BUFFER DESIGN FOR MULTICLASS TRAFFIC IN ATM NETWORKS

by

G. Kesidis and J. Walrand

Memorandum No. UCB/ERL M92/45

7 May 1992

BUFFER DESIGN FOR MULTICLASS TRAFFIC IN ATM NETWORKS

by

G. Kesidis and J. Walrand

Memorandum No. UCB/ERL M92/45

7 May 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

BUFFER DESIGN FOR MULTICLASS TRAFFIC IN ATM NETWORKS

by

G. Kesidis and J. Walrand

Memorandum No. UCB/ERL M92/45

7 May 1992

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Buffer Design for Multiclass Traffic in ATM Networks*

G. Kesidis[1,2] J. Walrand[1]

1. EECS Dept., University of California, Berkeley, CA94720.
2. Electrical Eng. Dept., University of Waterloo, Waterloo, ON, Canada, N2L 3G1.

*Supported by: NSERC of Canada, Pacific Bell, Micro Grant of the State of California, ARO Grant Number DAAL03-89-K-0128

Abstract

In ATM, buffering is required to reduce cell loss and increase bandwidth utilization in the presence of bursty traffic. Several types of traffic with different performance requirements (e.g., on delay, loss) will share buffer resources. Buffering disciplines that can satisfy different cell loss requirements (loss priority) are described. A shared buffer/linked list implementation is described and is compared to a segregated buffer approach in terms of speed (or fan-in), flexibility to handle future applications, and buffer utilization.

1 Introduction

Several kinds of traffic, consisting of streams of 53-byte cells on time-slotted channels, will share the resources of an ATM network. For example, voice and data will share the buffers in the network's switches. Since the traffic is bursty, buffering is required to achieve an extremely small cell loss probability and an efficient bandwidth utilization.

Different kinds of traffic will have different performance requirements from the network. For instance, a voice call may require a small end-to-end delay (delay sensitive) and may tolerate significant cell loss[1]. A data call, however, may require an extremely small cell loss probability (loss sensitive) and may tolerate significant delay. The purpose of this paper is to describe buffering and service disciplines and buffer allocation methods that can achieve the performance requirements of a mixture of different traffic streams.

Consider the case of two traffic streams multiplexed onto a time-slotted channel (fiber) with a bandwidth of c cells/s . One stream (data) is more loss sensitive that the other, and the other stream (voice) is more delay sensitive. Let λ_v (resp. λ_d) be the average cell arrival rate of the stationary voice (resp. data) traffic stream. The buffering and service disciplines described below can be modified to handle more than two traffic "types" (traffic statistics and sensitivities) in obvious ways.

[1]Voice is actually an isochronous traffic stream requiring small delay *jitter*, but by reducing the maximum end-to-end delay, jitter can be eliminated by a small buffer at the destination.

2 Buffering Disciplines for Loss Sensitive Traffic

We now briefly describe two buffering schemes to satisfy loss sensitive traffic constraints. Voice and data traffic share a buffer of size B cells in the first two (sharing) schemes.

2.1 Partial Buffer Sharing

A threshold b is chosen ($0 < b < B$) and voice cells can occupy at most b spots in the buffer. That is, an arriving voice cell is discarded if either the buffer contains b voice cells or the buffer is full. On the other hand, an arriving data cell is discarded only if the buffer is full [5]. In Figure 1 the shaded area represents the allowable states of buffer occupancy.

Note that, if the service discipline is FCFS, then this buffer can be simply implemented with a FIFO. Also note that the buffer occupancy process of partial buffer sharing does not agree with that of a regular buffer with the same (coupled) arrivals: partial buffer sharing is not work conserving.

2.2 Shared Buffer with Bumping Rules

In this scheme, an arriving data cell is discarded only if the buffer is full and the buffer contains no voice cell. If a data cell arrives to a full buffer with voice cells, it will replace, or *bump*, a voice cell out of the buffer; the bumped voice cell is discarded. An arriving voice cell is discarded if the buffer is full. We assume that the server is never idle when cells are queued in the buffer.

We can express the cell loss probability, due to buffer overflow, of the voice and data streams under the bumping rule (F_v^{b} and F_d^{b}, "b" for bumping) in terms of the cell loss probabilities without the bumping rule (F_v and F_d for a regular buffer). Let the cell loss probability for the aggregate traffic be F^{b} and F. Since the buffer occupancy process with or without the bumping rule is the same given the same arrivals (work conserving), $F^{\mathrm{b}} = F$. For the shared regular buffer (without bumping), let L_d be the average number of data cells that are lost in a busy cycle of the regular buffer occupancy, and let C_d be the average number of data cells that arrive in a busy cycle of the regular buffer occupancy. Similarly define L_v and C_v for voice, and L_v^{b}, L_d^{b}, C_v^{b}, and C_d^{b} under the bumping rule. Finally, let F_d^* be the cell loss probability of the data stream if there were *no* voice traffic at all through the regular buffer (i.e. $\lambda_v = 0$).

By Neveu's cycle formula (the "inversion" formula in [1]):

$$F = \frac{L_v + L_d}{C_v + C_d}, \quad F_v = \frac{L_v}{C_v}, \quad F_d = \frac{L_d}{C_d}, \quad \text{and} \quad \frac{C_v}{\lambda_v} = \frac{C_d}{\lambda_d}.$$

Claim 1:

$$F_v \; < \; F_v^{\mathrm{b}} \; < \; F_v + \frac{\lambda_d}{\lambda_v} F_d \quad \text{and} \quad F_d \; > \; F_d^{\mathrm{b}} \; \geq \; F_d^*.$$

Proof: Clearly the bumping rule increases the cell loss probability for voice and decreases that of data. Also, F_d^* is the cell loss probability of data when the data traffic is completely uninhibited by the voice traffic in the sense of buffer occupancy. Thus the cell loss probability for data can be no smaller than F_d^*. What remains to show is that $F_v < F_v + \frac{\lambda_d}{\lambda_v} F_d$.

Consider a typical busy cycle of the buffer occupancy process. Clearly, $C_v^{\mathrm{b}} = C_v$ and $C_d^{\mathrm{b}} = C_d$ (work conservation). Thus

$$F_v^{\mathrm{b}} = \frac{L_v^{\mathrm{b}}}{C_v^{\mathrm{b}}} = \frac{L_v + \epsilon}{C_v},$$

where ϵ represents the average number of voice cells lost due to bumping by data cells in a busy cycle. Since $\epsilon < L_d$,

$$F_v^{\mathrm{b}} < \frac{L_v}{C_v} + \frac{L_d}{C_v} = F_v + \frac{L_d}{\frac{\lambda_v}{\lambda_d} C_d} = F_v + \frac{\lambda_d}{\lambda_v} F_d,$$

as desired.

—

Note that if $F_d << F_v$ and B is large, $\epsilon \approx L_d$ (i.e. an arriving data cell to a full buffer will very likely find a voice cell to bump). Thus

$$F_d^{\mathrm{b}} \approx F_d^* \quad \text{and} \quad F_v^{\mathrm{b}} \approx F_v + \frac{\lambda_d}{\lambda_v} F_d. \tag{1}$$

On the other hand if B is small and the service discipline heavily favors voice, then $\epsilon \not\approx L_d$ to that the approximations above (1) do not hold. This is so because if the service discipline heavily favors voice, the sojourn time of voice cells will decrease significantly so that the average number of voice cells in the queue will decrease by Little's theorem [9] (fewer voice cells to bump). See the claim in section 3.2 for a more precise statement of this phenomenon.

· The benefit of using bumping over partial buffer sharing is the existence of algorithms that can estimate F_v and F_d in real time [2] as well as the existence of equivalent bandwidth formulas, for certain models, which allows for immediate computation of F_v and F_d (see [4] for an equivalent bandwidth formula for multi-class, on-off Markov fluids sharing a buffer). Equation (1) allows us to use these results to estimate F_d^{b} and F_v^{b}.

We can also define partial bumping schemes whereby an arriving data cell, to a full buffer, will bump a voice cell with some probability less than one. Bumping rules cannot be implemented by a FIFO. A linked list approach is required instead.

2.3 Segregated Buffers

In the segregated buffer scheme, voice and data occupy logically separate (segregated) buffers of size B_v and B_d respectively. The server is shared between the two buffers. A voice cell is lost if it finds a full voice buffer upon arrival and likewise for a data cell.

3 Service Disciplines for Delay and Loss Sensitive Traffic

We now describe nonpreemptive ways to satisfy delay sensitive traffic constraints. Voice and data traffic share a buffer of size B cells. Note that the cells of a virtual circuit in ATM may not be delivered to the destination in an order different from the one in which they were received by the network from the source.

3.1 Alternating Voice and Data Service

In this service scheme [7], up to cT_v voice cells are initially served (cT_v or until the buffer is exhausted of voice cells). After this, up to cT_d data cells (or data *and* voice cells, according to FCFS) are served, and then up to cT_v voice cells are served, and so on in an alternating manner. The operation of this simple round-robin service protocol would be suspended in order to serve signaling cells as they arrive.

3.2 Budding Rules

In this scheme, in every service epoch an independent Bernoulli random variable δ with parameter p_v ($0 < p_v < 1$) is generated. That is, $\mathrm{P}\{\delta = 1\} = p_v$ and $\mathrm{P}\{\delta = 0\} = 1 - p_v =: p_d$. If both voice and data cells are queued in the buffer and the outcome is $\delta = 1$ then a voice cell is served. If both voice and data cells are queued in the buffer and the outcome is $\delta = 0$ then a voice cell is served. If only voice (resp. data) cells are queued, then a voice (resp. data) cell is served.

Consider a bumping rule shared buffer using this budding rule.

Claim 2:

$F_d^{\mathrm{b}}(p_d)$ is a nonincreasing function of p_d.

Proof: See Appendix A.

A particular case of the budding rule is when buffered voice cells are *always* served before any buffered data cells ($p_v = 1$). Assuming that the voice and data streams are Poisson and the server is FCFS, one can derive expressions for the average queuing delay until service, Δ, for the budding rule using Little's theorem (see [10], p. 385):

$$\Delta_v = \frac{(\lambda_d + \lambda_v)c^2}{2(1 - \frac{\lambda_d+\lambda_v}{c})} \text{ and } \Delta_d = \frac{(\lambda_d + \lambda_v)c^2}{2(1 - \frac{\lambda_d+\lambda_v}{c})(1 - \frac{\lambda_v}{c})}.$$

These expressions can be generalized to M/GI/1 queues.

Budding is clearly work conserving and requires a linked list type of implementation. Given two traffic types with the same delay sensitivity, an equitable round-robin service discipline could be used.

3.3 Budding Rules for Delay and Loss Sensitive Traffic

This scheme is a generalization of the previous one wherein p_d is any nondecreasing function of the number of data cells queued. Data would be given almost all of the service bandwidth ($p_d \approx 1$) only when a very large number of data cells are queued. Otherwise $p_v > p_d$. The previous claim has a trivial generalization.

Claim 3:

For a bumping rule, shared buffer, if p_d and $\hat{p}_d$ are nonincreasing functions of the number of voice cells queued and $p_d \geq \hat{p}_d$ then $F_d^{\mathrm{b}}(p_d) \leq F_d^{\mathrm{b}}(\hat{p}_d)$.

We conducted the following simulations to illustrate this claim. The data and voice traffic were each modeled as independent on-off Markov-modulated Poisson processes (MMPPs). Each such process is specified by three parameters: q^{on}, q^{off}, and Λ. For each MMPP, a continuous time, two-state ("on" and "off") Markov process is simulated. The amount of time spent in the on (resp. off) state, before transition to the off (resp. on) state, is an exponential random variable with mean $1/q^{\mathrm{on}}$ (resp. $1/q^{\mathrm{off}}$). When in the off state, no cells are generated. When in the on state, however, cells are generated according to a Poisson process with rate Λ. Note that the average cell arrival rates are

$$\lambda_i = \frac{q_i^{\mathrm{off}}}{q_i^{\mathrm{off}} + q_i^{\mathrm{on}}} \Lambda_i$$

where $i = v, d$. These parameters were chosen so that the mean traffic intensity $(\lambda_v + \lambda_d)/c = 0.85$. Also, in order to make cell accumulation in the buffer possible, $\Lambda_v + \Lambda_d > c$.

We conducted several simulations using traffic parameters satisfying the above conditions with $p_v = 1 - p_d$ and

$$p_d = \begin{cases} 0.33 & \text{if } X_d < X_d^{\mathrm{max}} - b \\ 1 & \text{else} \end{cases},$$

where X_d is the number of data cells queued and X_d^{max} is the maximum number of data cells queued possible ($X_d^{\max}$ equals B^{sh} for a shared buffer), and the "data service threshold" b.

For example, we took the buffer size $B = 550$ cells, $(q_v^{\mathrm{on}}, q_v^{\mathrm{off}}, \Lambda_v) = (10\mathrm{s}^{-1}, 10\mathrm{s}^{-1}, 7000$ cells/s) and $(q_d^{\mathrm{on}}, q_d^{\mathrm{off}}, \Lambda_d) = (2\mathrm{s}^{-1}, 2\mathrm{s}^{-1}, 5000$ cells/s). Figure 2 is a graph of the logarithm of the fraction of data cells lost as a function of the data service threshold b. For all values of b in this simulation, we found that $F_v^{\mathrm{b}} \approx 10^{-2}$ and that the average delay for of a voice cell that is eventually served was about 3ms.

Note that $\log F_d^{\mathrm{b}}$ appears to be linearly decreasing in b. We offer the following heuristic explanation. Let $X_d(t)$ be the number of data cells in the buffer at time t. Assume that the data traffic is Poisson. When $0 < X_d < X_d^{\max} - b$ the data traffic experiences a service rate of either $0.33c$ or c depending on whether voice cells are queued or not. Thus the probability that $X_d \geq X_d^{\max} - b$ at some point in a busy cycle is $\Pi_{B-b} \approx (\hat{c}/\lambda_d)^{-(B-b)}$, for some $\hat{c}$ not a function of b satisfying $0.33c < \hat{c} < c$. The probability that a data cell is lost before $X_d < X_d^{\max} - b$ again, *given* that $X_d = X_d^{\max} - b$ initially, is $\pi_b \approx (c/\lambda_d)^{-b}$. Thus,

$$\begin{aligned} \log F_d^{\mathrm{b}}(b) &\approx \Pi_{B-b}\pi_b \approx -b \log \frac{c}{\lambda_d} - (B-b) \log \frac{\hat{c}}{\lambda_d} \\ &= -b \log \frac{c}{\hat{c}} - B \log \frac{\hat{c}}{\lambda_d}; \end{aligned}$$

i.e. $\log F_d^{\mathrm{b}}$ is linearly decreasing in b.

4 Buffer Allocation for Loss Sensitive Traffic

In this section we will discuss a simulation study that indicated that the shared buffer with bumping rule discipline results in a more efficient utilization of the buffer memory than the segregated buffers approach (see table IV in [6]). The budding rule service discipline described in the simulation example above (section 3.3) was used with data service threshold $b = 4$ ($X_d^{\max} = B_d^{\text{seg}}$ for segregated buffers). It was assumed that the voice (resp. data) traffic requires a cell loss probability no greater than $F_v = 10^{-2}$ (resp. $F_d = 10^{-5}$). The deterministic service rate c of the buffer was taken to be 10^4 cells/s.

We conducted several simulations using traffic parameters satisfying the conditions of those described in the previous section and found that the required $B^{\text{sh}} \leq B_d^{\text{seg}} = B^{\text{seg}} - B_v^{\text{seg}}$ in each case. Using the traffic parameters of the above example, the required shared buffer memory was found to be $B^{\text{sh}} = 550$ cells and the required segregated buffer memory $B^{\text{seg}} = B_v^{\text{seg}} + B_d^{\text{seg}} = 630 + 70 = 700$ cells. Note that the extra memory required to implement linked lists in the shared buffer (see next section) is $2B^{\text{sh}} \log_2(B^{\text{sh}})/(53 \times 8) = 24$ cells.

By flushing out the voice cells in the buffer, the bumping rule is allowing the data traffic to capture more service bandwidth. The statistical multiplexing of the bursty voice and data traffic in the shared buffer results in memory savings as well.

In [3], the problem of sharing memory optimally is considered. The authors assume that the buffer (memory) is of fixed size and that there are several independent and different memoryless servers. They minimize a penalty function (cell loss probability) over all possible coordinate-convex buffer sharing rules. In our set-up, we assume that the required cell loss probabilities are prespecified and that there is one server. We minimize buffer resources required to achieve the prespecified cell loss probabilities under to the bandwidth constraints. Also, we do not consider a general class of coordinate convex buffer sharing rules.

In Appendix B we consider the shared buffer architecture to compare its speed with that of a segregated buffer architecture; a segregated buffer architecture being a collection of cell FIFOs operating in parallel where all cells in any given FIFO are of the same traffic type. We find that the number of memory cycles per cell that is required for a shared buffer can be reduced by pipelining to a maximum of two. Thus, the segregated buffer architecture is twice as fast as the shared.

5 Discussion

Assume we have committed ourselves to a fixed amount of cell memory but we can replace the FSM/LPR chip to add different traffic types. There is a trade-off in the two architectures considered above. The advantage of the segregated buffer approach is that the fan-in R is significantly larger. The advantage of the shared buffer is a greater flexibility to handle new applications and a more efficient utilization of the existing cell buffer so that new applications can be accommodated by changing the FSM/LPR.

The issue of *selective* discarding of cells is also motivated by "layered" compression algorithms. For example, a voice compression algorithm could create cells in pairs; in every pair, one cell contains more significant information than the other cell ("MSB" and "LSB"). The less significant voice cells are less loss sensitive and are bumped before the more significant voice cells [8]. The MSB and LSB cells would have the same delay sensitivity. Thus, two virtual circuits would be created: MSB and LSB. The salient property of layered compression algorithms is the ability to reconstruct a (degraded) signal when only the more significant information (higher layer) is available.

Acknowledgement

We thank Richard Edell for several helpful discussions on implementation issues.

References

[1] F. Baccelli and P. Bremaud. *Palm Probabilities and Queues.* Springer Lecture Notes is Stat., New York, NY, 1986.

[2] C. Courcoubetis, G. Kesidis, A. Ridder, J. Walrand, and R. Weber. Admission control and routing in ATM networks using inferences from measured buffer occupancy. *submitted to IEEE Trans. Comm.*

[3] G.J. Foschini and B. Gopinath. Sharing memory optimally. *IEEE Trans. Comm.*, 31 No. 3:352–360, March 1983.

[4] R.J. Gibbens and P.J. Hunt. Effective bandwidths for multi-type UAS channel. *submitted to QUESTA.*

[5] I.W. Habib and T.N. Saadawi. Controlling flow and avoiding congestion in broadband networks. *Commun. Mag.*, 29 No.10:46–53, Oct. 1991.

[6] D. Mitra. Asymptotically optimal design of congestion control for high speed data networks. *IEEE Trans. Comm.*, 40 No.2:301–311, Feb. 1992.

[7] K. Sriram. Dynamic bandwidth allocation and congestion control schemes for voice and data integration in wideband packet technology. *Proc. IEEE ICC'90, Atlanta, GA*, 3:1003–1009, Apr. 1990.

[8] K. Sriram, R.S. McKinney, and M.H. Sherif. Voice packetization and compression in broadband ATM networks. *IEEE JSAC*, 9 No.3:294–304, Apr. 1991.

[9] J. Walrand. *An Introduction to Queuing Networks.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[10] J. Walrand. *Communication Networks: A First Course.* IRWIN, Asken Assoc. Inc. Publ., Homewood, IL, 1991.

[11] Z.-X. Zhao, S.S. Panwar, and D. Towsley. Queueing performance with impatient customers. *preprint.*

6 Appendix A: Proof of Claim 2

Consider a bumping rule, shared buffer using this budding rule.

To show $F_d^{\mathrm{b}}(p_d)$ is a nonincreasing function of p_d, choose p_d and $\hat{p}_d$ such that $1 > p_d > \hat{p}_d > 0$. Consider two stationary bumping rule, shared buffers with coupled data and voice arrivals. Let the number of voice cells at time t in the p_d (resp. $\hat{p}_d$) buffer be $X_v(t)$ (resp. $\hat{X}_v(t)$). Note that, with coupled arrivals, the right-continuous buffer occupancy processes of both buffers are identical. For each cell served when both data and voice are queued, to determine whether it's voice or data an independent random variable δ is generated with uniform distribution on the interval $[0,1]$. If $\delta < p_d$ (resp. $\delta < \hat{p}_d$) a data cell departs from the p_d (resp. $\hat{p}_d$) buffer. Otherwise a voice cell departs. Consider a busy cycle of the buffer occupancy process, starting at time S, during which data cells arrive to a full buffer; let $T_i > S$ be the arrival time of the i^{th} such data cell.

We first show that $X_v(T_1-) \geq \hat{X}_v(T_1-)$. If there are no departures in the interval $I_1 := [S, T_1)$, then $X_v(T_1-) = \hat{X}_v(T_1-)$; so assume cell departures occur in I_1. Let the departure time of the j^{th} such cell be $D_j \in I_1$, $j = 1, ..., J$. Note that $X_v(S) = \hat{X}_v(S) \Rightarrow X_v(D_1-) = \hat{X}_v(D_1-)$. If $\hat{X}_v(D_1-) = 0$ or $\delta \leq \hat{p}_d$ then a data cell departs from both buffers at D_1. If $\hat{X}_v(D_1-) > 0$ and $\delta \geq p_d$ then a voice cell departs from both buffers at D_1. Finally, if $\hat{X}_v(D_1-) > 0$ and $\hat{p}_d < \delta < p_d$ then a voice cell departs from the $\hat{p}_d$ buffer and a data cell departs from the p_d buffer at D_1. Thus $X_v(D_1) \geq \hat{X}_v(D_1)$. By induction, $X_v(D_J) \geq \hat{X}_v(D_J)$. By definition, there are no departures in the interval (D_J, T_1), which implies that $X_v(T_1-) \geq \hat{X}_v(T_1-)$,as desired.

Therefore, if a voice cell is bumped in the $\hat{p}_d$ buffer at time T_1 (i.e. $\hat{X}_d(T_1-) > 0$), then a voice cell is also bumped in the p_d buffer at time T_1. Also, the inequality

$X_v(T_1) \geq \hat{X}_v(T_1)$ is preserved (the only way a voice cell is not bumped in the $\hat{p}_d$ buffer is if $\hat{X}_v(T_1-) = 0$). By induction, $X_v(T_i-) \geq \hat{X}_v(T_i-)$ for all i. We can conclude that if an arriving data cell bumps in the $\hat{p}_d$ buffer then it bumps in the p_d buffer as well.

Therefore, the average number of voice cells bumped per busy cycle $\epsilon(p_d) \geq \epsilon(\hat{p}_d)$ (i.e. $\epsilon(p_d)$ is a nondecreasing function of p_d). Recall from claim in section 2.2 that

$$F_v^{\mathrm{b}}(p_d) = \frac{L_v + \epsilon(p_d)}{C_v}.$$

Thus $F_v^{\mathrm{b}}(p_d)$ is nondecreasing too. By Neveu's cycle formula, the aggregate cell loss probability

$$F = \frac{\lambda_v}{\lambda_v + \lambda_d} F_v^{\mathrm{b}}(p_d) + \frac{\lambda_d}{\lambda_v + \lambda_d} F_d^{\mathrm{b}}(p_d)\ .$$

Note that F is not a function of p_d! Thus $F_d^{\mathrm{b}}(p_d)$ is nonincreasing.

7 Appendix B: Buffer Implementation

We now describe the simple shared buffer design that can realize the buffering and service disciplines described above for many traffic types. The buffer resides in a RAM of size $B = 2^\beta$ by $53 \times 8 + \beta$ bits. The buffer is split between the cell memory (CM) of size B by 53×8 bits, and the "next-pointer" memory (NPM) of size B by β bits. The NPM is a dual-port RAM that can be accessed independently of the CM. It contains pointers to other cells in the buffer in the manner of a linked list. There may be several linked list in the buffer at any one time. The pointers in the NPM point to the next-to-be served cell of the same type in the buffer.

Also, there is a FIFO (UPF) of size B by β containing pointers to the unused portions of the buffer and a bank of registers (LPR) containing pointers to the head and tail of each linked list in the buffer (e.g. head_voice and tail_data). The LPR resides on chip with the finite state machine which controls the memories. The output of the UPM is registered.

Cells or cell fragments are placed by the switch fabric into a register. The type of cell is determined from a field in the cell's header. A sequence of memory operations will follow that depend on the type of cell to be handled and the state of the buffer. Three such memory sequences will now be discussed.

Assume a data cell has arrived from the switch fabric and the UPF is not empty. Two memory cycles ensue:

1. Get unused and tail_data pointers (read UPF and LPR).
2. - unused.cell = new data cell (write CM)
   - tail_data.next_ptr = unused (write NPM)
   - tail_data = unused (write LPR)

Now assume a voice cell is to be served (transmitted). Three memory cycles ensue:

1. Get head_voice pointer (read LPR).

2.
    - Read head_voice.cell (read CM).
    - Write head_voice to UPF.
    - head_voice = head_voice.next_ptr (read NPM)
3. head_voice = head_voice.next_ptr (write LPR)

Finally, assume a data cell has arrived, the buffer is full (UPF is empty) and there are more than one voice cells queued. The arriving data cell bumps the oldest [11] voice cell. Three memory cycles ensue:

1. Get head_voice and tail_data pointers (read of dual-port LPR).
2.
    - head_voice.cell = new data cell (write CM)
    - tail_data.next_ptr = head_voice (write NPM)
    - head_voice = head_voice.next_ptr (read NPM; see next memory cycle)
    - tail_data = head_voice (write LPR)
3. head_voice = head_voice.next_ptr (write LPR)

We will now compare the speed and flexibility of this shared buffer architecture with that of a segregated buffer architecture; a segregated buffer architecture being a collection of cell FIFOs operating in parallel where all cells in any given FIFO are of the same traffic type. First note that the number of memory cycles required for a shared buffer can be reduced to a maximum of two (instead of three) by pipelining the operations involving the LPR for consecutive cells.

Figure 3 is a block diagram of this shared buffer. R0 represents a cell fragment accumulator; depending on the switch fabric, R0 may or may not be necessary. R1 and R2 are 53 byte (one cell) registers.

Given an LPR with pipelined operations and assuming that memory access is the bottle neck, we get that the maximum amount of time, M^{sh}, needed to handle a cell (being received or transmitted) is equal to one FIFO cycle time and one RAM cycle time. The amount of time, M^{seg}, needed to handle a cell for the segregated buffer architecture is equal to one FIFO cycle time. Let R be the maximum number of cells that can arrive to the buffer from the switch fabric in c^{-1} seconds (R is the "fan-in" of the switch). The following inequality must hold:

$$M(R+1) \ \leq \ c^{-1}, \tag{2}$$

where the figure $R+1$ includes a transmission operation (actually, both M and c^{-1} have dimensions seconds/cell).

For example, assume the FIFO and RAM cycle times are 25 ns, and the fibers are transmitting at 155 Mbps which corresponds to $c = 155 \times 10^6/(53 \times 8)$cells/s. Thus $M^{\mathrm{sh}} = 50$ns and $M^{\mathrm{seg}} = 25$ns. The fan-in inequality (2) implies that $R^{\mathrm{sh}} = 54$ and $R^{\mathrm{seg}} = 108$.

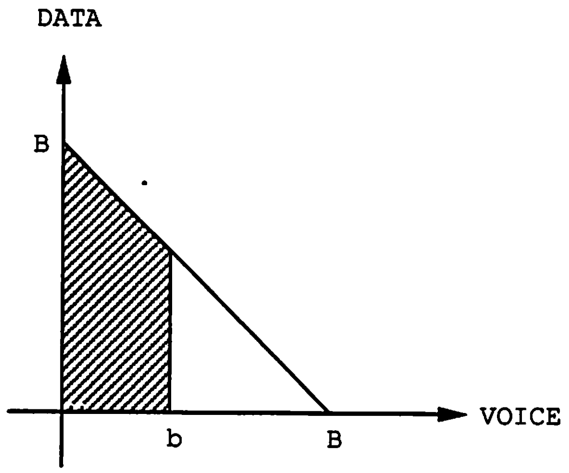

Figure 1: Partially Shared Buffer

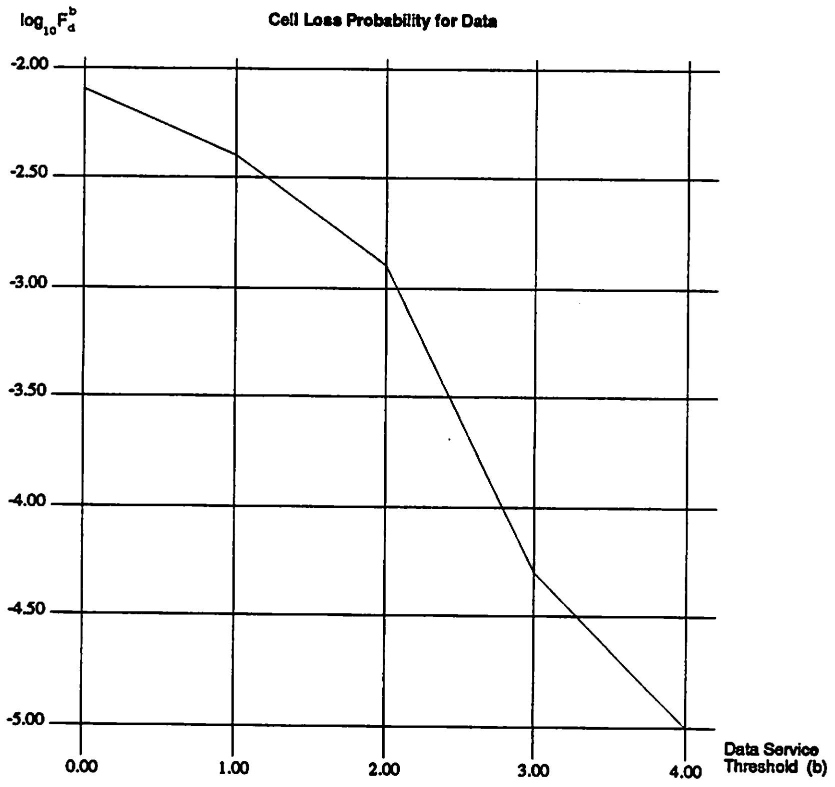

Figure 2: Budding Rule for Delay and Loss Sensitive Traffic

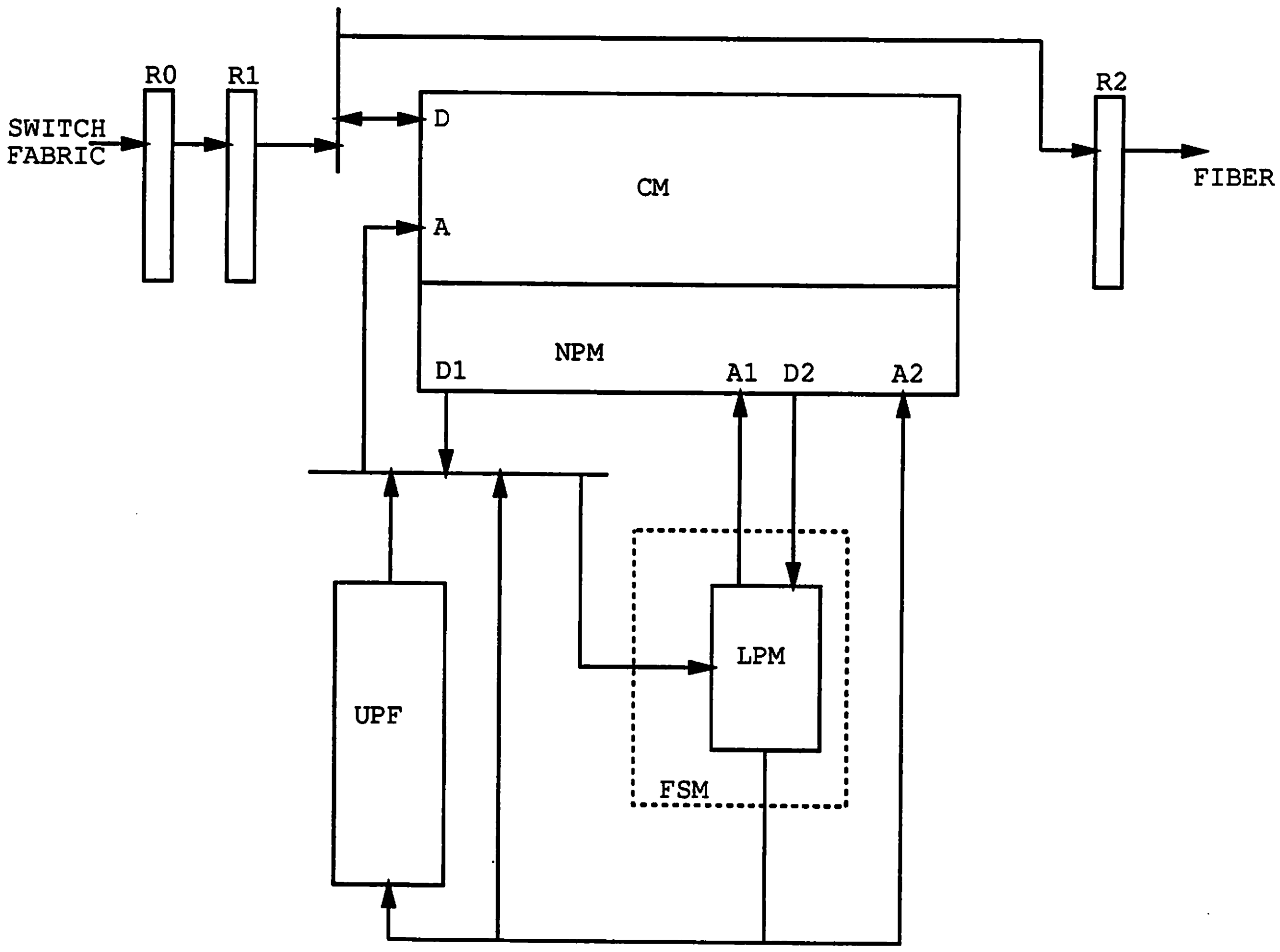

Figure 3: Shared Buffer