel to specify which type of channel it was. In this case, it would be sensible to use the same field in the map memory for a channel's priority and its *repaymentMultiplier*, as these parameters fulfill essentially the same role in the two approaches.

The switch will also have to ensure that more than 100% of its throughput is not reserved. Like the calculation of *repaymentMultiplier*, this only occurs at the time the channel is established through the switch, and is the responsibility of the node's administrative software, and not the hardware.

## **References**

**Bleazard, 1979**

“Why Packet Switching?” Bleazard, G.B.; NCC Publications, 1979

**Lyons and McGregor, 1987**

"MasseyNet: A University-oriented Local Area Network"; P.J. Lyons and A.J. McGregor, MICROS PLUS: Educational Peripherals; Proceedings of the IFIP Working Conference on the Educational Implications of Connecting Tools and Devices to Micro-computers, August 1986, 155 - 167

**Lyons, McGregor and Moretti, 1996**

“The Associative Chandelier - Fair, Efficient Prioritised Buffering in ATM switches” , P.J. Lyons, A.J. McGregor and G.S. Moretti; Proceedings of the First New Zealand ATM and Broadband Workshop, February 1996, 127 - 151

**Lyons, McGregor and van Oeveren, 1987**

"Performance Measurements in the MasseyNet Local Area Packet Switching Network"; P.J. Lyons, A.J. McGregor, and E. van Oeveren, Proceedings of the 1987 NZ Computer Society Conference, R126-R135

**Lyons and McGregor, 1990**

"Multipath Local Area Network"; Paul J. Lyons and Anthony J McGregor, United States Patent Number 4,953,162, August 28, 1990

**Woodson, 1982**

“Human Factors Design Handbook”; Woodson, 1982