ISRN SICS-R--95/02--SE

# Capacity Reservation in ATM Networks

by
Gunnar Karlsson

SICS Research Report
R95:02
ISSN 0283-3638

Report R95:02

ISRN : SICS-R--95/02--SE
ISSN : 0283-3638

# Capacity Reservation in ATM Networks

BY

Gunnar Karlsson

March 1995

gk@sics.se

Swedish Institute of Computer Science
Box 1263, S-164 28 KISTA, SWEDEN

---

ABSTRACT

The research on traffic control has not yet lead to a consensus on how to properly allocate resources in ATM networks. There are consequently no practicable methods available now when the initial deployment of ATM switches and terminals is under way. Yet, many of the applications which motivate the deployment uses multi-media and will thus require some degree of performance guarantees on the information transfer. Here we suggest a readily applicable method for reserving capacity in ATM networks. Cells using the reserved capacity may only depart at fixed time-instances, and have reserved buffers in the network nodes. The scheme gives a lossless performance, can be implemented with low complexity; it simplifies the call acceptance and allows ``best effort'' traffic to use slack in the reserved and all of the non-reserved capacity.

Keywords: ATM, traffic control, resource allocation, scheduling, quality of service

# Capacity Reservation in ATM Networks

**Gunnar Karlsson**
**Swedish Institute of Computer Science**
**Box 1263**
**S–164 28 Kista**
**Sweden**
**Internet: gk@sics.se**

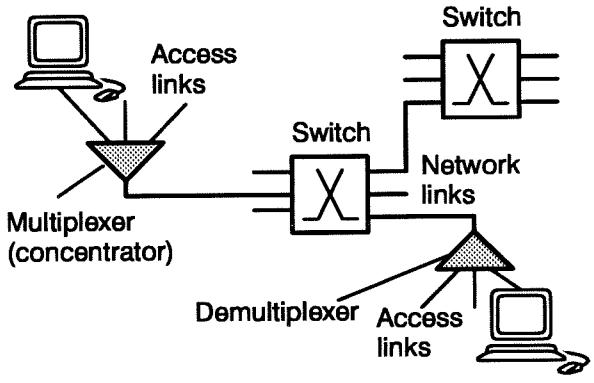
*The research on traffic control has not yet lead to a consensus on how to properly allocate resources in ATM networks. There are consequently no practicable methods available now when the initial deployment of ATM switches and terminals is under way. Yet, many of the applications which motivate the deployment uses multi–media and will thus require some degree of performance guarantees on the information transfer. Here we suggest a readily applicable method for reserving capacity in ATM networks. Cells using the reserved capacity may only depart at fixed time–instances, and have reserved buffers in the network nodes. The scheme gives a lossless performance, can be implemented with low complexity; it simplifies the call–acceptance and allows "best effort" traffic to use slack in the reserved and all of the non–reserved capacity.*

## 1 Introduction

The asynchronous transfer mode (ATM) is the International Telecommunication Union's recommended suit of protocols for broadband integrated services digital networks. The mode combines the circuit switched routing of telephone networks with the statistical multiplexing of packet switching. This is accomplished by establishing a connection (fixed route) through the network before accepting any traffic. The information is then sent in 53–octet long cells. The switching nodes, which interconnect network links, route the cells according to address information contained in their 5–octet headers. The traffic on a particular link will consist of a random mixture of cells belonging to different calls. The network guarantees that all cells of a call follow the same route and, hence, get delivered in the same order as sent.

We will consider a network consisting of access and network links. The access links connect terminals to multiplexers (concentrators) and to demultiplexers; the network links connect multiplexers, demultiplexers and switches. The network structure is illustrated in Fig. 1 for one way from terminal to terminal.

*Fig. 1 The ATM network structure.*




It is possible that the aggregate capacity of the access links of a multiplexer exceeds the capacity of the network link (concentration). The statistical variations of the arrivals may therefore lead to temporary overflow of the multiplexer queue even when the average incoming rate is less or equal to the outgoing rate.

When a connection is being established, the network control chooses a route over which the call can be added to obtain the capacity and quality required by the user, without violating the requirements of existing calls. The call is blocked if no such route is available.

The call–acceptance decision can be stated as follows. The average total capacity of $N$ established connections over a link may at most be equal to the link capacity,

$$E\left[\sum_{j=1}^{N} C_j(t)\right] \leq C_{link} . \qquad (1)$$

($E[\cdot]$ is the expected value of the argument, $C_j(t)$ is the time–varying capacity used by connection $j$, and $C_{link}$ is the link–capacity and therefore constant.) The offered quality of service for the link (denoted $Q_{link}$) at this load is at least as good as any user requires. Hence,

$$Q_{link} \geq \max_{1 \leq j \leq N} Q_j , \qquad (2)$$

where $Q_j$ is the required service–quality of connection $j$. Is it now possible to establish a new connection over the link, so that

$$E\left[\sum_{j=1}^{N+1} C_j(t)\right] \leq C_{link} \text{ and } Q_{link} \geq \max_{1 \leq j \leq N+1} Q_j ? \qquad (3)$$

The posed question is difficult to answer when the traffic streams $C_j(t)$ are stochastic processes [7]. The user is therefore expected to characterize the traffic and the expected quality of the call in some way that the network controller can use for the call–acceptance decision. The more precise the characterization is, the better the network can be controlled. Improved control gives better utilization of the network resources for a given quality level. The quality on a link is given by the call with the most stringent requirement; all other calls get the same level of service whether they need it or not, as shown in Eq. (2). A call that requires high quality may raise the quality level on all traversed links and thereby cause the capacity on this path to be meagerly utilized.

The traffic characterization is normally based on some stochastic model of the particular source for which the user has to estimate the parameters (like peak and average rates). The model may be a poor estimate of the source's actual behavior. Sources for data and variable–rate coded video seem especially problematic to model accurately (see [5] and [2]). It may also be difficult for a user to assess the parameters of the traffic before the call is initiated.

The question in Eq. (3) has to be answered positively for all links along a route in order to establish the connection. Multiplexing may, however, modify the characteristics of a flow so that it no longer is described by the user–supplied model and its parameters [1]. This exacerbates the call–acceptance problem.

The expected quality of service for a call is hard to define quantitatively. The probability of cell–loss, the maximum total transfer–delay and its variations are frequently suggested quality criteria. Delay could be mapped into loss if cells not delivered within a specified time–limit are considered to be lost. The probability of cell–loss will, of course, only be guaranteed over an infinite call–duration and can thus not be verified by the users. There are indications that cell–loss due to transient phenomena might be orders of magnitude higher than the steady–state loss [1]. This means the transient behavior of the network must be considered and that the loss–limits should be specified for finite time–periods.

Uncertainties in the information about a source's behavior before and after multiplexing may require the network control to over–allocate capacity in order not to violate accepted quality contracts. The most careful solution is to allocate capacity according to the call's maximum transmission rate. The peak–rate is specified as the minimum inter–arrival time between cells and are thus ratios of one cell per $T$ seconds. The main disadvantage of peak–rate allocation is the low utilization that follows. All connections must be peak–rate controlled and surplus capacity is wasted since statistical multiplexing is precluded.

Traffic control for ATM is a highly active research field. Yet there is no clear indication that the suggested methods for statistical multiplexing with quality guarantees are practicable [1][4][7]. This article describes an alternative and readily applicable way of reserving capacity for virtual circuits and paths in ATM networks. The reserved capacity is carried without loss in the network with regard to contention (bit–errors might still occur). The left–over, non–reserved capacity can be used by anyone as "best effort" and could also be allocated statistically with quality guarantees.

The call–acceptance problem, as stated above, is simplified by bounding the flows deterministically and

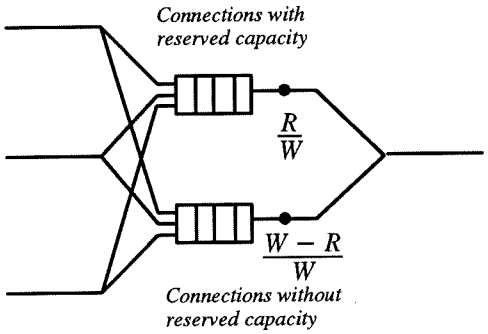
by defining the quality to be complete absence of cell–loss. The method is based on a pacing–mechanism, which controls the departure instances of the reserved cell–stream, and on reserved buffers which are guaranteed never to overflow.

The proposed method is developed to provide a service with deterministic quality in addition to statistical guarantees. It may serve well for traffic that has some performance requirements until the statistical resource allocation methods can be put to practice.

## 2 The capacity reservation scheme

Connections which require reserved capacity are treated separately from connections without reserved capacity (so called "best effort" traffic). The reserved capacity could be a part or the full (maximum) capacity needed by the call. Each of these two traffic classes has its own buffers in the network nodes, as shown in Fig. 2. The classes are separated by the addresses (VCI/VPI). This article concerns primarily connections with reserved capacity. The cells in this class could be given high or low priority, which is distinguished by the cell–loss priority (CLP) bit in the ATM cell header.

*Fig. 2 Separation of traffic classes into different FIFO–queues.*




Capacity is only reserved for the high priority cells. The low priority cells may use left–over portions of the reserved and non–reserved capacity but has no guarantees against loss. It is therefore up to the user to decide how much capacity to reserve and how much to vie for when a connection is requested. The surplus traffic with low priority is allowed on a connection to increase the flexibility—the reserved rate can be exceeded at own risk—and to better the utilization (*eg*, when capacity is reserved but little of it used and there is no "best effort" traffic).

We define the reserved rate of a connection as the number of cells, $R$, that may maximally be sent during a window of $W$ slots. The reserved rate may be any ratio $R / W$ of the access–link's capacity ($R$ and $W$ can be arbitrarily large). For example, a reservation of 750 kb/s gives a ratio of 3/8 over a 2 Mb/s link and 3/136 over a 34 Mb/s link. The connections without reservations get full use of the remaining portion $(W–R) / W$ of the link's capacity. Cells are serviced from one queue exclusively if the other is empty. Unused potions of the reserved capacity is consequently not wasted since it may be used by connections without reservations.

If we assume $N$ connections to a multiplexer and that all access–links have the same capacity, then

$$C_{acc} \sum_{j=1}^{N} \frac{R_j}{W_j} = C_{net} \frac{R_{out}}{W_{out}} . \qquad (4)$$

($C_{acc}$ and $C_{net}$ are the capacities of the access and the network links, respectively.) The departures of the $R_{out}$ cells may well be clustered within the $W_{out}$–window. The equation shows that

$$W_{out} \leq \prod_{j=1}^{N} W_j . \qquad (5)$$

At each stage of multiplexing the values of $W_{out}$ and $R_{out}$ for the outgoing stream may consequently increase dramatically. (Equality is reached when $W_i$ and $W_j$ are relative prime for all $i \neq j$.) Since the cells may depart back–to–back, a higher value of $R_{out}$ means that more buffer–space is needed to avoid cell–loss. Also, if the connections with reservations are served exhaustively before non–reserved traffic is served, then the latter may be unduly delayed.

Multiplexing may thus severely increase the burstiness of a traffic stream. There are two immediate solutions to this problem, of which we have chosen the latter. The first is to require $W_j = W_{out}$, for all $j$. (This is basically the approach of the Stop–and–Go scheme [3].) If the value of $W$ is low, then the resolution of the specified rates is limited (*eg*, none, half, or all of the capacity for a two–slot window). When the value is high, non–reserved traffic may still be considerably delayed. We have instead chosen to alleviate the batch arrival problem by enforcing an even distribution of the cell departures within a window. Non–reserved traffic will be served in between departures of reserved cells.

The first issue is consequently to pace $R$ cell–departures evenly during a $W$–slot window for arbitrary values of $R$ and $W$ ($R \leq W$). The second issue is to determine the queue sizes needed to guarantee an operation free from cell–loss.

The remainder of the paper is organized as follows. Section 3 gives the pacing function with a suggested software implementation. Section 4 contains the buff-

er dimensioning for multiplexers, demultiplexers and switches. The single criterion is to ensure enough buffers for paced traffic streams so that cell–loss is avoided. Section 5 contains a discussion on delay limits for the needed buffer–sizes. The call–acceptance for the connections with reservations is discussed in Section 6. In Section 7, we compare the proposed schemes with two other schemes from the literature. Section 8 concludes with a summary and some remarks.

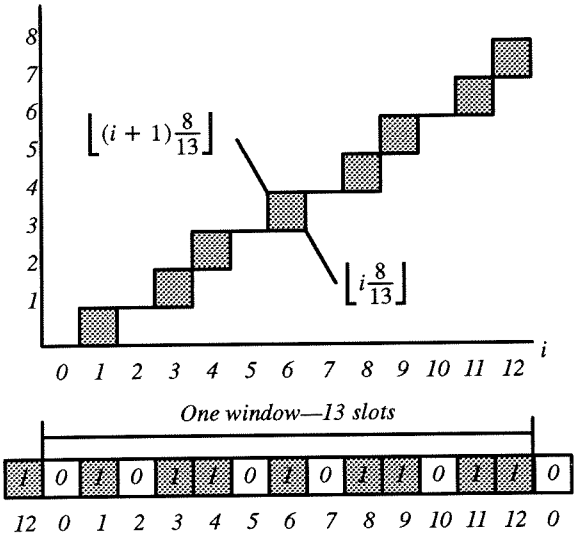
## 3 Pacing cell departures

Let $\pi(\cdot)$ be a binary–valued function which is one for reserved slots and zero for non–reserved slot during a window $W$. Let $i$ indicate a modulo–$W$ counter, then $\pi(i)$ is given by

$$\pi(i) = \left\lfloor (i+1)\frac{R}{W} \right\rfloor - \left\lfloor i\frac{R}{W} \right\rfloor . \qquad (6)$$

($\lfloor \cdot \rfloor$ denotes the integer–part of the argument.) The function is illustrated in Fig. 3 for $R=8$ and $W=13$.

*Fig. 3 The pacing function for R=8 and W=13*




The pacing can be easily implemented without multiplications, divisions or truncations, as shown later. The distance between any two reserved slots is $\leq \lfloor W/R \rfloor$. If a reserved slot has not been used then a subsequent reserved cell can depart immediately; the pacing is restarted at its departure. For example, $R = 1$ and $W = 3$ would mean that cells would depart every third slot. Suppose the first cell has departed and the second cell comes four slots later, then it can depart immediately, but a third cell could depart no sooner than the third slot after the second cell. The pacing hence controls the minimum distance between cells; it does not enforce fixed departure instances for larger inter–cell distances.

The service algorithm for a multiplexer or switch output is given by the following C–code. One pass through the code schedules a full window ($W$ slots).

```
start:
{

 get_new_values();
/* Update R and W if needed */
 i = 0;          /* Reset window-counter */
 do              /* Start a new window. */
 {
  i += R;        /* iR */

  /* Is the integer-part of the ratio < 1? */
  if (i < W)
  {
   /* Try to send the next cell from the non-
      reserved queue. */
   if (!send_no_res())
    /* Send from reservation queue if the other
       is empty and the first cell has low
       priority (not paced) */
    send_res (!CLP);
  }
  else
   /* Send the first cell from the reservation
      queue (regardless of priority) */
   if (send_res (CLP))
    /* Remove the integer-part of the ratio */
    i -= W;
   else
   {
    /* Send from other queue, if empty */
    send_no_res();
    /* Compensate counter value */
    i -= R;
   }
 }
 while (i);  /* Repeat until end of window */
}
goto start;
```

The functions "`send_no_res()`" and "`send_res()`" transmit the first cell in their respective FIFO–queues and they immediately return the value zero if the particular queue is empty. The function "`send_res()`" takes two parameter values: "`CLP`" is passed when the slot is reserved to indicate that a high–priority cell at the head of the queue may be sent; otherwise "`!CLP`" is passed and only a cell with low–priority may be sent. The function "`get_new_values()`" is just an indication that $R$ and possibly also $W$ need to be changed whenever a reservation has been added or cancelled. Note that the counter may not exceed the value $W$ by more than $R–1$. Thus the compensation in the `else-else` clause.

## 4 Dimensioning of reserved buffer–space

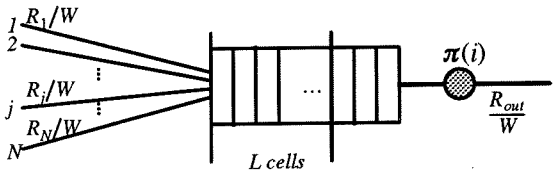
The pacing gives a deterministic bound on the arrival process on each link. The needed buffer–space to ensure a lossless operation may thus be computed. The

only uncertainty about the arrival processes is their relative time–alignment at the inputs to a multiplexer or switch. The worst case is when the reservation ratios on all inputs are equal and the windows are perfectly aligned. The cells will then arrive at the same time instances on all input links. The probability for it to happen is low but the scheme is guaranteed to handle even the worst case without loss.

## MULTIPLEXING

The model of a multiplexer that we use is depicted in Fig. 4. Assume that all links—incoming as well as the outgoing—have the same capacity, that the traffic on all connections is paced, and that their reservations are fully used. The number of cells that have arrived on input link $j$ to the multiplexing buffer at a time $n$ during a window is given by

*Fig. 4 A multiplexer with N input links.*




$$B_j(n) = \left\lfloor (n + \Delta_j)\frac{R_j}{W} \right\rfloor, \quad 1 \le n \le W, \qquad (7)$$

where

$$0 \le \Delta_j < \left\lceil \frac{W}{R_j} \right\rceil. \qquad (8)$$

($\lceil \cdot \rceil$ denotes the integer next–larger than the argument.) The $\Delta_j$'s are constant, integer slot–shifts and determine the arrival time of the first cell in the window on each link. Note that

$$B_j(n) \le \left\lfloor n\frac{R_j}{W} \right\rfloor + 1. \qquad (9)$$

All rates $R_j$ are expressed over a common window $W = W_{out} = W_1 \times \ldots W_N$ for simplicity. The only effect of this is that the cell arrival pattern will repeat itself also within the window when $R_j$ and $W$ have common factors. The reservation–ratios are normalized with respect to the link–capacities: $R_j \leftarrow R_j\,(C_{acc}/C_{net})$ (*cf.* Eq. (4)).

The in and out flows must be equal when summed over one window so that the finite multiplexing buffer is guaranteed not to overflow. This means that for $N$ input links which carry reserved traffic, we have

$$R_{out} = \sum_{j=1}^{N} R_j \le W \qquad (10)$$

and consequently

$$\sum_{j=1}^{N} B_j(W) - B_{out}(W) = 0, \qquad (11)$$

where

$$B_{out}(n) = \left\lfloor n\frac{R_{out}}{W} \right\rfloor, \quad 1 \le n \le W. \qquad (12)$$

We have assumed $\Delta_{out} = 0$ for the output link which means that the first cell will depart at the latest time possible (worst case). Given this, the maximum buffering requirement is given by

$$L = \max_{1 \le M \le W} \sum_{j=1}^{N} B_j(M) - B_{out}(M). \qquad (13)$$

Thus, to avoid cell loss there has to be at least $L$ locations reserved in the buffer for priority cells.

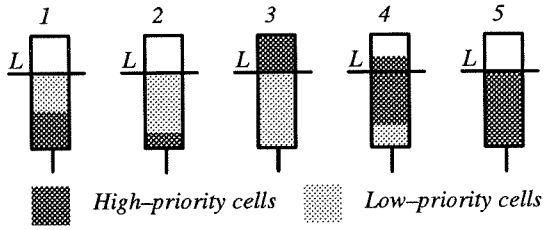
**Proposition 1** The maximum buffering requirement for reserved traffic in a multiplexer is one cell–position per input link, *i.e.* $L = N$.

**Proof**

$$\begin{aligned} L &= \max_{1 \le M \le W} \sum_{j=1}^{N} B_j(M) - B_{out}(M) \\ &= \max_{1 \le M \le W} \sum_{j=1}^{N} \left\lfloor (M + \Delta_j)\frac{R_j}{W} \right\rfloor - \left\lfloor M\frac{R_1 + \cdots R_N}{W} \right\rfloor \\ &= \{\textit{use Eq. (9) with equality}\} \\ &= N + \max_{1 \le M \le W} \sum_{j=1}^{N} \left\lfloor \frac{MR_j}{W} \right\rfloor - \left\lfloor \frac{MR_1}{W} + \cdots \frac{MR_N}{W} \right\rfloor \\ &= \{\lfloor a_1 + \cdots a_N \rfloor \ge \lfloor a_1 \rfloor + \cdots \lfloor a_N \rfloor,\ \forall a_j \ge 0\} \\ &= N. \end{aligned} \qquad (14)\square$$

The queue has consequently to have at least $N$ cell locations dedicated to reserved traffic. Low–priority cells are not accepted into the queue whenever the number of free queue–spaces is below that.

*Fig. 5 Transient behavior at an overload situation: low–priority cells are initially accepted into the buffer; as it fills up, less of them get admitted. Eventually the buffer is constantly filled to the threshold with high–priority cells.*

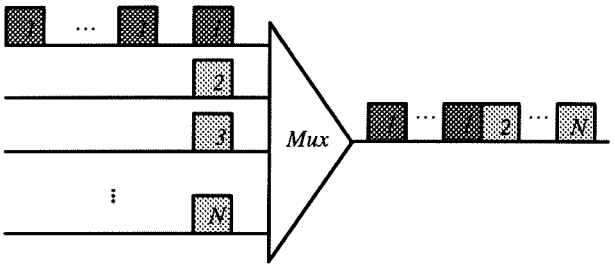



The total amount of arriving high–priority and low–priority cells might exceed the reserved output capacity. Remember that the reservation is only for

high–priority cells. As Fig. 5 illustrates, both types of traffic may initially be accepted (1). Eventually the queue is filled up to the threshold and low–priority cells are no longer allowed entry (2). The reserved buffer–space can hold all arriving high–priority cells (3). Still, the buffer level might occasionally go below the threshold and a few low–priority cells get accepted. This, in turn, leads to a higher number of high–priority cells kept in the queue (4). When stationarity is reached, the queue is constantly filled to the threshold with high–priority cells (5). High–priority cells are then accepted without loss but all low–priority cells are discarded.

The multiplexing will not maintain the pacing of the individual input streams. Consider the (worst) case when, say, $R_1 = W-N+1$ and $R_2 = \ldots = R_N = 1$. If the first cell on each input line arrives at the start of a window, and lines 2 to $N$ get serviced before line 1, then all cells that arrived on line 1 will depart back to back. They will consequently need to be paced again after the aggregate flow has been demultiplexed, as will be analyzed next. This issue is illustrated in Fig. 6.

*Fig. 6 Multiplexing may destroy the pacing of an input stream.*




## DEMULTIPLEXING

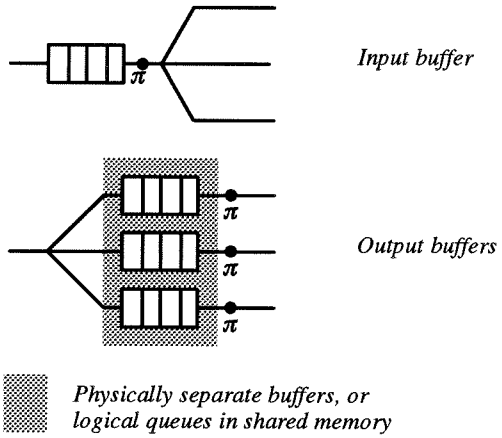
A demultiplexer takes an aggregate input flow and splits it into several output flows (of which each may still be an aggregate of connections). The demultiplexed flows may have any arrival distribution of cells within a window. In the worst case they will be clustered, as shown in Fig. 6, and thus require buffering and pacing. This cannot be done before the demultiplexing, so called input buffering (Fig. 7). The reason is that the pacing will greatly limit the throughput of the system through "head of the line" blocking. The worst case appears when cells destined to an output arrive as a contiguous pair: the first cell departs immediately, the second is delayed for half a window. More precisely, the minimum throughput (given full load to the demultiplexer) is

$$\varrho_{min} = \frac{2}{\lceil W/2 \rceil + 1}. \qquad (15)$$

This assumes that the window is short, $\lceil W/2 \rceil \leq N$ ($N$ is the number of output ports). Due to the throughput limitation, we will only consider output buffering, after the demultiplexing (Fig. 7).

The bound on the buffer space for the output lines of a demultiplexer, needed to evenly pace the output flows without overflow, is given below. There are two cases: a dedicated buffer on each output line, and a common buffer to be shared by all output lines.

*Fig. 7 Demultiplexer architectures.*




**Proposition 2** The maximum buffering requirement for reserved traffic in a demultiplexer is $L = \lceil \frac{W}{4} \rceil$ cells per output line.

**Proof**

After the demultiplexing, the clustered cells will in the worst case arrive according to the cumulative function

$$B_{in}(n) = \begin{cases} n, & 1 \leq n \leq R \\ R, & R < n \leq W \end{cases} \qquad (16)$$

and will depart evenly paced as

$$B_{out}(n) = \left\lfloor (n + \Delta)\frac{R}{W} \right\rfloor, \quad 1 \leq n \leq W, \qquad (17)$$

where we choose $\Delta = 0$ to delay the first cell departure maximally. The maximum buffering is then given by

$$L = \max_{1 \leq M \leq W} B_{in}(M) - B_{out}(M) \qquad (18)$$

which reaches its maximum at $M = R$ since $B_{in}$ increases faster than $B_{out}$ up to that point (since $\frac{R}{W} \leq 1$). So, we have

$$L = \max_{0 \le R \le W} \left\{ R - \left\lfloor \frac{R^2}{W} \right\rfloor \right\}. \qquad (19)$$

The real–valued function $x - \frac{x^2}{W}$ has its maximum at $x = \frac{W}{2}$ and the value at that point is $\frac{W}{4}$. However, $R$ is a positive integer and the maximum is therefore at $R = \lfloor \frac{W}{2} \rfloor$ (or at $R = \lceil \frac{W}{2} \rceil$ since the function is symmetric). We then have that

$$L = \max \left\lfloor \frac{W}{2} \right\rfloor - \left\lfloor \left\lfloor \frac{W}{2} \right\rfloor^2 \frac{1}{W} \right\rfloor = \left\lceil \frac{W}{4} \right\rceil. \qquad (20)$$

To see this, let $W = 4w_2 + 2w_1 + w_0$ where $w_2 \in \mathcal{N}$ and $w_0, w_1 \in \{0,1\}$. For any value of $W$, not divisible by $4$, we get that

$$2w_2 + w_1 - \left\lfloor \frac{2w_2 + w_1}{2 + \frac{w_0}{2w_2 + w_1}} \right\rfloor \le w_2 + 1. \qquad (21) \ \square$$

If we have $N$ output lines with a buffer each, the total requirement is simply $L = N\lceil W/4 \rceil$. A more efficient way of utilizing the buffer–space is to share it among the output lines.

**Proposition 3** The maximal need of shared output buffers is $L_s = \lfloor N/2 \rfloor (\lceil W/N \rceil + 1)$ cells in total.

**Proof**

Assume that the shared buffer is first filled with the cells for output 1 and then by the cells for output 2 and so forth. The departures are spread over a window $W$ on each of the outputs. We then have

$$B_{in}(n) = \left\lfloor (n + \Delta) \frac{R}{W} \right\rfloor, \quad 1 \le n \le W, \qquad (22)$$

and the number of cells departed on output $i$ is

$$B_i(n) = \left\lfloor (n - \Theta_i)^+ \frac{R_i}{W} \right\rfloor. \qquad (23)$$

The function $(\cdot)^+ = \max\{0, \cdot\}$, and $\Theta_i$ is the smallest integer such that

$$B_{in}(\Theta_i) \ge \sum_{l=0}^{i-1} R_l \quad (R_0 \equiv 0). \qquad (24)$$

This means that enough cells have arrived at time $\Theta_i$ to fill the quotas of the previous outputs. The output rate is below or equal to the arrival rate for the entire window and the highest accumulation of cells in the buffer occurs when they arrive at the maximal rate. The number of cells which have arrived is therefore $B_{in}(n) = n$, $1 \le n \le W$. The accumulation of cells in the buffer is thus

$$L_s = \max_{1 \le M \le W} B_{in}(M) - \sum_{i=1}^{N} B_i(M). \qquad (25)$$

$B_{in}$ grows at least as fast as $\sum_i B_i$ for all $n$ and the highest number of cells in the buffer is reached at the end of the input window,

$$L_s = W - \min \left\{ R_1 + \left\lfloor R_2 - R_1 \frac{R_2}{W} \right\rfloor + \cdots \left\lfloor R_N - (R_1 + \cdots R_{N-1}) \frac{R_N}{W} \right\rfloor \right\}. \qquad (26)$$

Given that $\sum_i x_i = 1$, the real–valued function $x_1 x_2 + \cdots (x_1 + \cdots x_{N-1}) x_N$ has its maximum at $x_1 = \cdots = x_N = 1/N$ ($\max_{x_1 + x_2 = 1} x_1 x_2 \Rightarrow x_1 = x_2$ which is easily generalized inductively). This means that the minimum in Eq. (26) is attained when all $R_i = W/N$ and integer. We then get

$$\frac{W}{N} + \left\lfloor (N-1)\frac{W}{N^2} \right\rfloor + \cdots \left\lfloor \frac{W}{N^2} \right\rfloor \ge \frac{W}{N} + \left( \frac{W}{N} - 1 \right) \left\lfloor \frac{N}{2} \right\rfloor, \qquad (27)$$

since $\lfloor jW/N^2 \rfloor + \lfloor (N-j)\, W/N^2 \rfloor = (W/N) - 1$ for integer ratios $W/N$. When the right–hand side of (27) is substituted in to Eq. (26), we find that the needed buffer–space is bounded by

$$L_s = \left\lfloor \frac{N}{2} \right\rfloor \left( \left\lceil \frac{W}{N} \right\rceil + 1 \right). \qquad (28) \ \square$$

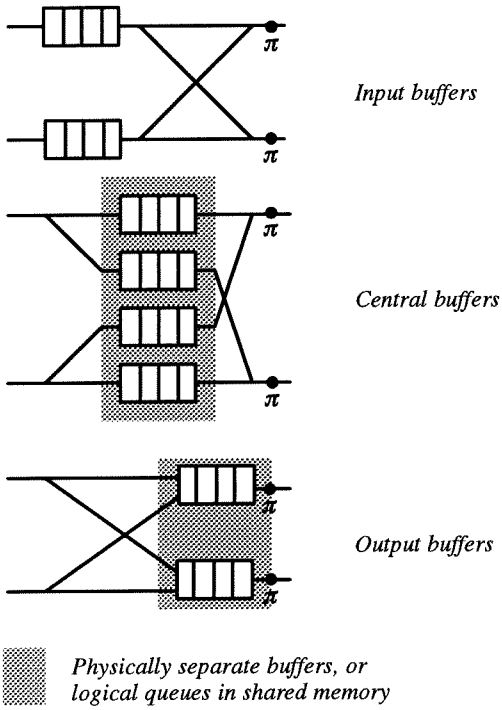
It should be noted that the pacing for an output link is only needed when the flows later are going to enter a multiplexer. A receiving terminal may, however, handle the arrival of clusters of cells. Pacing and buffering can then be avoided for the final demultiplexer in a network configured as in Fig. 1.

## SWITCHING

A switch can be viewed as a stage of demultiplexers, one per input, followed by a stage of multiplexers, one per output. The buffers could be placed before the demultiplexers (input buffers), after the demultiplexers but before the multiplexers (central buffers), or after the multiplexers (output buffers). These cases are illustrated in Fig. 8. Analogously to the reasoning against input buffers for a demultiplexer, with the

throughput given in Eq. (15), only the latter two cases will be considered here.

*Fig. 8 The alternatives for buffer placement in switches.*




Central buffering after the demultiplexing will simply require $N$ times the buffer–space needed for a single demultiplexer. Buffers shared across all $N^2$ lines is the same case as the shared output–buffer covered below.

When the buffering is done at the multiplexing (the switch outputs), it is assumed that there is no pacing of the flows after the demultiplexing. The arrivals can thus be clustered. The amount of reserved space for dedicated buffers is derived first, and the amount when a pool of memory is shared among the outputs is determined after that.

**Proposition 4** The maximal need of output buffers per port is $L = W - \left\lceil \frac{W}{N} \right\rceil$, $N > 1$.

**Proof**

The worst case is when all inputs simultaneously have cells destined to the same output port. The buffer will then get the highest momentary load:

$$B_{in}(n) = \begin{cases} nN, & 1 \le n \le \lfloor R/N \rfloor \\ R, & \lceil R/N \rceil \le n \le W \end{cases} \qquad (29)$$

The output will be serviced, according to

$$B_{out}(n) = \left\lfloor (n + \Delta)\frac{R}{W} \right\rfloor, \quad 1 \le n \le W. \qquad (30)$$

Again, $\Delta = 0$ is chosen to delay the first cell departure. The buffer requirement is as stated in Eq. (18). The maximum is found at $M = \lceil R/N \rceil$ (not at $\lfloor R/N \rfloor$) and is

$$L = \max R - \left\lfloor \left\lceil \frac{R}{N} \right\rceil \frac{R}{W} \right\rfloor. \qquad (31)$$

There are two cases to consider: $N > R$ and $N \le R$. The first case is $L = W - 1$ (and $L = R$ for $R < W$) since $\lceil R/N \rceil = 1$. For the second case we notice that the real–valued function $x - \frac{x^2}{NW}$ has maximum at $x = \frac{NW}{2}$, and for the integer function, the maximum is at $R = \left\lfloor \frac{NW}{2} \right\rfloor$ (for $N = 1$ we get the demultiplexer case of Eq. (20)). However, $R \le W$ so, for $N > 1$, we get

$$L = W - \left\lceil \frac{W}{N} \right\rceil. \qquad (32) \quad \square$$

The last switch–case to consider is when all the output buffers are shared. Although the in and out flows are balanced there can be a temporary storage of cells in the buffer to time–shift the input processes: picture that all inputs first deliver cells only to the first output then, when its quota has been delivered, to the second output and so on. Then it is clear that initially only the first output will service traffic since the other outputs do not have anything to deliver. (This is when the buffer is initially empty; in the following windows there will be cells to service from previous windows, which means that the output windows have been shifted in respect to each other.) We then have:

**Proposition 5** The maximum number of cell locations in a block of memory used as shared output queues in a switch is $L_s = \left\lfloor \frac{N}{2} \right\rfloor (W + 1)$.

**Proof**

The aggregate arrival process from all $N$ inputs to the shared buffer is given by

$$B_{in}(n) = \sum_{j=1}^{N} \left\lfloor (n + \Delta_j)\frac{R_j}{W} \right\rfloor, \quad 1 \le n \le W, \qquad (33)$$

where $0 \le \Delta_j < \lceil W/R_j \rceil$. We will assume that the destinations of the cells are such that $R_i$ cells are delivered to output $i$ before any traffic is going to output $i+1$. The number of cells served from output $i$ is given by Eq. (23).

The highest accumulation of cells in the buffer occurs when they arrive at the maximal rate, and each output gets $W$ cells consecutively. The number of cells serviced by output $i$ is thus

$$B_i(n) = \left(n - \left\lceil \frac{(i-1)W}{N} \right\rceil \right)^+. \qquad (34)$$

The maximal amount of buffering is reached at the end of the window:

$$\begin{aligned} L_s &= \max NW - \sum_{i=1}^{N} B_i(W) \\ &= \max \left\lceil \frac{W}{N} \right\rceil + \cdots \left\lceil (N-1)\frac{W}{N} \right\rceil . \end{aligned} \qquad (35)$$

Note that

$$\left\lceil \frac{jW}{N} \right\rceil + \left\lceil \frac{(N-j)W}{N} \right\rceil \leq W + 1. \qquad (36)$$

With $N = 2n_1 + n_0$ $(n_1 \in \mathcal{N},\ n_0 \in \{0,1\})$, we get

$$\begin{aligned} &\left\lceil \frac{W}{N} \right\rceil + \cdots \left\lceil (N-1)\frac{W}{N} \right\rceil = \\ &(n_1 + n_0 - 1)(W+1) - \\ &(n_0 - 1)\left\lceil \frac{n_1 W}{2n_1 + n_0} \right\rceil \leq \\ &(W+1)\, n_1. \end{aligned} \qquad (37)$$

The maximum in Eq. (35) is then

$$L_s = \left\lfloor \frac{N}{2} \right\rfloor (W+1). \quad (38)\ \square$$

Many modern ATM switches combine input queuing with the more efficient output queuing. The buffer requirements for reserved traffic in this case have already been derived implicitly through the previous cases. The output queues buffer batches of cells for the pacing to avoid head–of–the–line blocking, while the input queues time–shift the arrival processes.

Worst case for the output queues is when the cells arrive in batches to the outputs and consequently need pacing. The case is the same as for demultiplexers in Proposition 2 and the cells must be transferred to the output buffers to avoid "head of the line" blocking. This means that the outputs need either $\lceil W/4 \rceil$ cells of buffer–space each, or $\lfloor N/2 \rfloor (\lceil W/N \rceil + 1)$ cells of shared buffer–space.

The worst case for the input queues is the one considered above in Proposition 4. Each input queue must consequently have $L = W - \lceil W/N \rceil$ cells of reserved spaces (Eq. (32)). To see this, assume that all inputs receives $W/N$ cells for the first output, then $W/N$ to the second, and so on. If we first serve input 1 for $W/N$ slots, then all other inputs are blocked. We can then serve both input 1 and 2 for the next $W/N$ slots (input 1 sends to output 2, and 2 sends to 1). Eventually we have an arrangement where all inputs are serviced at full rate (input 1 sends to output N, 2 to N–1 and so on). The input–buffer with most contents will contain almost a full window of cells.

The output–buffers may allow $N$ cells to be transferred per slot until filled. This means that $\lfloor N/2 \rfloor (\lceil W/N \rceil + 1)$ cells may move directly from the inputs to the shared output buffer. This amount may be subtracted from the needed input buffers and the requirement in then roughly $L = W - (3W/2N)$.

## 5 Delay

There are applications which require a bounded transfer delay. Video–data, for example, may be transferred over a network for instantaneous viewing and require timely delivery to be displayed in a perceptually continuous sequence. Audio has similar requirements and in both cases they result from the isochronal sampling rate that has to be maintained at the receiver.

Queueing delays in the network vary dynamically from cell to cell for a given route and must consequently be equalized before audio and video streams can be played–out. It is possible to limit the delay–jitter in the network, to remove it from an arriving cell stream at the ATM adaptation layer or at the application layer. Jitter control in the network comes at a cost which is either high minimum delay or more complex scheduling. Since not all applications require jitter–free service, we find that jitter removal is better done at the application layer where the bit–stream can be synchronized to the end–device [6].

Some delay variations are in our scheme removed by the buffering and pacing at the demultiplexers. The variations depend partly on the surplus traffic with low priority which uses the unused portions of the reserved capacity. These variations may be bounded by the amounts by which the buffer–sizes exceed the limits derived in the propositions.

There are also bounds on the maximal transfer delay from microphone and camera to loudspeaker and monitor when the audio and video are part of an interactive conversation or conference. Delay limits for conversational applications are set by human perception and determine when the delay starts to impede the information exchange. The limits follow those for voice conversations (ITU–T G.114: no impact below 100 ms and serious impact above 400 ms). There are

other applications for which delay matters; distributed computing and interactive games are two examples. There are, however, no clear guidelines to follow concerning the maximum permissible delay for such applications.

The maximum queuing delays for the capacity reservation scheme are as follows. The maximum multiplexing delay for a paced input stream with reserved rate $R_j/W$ is

$$D_{max} = \left\lceil \frac{W}{R_j} \right\rceil . \qquad (39)$$

A switch or demultiplexer output with reserved rate $R/W$ has maximum delay

$$D_{max} = W - \left\lceil \frac{R}{N} \right\rceil . \qquad (40)$$

$N$ is the number of inputs that send traffic to the considered output (always one for a demultiplexer).

The minimum delay is determined by the possible shift between the locations of reserved slots on an input link and an output link. It is zero if a cell can depart at the same time as it is being received (so called 'cut–through'), and cannot, in any case, exceed $\lceil W/R \rceil$ cells.

There is an engineering trade–off in determining the maximal length of $W$ between needed buffer–space and maximally incurred delay on one hand, and limitations in the granularity of reservations on the other. A limit of 1000 slots would mean that a switch with 16 ports would need roughly 8000 cells of buffers–pace totally, and that the delay could be 2.6 ms at 150.3 Mb/s port–rate (net rate of an SDH STM–1 channel). Reservations could be made in increments of 150 kb/s. This indicates that a lack of buffer–space in a switching–chip might be the main restriction on the length of $W$.

The limit on $W$ need not be uniform throughout the network. A long window may be needed at the network access to match closely the capacity that is actually reserved to the requested reservation. The maximal length could be shorter in the network where the reservation ratio is for an aggregate of connections.

## 6 Call–acceptance

The goal with the proposed traffic control is to simplify the call–acceptance decision, formulated in Eq. (3). The complexity of the acceptance procedure is greatly reduced by the deterministic upper–bounds on all traffic streams and by presetting the quality–level for all calls to absence of cell–loss. The call–acceptance is thus reduced to finding a path where the unreserved portion of capacity is larger than the capacity request of the call.

Although decision to accept or reject the establishment of a new connection is quite simple and could be taken fast, there is still delay incurred by updating the parameter values in all pacers along the route of the connection. We therefore sketch a speedier procedure.

The network controller reserves some portion $R_{res.}/W_{res.}$ of the capacity on each network link (possibly different for each one). This capacity is not wasted since any surplus capacity may be used by other connections (as described in Sec. 2).

To accept a call, the network control has to find a route to the destination such that on each link

$$\frac{R_{req.}}{W_{req.}} + \frac{R_{ex.}}{W_{ex.}} \le \frac{R_{res.}}{W_{res.}} . \qquad (41)$$

(The ratio denoted *'res.'* is the reserved, pre–allocated portion of the link's capacity; *'ex.'* is the existing, used part of that capacity and *'req.'* is the requirement of the new call.) If the inequality holds for all links on the route, then the call is accepted and the used capacity is updated:

$$\frac{R_{ex.}}{W_{ex.}} \leftarrow \frac{R_{ex.}}{W_{ex.}} + \frac{R_{req.}}{W_{req.}} . \qquad (42)$$

The call could be blocked if no such route exists. The pre–allocated capacity could alternatively be updated on one path, according to

$$\frac{R_{res.}}{W_{res.}} \leftarrow \frac{R_{res.}}{W_{res.}} + \frac{\varkappa}{W_{res.}} , \qquad (43)$$

for some integer $\varkappa$, so that Eq. (41) now can be met.

The increment of the pre–allocation in Eq. (43) may be done in anticipation of new connections to avoid blocking. When the connection–request of a rejected call is retried, there might in the meantime have been an up date in the reservations so that the call can be accepted. The pre–allocated capacity could of course be reduced if only a small portion of it is used, and no drastic increase is anticipated. (Increasing the ratio on one link normally implies that the ratio has to be lowered on another link entering the same multiplexer.)

The pre–allocation lowers the need for parameter updates in the pacers since most new calls may be handled according to Eq. (41). The update of $R_{ex.}/W_{ex.}$ in

(42) is only done at the network controller (the pacers operate with $R_{res.}/W_{res.}$). The time to set up a connection may thus be reduced, especially if the route partially uses a virtual path since that eliminates updates of address–tables in the switches.

The proposed procedure disassociates the call–acceptance decision from the actual reservation of capacity in the network. The time–consuming decision of how to balance loads over the network does not have to be determined each time a connection is requested.

## 7 Comparison with other schemes

There are two proposals for scheduling with deterministic performance guarantees in the literature that have received much attention: Packet–by–Packet Generalized Processor Sharing (PGPS) by Parekh and Gallager [8][9], and Stop–and–Go Queuing by Golestani [3]. We describe the two proposals briefly and point out any kinship to our proposal. (The reader is referred to [7] for a more extensive survey of techniques for providing quality guarantees.)

PGPS works as follows for ATM. A cell arriving at a switch is stamped with the time it would complete service according to Generalized Processor Sharing (GPS). The cells are scheduled to be transmitted in order of increasing time–stamps. The GPS assumes traffic and service rates to be infinitely divisible. Each connection has an associated weight $\Phi_j$ which guarantees a portion equal to $\Phi_j / \sum_n \Phi_n$ of the total capacity (the sum is over all connections on the link). When a connection does not use its allocation, the slack may be shared by the queued flows according to their weights. A "best effort" traffic class is assigned a weight and treated like any other flow.

The idea is to emulate a system where each flow has its own dedicated queue that is serviced at the allocated rate. Performance guarantees are possible for flows that are controlled by leaky–buckets at the network ingress. The papers give delay–bounds and buffer–bounds for both single–node and multinode networks.

Although PGPS gives tight buffer and delay bounds, we have nevertheless decided against using it because of the limited scalability of its realization. The complexity of implementation increases with the number of connections. There has to be one logical queue (linked–list) for each connection to hold the cells scheduled for departure. The scheduling decision requires computation of the completion time for each cell, and may therefore not scale well with increased link–capacity. The list of time–stamps for the cells awaiting service must be searched to insert the stamps of newly arrived cells. Most switches route cells according to physical port addresses added to the cells. The separation of flows would, however, require the switches to examine the VCI and VPI fields inside the cells. Note also that the weights for all links along a chosen route must be recomputed when a new connection is accepted.

The method we have proposed is akin to the Stop–and–Go queueing suggested by Golestani [3]. Stop–and–Go relies on a framing strategy: A connection is restricted to send at most $B$ cells of data in a period of length $T$ seconds (a frame in the author's terminology). The frame–length is the same throughout the network. A cell arriving in frame $i$ at a node will not be serviced until the start of the next frame, $i+1$. The service is consequently not work–conserving. The delayed service ensures that the flow does not get more clustered in the network. Note the difference to our capacity reservation scheme which does not have a fixed frame–size. Instead it has windows which may be different for each link, depending on the total capacity and portion of reserved capacity.

The minimum and maximum queuing delays are no more than $H \times T$ seconds apart, for a route $H$ hops long. The jitter is tightly bound within $2T$. The frame–length thus determines the worst–case delay through the network. A small $T$ gives a low delay, but limits the granularity of the capacity allocations which is one cell per frame. It may be alleviated by allowing multiple frame–lengths at the cost of increased complexity. There is one buffer needed per frame–length in every multiplexer and every port of a switch. Traffic belonging to a short frame is given non–preemptive priority over traffic in longer frames. We believe the pacers in our scheme to be simpler to implement than the servers for multi–frame Stop–and–Go, and we only require two buffers, one per traffic class.

Best effort traffic in Stop–and–Go may be serviced during the periods when controlled cells wait for the next frame to commence and receives a service which clusters the cells at the end of the frames. This may make the traffic more bursty and worsens its delay and loss performance.

A network operator may initially want to offer communication services with deterministic guarantees and best effort. Then gradually, further traffic classes may be introduced with statistical guarantees that replace the mere best–effort service. This might prove itself difficult with Stop–and–Go: the bursty service

that uncontrolled traffic gets exacerbates the already hard task of offering statistical guarantees. To avoid this problem, our capacity reservation scheme requires reserved cells to be evenly paced. This gives non–reserved connections a smooth service which would allow statistical guarantees to be offered in addition to the deterministic one.

The smooth pacing also reduces the needed buffer space from roughly $3W$ cells per output for Stop–and–Go to $W/2$ cells in our case. Golestani states that a buffer of length $3 \times T_{max} \times C_{link}$ bits is enough to avoid loss ($T_{max}$ is the longest of the frame lengths) and one may take $W = \frac{T \times C_{link}}{53 \times 8}$ to make the comparison. Buffer–space may be severely restricted when a switching element is implemented as a single or as a few VLSI–chips. Tight bounds are thus of merit.

## 8 Summary and final remarks

We have presented a capacity reservation scheme for ATM networks. A reservation ratio of $R/W$ of a link's capacity means that $R$ cells may maximally be sent over a window of $W$ slots. The cell departures are spaced evenly over the window. Since all flows are thus bounded, the minimum buffer–sizes could be determined to ensure that reserved traffic does not suffer from cell–loss due to multiplexing overloads.

ATM has since its conception been intended to offer statistical multiplexing with quality guarantees. It may now seem a step backwards to propose a capacity reservation which enforces a deterministic service rate for a source. First, it should be kept in mind that the reservations do not preclude the use of statistical multiplexing. Second, statistical multiplexing is not an end in itself. Its prime motivation has been to efficiently use the network capacity, a resource which now appears less scarce. Consider instead some of the advantages with capacity reservation over statistical multiplexing with quality guarantees:

- Implementation with low complexity
- Straight–forward call–acceptance procedure
- Loss–free operation with bounded delay
- High utilization of network capacity
- Sound basis for charging

The pacing function is performed per link and not per virtual circuit. The complexity of the implementation is thus independent of the number of simultaneously open circuits. The dual–buffers for the two traffic classes may be logical queues in shared memory, or accomplished by multiplexer or switching circuits operating in parallel, with the pacer as the arbiter.

Since the buffers are separate, they may be dimensioned differently. The buffer for connections with reservations must have a size in accordance with the limits previously derived. Any additional space should be included only if the delay limits allow it. Best effort connections, without reservations, could instead be given ample buffer–space to minimize the probability of cell–loss.

The user may send any amount of surplus traffic over the connection at own risk. Only the reserved amount of capacity is guaranteed. The surplus is allowed to soften the limit of the reservation at temporary capacity peaks. The reservations can use all of the network's capacity, if need be, but the reservation scheme allows "best effort" traffic to use slack in the reserved capacity and all of the non–reserved capacity. The network may therefore be well utilized. Note that the paced service of reserved traffic gives a smooth service and—in the absence of statistical guarantees—a truly best effort service to non–reserved traffic. In addition, the scheme does not hinder statistical guarantees to be given for the the unreserved part of the network's capacity.

The charge for a call may be based on the amount of reserved capacity, its duration and, possibly, the length of the route. Cells with no priority that are sent in addition over the connection would not be counted. This basis is more attractive than the charge after behavior (peak–to–mean ratio) that may result from the statistical traffic control.

The capacity reservation is most suited to near isochronal sources which have predictable rates. The performance guarantees are also most valuable for such sources, which include most video and audio sources, since the needed real–time delivery makes retransmission infeasible. Data, on the contrary, is commonly offered a "best effort" service and the reliability is added by retransmission at the transport layer. This service is offered by the non–reserved part of the network in the proposed scheme. Note, however, that the simplified call–acceptance procedure may allow sufficiently fast connection establishment to handle transfer of bulk–data, such as images.

Another use is to overlay TCP/IP on ATM. Since the retransmission unit is a packet but the loss unit is a cell, there is a risk that the network delivers cells but few complete packets [10]. The system may then be

unstable: cell–loss gives retransmissions which in turn increase the load and potentially also the cell–loss. A reservation for TCP/IP traffic would ensure that loss in the system would only be at the packet–level in the routers that are interconnected by the ATM–network. A similar use is to reserve capacity for signalling channels so that necessary commands may be get through even during periods of congestion.

The criticism given by Lea [4] of the pretension to offer statistical multiplexing in ATM with performance guarantees is convincing and forms the justification for our proposal of deterministic service and quality. Also the study of statistical multiplexing done by Bonomi et al. [1] reveals the challenges offered by multiplexing of heterogeneous sources, by transient behavior and difficulties in characterizing flows after multiplexing.

The described procedure for capacity reservation is intended to be a practicable method for traffic control. We have strived for simplicity of implementation and deterministic quality guarantees. Much of this is enabled by the slot–like operation of ATM, with its fixed–sized cells which simplifies scheduling. We have also attempted to give the non–reserved traffic a smooth and truly best–effort service, without delays due to reserved traffic, which ensures that statistical guarantees may be offered in the future.

## 9 References

[1] F. Bonomi, et al., "A Further Look at Statistical Multiplexing in ATM Networks," Computer Networks and ISDN Systems, Vol. 26, No. 1, September 1993, pp. 119–138.

[2] M. W. Garrett and W. E. Leland, "Analysis, Modeling and Generation of Self–Similar VBR Video Traffic," ACM Computer Communications Review, Vol. 24, No. 4, October 1993, pp. 269–280.

[3] S. J. Golestani, "A Stop–and–Go Queueing Framework for Congestion Management," in *Proceedings of ACM Sigcomm '90*, Philadelphia, Pennsylvania, September 1990, pp. 8–18.

[4] C–T Lea, "What Should Be the Goal for ATM," IEEE Network, September 1992, pp. 60–66.

[5] W. E. Leland, et al., "On the Self–Similar Nature of Ethernet Traffic," ACM Computer Communications Review, Vol. 23, No. 4, October 1993, pp. 183–193.

[6] G. Karlsson, "ATM Adaptation for Video," in *Proceedings of Sixth International Workshop on Packet Video*, Portland, OR, September 26–27, 1994, pp. E3.1–5.

[7] J. Kurose, "Open Issues and Challenges in Providing Quality of Service Guarantees in High–Speed Networks," ACM Computer Communication Review, Vol. 23, No. 1, January 1993, pp. 6–15.

[8] A. K. Parekh and R. G. Gallager, "A Generalized Processort Sharing Approach to Flow Control in Integrated Services Networks: The Single–Node Case," IEEE/ACM Transactions on Networking, Vol. 1, No. 3, June 1993, pp. 344–357.

[9] A. K. Parekh and R. G. Gallager, "A Generalized Processort Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case," IEEE/ACM Transactions on Networking, Vol. 2, No. 2, April 1994, pp. 137–150.

[10] A. Romanow and S. Floyd, "Dynamics of TCP Traffic over ATM Networks," ACM Computer Communications Review, Vol. 24, No. 4, October 1994, pp. 79–88.