Contention Resolution for Different Traffic Categories in Large Input Buffered ATM Switches

Andreas Kirstädter
Institute for Communication Networks
Technical University of Munich
Arcisstr. 21 80290 München, Germany
phone: +49–89–289–23510, fax: +49–89–289–23523
andreas@lkn.e–technik.tu–muenchen.de

Abstract

The current discussion about ABR and LAN emulation shows the important role that data traffic is assumed to play in future ATM networks. In order to correctly support its characteristics like high burstiness and asymmetric load patterns, ATM switches have to provide large input buffers together with effective and flexible contention resolution mechanisms. This paper first presents the principles of a distributed approach for contention resolution. Based on shared medium access the communication between the single input buffers is integrated with the prevention of cell collisions at the switch output ports. In the second part flexible extensions for the support of priorities and multicasting are discussed. The paper concludes with the results of a VHDL–based feasibility study using a 0.7 µm CMOS standard cell library.

1 Introduction

Considering the current discussion about and standardization efforts for ABR and LAN emulation it seems that the replacement and extension of legacy LANs will be the primary introduction strategy for ATM. The data communication applications prevalent in this area use transport–layer protocols like TCP that react very sensitive to cell losses [Roma 95]. The segmentation of TCP packets into ATM cells together with retransmission strategies like go–back–N waste a lot of bandwidth by the transmission of already useless packets in the case of cell losses. The bursty nature of LAN traffic together with highly asymmetric client–server scenarios complicate these problems even more.

It has been shown for example [Kaly 96] that with current transport and ATM layer protocols like TCP, UBR, and ABR these problems can only be avoided by providing enough buffer capacity in the ATM switches. In the case of ABR necessary amounts of buffering are in the order of several times the round trip time of the corresponding TCP connections. For UBR the required buffer length even rises with the number of TCP sources: it becomes more or less equal to the product of the number of sources and their TCP receiver windows.

2 Distributed Contention Resolution

Shared memory and output buffered switching modules alone are not able to provide the above mentioned amounts of buffering since the size of on–chip buffers is limited by economic and feasibility reasons. Input buffers can be easily implemented with standard D–RAMs up to arbitrary sizes since they only have to work at line speed. But they have to be coordinated: i.e. at each T_{cell} it has to be decided again, which of the contending input buffers is allowed to send a cell to a certain output. The necessary contention resolution mechanism has not only to arbitrate as fair and waste–free as possible. It also has to offer efficient and flexible mechanisms for handling multicast and prioritized traffic and to support sub–buffering or similar methods to prevent head–of–line blocking. Last but not least it has to be scaleable since large switching modules have to be coordinated in order to limit excess hardware expenses and bandwidth limitations resulting from multistage arrangements.

A number of different mechanisms have been proposed for contention resolution (e.g. [Obar 92], [LaMa 94], [Main 95], or [Ande 93]). Centralized schedulers and the output scheduling often suffer from limited scalability or performance. The reason behind this is that in addition to the decision process itself a large amount of information (about the state of all sub–buffers within all input buffers) has to be transferred between the scheduler(s) and the input buffer controllers at each cell duration.

A new distributed approach has been introduced in [Kirs 96] that (similar to the well known ring reservation [Bing 88]) integrates the tasks of information transfer and evaluation (figure 2–1): the authoriza-

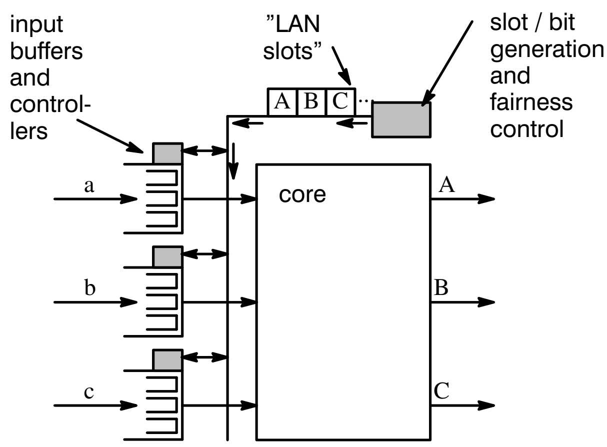


Figure 2–1: Principle of LAN–based contention resolution.

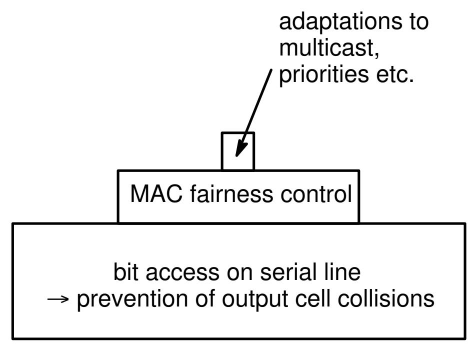


Figure 2–2: Layers of LAN–based contention resolution.

tion for sending a cell into the core is tied to the successful setting of a certain bit position on a serial line that like a LAN connects all input buffer controllers (IBCs). The fairness between contending inputs is then guaranteed by adapted medium access control (MAC) protocols from the LAN domain. Especially designed for distributed fairness control they are ideally suited for this task in the above architecture.

The contention resolution is now split into three layers (figure 2–2). Most of the coordination is done by the basic bit access. Fairness control and the adaptation to different traffic categories (see section 3 below) can thus operate at much lower frequencies, leading to economic feasible implementations even at very large switch module sizes. With this kind of contention resolution sub–buffering can be used without increasing the coordination traffic.

Requirements like low bandwidth consumption, low processing overhead, and slotted transmission resulted in using a fairness mechanism derived from high–speed LANs like the Orwell Ring or the IBM MetaRing [Cido 93] in this project (see [Kirs 96] for a more general discussion of these requirements and the description / comparison of other MAC protocols).

It assigns per–output transmit quotas to the IBCs. Its corresponding quota counter is then decremented by 1 each time the IBC successfully sets a bit position for an output on the serial line B (figure 2–3).

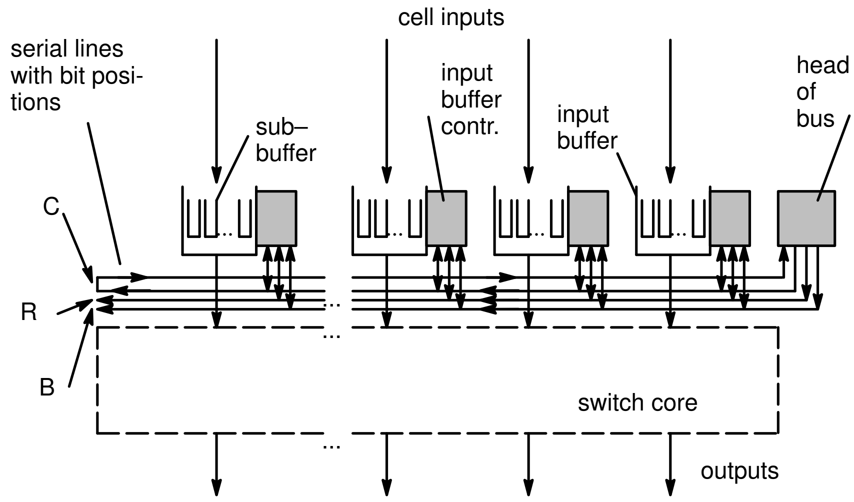


Figure 2–3: Architecture for quota–based fairness mechanism.

The quotas belonging to an output are refreshed by the head–of–bus (HOB) that sets a corresponding bit on the serial line R to 0 if the quotas of all willing stations (i.e. those that have at least one cell to this output in their buffers) are exhausted. This state is signaled to the HOB by means of the line C: the corresponding bit is initialized to 0 by the HOB; willing stations that still have a quota greater than zero to the considered output set this bit to 1. Thus a C–bit arriving back with the state 0 indicates the HOB that the quota counters need a refresh.

In time–discrete simulations at the cell level it has been shown [Kirs 96] that this approach is able to provide absolute round–robin fairness even in the case of highly asymmetrical load patterns (figure 2–4). Furthermore the throughput–delay performance of input sub–buffering with shared–medium

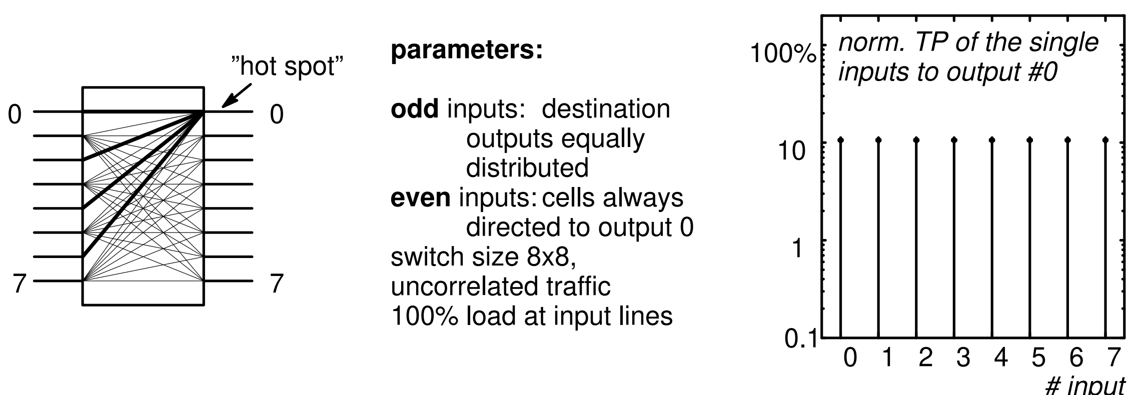


Figure 2–4: Fairness at asymm. load pattern.

contention resolution gets very close to that of an ideal (output buffered) switch as shown in figure 2–5.

3 Contention Resolution for Different Traffic Categories

The main purpose of this paper is to show how this approach can easily be extended to support different traffic categories.

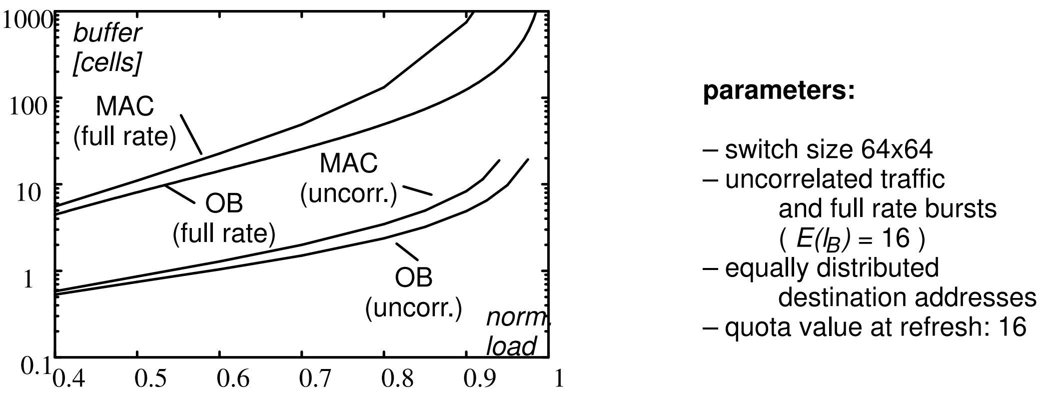


Figure 2–5: Delay performance of MAC–based CR versus ideal switch.

3.1 Per–VC Fairness

Contention resolution has been mainly regarded as a problem of distributing throughput fairly to all requesting input links. But it may be argued that the throughput of a single user must only be a fair share of the common bottleneck. Especially it has to be independent of the pattern how the contending users (VCs) are distributed over the inputs of the bottleneck switch. Figure 3–1 shows the case where 4 VCs

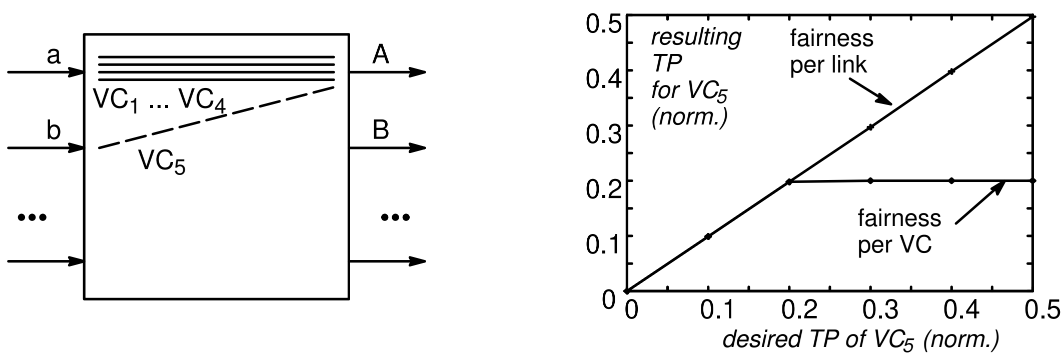


Figure 3–1: Traffic pattern and resulting throughput for per–VC fairness.

(VC_1 ... VC_4) on input *a* share the bottleneck output *A* with VC_5 on input *b*. With conventional per–link fairness VC_5 gets up to 50% of the total throughput during overload phases on *A*.

The quota–based fairness mechanism can be extended to provide also per–VC fairness by simply setting the refresh values of the quotas at the single inputs proportional to the number of VCs active at the corresponding input. The sub–buffers of the single VCs at one input are then served in a round–robin policy. Except for the administrative overhead for per–VC buffering nearly no additional hardware is required. Figure 3–1 also shows the perfect per–VC fairness of this distributed approach (reaching exactly the same quality of fairness as the centralized solution in [Lund 96]): VC_5 does not get more throughput at overload than the other 4 VCs.

3.2 Priorities

The shared–media based contention resolution allows the implementation of different priority levels by simply splitting and reconnecting the lines C and R (figure 3–2). Since line B is left intact, the IBCs are split into priority groups: fairness is provided between the members of a group (between the interruption points) while a clear top–down priority ranking exists between the groups.

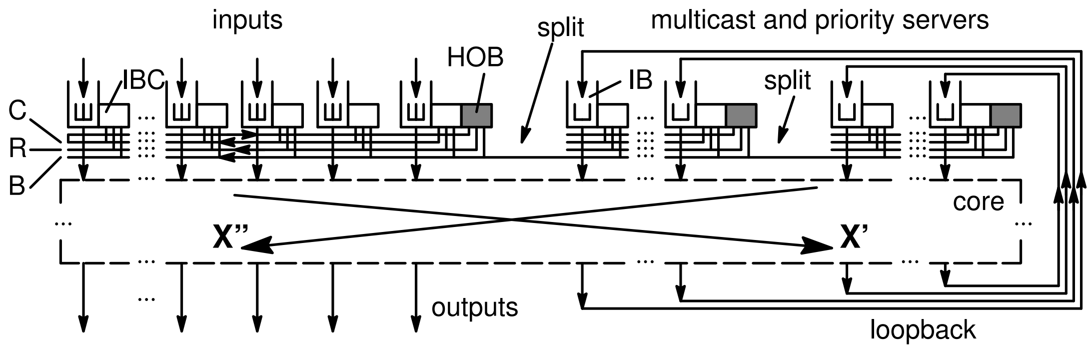


Figure 3–2: Server–based contention resolution for different traffic categories.

This mechanism can now be used within a server–based architecture for the handling of an arbitrary number of priority levels: arriving cells are stored in the buffers at the switch inputs. Cells of the lowest priority level are directly routed to their destination outputs (if the corresponding LAN–slots are still unoccupied). Higher prioritized cells are stored in different sub–buffers at the inputs. By routing them to special priority servers located near the bus head end (path X" in figure 3–2) they can set the necessary bit positions before the lower priorities get a chance to do so. These servers are arranged into groups according to the expected traffic within the different priority levels. Since the access (from the input buffers) to the servers is controlled by designated bit positions on the serial lines, each server can control its own input load. This server backpressure not only allows the usage of small buffers within the servers. It also enables any reconfiguration of the priority groups at run–time.

3.3 Multicast

Many different applications like signaling (e.g. for address resolution), video distribution and multimedia conferencing require multicasting within ATM networks with a vast range of bandwidth and fanout scenarios that have to be supported by the switching architecture. One possibility to handle cell multicasting within the switches is to duplicate the cells at the switch inputs and feed them serially into the core (**time multicasting**, figure 3–3).

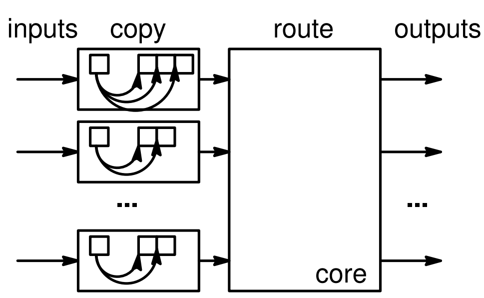


Figure 3–3: Time multicasting.

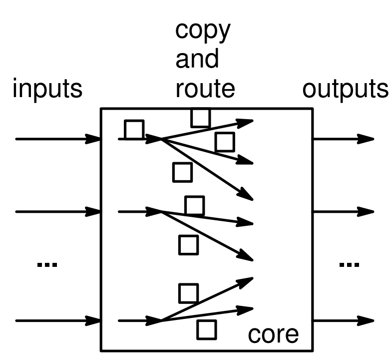


Figure 3–4: Space multicasting.

Since the resulting cell copies are treated like normal unicast cells neither the switch core nor the contention resolution mechanism have to be modified. But the time multicasting principle gets quite inefficient at asymmetrical load patterns. The simulation results in figures 3–5 and 3–6 show for a generic time multicast switch ($N = 16$ inputs and outputs) the dependency of the mean cell delay on the normalized total switch throughput at different mean fanout values of $F = 2$, 8, and 12. An uncorrelated and

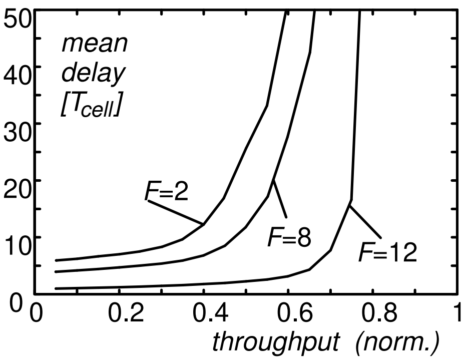


Figure 3–5: Time multicast at symm. input loading.

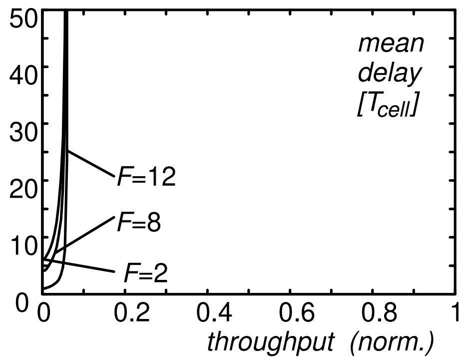


Figure 3–6: Time multicast at asymm. input loading.

symmetric model for the generation of multicast copies had been used: i.e. for each generated input cell the destination outputs were determined individually with equal probability P. P had been calculated such that the resulting bernoulli distribution of copies had a mean value of F. Contention resolution was achieved by the above mentioned LAN–based mechanism. In the case of asymmetric input loading (cells were only fed into one switch input, figure 3–6) the achievable switch throughput is limited to $1/N$.

The other more interesting alternative is to copy the cells within the switch core itself (**space multicasting**, figure 3–4). Then a single switch input can no longer be the bottleneck of the whole system. Many cell–duplicating switch core architectures have been presented for this purpose in the past. But often the contention resolution (CR) problems arising from the varying duplication patterns have been neglected. Since the copies of the multicast cells now are only generated within the switch core, the CR mechanism now has to consider up to N destination addresses per cell (instead of one per cell at unicast). This gets even more complicated by the effects of fanout–splitting: often all the desired copies of a MC cell cannot be generated within a single T_{cell} because some of the output ports are being used by other (multicast) cells during the same T_{cell}. Thus centralized CR architectures can no longer be used because of the strongly increased information flow between the IBCs and the CR processor.

The **server–based** extension of the shared–medium based contention resolution shown above offers a convenient way of combining cell duplication with a complete contention resolution. According to the expected throughput and priority requirements some of the uppermost priority servers become multicast servers: before sending the multicast cell into the core they set as many bit positions as required by the fanout of the cell that then can be duplicated within the core without any output port collisions. Figure 3–7 shows the simulation model of the architecture that is investigated in the rest of this section: in order to assure an equally distributed loading of the N_{MC} servers each of the $N=16$ input ports receives its MC cells from N_{MC} independent sources (N_{MC} = 1...8). Cell delay of a MC copy is measured from the point the MC cell enters an input buffer until the time the copy leaves one of the output ports. It therefore includes the waiting time in both the input buffer and the buffer of the MC server and the increased service time in the MC server because of fanout–splittings.

The **optimal number of multicast servers** in this architecture depends not only on the delay (and budget) requirements but also on the expected fanout F of the multicast load. The mean cell delay at different numbers (N_{MC} = 1, 2, 4, and 8) of MC servers is shown in the figures 3–8 (F = 4) and 3–9 (F = 12). As it could be expected, an increased number of MC servers leads to a greater reduction of delay in the case of small fanouts. Since each MC server tries to emit a mean number of F copies per T_{cell} it is a good guess to provide $N_{MC} = N/F$ servers. Especially at at large F the performance improvement of a larger number of servers is also always impaired by an increased rate of fanout–splittings.

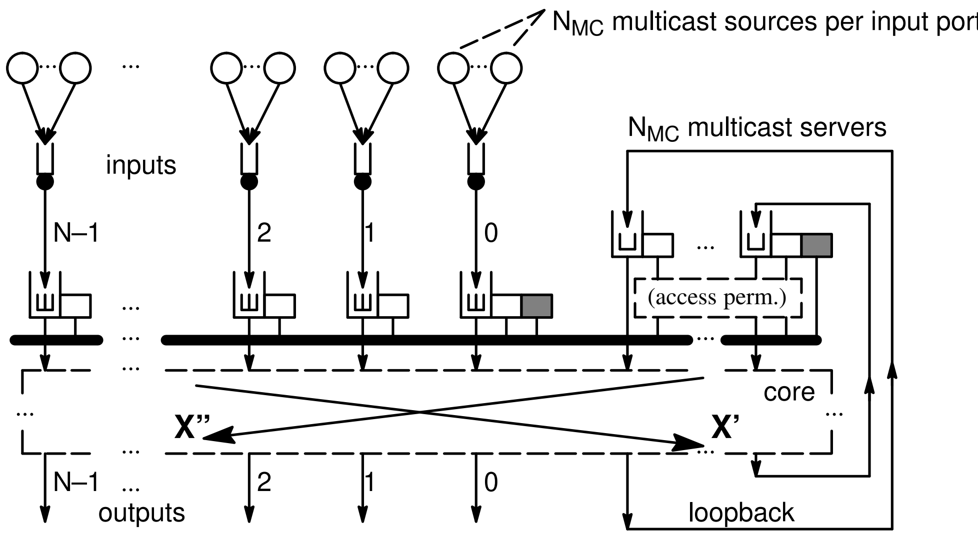


Figure 3–7: Simulation model for multicast behavior.

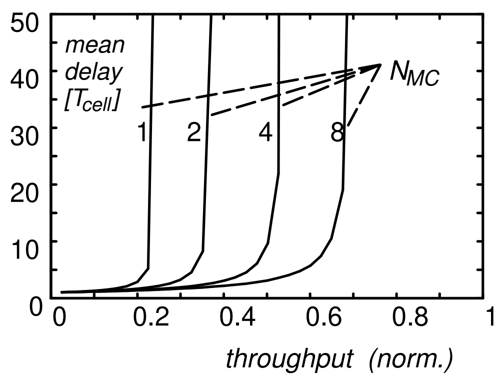


Figure 3–8: Influence of N_{MC} (F=4).

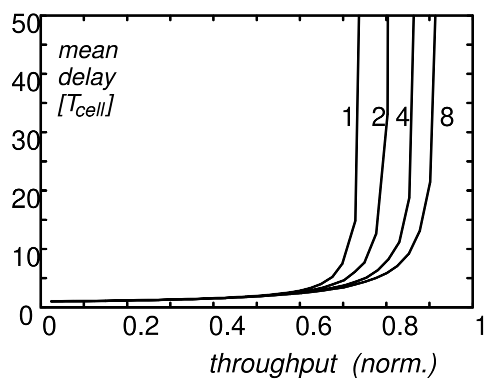
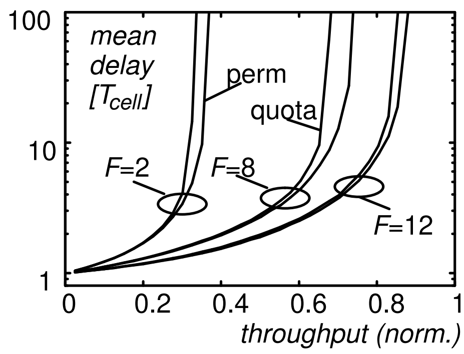


Figure 3–9: Influence of N_{MC} (F=12).

In the case that several multicast servers co–exist within a priority group, the fairness between these servers can principally be controlled either by **permutating** (by hardware) their access to the lines B, R, and C from T_{cell} to T_{cell}. Or the **quota–based** fairness mechanism is applied also to the multicast servers. Figure 3–10 shows (for N_{MC} = 4) that the second alternative nearly performs as well as the server permutation. The reason behind the small difference is that the linear server arrangement with quota–based fairness tends to favor upstream servers in the case of fanout–splittings. At low and medium server loads MC cells arriving at upstream servers more often see more free slots for their copies than those at the downstream ones. The head–of–line cells in the downstream servers then have to wait until the fairness mechanism provides them with a sending opportunity, before they can emit their remaining fanout. Although at this time the upstream servers have to deferred their access they often have already sent all their copies and thus are not affected. Thus downstream servers generally experience larger service times T_s at low and medium server loads while upstream servers may operate better than in the case of server permutation. At high loads always some of the servers must defer and the effect vanishes. This effect slightly increases the mean as well as the variance of T_s at the downstream servers although (at the same time) the quota mechanism still equally distributes the capacity of the single outputs fairly between contending servers. Figure 3–11 shows the distributions of T_s at the servers with the indexes 0 (near the head of bus), 3, and 7 (N_{MC} = 8) for a mean fanout of F = 2 and a total switch



Parameters:

switch size:	16x16
number of MC servers:	4
mean fanout:	2, 8, 12
quota after refresh:	1

Figure 3–10: Server permutation versus quota–based fairness.

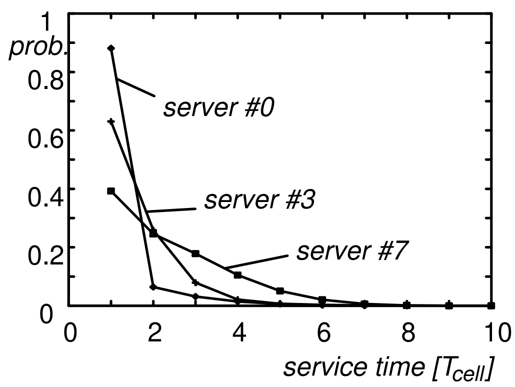


Figure 3–11: Distribution of T_s for different servers.

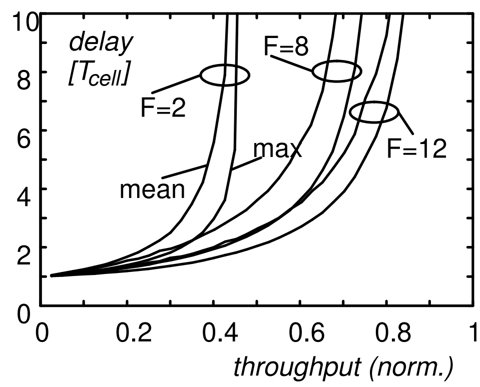


Figure 3–12: Resulting delay behavior.

throughput of 0.5. Higher service times T_s together with a larger variance of the service process increase the delay experienced by MC cells in the buffers of the downstream MC servers. Figure 3–12 shows the differences between the mean delay value of all servers and the delay of the worst–case (i.e. last) server. Since the mean delay values of a single server depend on T_s (and the resulting effective MC load ϱ_{eff}) in a nonlinear manner (equation 3–1), the increased delay of the downstream servers is not fully

$$delay = \text{const} \cdot \frac{\varrho_{eff}^2}{1-\varrho_{eff}} = \text{const} \cdot \frac{(\varrho \cdot T_s)^2}{1-\varrho \cdot T_s} \qquad (3\text{–}1)$$

compensated by the advantage of the upstream servers. Thus the mean delay of the quota–based fairness mechanism is slightly higher than with the server permutation.

4 Implementation and Scalability

The generic blocks within an input buffer controller are shown in figure 4–1. A multiplexer and flipflop block separates the high speed bit access control (HS–core) part from the rest of the main components. Figure 4–2 shows a design for the HS–core that resulted from a VHDL–based logic synthesis. Apart from the flipflops for the synchronous handling of the serial lines it mainly consists of a few gates responsible for the correct and fast reaction at the quota refreshs and the setting / reading of bit positions. Used by all MAC FSMs in common, it can easily be optimized for high speed in a full custom design.

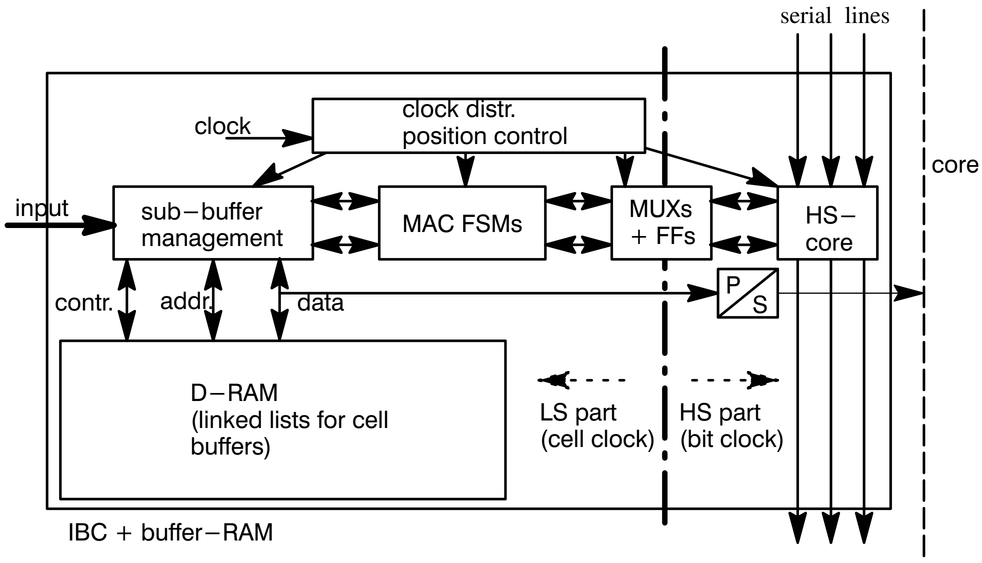


Figure 4–1: Architecture of an input buffer controller.

Figure 4–2: High–speed part of an input buffer controller.

The other components (including the MAC FSMs) operate only on the cell level so that area and power optimized standard cell techniques can be used for the greatest part of the IBC.

The main parts of this shared–medium based contention resolution architecture are currently being investigated by CMOS design techniques. It has been shown [Kirs 97] that using a 0.7 µm CMOS standard cell library and VHDL–based logic synthesis the input buffer controllers operate correctly up to a bit rate of 150 Mbit/s on the serial lines B, R, and C. Thus either up to 200 switch ports with a speed of 155 MBit/s or 50 ports at a speed of 622 Mbit/s can be coordinated leading to a total switch throughput of 30 Gbit/s. It can be expected that the usage of a state–of–the art BICMOS technology for the speed–critical parts together with full custom design techniques will allow switch throughputs in the Terabit/s range. In the case of a 32x32 switch about 7 mm^2 of chip area (including 4 mm^2 for the buffer length counters) are required to integrate the MAC–based CR circuitry on a buffer controller ASIC.

The author wishes to thank Mr. Werner Biebl for his simulations of the multicast and per–VC–fairness performance.

References:

[Ande 93] Anderson, Thomas E. et al.: "High Speed Switch Scheduling for Local Area Networks" in: ACM Transactions on Computer Systems, Vol. 11, No. 4, Nov. 1993, S. 319 – 352

[Bing 88] Bingham, B. / Bussey, H.: "Reservation–Based Contention Resolution Mechanism For Batcher–Banyan Packet Switches", in: Electronics Letters, Vol. 24, No. 13, pp. 772 – 773

[Cido 93] Cidon, Israel / Ofek, Yoram: "MetaRing – A Full–Duplex Ring with Fairness and Spatial Reuse", in: IEEE Transactions on Communications, Vol. 41, No. 1, Jan. 1993, pp. 110 – 120

[Kaly 96] Kalyanaraman, Shiv / Jain, Raj et al.: "Performance of TCP / IP over ABR Service on ATM Networks", in: IEEE GLOBECOM '96

[Kirs 96] A. Kirstadter: "Fair and Flexible Contention Resolution for Input Buffered ATM Switches Based on LAN Medium Access Control Protocols", in: Proceedings of 1996 Int. IFIP IEEE Conference on Broadband Communications, pp. 370 – 381, Montreal, 1996.

[Kirs 97] A. Kirstadter, T. H. Nguyen: "Distributed Contention Resolution in Large Input Buffered ATM Switches using Adapted LAN Medium Access Control Mechanisms" in: Proceedings of 1997 Asia Pacific Symposium on Information and Telecommunication Technologies, Hanoi, March 12–14, 1997.

[LaMa 94] LaMaire Richard O. / Serpanos, Dimitrios N.: "Two–Dimensional Round–Robin Schedulers for Packet Switches with Multiple Input Queues", in: IEEE/ACM Transactions on Networking Vol. 2, No. 5, Oct. 1994, pp. 471 – 482

[Lund 96] Lund, Carsten / Phillips, Steven / Reingold, Nick: "Fair Prioritized Scheduling in an Input Buffered Switch", in: Proceedings of 1996 Int. IFIP IEEE Conference on Broadband Communications 96, pp. 358 – 369, Montreal, 24. – 26. April 1996

[Main 95] Main, Jonathan / Sarkies, Ken: "Cell Scheduling Using Status Arrays in Input Buffered ATM Switches" in: Proceedings of the first IEEE Workshop on Broadband Switching Systems, Poznan, Poland, April 19 – 21, 1995

[Obar 92] Obara, H. / Hamazumi, Y.: "Parallel Contention Resolution Control for Input Queueing ATM Switches", in: Electronics Letters, Vol. 28, No. 9, 23rd April 1992, pp. 838 – 839

[Roma 95] Romanow, Allyn / Floyd, Sally: "Dynamics of TCP Traffic over ATM Networks" in: IEEE Journal on Selected Areas in Comm., Vol. 13, No. 4, May 1995, pp. 633 – 641