Multi-Queue Management and Scheduling for Improved QoS in Communication Networks

Manolis Katevenis, Dimitrios Serpanos, Evangelos Markatos *

Institute of Computer Science (ICS)
Foundation for Research & Technology – Hellas (FORTH)
P.O.Box 1385, Science and Technology Park of Crete,
Heraklion, Crete, GR-711-10 GREECE
email: katevenis@ics.forth.gr, tel.: +30 (81) 391664 fax: +30 (81) 391661
URL: http://www.ics.forth.gr/proj/avg/asiccom.html

Abstract. Quality of Service (QoS) and gigabit networking are the new dimensions that modern VLSI technology is bringing into communications, today. Highly-integrated building block components are the key in bringing this new technology to the users at low cost. Based on OMI-developed HIC/HS gigabit links, we are developing two such components. ATLAS I, a 20 Gbps single-chip ATM switch is being implemented within ACTS. This paper describes the second component, MuqPro. This Multi-Queue Processor chip will handle thousands of cell queues, and will schedule cell departures in hardware, ensuring that sophisticated QoS guarantees are met. An FPGA-based prototype is already under design; we use simulation to evaluate the performance of its scheduling algorithm. ATLAS and MuqPro achieve per-connection QoS guarantees that networks of the past could not provide: minimum throughput, maximum delay, instantaneous access to all available throughput for bursty traffic, and full utilization of the remaining capacity by low-priority traffic. We predict that within a few years, successful new products in the booming networking business will have to support such guarantees.

1 Introduction

Communication of information is at least as important as storage and processing of information. For the latter, highly-integrated, general-purpose, sophisticated building blocks already exist: SRAM, DRAM, and microprocessor chips. The former area –communication and networking– is

*The authors are also with the University of Crete.

in turmoil today: architectures and performance levels change at a pace much faster than in the memory and processor business. Under such circumstances, profit potentials are maximized for those who are willing to take risks.

Quality of Service (QoS) is the new dimension that modern VLSI technology is bringing into communication and networking today. Besides the everlasting goals of low latency and high throughput, modern communication networks also need QoS guarantees *per connection* (minimum throughput, maximum delay), instantaneous access to all available throughput for bursty traffic, and full utilization of the remaining capacity by low-priority traffic. In the past, not all of these could be provided by a single network at once. Modern VLSI technology, however, makes that possible, as discussed in this paper. Given the obvious importance that these features have for users, and given their technical feasibility, their appearance in commercial products is only a matter of time (for example, FORE already has such hardware components [2]).

Currently, the market for networking equipment that provides QoS guarantees –ATM in particular– is limited, because of the relatively high cost of such systems. ATM systems are complex, so their cost can be reduced to levels that will allow this market to grow significantly only if highly integrated ATM parts are introduced. Semiconductor companies are reluctant to make the major investment required to produce such chips as long as the market is perceived to be small (this is particularly true for 622 Mb/s and higher speeds). By demonstrating the feasibility of highly integrated commodity silicon implementations for networking with QoS guarantees, we help increase the confidence in the near-term growth of this market, thereby encouraging the investment that will bring this about. We predict that within a few years, when users will have seen in practice the importance of and benefits from QoS guarantees, all successful new products in the booming networking business will have to support such guarantees.

We are developing general-purpose building block chips for wide, local, and system area networking that will satisfy the above requirements. They are based on the OMI-developed HIC/HS technology, which provides the prerequisite: gigabit-per-second links. The first such chip, ATLAS I, is currently being implemented within the "ASICCOM"[1] project, funded by the European Union "ACTS" Programme. ATLAS I was presented at the EMSYS'96 Conference [3]. It is a 4-million-transistor 0.35-micron CMOS chip, that uses 16 OMI HIC/HS link transceiver macrocells to provide 20 Gbit/s aggregate I/O throughput. Figure 1 gives an overview of that chip.

This paper presents the architecture of the second component: the *Multi-Queue Processor*, or *MuqPro* for short (pronounced *Mac Pro*). ATLAS I provides switching and is optimized for short links; MuqPro will provide adaptation to long links and end-user stations. In order to satisfy the QoS requirements listed above, ATLAS and MuqPro offer a number of features in addition to what is standard today in ATM. MuqPro must manage many thousands of (per-VC) cell queues; separate queues per connection are required so that the bursts of one connection do not compromise

[1] the ASICCOM Consortium consists of industrial partners (INTRACOM, Greece; SGS THOMSON, France and Italy; BULL, France), telecom operators (TELENOR, Norway; TELEFONICA, Spain), and research institutes (FORTH, Greece; SINTEF, Norway; Politecnico di Milano, Italy; NCSR Democritos, Greece).

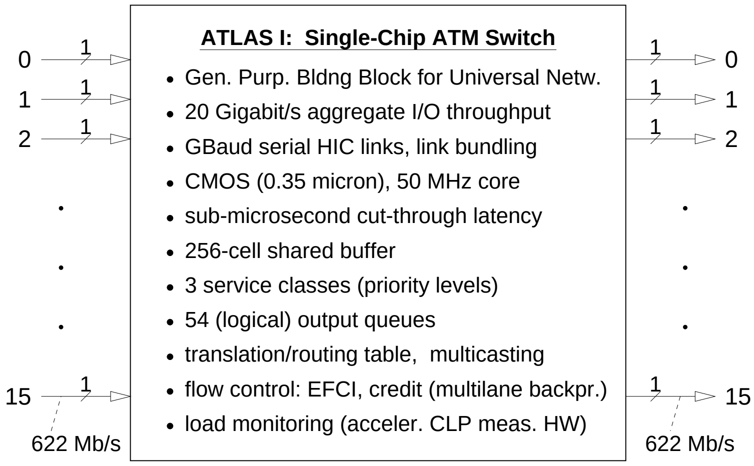


Figure 1: *Overview of the ATLAS I switch chip*

the latency guarantees of another. Transmission of cells over the outgoing link must be carefully scheduled, in order to provide fairness and throughput guarantees in the presence of bursty and low-priority saturating traffic. Since customers pay different rates for different levels of service each, weighted round-robin scheduling is needed, where the weights correspond to the service levels.

Section 2 describes the architecture of MuqPro, its interfaces, and the ways of using it. Reference is made to MuqPro I, a 155-Mbps, partial-functionality, FPGA-based prototype, which is currently under design, and to MuqPro II, the 622-Mbps, full-functionality ASIC, which is being planned. Section 3 evaluates our weighted round-robin scheduling algorithm. This algorithm is easily implemented in hardware, and guarantees that it will periodically service each ready queue, at intervals never exceeding a number that is inversely proportional to the weight factor of the queue (i.e. to the service level that the user has bought). Our simulations show that service interval variation is less than a few percent of the service interval itself, thus providing excellent delay guarantees to the user.

2 MuqPro Design

In the emerging gigabit-per-second networks, the sophisticated mechanisms that were prescribed above must operate at the cell rate which is several millions per second. This requires hardware support, in the form of a MuqPro ASIC. The interfaces of such a MuqPro chip, and its main functional blocks are shown in figure 2; we refer to this ASIC as *MuqPro II*, in order to distinguish it from MuqPro I, the partial-functionality, FPGA-based prototype that we are currently designing. MuqPro has two bidirectional ports, and handles bidirectional ATM traffic between these ports, with optional multi-queue buffering of the cells in both directions, and cell forwarding in an order that is potentially different from the arrival order. Each MuqPro port can be configured to operate

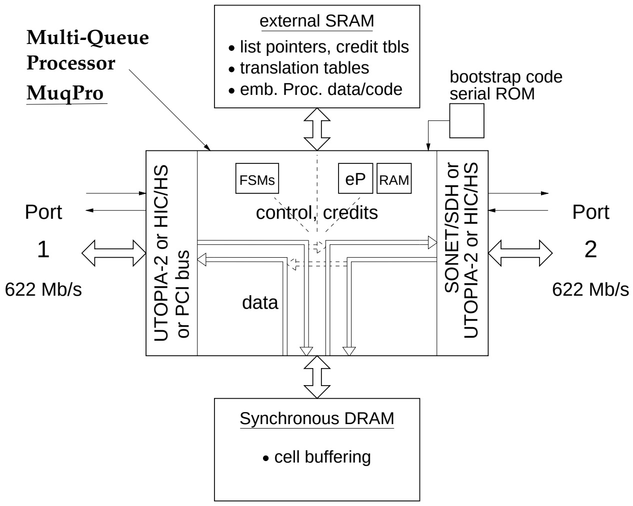


Figure 2: *MuqPro II overview and interfaces*

according to one of a number of different standards: HIC/HS, or UTOPIA-2; port-1 can also be configured to operate as a PCI bus port (32 bits). Both ports of MuqPro will operate at the SDH/STM4c or SONET/OC12 rate of 622 Mbits/s in each direction. When port-1 is configured as a PCI bus, rather than normal network port, it may be a slave device (cells written from the computer to the MuqPro chip), or as a master device (cells read by MuqPro from the computer's memory, via DMA); in the latter configuration, MuqPro will provide multiple DMA channels, with capability for protected DMA initiation at user-level, without OS kernel invocation overhead [4].

An important function of MuqPro II will be the provision of large buffer space, and large VP/VC translation tables; this requires high-density, off-chip memories. MuqPro will use external DRAM to store ATM cells, while it will use external SRAM to store management information: VP/VC translation tables, queue tables, credit tables, linked list pointers, data and code for the embedded processor, etc. Dynamic RAM is appropriate for cell storage, because large capacity at reduced cost is required, and because accesses are quite predictable: all accesses are to 53-byte blocks (better locality than random-word accessing), and cell addresses can be pre-calculated, thus allowing the use of pipelining. Management information, on the other hand, calls for static RAM: dynamic data structures require fast pointer chasing, so shorter access time is needed. The need for memory partitioning with different requirements has been identified by industry [5].

The cell-body datapath of the MuqPro II is simple, as illustrated in figure 2. Incoming cells are, in general, buffered in the DRAM, and outgoing cells originate from that same buffer memory. The option is provided, however, to forward cells directly from input to output, without buffering, in one or in both directions. Unbuffered forwarding in both directions appears when MuqPro is used merely for physical layer conversion (e.g. HIC to SONET/SDH). When buffering is performed in both directions, MuqPro requires a DRAM throughput of 4 times the link rate of 622 Mbps, i.e. 2.5 Gbps. When buffering is performed in one direction only, DRAM needs 1.24 Gbps of throughput. To achieve such high throughput, we envision the use of a couple synchronous DRAM chips;

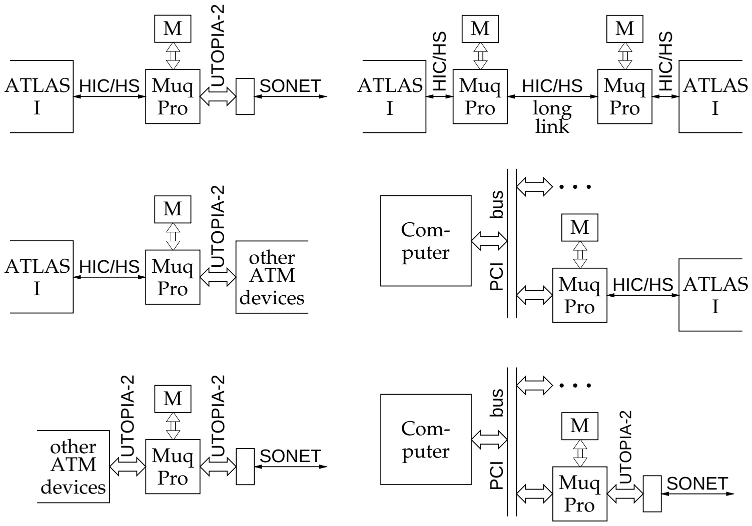
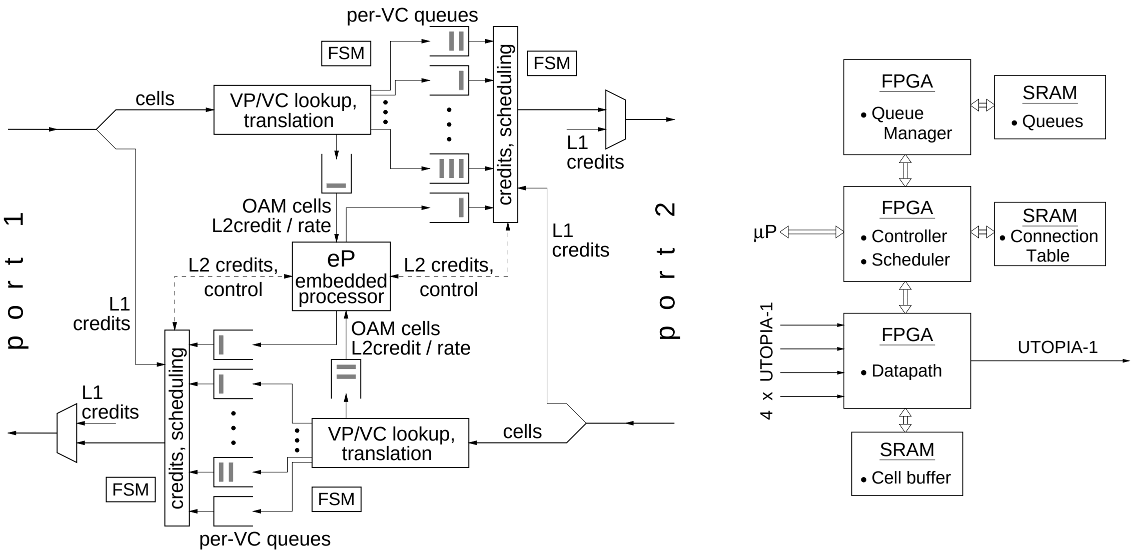


Figure 3: *Possible Uses of MuqPro II*

commercially available synchronous DRAM's offer a 16-bit wide off-chip data path, operating at up to 100 MHz, thus providing a peak throughput of 1.6 Gb/s and a sustained throughput of 1.2 Gb/s per chip.

Given the considerable port configuration flexibility of MuqPro II, figure 3 gives examples of its possible uses in various systems. The three left configurations demonstrate the use of MuqPro as a "bridge" between ATM subsystems using HIC/HS and/or UTOPIA-2 physical connections. The right column illustrates the use of MuqPro to accommodate long HIC/HS links, while the remaining two configurations show the use of MuqPro in network adapters to provide HIC/HS or SONET (or other ATM FORUM standardized physical protocol) connectivity.

Figure 4(a) shows a logical block diagram of the MuqPro II functions. The two directions of flow are symmetrical. First, the header of each incoming cell is inspected, determining its translation and the queue where it should be placed; queue selection implicitly determines whether the cell is to be forwarded out or is destined for the embedded processor (e.g. OAM cell, rate-based FC control cell, QFC cell). On the output side of the multiple queues, a sophisticated transmission scheduling circuit exists; its basic algorithm is described in section 3. This scheduler takes into consideration queue status (empty/non-empty), credit availability, VP/VC flow groups, and rate control information, in order to decide which cell to forward next to the output. MuqPro II will be able to handle two levels of credit information: level-1 (L1) is the ATLAS I "hardware" credit mechanism, while level-2 (L2) are the "long-distance, software" credits, like e.g. QFC credits [1]. Rate-based flow control will be handled by MuqPro as a special case of credit flow control: the scheduler ignores credit availability, and just schedules the servicing of VC's based on their service class weight factor. Low-level queue manipulation will be handled by hardware FSM's, while higher-level queue management tasks will be performed by the embedded processor. To get a rough idea of the required external-SRAM throughput, assume that each cell arrival or cell



(a) (b)

Figure 4: *(a) MuqPro II Log. Block Diagram; (b) MuqPro I: FPGA-based prototype*

departure event needs an average of 6 accesses to this SRAM. Then, the SRAM must support about 24 accesses (4 events) per cell-time (700 ns), i.e. one access every 30 ns approximately; this is a quite reasonable rate, and it actually leaves a good safety margin for some additional accesses.

Figure 4(b) shows a block diagram of *MuqPro I*, a 155-Mbps, partial-functionality, FPGA-based prototype that we are currently designing. It consists of 3 FPGA and 8 SRAM chips. Four incoming UTOPIA-1 links are provided, so that N such boards can form an $4 \times N$ switch. The cell buffer is implemented using SRAM for simplicity; in another prototype that we are building, synchronous DRAM is being used. Only a single-level of credit-based flow control, which is QFC compatible, is being included in MuqPro I. The scheduler will implement one or more of the selection algorithms described in section 3.

3 Queue Service Evaluation

In general, at any point in time, several different VCs will contend for the use of the MuqPro's outgoing link. These VCs represent various connections that have been promised various quality of service guarantees. For simplicity, we assume that the ATM service provider will divide the bandwidth of each link into a large number of "base-bandwidth" channels (muck like the ISDN channels); each user (VC) will purchase one or more of the base-bandwidth channels. We expect that the number of channels that each customer is allowed to purchase is chosen from a menu of –say– a dozen different service classes. For example, assume a 155 Mbps ATM link. The ATM service provider may divide this link into 2400 base-bandwidth channels of 64 Kbps each. Each customer may be given a choice of 64 Kbps (one base-bandwidth share), or 256 Kbps (4 shares), or 640 Kbps (10 shares), and so on.

At any point in time, several VCs that belong to different service classes may be ready to transmit. Choosing the one to transmit next is a complicated decision that must balance several factors:

- Each VC should get at least the bandwidth it has purchased.
- Successive services to the same VC (and hence service class) should be evenly spread out in time –otherwise, the traffic will appear bursty. Bursty traffic will impose higher pressure in the MuqPro's buffer space, and will increase the time each VC waits to be served.
- If the link is not 100% utilized, the extra bandwidth should be fairly divided among competing service classes, in proportion to the number of VC's in each and the weight factor of each VC.

We describe a scheduling framework in order to achieve the above goals.

3.1 Scheduling Algorithms

Assume that each service class i has N_i VCs. For each service class i there is a weight w_i associated with its VC's. The weight represents the number of base-bandwidth channels that each VC corresponds to. We assume the existence of a counter c_i for each service class i. The initial value of all counters is zero. At each cell time the following happens:

INCREMENT: For all i, counter i is incremented by $w_i \cdot N_i$.

SELECT: One non-negative counter is selected according to some algorithm. The counter is decremented by N, where $N = \sum_{i=1}^{n} w_i \cdot N_i$. The service class that corresponds to this counter is serviced: one ATM cell is selected for transmission from the next VC in the class, in a round-robin fashion.

We have considered four scheduling algorithms: Round-robin, Highest-value-first, Highest-priority-first, and Virtual Clock:

Highest-value-first (`HVF`): Select the counter with the highest value. This policy selects the counter that has accumulated more "tokens" than anyone else, and thus deserves to be served.

Round-Robin (`RR`): Select the next non-negative counter in a round-robin fashion. This policy attempts to achieve some fairness among all competing service classes.

Highest-priority-first (`HPF`): Select the highest priority class (where the priority of class i is its weight w_i) that has a non-negative counter. This policy favors high-priority classes even if there exist low-priority classes with more tokens.

Virtual Clock (`VCLOCK`): This is the algorithm defined in [6].

3.2 Performance Comparison of VC Scheduling Policies

3.2.1 Experimental Environment

We simulated a system with 6 service classes. Each service class contains a number of VCs, which always have a cell to transmit; A VC in class i has weight $w_i = 2^{i-1}$. The number of VCs in each class is a parameter of the experiments. At each cell time, the simulated algorithm chooses the service class that will transmit a cell; inside a class, VCs are served in a round-robin fashion.

3.2.2 Performance Metric

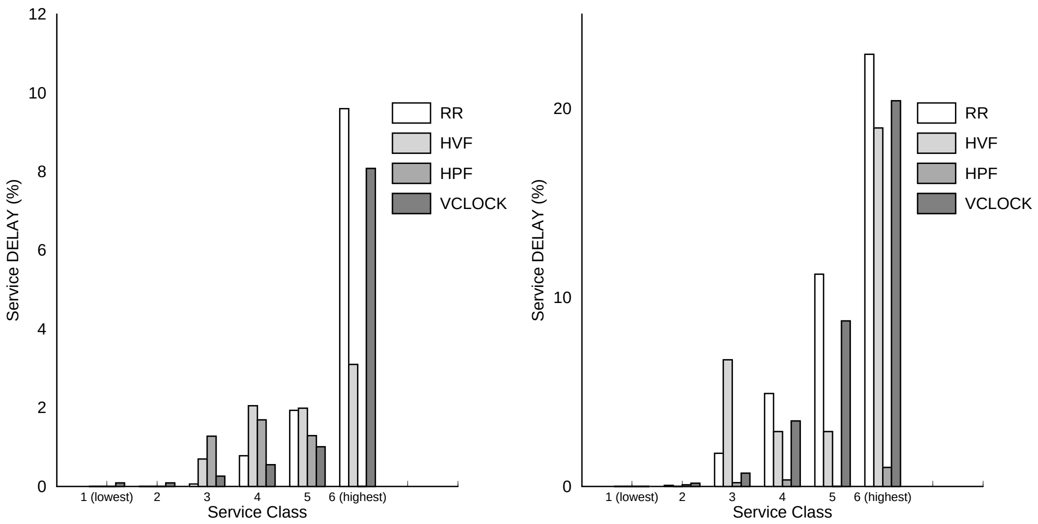
It can be easily proved that all the above scheduling algorithms distribute the entire link throughput to all serviced VC's in proportion to the weight factor of each. Thus, the throughput received by each VC is not an interesting performance metric, because all algorithms yield the correct ideal result. On the other hand, the algorithms differ from each other with respect to the *jitter* they introduce in the time intervals of the periodic services; we measure this jitter. In an ideal case each virtual circuit i would be serviced every avg_i cell times. However, since avg_i is not always an integer, and since each algorithm may give different priorities to different classes, the actual time between successive servings of a given VC will sometimes be shorter and sometimes longer than avg_i. Each time a VC transmits a cell, we measure the number of cell times that have elapsed since the last cell transmission of the same VC: suppose that $s_{i,j}$ is the number of cell times that have passed between the $(j-1)_{th}$ and the j_{th} cell transmissions of VC i. If $s_{i,j}$ is larger than avg_i, then $s_{i,j} - avg_i$ is the number of cell times that the service of VC i has been delayed. Similarly, $(s_{i,j} - avg_i)/avg_i$ is delay experienced by VC i as a percentage of its average service avg_i. We call the average (for non-negative $(s_{i,j} - avg_i)/avg_i$, over all j and over all VCs of the same service class) percentage delay as DELAY, and we use it as the performance metric in our studies.

Uniform Workload

In the first experiment we assume that each service class has a number of VCs that is a uniform random variable between 1 and 20. Figure 3.2.2(a) shows the DELAY for each service class. We see that HPF has the best performance for the VCs of the highest priority class (practically zero DELAY), and has no more than 2% DELAY for the rest of the classes. All the other policies have similar performance for low priority classes, but for the highest priority class they impose a DELAY as high as 10%.

Non-Uniform Workload

In the second experiment we assume that each service class has only one VC, with the exception of the lowest priority class, which has a number of VCs that is a random variable between 1 and 1000. This represents a system with a large number of low-priority VCs and a small number of



(a) (b)

Figure 5: *Scheduling performance: (a) uniform workload, (b) non-uniform workload*

high-priority VCs. Figure 3.2.2(b) shows the DELAY for each service class. `HPF` is always the best policy, having a DELAY less than 1%. The other policies may impose a DELAY as high as 20% for the highest priority class, and noticeable DELAYs for the lower priority classes. The reason is that all of `RR, HVF`, and `VCLOCK` tend to be fair to all service classes, and may delay the service of high priority VCs. Since high priority VCs need to transmit cells more frequently than other VCs, any delay in their service results in higher (percentage) DELAY.

4 Conclusions

Gigabit networking with advanced Quality-of-Service (QoS) guarantees is coming. The booming networking market will soon need gigabit-per-second speed, and will soon realize that QoS is of utmost importance. Within a few years, successful networks will demand per-connection guarantees for minimum throughput, maximum delay, instantaneous access to all available throughput for bursty traffic, and full utilization of the remaining capacity by low-priority traffic. We are working on VLSI components that will offer these. Based on OMI-developed HIC/HS gigabit links, we are implementing a single-chip ATM switch, ATLAS I, and we are planning an interface chip, MuqPro, that will be the building blocks for such networks. We outlined the architecture of MuqPro, the Multi-Queue Processor, which handles thousands of cell queues, and schedules cell departures according to a weighted round-robin algorithm that ensures fairness. We evaluated this algorithm by simulation, and showed that the service interval varies by no more than a few percent, thus yielding excellent delay guarantees to the user.

References

[1] Quantum Flow Control Alliance. Quantum Flow Control: A cell-relay protocol supporting an Available Bit Rate Service, July 1995. Version 2.0, http://www.qfc.org.

[2] J.C.R. Bennett, D.C. Stephens, and H. Zhang. High Speed, Scalable, and Accurate Implementation of Fair Queueing Algorithms in ATM Networks. In *Proceedings of International Conference on Network Protocols*, 1997. Available from ftp://ftp.cs.cmu.edu/user/hzhang/ICNP97.ps.Z.

[3] M. Katevenis and P. Vatsolaki. ATLAS I: A Single-Chip ATM Switch with HIC Links and Multi-Lane Back-Pressure. In *Proceedings, EMSYS 96 Conference (ESPRIT OMI 6th Annual Conference: Embedded Microprocessor Systems)*, pages 126–136, 1996. Berlin, Germany, IOS Press, ISBN 90 5199 300 5, available from file://ftp.ics.forth.gr/tech-reports/1996/1996.EMSYS96.ATLAS_I_ATMswitchHIC.ps.gz.

[4] E. P. Markatos and M. G.H. Katevenis. User-Level DMA without Operating System Kernel Modification. In *Proc. of the 3rd International Symposium on High Performance Computer Architecture*, pages 322–331, Feb 1997. URL: http://www.csi.forth.gr/proj/aavg/papers/1997.HPCA97.user_level_dma.ps.gz.

[5] H. Meleis and D. Serpanos. Designing Communication Subsystems for High-Speed Networks. *IEEE Network Magazine*, 6(4), July 1992.

[6] L. Zhang. Virtual Clock: A New Traffic Control Algorithm for Packet Switching Networks. *ACM Transactions on Computer Systems*, 9(2):101–124, May 1991.