Output-Buffer ATM Packet Switching for Integrated-Services Communication Networks

G. Kesidis
E & CE Dept
University of Waterloo
Waterloo, ON, Canada N2L 3G1
g.kesidis@eandce.uwaterloo.ca

N. McKeown
Elec. Eng. Dept
Stanford University
Stanford, CA
nickm@ee.stanford.edu

Abstract— In this paper, we give an overview of the basic design principles and trade-offs of output-buffer ATM switching. Output-buffer switches give optimal performance in terms of offering bandwidth guarantees to individual flows. Bandwidth scheduling and memory bandwidth requirements are also described.

I. Introduction

Many existing ATM packet switches[1] are primarily concerned with maximizing throughput to the output links subject to the tightest possible memory bandwidth restrictions. Such switches are well-suited for the transport mechanism of data networks such as the Internet, i.e., "TCP/IP over ATM". For integrated-services networks, however, the quality-of-service (QoS) requirement of individual real-time connections, such as video teleconferencing, is a great concern. Output-buffer switches can accommodate individual QoS requirements because they are able to erect bandwidth "firewalls" between individual connections or classes of connections. That is, output-buffer switches facilitate the ability to offer a bandwidth *guarantee* to a traffic flow. The drawback of output-buffer switches is that they suffer from significant memory bandwidth and/or scalability limitations at high transmission speeds. Also, associated with high memory bandwidth requirements are high switch fabric speed requirements.

This paper is concerned with the design of output-buffer, ATM packet switches. We assume here that the better accountability of QoS that is possible under output-buffer switching is worth the possible reductions in speed of operation. Good surveys of ATM switching are given in [5], [13], [1].

A generic single-stage ATM packet switch is initially defined. The condition for output-buffer switching is then specified. Why output-buffer switches are ideal from a QoS management point-of-view will then be explored. The memory bandwidth limitation of output-buffer switches is explained and simple memory bandwidth/scalability tradeoffs are described. The implementation and performance issues of TDM bandwidth schedulers are compared against those that use local time stamps, e.g., PGPS and Virtual Clock. Finally, conclusions are drawn.

G. Kesidis is supported by the NSERC of Canada.

[1] In this paper, we focus on packet switching as opposed to wavelength division multiplexed (WDM) optical switches. ATM uses fixed-length packets, called "cells", of 53 bytes.

II. Single-Stage Packet ATM Switches

A generic single-stage packet ATM switch is illustrated in Figure 1. The queueing stage consists of a bank of logically separate queues distributed among "blocks" of memory. Each block of memory has a separate I/O bus and, therefore, can operate independently from the other blocks.

Every unit of time (transmission time of a cell): at most one cell arrives at each input port, at most one cell departs from each output port, the input-side fabric removes (at most N) cells from the input ports and places them in the queueing stage, and the output-side fabric removes (at most N) cells from the queueing stage and places them in output ports for transmission onto the output links.

We assume switch fabrics are nonblocking, i.e., cells are never dropped passing "through" a fabric. A cell may be dropped by the queueing stage if, for example, it arrives to a full queue.

Each queue typically handles a connection or a class of connections. If there is a separate queue for each connection, the resulting switching is called per-virtual-channel (per-VC) or per-virtual-circuit.

III. Output-Buffer Switches

A defining condition for output-buffer switching is: no two cells that are destined for different output ports use the same memory block. So, there is at most one read operation per unit time for each memory block and each memory block is associated with only one output port.

A processor sharing node (PSN) is associated with each output port. A PSN consists of a bank of synchronized, slotted, FIFO queues that share the output link transmission bandwidth via some *bandwidth scheduling* policy. FIFO queues of a PSN may reside on different blocks of memory but FIFO queues of two different PSNs cannot reside on the same block of memory. The N independent bandwidth scheduling policies constitute the output-side switch fabric of an $N \times N$ output-buffer switch.

In the general context of single-stage switches, output-buffer switches form one group. Another group are input-buffer switches in which each memory block has only one associated input port. The third group are shared-memory switches in which each memory block has more than one associated input port and more than one associated output port. Each group has drawbacks: multicast is a significant complication of shared-memory and input-buffer switches and, in addition, input-buffer switches suffer from bandwidth contention at the input ports resulting in poorer guaranteed service properties. Output-buffer switches suffer from memory bandwidth limitations as described below.

Note that a switch that has two queueing stages (input-and-output-buffer switches) can be realized by using two generic single-stage switches in tandem.

A. Performance Advantages of Output-Buffer Switches

Output-buffer switches have the following performance advantages. They enable "bandwidth firewalls" to be simply erected between the FIFO queues, c.f., the "minimum-bandwidth" property of Section 4. Nonblocking fabrics help to ensure that "the adequacy of service delivered to a particular user should not depend on the detailed behavior of other users" [10] (this is basically a notion of fairness). Deterministic bounds on end-to-end delay and end-to-end buffer sizing results for lossless transmission are available under output-buffer switches; these results can be extended to arbitrary virtual path connection (VPC) structures [9]. Consequently, the use of output-buffer switches simplifies network resources management to achieve *individual* QoS requirements. Finally, because output-buffer switches do not suffer from contention at the input ports, they can achieve 100% throughput to each output link.

B. Memory Bandwidth Requirements

Consider an $N \times N$ output-buffer switch in which a single memory block is associated with each output port. In the worst-case, $N+1$ read/write operations per unit time are required for any given block of memory. This is the memory bandwidth requirement of this output-buffer switch. For example, for the 2×2 switch of Figure 2, up to two write and one read operations per unit time are possible.

C. Simple Memory Bandwidth/Scalability Tradeoffs

Consider the 2×2 switch of Figure 3 which uses a separate memory block for all cells having the same input/output-port pair. Although this switch is typically thought of as an output-buffer switch, clearly it is *both* output- and input-buffered because each memory block experiences at most one read and at most one write operation per unit time. This switch has a scalability problem: the number of memory blocks is a *quadratic* function (N^2) of the number of ports (N). Also, with a separate memory block for each input/output-port pair, the amount of statistical multiplexing and the structure of VPCs may be restricted.

One way to reduce memory-bandwidth requirements is to use multiple-stage implementations [13], see Figure 4. With multiple-stage implementations, there is more statistical multiplexing and fewer restrictions on VPC structures than the N^2-memory-block, single-stage switches but each connection experiences more "hops." Also, the multistage switch will still have N^2 memory blocks organized around N^2 2×2 fabrics, instead of one $N \times N^2$ fabric and one $N^2 \times N$ fabric.

For single-stage switches, we now describe simple ways to improve scalability when additional memory bandwidth is available.

If $k > 2$ read/write memory operations per unit time are possible: one can use a single memory block per each output port and per $k-1$ input ports. So, the required number of memory blocks would be $N^2/(k-1)$. For example, consider the switch of Figure 2 with $k=3$.

Alternatively, if $k > 2$ read/write memory operations per unit time are possible and N^2 memory blocks are used (as in the switch of Figure 3), then the memories can operate on a clock cycle of $1/k$ of a unit of time. A read or write of $2/k$ of a cell ($53 \times 8 \times 2/k = 848/k$ bits) occurs each clock cycle. So, each memory I/O bus is $2/k$ of a cell wide ($848/k$ wires) instead of one cell wide (424 wires).

D. Multicast Connections

A multicast connection has a single network source but has more than one network destination. For example, a teleconference among three parties can be implemented with three multicast connections each having one source and three destinations. Multicast can be implemented by input ports and input-side fabrics that copy multicast cells to more than one PSN simultaneously. Note that multicast is not a significant complication to output-buffer switches.

IV. The Processor Sharing Node (Switch Output Port)

The role of the bandwidth scheduler in a PSN is to determine which queue to serve at each departure epoch. Each cell of a connection with a *bandwidth guarantee* has a desired *deadline* by which it should be transmitted by the switch. Note that this is the desired deadline — a cell may actually depart before, at or after its desired service deadline.

For the n^{th} queue having bandwidth allocation ρ_n cells per unit time,[2] the deadline of the i^{th} cell is given by the following recursive formula:

$$\begin{aligned} \mathcal{F}_i^n &= \max\{\mathcal{F}_{i-1}^n, a_i^n\} + \rho_n^{-1} \quad \text{with} \\ \mathcal{F}_0^n &= 0. \end{aligned}$$

Note that $\lceil \mathcal{F}_i^n \rceil$ units of time is the departure time of the i^{th} cell from a queue having the same arrival process as the n^{th} queue of the PSN and *exactly* ρ_n cells per unit time service bandwidth.

Let d_i^n be the departure time from the PSN of the i^{th} cell of the n^{th} queue. If

$$d_i^n \quad \le \quad \mathcal{F}_i^n + \mu_n \quad \text{for all } i,$$

then μ_n is called the "minimum-bandwidth" (a.k.a. "guaranteed rate" or "guaranteed service") parameter of the n^{th}

[2] Recall that a unit of time is the transmission time of a cell.

queue [15], [4], [6], [9]. That is, μ_n is a measure of how well the departure times from a queue of the node ($\{d_i^n\}$) track those of a reference queue ($\{\mathcal{F}_i^n\}$) in order to provide the node queue with a minimum amount of bandwidth.

We now consider representatives of two major classes of bandwidth schedulers. One class of schedulers is called time-division multiplexed (TDM) and the other is based on local time stamps.

A. Hierarchical Round Robin

Weighted Round Robin (WRR) is a TDM scheduler with fixed-size frames. Each FIFO queue is reserved a certain number of slots per frame consistent with its bandwidth allotment. In the example of Figure 5, the frame size is $f = 8$ slots and the reserved slots are indicated by the index of the queue. FIFO 1 is allotted $\rho_1 = 1/10$ cells per unit time but receives 1/8 cells per unit time.

The bandwidth granularity[3] of WRR is 1/f and the minimum-bandwidth parameter of a queue allocated ρ cells per unit time is [7]:

$$\mu \quad = \quad (1-\rho)f - \rho^{-1}. \qquad (1)$$

One can attempt to improve the bandwidth granularity of WRR by increasing its frame size f. Because WRR serves all of the FIFO queues in a round-robin manner within each frame, increasing f increases the minimum-bandwidth property parameters, see Equation (1). Hierarchical Round-Robin (HRR) [8] is a TDM scheduler that attempts to allocate bandwidth more "evenly" than WRR and thereby reduce the minimum-bandwidth property parameters. So, HRR has a better tradeoff between bandwidth granularity and minimum-bandwidth parameter μ [7], [9].

B. Virtual Clock

Under Virtual Clock, each cell is given a local time stamp when it arrives at the switch, namely its service deadline $\mathcal{F}_i^n$. Head-of-queue cells are served according to the numerical order of their local time stamps. The bandwidth granularity of Virtual Clock is arbitrarily close to zero. Also, all Virtual Clock queues have a minimum-bandwidth property parameter $\mu = 0$ [3], [6], [14]. There is also a non-work-conserving version of Virtual Clock [6] (called Idling Virtual Clock) that attempts to better control cell delay jitter.

V. Implementation Issues

In general, when designing an ATM switch one should consider worst-case per-unit-time complexity instead of worst-case per-cell complexity. For output-buffer ATM switches, there is significantly more enqueueing (cells being written to a memory block) than dequeueing [2]. With this in mind, we will consider issues related to the implementation of Virtual Clock and HRR in this section. The PSN of Figure 6 has the following general parameters:

[3] The bandwidth granularity of a bandwidth scheduler is the smallest non-zero amount of bandwidth it can allocate to a queue.

- T : the cell transmission time or "unit" of time. For example, at 155 Mbps, $T = 2.735$ μs.
- K : the maximum number of "connections" per output node
- ϕ : the worst-case fan-in from the input-side switch fabric to an output node ($\phi \leq N$)
- m : the number of memory read or write cycles per cell
- R_w : the time required for a memory read/write cycle

So, $m \times R_w$ is the memory read or write time of an entire cell.

A. Using Linked Lists to Implement FIFO Queues

In each memory block, individual FIFO queues (connections) can be implemented as singly-linked lists (SLLs) with pointers to the head and tail. Each FIFO queue could have a certain amount of memory reserved for it.

Alternatively, in [11] (for the PGPS bandwidth scheduler) a single doubly-linked list (DLL) is proposed for all queued cells of an output node. Cells are arranged in the DLL in their order of departure and, thereby, the FIFO queues of the node are "merged" into a single DLL. The drawbacks of this approach are that inserting a cell into the middle of a DLL requires three memory cycles and that a search of the queued cells is required to find the appropriate position to insert each arriving cell. An advantage of this approach is that the next cell to be served is always the one at the head of the DLL.

For each output node, the number of connections K will typically be much greater than the number of memory blocks N/ϕ; consequently, more than one FIFO queue will be implemented on each memory block. Using a separate SLL for each FIFO queue is motivated by the following reasons. All operations on the SLL occur at either the head or tail of the SLL and, therefore, each SLL operation requires only two memory cycles. In the case of TDM bandwidth scheduling (see Section V-B below), to determine which cell to serve requires very little overhead. In the case of bandwidth schedulers using time-stamps (see Section V-C below), to determine which cell to serve requires comparing the time-stamps of just the (at most K) *head-of-queue* cells and this operation is required only once per unit time.

One way to reduce the "$\log_2 K$" compares is to use "service" queues [12]. That is, FIFO queues with the same bandwidth allotment are grouped into a single service queue. The hierarchical scheduler works so that the bandwidth allocated to a service queue is distributed to the corresponding, individual FIFO queues in a round-robin manner.

B. Implementation of HRR

Under WRR or HRR, to determine which cell to serve next, a pointer to the next cell to be served is simply fetched. Assume this operation requires p seconds; $p = O(L)$ where L is the number of frame levels of the HRR scheduler. By interleaving fetch (prefetch) with memory input/output, we arrive at the following simple requirement:

$$\max\{2(\phi+1)\times mR_w,\ p\} \quad \leq \quad T.$$

In practise, the memory input/output time $2(\phi+1)\times mR_w$ determines the maximum; so, HRR will not worsen the memory bandwidth constraints of the output-buffer switch.

C. Implementation of Virtual Clock

Since Virtual Clock computes and compares local time stamps (cell service deadlines), the following parameters are also considered.

C : time required to compare two time stamps

C^+ : time required to calculate a local time stamp[4]

Interleaving the memory input/output and comparison and generation of time stamps could yield the following tradeoff formula:

$$\max\{C\times\log_2 K,\ 2(\phi+1)\times mR_w,\ \phi\times C^+\} \quad \leq \quad T.$$

The term "$C\times\log_2 K$" represents the time required to compare K time stamps using binary comparators arranged in a binary tree with height $\log_2 K$. The performance penalty of interleaving operations is an increase in cell delay of two units of time, i.e., the minimum-bandwidth parameter increases to $\mu = 2$, see Section V of [6].

D. Connection Set-Up Considerations

As new connections are set-up or terminated, the bandwidth allotments of a PSN change. For Virtual Clock, the bandwidth allotment of a queue is modified by simply changing contents of registers containing the bandwidth allotments ρ that are used to determine local time stamps. For HRR or WRR, however, a permutation of the queue assignments in the frame(s) or a reorganization of the frame structure may be required, see [8] and Chapter 3 of [5]. Clearly, a drawback of certain schedulers (e.g., PGPS) is that floating point operations may be required for bandwidth redistribution.

VI. Summary

Output-buffer ATM switches were defined and their ability to provide bandwidth guarantees to individual flows was discussed. The memory bandwidth versus scalability tradeoff was explored. Bandwidth schedulers based on round-robin disciplines and those based on local time-stamps were compared in terms of bandwidth granularity, minimum-bandwidth performance, worst-case per-unit-time computational complexity, and bandwidth redistribution costs.

Acknowledgements

We would like to acknowledge the helpful comments of Dr. O. Aboul-Magd of Nortel Ltd, Ottawa.

[4] Clearly, an integer addition to compute a time stamp would be preferable to a floating point add.

References

[1] R.Y. Awdeh and H.T. Mouftah. Survey of atm switch architectures. *Computer Networks and ISDN Systems*, Vol. 27:1567–1613, 1995.

[2] J.C.R. Bennett. Implementation of output-buffer switches (talk). *Workshop on Packet Scheduling Algorithms*, Xerox PARC, CA, Aug. 27, 1996.

[3] N.R. Figueira and J. Pasquale. An upper bound on delay for the VirtualClock service discipline. *IEEE/ACM Trans. Networking*, Vol. 3, No. 4:pages 399–408, Aug. 1995.

[4] P. Goyal, S. Lam, and H. Vin. Determining end-to-end delay bounds in heterogeneous networks. In *Proc. of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video, Durham, NH*, Apr. 1995.

[5] J.Y. Hui. *Switching and Traffic Theory for Integrated Broadband Networks*. Kluwer Acad. Publ., Boston, 1990.

[6] A. Hung and G. Kesidis. Bandwidth scheduling for wide-area ATM networks using virtual finishing times. *IEEE/ACM Trans. Networking*, Vol. 4, No. 1:pp. 49–54, Feb. 1996.

[7] A. Hung and G. Kesidis. Performance evaluation of hierarchical round-robin bandwidth scheduling for ATM. In *Proc. ITC-15, Washington, DC*, June 1997.

[8] C.R. Kalmanek, H. Kanakia, and S. Keshav. Rate controlled servers for very high-speed networks. In *Proc. IEEE Globecom*, 1990.

[9] G. Kesidis. *ATM Network Performance*. Kluwer Academic Publishers, Boston, MA, 1996.

[10] C. Lefelhocz, B. Lyles, S. Shenker, and L. Zhang. Congestion control for best-effort service: why we need a new paradigm. *IEEE Network*, Vol. 10, No. 1:pages 10–19, Jan./Feb. 1996.

[11] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single node case. *IEEE/ACM Trans. Networking*, Vol. 1, No. 3:pages 344–357, June 1993.

[12] J. Rexford, A. Greenberg, and F. Bonomi. Hardware-efficient fair queueing architectures for high-speed networks. In *Proc. IEEE INFOCOM*, pages 638–646, 1996.

[13] F.A. Tobagi. Fast packet switch architectures for broadband integrated services digital networks. *Proc. of the IEEE*, Vol. 78, No.1:pp. 133–167, Jan. 1990.

[14] G.G. Xie and S.S. Lam. Delay guarantee of Virtual Clock server. *IEEE/ACM Trans. Networking*, Vol. 3, No. 6:pages 683–689, Dec. 1995.

[15] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proc. of the IEEE*, Vol. 83, No. 10, Oct. 1995.

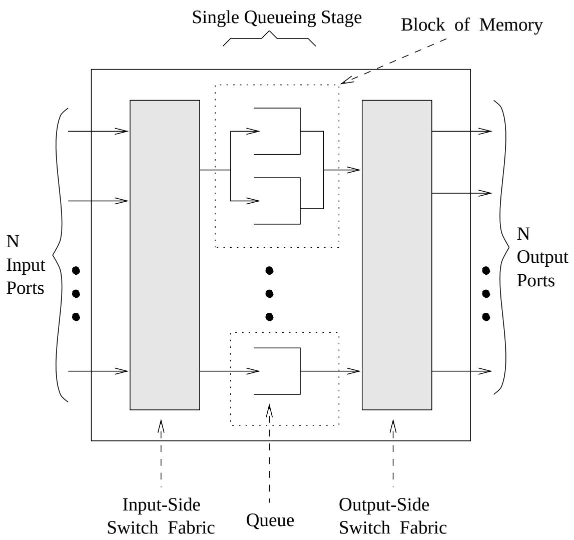

Fig. 1. A Single-Stage Packet Switch

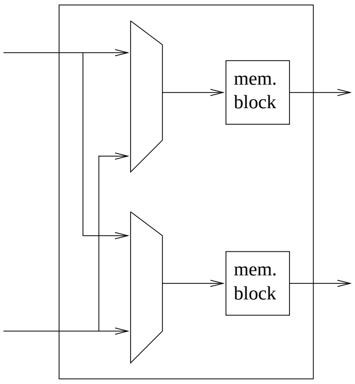

Fig. 2. Memory Bandwidth Limit of an Output-Buffer Switch

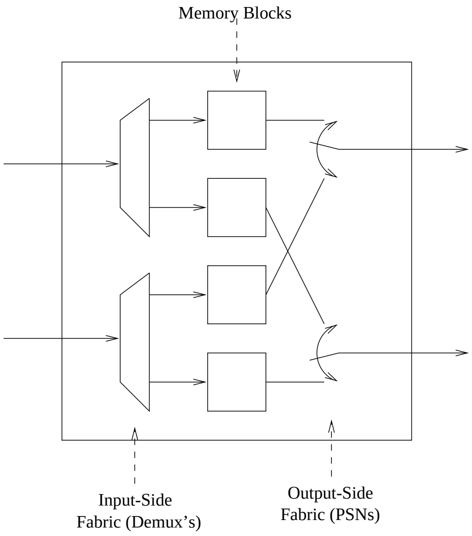

Fig. 3. An Input- and Output-Buffered Switch

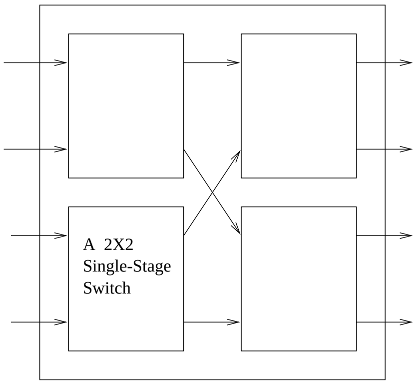

Fig. 4. A Multi-Stage 4×4 Switch

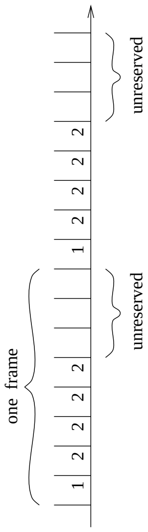

Fig. 5. Weighted Round Robin

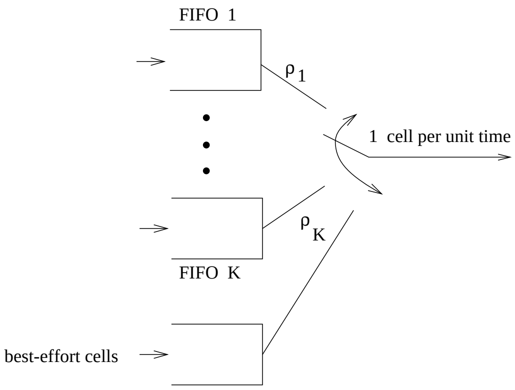

Fig. 6. A Processor Sharing Node