Tustralian Broadband Symposium 7/92

Bandwidth Management in ATM Networks Using Fast Buffer Reservation

Jonathan S. Turner
Washington University

June 3, 1992

ABSTRACT

This paper describes a method for managing the allocation of bandwidth and controlling congestion in ATM networks. The method described provides a complete solution to the problem of efficient resource management in the presence of bursty traffic. The method relies on a technique called *fast buffer reservation* in which space in link buffers is allocated "on-the fly" to user information bursts at the time the bursts occur, ensuring that a burst, if accepted, has extremely high probability of being delivered intact. We also describe a fast call acceptance algorithm that ensures that the probability of burst rejection is acceptably small. We conclude that the method is practical and adds only a small incremental cost to an ATM switching system. We restrict ourselves here to the method's application to point-to-point and one-to-many virtual circuits, but note that it can also be extended to many-to-many virtual circuits (that is, virtual circuits with multiple transmitters, as well as multiple receivers).

I. INTRODUCTION

A central objective in ATM networks is to provide virtual circuits that offer consistent performance in the presence of stochastically varying loads on the network. This objective can be achieved in principle, by requiring that users specify traffic characteristics when a virtual circuit is established, so that the network can select a route that is compatible with the specified traffic and allocate resources as needed. While this does introduce the possibility that a particular virtual circuit will be blocked or delayed, it allows established virtual circuits to receive consistent performance as long as they remain active.

Bandwidth management and congestion control is a topic that has received a tremendous amount of attention of late. While we cannot give a complete survey, a small sample of the recent progress may be useful. The central problem of understanding the queueing behavior of bursty data traffic has been studied by many authors. Anick, Mitra and Sondhi [1] made one of the earliest contributions, obtaining analytical models for homogeneous traffic by treating the flow of cells from each active bursty source as a continuous fluid flow. Others have extended this work to multiple source types and to different bursty source models. A few examples of such work can be found in [5, 8, 11, 12]. While such methods are useful, computational constraints prevent their use in making virtual circuit multiplexing decisions in real-time. Several researchers have attempted to develop more comprehensive bandwidth management schemes for high speed networks (see, for example [2, 3, 6, 7, 9]), but none of the proposals to date offers a complete solution to the problem.

Ideally, a bandwidth management and congestion control mechanism should satisfy several competing objectives. First, it should provide consistent performance to those applications that require it, regardless of the other virtual circuits with which a given virtual circuit may be multiplexed. Second, it should allow high network throughputs even in the presence of bursty traffic streams. (For typical file transfer applications, a single burst may be hundreds of kilobytes or megabytes long; that is, there may be more than 10,000 ATM cells in a burst, while the buffers in the switching systems will typically contain room for only a few hundred of cells.) Third, the specification of traffic characteristics should be simple enough that users can develop an intuitive understanding of the specifications and flexible enough that inaccurate specifications don't have seriously negative effects on the user. Fourth, it should not artificially constrain the characteristics of user traffic streams; the need for flexibility in ATM networks makes it highly desirable that traffic streams be characterized parametrically, rather than by attempting to fit them into a pre-defined set of traffic classes. Fifth, it must admit a simple realization for reasons of economy and reliability. All proposals we have seen for connection

This work was supported by Bell Communications Research, Bell Northern Research, Digital Equipment Corporation, Italtel SIT, NEC America, NTT and SynOptics Communications.

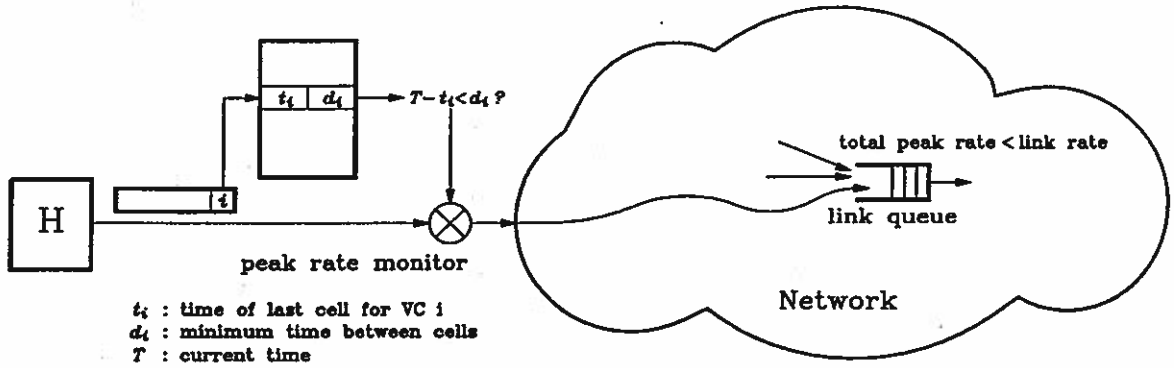


Figure 1: Peak Rate Allocation

management in ATM networks have serious deficiencies with respect to at least one of these objectives.

In the remainder of this section, we review three approaches to the bandwidth management problem that have been proposed and studied by various groups. Our purpose here is to illustrate three distinctly different approaches, identify their strengths and weaknesses so that we can make use of them in synthesizing a new approach, which will be described in detail in the body of the paper.

Peak Rate Allocation

In this approach, the user simply specifies the maximum rate at which cells are to be sent to the network, and the network routes virtual circuits so that on every link, the sum of the peak rates of the virtual circuits using that link is no more than the link's maximum cell rate. The network also must supply a mechanism to monitor the rate at which the user actually sends cells and in the event that the user exceeds the specified rate it may do one of three things; discard the offending cells, mark them by setting a bit in the header informing switches along the virtual circuit path that the marked cell can be discarded if the presence of congestion requires that something be discarded, or flow control the user. Figure 1 illustrates this method. Note that in the peak rate monitor, the illustrated lookup table records, for virtual circuit i, a minimum inter-cell spacing d_i and the time the most recent cell was transmitted, t_i. By subtracting t_i from the current time T, the monitor can decide whether or not to pass the cell unchanged to the network. Note that in this simple approach, the user's peak rate must be of the form R/j where R is the link rate (typically 150 Mb/s) and j is an integer. This restriction can be easily eliminated (see [14]).

Peak rate allocation offers a very strong performance guarantee, is easy for users to understand and specify and admits a very straightforward implementation. Its obvious drawback is that it makes poor use of network bandwidth in the presence of bursty traffic.

Minimum Throughput Allocation

In this approach, the user specifies the throughput that is needed when the network is congested. The user is free to exceed this rate whenever desired, but the network guarantees only the specified throughput. One way to implement this is for the network to allocate slots in link buffers for virtual circuits in direct proportion to their required throughput. Thus if a given virtual circuit requires 20% of the link's bandwidth, it is allocated 20% of the buffer slots. This allocation only comes into play during overload periods. During those overload periods, each virtual circuit has access only to its buffer slots and any excess cells may be discarded.

Virtual circuit routing ensures that the sum of the minimum required throughputs does not exceed the link bandwidth. To implement the scheme, it is necessary to track the number of buffer slots in use by each virtual circuit and mark cells if the number of buffer slots already in use is equal to the virtual circuit's allocation. The buffer controller must also have the ability to discard marked cells if an unmarked cell arrives at a time when the buffer is full. This is illustrated in Figure 2, where the table entry B_i is the number of buffer slots allocated to virtual circuit i and b_i is the number of buffer slots currently in use by unmarked cells belonging to virtual circuit i.

This approach is easy to specify and can provide high efficiency. The implementation, while more complex than peak rate allocation need not be excessively complex. The drawback is that the performance guarantee is rather weak. The network can't promise anything about throughput in excess of that requested, since it has no advance knowledge of to what extent users will transmit information in excess of their required throughput. Users with bursty traffic streams, but needing all or almost all of their data to get through, regardless of other traffic, can specify a high required throughput, but this essentially leads

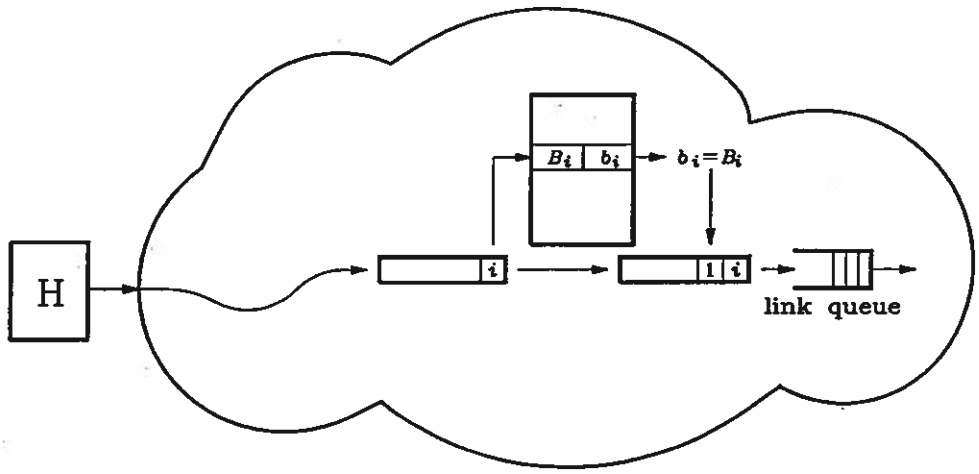


Figure 2: Minimum Throughput Allocation

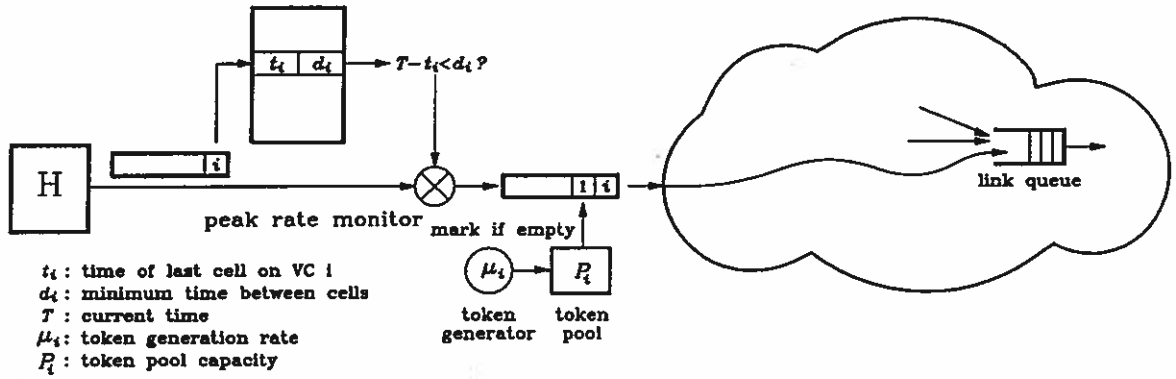


Figure 3: Bursty Traffic Specification and Allocation

to peak rate allocation.

Bursty Traffic Specification and Allocation

In this approach, the user specifies a peak cell rate, an expected average cell rate and a maximum burst size. The network uses these parameters to configure a peak rate monitor (as described above), and a per virtual circuit token pool at the interace to the network; this is shown in Figure 3. Whenever the user transmits a cell, a token is consumed from the token pool. If there are no tokens available, the cell can be marked for preferential discarding in the event it encounters congestion and a flow control cell returned to the user, allowing the user to defer further transmissions. Tokens are replenished at the user's specified average rate with the maximum number of tokens in the token pool limited by the specified maximum burst size. When routing virtual circuits, the network must be able to decide if a given set of virtual circuits can be safely multiplexed together; that is, if multiplexing a given set of virtual circuits with known traffic parameters will result in an acceptable cell loss rate or not.

This approach allows performance guarantees to be traded off against link efficiency and traffic burstiness. It is somewhat more difficult for users to understand and specify, but is still reasonable since the consequences of an incorrect specification are not serious. The excess traffic is simply not guaranteed (but may still get through) and since the user is informed of the excess traffic, he can either modify the dynamic behavior of the traffic or request that the network adjust the traffic parameters. The main drawback of this approach is that there are currently no computationally effective ways to decide when a new virtual circuit can be safely multiplexed with other virtual circuits specified in this manner. (To allow rapid call setup, the determination of whether a given call can be safely added to a link must be made in at most a few milliseconds. Current computational methods

are at least several orders of magnitude away from this objective.)

Comments

There are two other deficiencies that the above approaches suffer from in varying degrees. First, because cell marking and discarding is done on a cell basis rather than on a burst basis, it is necessary to have very low cell loss rates in order to achieve acceptable burst loss rates. This is particularly problematic in the context of the very small cells that have been standardized for use in ATM networks. Since end-to-end protocols will typically operate on the basis of much larger data units, comprising many ATM cells, it would be desirable for an overloaded network to react by discarding cells from as few virtual circuits as possible, rather than discarding cells indiscriminately from all virtual circuits. This problem is illustrated in Figure 4, which shows a single lost cell in a burst, resulting in retransmission of the entire end-to-end data unit. A second limitation of the above schemes is that they are not sufficient in and of themselves to handle multicast virtual circuits in which there can be multiple sources. In such virtual circuits traffic streams from different virtual circuits can flow together and some explicit mechanism is required to monitor and control these converging flows.

In the remainder of the paper, we describe a collection of mechanisms that provides a complete, technically feasible and economically implementable approach to bandwidth management and congestion control for point-to-point and one-to-many virtual circuits. This approach can be extended to multi-source virtual circuits as well; while multi-source virtual circuits are beyond the scope of this paper, the interested reader can find details of this extension in [14].

II. Fast Buffer Reservation

In this section we describe a complete set of mechanisms for point-to-point virtual circuits and for multicast circuits with a single transmitter. As mentioned in the introduction, one crucial problem with most earlier approaches to bandwidth management and congestion control is that they do not directly address the need to allocate network resources to traffic bursts in order to preserve the integrity of the burst as a whole. One exception can be found in reference [2], which mentions a fast bandwidth reservation scheme to handle burst traffic with low peak rates. We have adopted a similar approach, but apply it to the more difficult case of bursty traffic with peak rates that can be a large fraction of the link bandwidth. We also adopt an implementation in which the reservation is made as the data is sent. This eliminates the need for explicit control messages, simplifying the implementation and allowing more rapid response to user traffic. Furthermore, we integrate the buffer reservation idea into a larger framework that provides a comprehensive solution to the bandwidth management problem.

To preserve the integrity of user information bursts, the network must detect and track activity on different virtual circuits. This is accomplished by associating a state machine with two states with each virtual circuit passing through a given link buffer. The two states are *idle* and *active*. When a given virtual circuit is active, it is allocated a prespecified number of buffer slots in the link buffer and it is guaranteed access to those buffer slots until it becomes inactive, which is signalled by a transition to the idle state. Transitions between the active and idle states occur upon reception of user cells marked as either *start-of-burst* or *end-of-burst*. Other cell types include *middle-of-burst* and *loner*, the latter is used to designate a low priority cell that is to be passed if there are unused buffer slots available, but which can be discarded if necessary. A forced transition from active to idle is also made if no cell is received on the virtual circuit within a fixed timeout period.

Figure 5 illustrates the buffer reservation mechanism. For virtual circuit i, the mechanism stores the number of buffer slots needed when the virtual circuit is active (B_i), the number of buffer slots used by unmarked cells (b_i) and a state variable (s_i: idle, active). The mechanism also keeps track of the number of unallocated slots in the buffer (B). The detailed operation of the state machine for virtual circuit i is outlined below.

When a start cell is received:

- If the virtual circuit is in the idle state and $B - B_i < 0$, the cell is discarded.
- If the virtual circuit is in the idle state and $B - B_i \geq 0$, s_i is changed to active, a timer for that virtual circuit is set and B_i is subtracted from B. If $b_i < B_i$, b_i is incremented and the cell is placed (unmarked) in the buffer. If $b_i = B_i$, the cell is marked and placed in the buffer.

If a start or middle cell is received while the virtual circuit is in the active state, it is queued and the timer is reset. The cell is marked, if upon reception, $b_i = B_i$, otherwise it is left unmarked and b_i is incremented.

If a middle or end cell is received while the virtual circuit is in the idle state, it is discarded.

If an end cell is received while the virtual circuit is active or if the timer expires, s_i is changed from active to idle and B_i is subtracted from B.

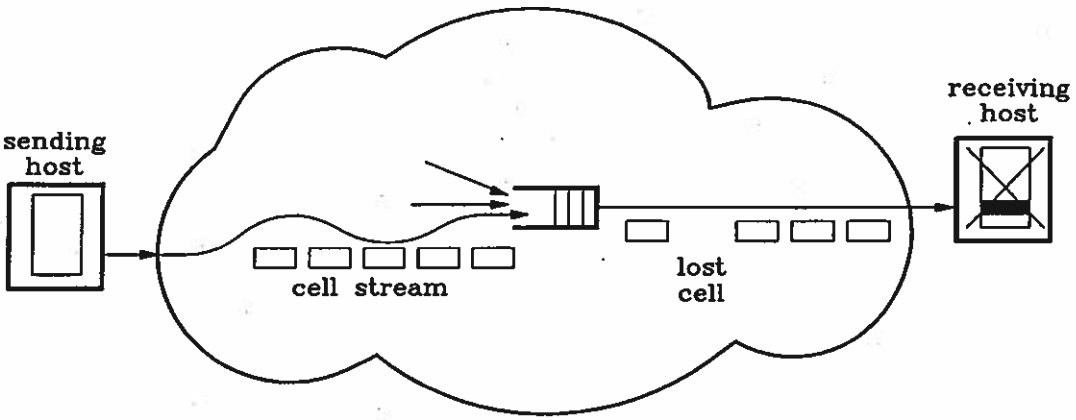
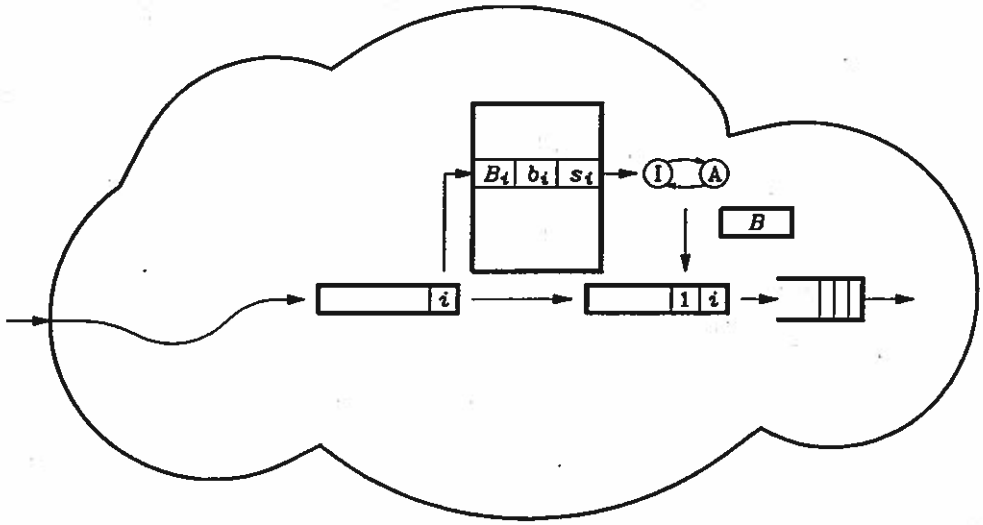


Figure 4: Effect of Cell Discarding on Data Bursts



B_i : number of buffer slots for VC i when active
b_i : number of buffer slots currently used by VC i
s_i : state of VC i (active=A or idle=I)
B : number of unallocated buffer slots

Figure 5: Fast Buffer Reservation

If a loner is received, it is marked and placed in the buffer.

Whenever a cell is sent from the buffer, the appropriate b_i is decremented (assuming the transmitted cell was unmarked).

The duration of the timeout which forces the eventual return to the idle state is determined primarily by the delay variation in the network. An interval of a few hundred cell transmission times appears to be about right. This translates to less than 1 ms for ATM cells and 150 Mb/s transmission links. Note that the choice of timeout puts a lower bound on the peak rate of those virtual circuits to which the fast buffer reservation mechanism is applied. We propose to use explicit buffer reservation only for virtual circuits whose peak rates exceed 1–2% of the link rate, and rely on simpler mechanisms for the remainder.

In the most common case, the end-to-end protocol would send a burst of the form `sm ...me` where `s` denotes a start cell, `m` a middle cell and `e` an end cell. Notice that in this case, if when the start cell arrives, there are not enough unallocated buffer slots to accommodate the burst, the entire burst is discarded. The state machine has been designed to allow other options as well. For example, a burst of the form `ss...se` is permissible. In this case, the state machine would attempt to allocate the buffer slots every time it received a start cell, so even if part of the burst was lost, at least the tail end of it is likely to get through. This type of behavior might be preferred for voice circuits, for example, where clipping of a talk spurt might be acceptable but deletion of an entire talk spurt would not be. Another allowed option is

a burst of the form **sm...msm...msm ...me**. A burst of this form could be used to transport a large file, where the end-to-end transport protocol performs retransmission on packets that are much larger than a single cell but smaller than the entire file.

Note that the buffer reservation mechanism can be applied directly to constant rate virtual circuits as well as to bursty virtual circuits. Such virtual circuits would simply be made active initially and remain active all the time. We can accomplish this effect without an explicit table entry for the constant rate circuits by simply subtracting B_i from the buffer-slots-available register B at the start of the call. For bursty virtual circuits with a small peak rate (say, less than 2% of the link rate), we can use a similar strategy.

When a virtual circuit is routed, the software that makes the routing decisions attempts to ensure that there is only a small probability that the instantaneous demand for buffer slots exceeds the buffer's capacity. This probability is called the *excess buffer demand probability* and might typically be limited to say .01 or .001. We'll show in the next section how fast virtual circuit routing decisions can be made, while bounding the excess buffer demand probability.

First however, it is worth considering a variation on the buffer reservation scheme. As described, the mechanism requires two bits of the ATM cell header to encode the cell type (loner, start, middle, end). The CLP and RES bits of the current ATM cell header can reasonably be redefined for this purpose.

Another approach is to avoid non-standard header bits by using just the CLP bit and interpreting it as follows.

- When in the idle state, a cell with the CLP bit set (indicating a discardable cell) is treated as a loner; a cell with CLP cleared is treated as a start cell.
- When in the active state, a cell with the CLP bit set is treated as an end cell and a cell with CLP cleared is treated as a middle cell.

The obvious advantage of this approach is that it avoids non-standard headers. The drawbacks are first, that loners (or more generically, low priority cells) cannot be sent during a burst and second, the network cannot filter out clipped bursts, meaning that the end-to-end transport protocols must receive and discard such bursts as appropriate. While we recognize both of these as legitimate approaches, we prefer explicit start, middle, end and loner cells.

III. Virtual Circuit Acceptance Algorithm

When selecting a route for a virtual circuit, it is necessary for the control software to decide if a new virtual circuit can be safely multiplexed with a given set of pre-existing virtual circuits. We consider two methods for making this decision. The first method considers a given set of virtual circuits to be acceptable if at a random instant, the probability is small that the set of virtual circuits requires more buffer slots than are available This is called the *excess demand probability*.

To make this precise, let λ_i denote the peak data rate of a given virtual circuit and let μ_i denote the average rate. If the link rate is R and the buffer has L buffer slots, the number of slots required by an *active source* with peak rate λ_i is defined to be $B_i = \lceil L\lambda_i/R \rceil$. Note that B_i is determined by the virtual circuit's peak rate, not the burst length. Since B_i buffers are allocated to a virtual circuit when it is active, the virtual circuit's instantaneous buffer requirement is either 0 or B_i. If we let x_i be a random variable representing the number of buffer slots needed by virtual circuit i at a random instant then

$$\Pr(x_i = B_i) = \mu_i/\lambda_i \qquad \Pr(x_i = 0) = 1 - \mu_i/\lambda_i$$

Consider then, a link carrying n virtual circuits with instantaneous buffer demands $x_1, \ldots, x_n$. Define $X = \sum_{i=1}^{n} x_i$. Note that X represents the total buffer demand by all the virtual circuits. The probability distribution for X can be most conveniently described using a generating function. Letting $p_i = \mu_i/\lambda_i$, $\overline{p}_i = 1 - p_i$ and

$$f_X(z) = \prod_{i=1}^{n} (\overline{p}_i + p_i z^{B_i}) = C_0 + C_1 z + C_2 z^2 + \cdots$$

we note that $\Pr(X = j) = C_j$, assuming that the x_i are mutually independent. The excess demand is then just $C_{L+1} + C_{L+2} + \cdots$. When deciding if a given set of virtual circuits can be safely multiplexed, we check to ensure that the excess demand probability is less than some fixed bound ϵ.

To compute the excess demand probability, we need to maintain the probability distribution for X in the form of a list of the coefficients $C_0, C_1, \ldots$ We can then decide if the given set of virtual circuits can be safely multiplexed by checking that $C_0 + \cdots + C_L \geq 1 - \epsilon$. (To allow for jitter in the network, we might choose to hold some buffer slots in reserve; this can be accomplished by initializing the available buffer slots (B) to some value $L' < L$ and requiring that $C_0 + \cdots + C_{L'} \geq 1 - \epsilon$.) We obtain the coefficients C_j in an incremental fashion, updating the coefficients whenever a new virtual circuit is added to a link. Suppose we are adding a new virtual

circuit with buffer demand x_{n+1} to a link carrying n virtual circuits with buffer demands $x_1, \ldots, x_n$ and total buffer demand X. If $X' = X + x_{n+1}$, then

$$f'_{X'}(z) = \prod_{i=1}^{n+1} (\overline{p}_i + p_i z^{B_i}) = C'_0 + C'_1 z + C'_2 z^2 + \cdots$$

is the desired distribution. Given the original coefficients C_j, we can easily compute the new coefficients C'_j using the equation

$$C'_j = C_j \overline{p}_{n+1} + C_{j-B_{n+1}} p_{n+1}$$

with the understanding that $C_j = 0$ for $j < 0$. To recover the old coefficients when virtual circuit $n+1$ is disconnected, use the equation

$$C_j = (C'_j / \overline{p}_{n+1}) - (C_{j-B_{n+1}} p_{n+1} / \overline{p}_{n+1})$$

computing C_0 first, then C_1 and so forth.

The time required to compute the new coefficients is determined primarily by the number of coefficients that must be maintained. Given that we accept calls only when the excess demand probability is small, the coefficients C_j with $j > L$ are necessarily small. Hence we can reasonably neglect coefficients C_j where $j >> L$. So for example, given a buffer of size 256, we can probably safely neglect the coefficients with $j > 1000$. In this case, we must perform about 1000 additions and 2000 multiplications to compute the new coefficients. To check the excess demand probability another 256 additions are required. This computation can be completed in under half a millisecond using a typical 10 MIPS processor.

Unfortunately, the excess demand criterion is subject to certain anomalies. For example, consider two virtual circuits, both of which require all the buffer slots when busy and suppose the first is active with probability .9, the second with probability .01. The excess buffer demand probability in this case is .009, but whenever the second virtual circuit becomes active, it finds the buffer busy 90% of the time, meaning that only 10% of its bursts are successful. If we used the excess buffer demand probability with a bound of .01 as our call acceptance criterion, this would be considered acceptable, whereas it would almost certainly be unacceptable to the second virtual circuit.

We define the *burst loss probability* of virtual circuit i to be the probability that when source i attempts to send a burst, the desired buffers are not available. This is bounded above by $\Pr(X - x_i > L - B_i)$. We call this quantity the *contention probability* for virtual circuit i. Note that

$$\begin{aligned} \Pr\{X = L\} &= \overline{p}_i \Pr\{X - x_i = L\} \\ &\quad + p_i \Pr\{X - x_i = L - B_i\} \end{aligned}$$

So,

$$\begin{aligned} \Pr\{X > L\} &= \overline{p}_i \Pr\{X - x_i > L\} \\ &\quad + p_i \Pr\{X - x_i > L - B_i\} \\ \Pr\{X - x_i > L - B_i\} &= \frac{1}{p_i} \Pr\{X > L\} \\ &\quad - \frac{\overline{p}_i}{p_i} \Pr\{X - x_i > L\} \\ &\leq \frac{1}{p_i} \Pr\{X > L\} \end{aligned}$$

Thus, so long as p_i is not too small, the excess demand probability is not too much larger than the contention probability, but as we have seen, the two can differ by a large amount if p_i is small. Fortunately, when p_i is small we can obtain tight bounds on the contention probability without too much additional computation.

As before, let $X = x_1 + \cdots + x_n$ and assume we have the corresponding coefficients C_j, but we want to determine if the contention probability for virtual circuit i is no more than some bound ϵ. For any $j \geq 0$,

$$\begin{aligned} \Pr\{X - x_i > j\} &= \frac{1}{\overline{p}_i} \Pr\{X > j\} \\ &\quad - \frac{p_i}{\overline{p}_i} \Pr\{X - x_i > j - B_i\} \end{aligned}$$

Consequently, for any $k \geq 0$,

$$\begin{aligned} &\Pr\{X - x_i > j\} \\ &\quad = \frac{1}{\overline{p}_i} \sum_{h=0}^{k-1} \left(-\frac{p_i}{\overline{p}_i}\right)^h \Pr\{X > j - hB_i\} \\ &\qquad + \left(-\frac{p_i}{\overline{p}_i}\right)^k \Pr\{X - x_i > j - kB_i\} \\ &\quad = (Z_k / \overline{p}_i) + (-1)^k R_k \end{aligned}$$

where $Z_k = \sum_{h=0}^{k-1} (-p_i/\overline{p}_i)^h \Pr\{X > j - hB_i\}$ and $R_k = (p_i/\overline{p}_i)^k \Pr\{X - x_i > j - kB_i\}$. If we let $k = 1 + \lfloor j/B_i \rfloor$, $R_k = 1$ and we're left with an exact expression for the $\Pr\{X - x_i > j\}$ in terms of the given distribution. If we substitute $L - B_i$ for j, we get the contention probability. To compute the contention probability in this way, it's helpful to have available $S_j = C_0 + \cdots + C_j$ for $j \leq L$. Using these, we have $Z_k = \sum_{h=0}^{k-1} (-p_i/\overline{p}_i)^h (1 - S_{j-hB_i})$.

To compute the contention probability for virtual circuit i then, we need to perform roughly $2L/B_i$ additions and the same number of multiplications. For virtual circuits with a large peak rate, L/B_i is small, so we can afford to perform the computation exactly; but for virtual circuits with a small peak rate, the time required for the exact computation may be excessive. Fortunately, we rarely need to perform the full computation. Note that for all even $k \geq 0$,

$$(Z_k / \overline{p}_i) + R_k = \Pr\{X - x_i > j\}$$

$$= (Z_{k+1}/\overline{p}_i) - R_{k+1}$$
$$(Z_k/\overline{p}_i) \leq \Pr\{X - x_i > j\} \leq (Z_{k+1}/\overline{p}_i)$$

and since

$$|Z_{k+1} - Z_k| = \left(\frac{p_i}{\overline{p}_i}\right)^k \Pr\{X > j - kB_i\} \leq \left(\frac{p_i}{\overline{p}_i}\right)^k$$

we can get a good approximation to the contention probability for modest values of k when p_i is not too large. Moreover, since all we need to do to make the call acceptance decision is determine if the contention probability is below some fixed bound ϵ, we can halt the computation early if any of the upper bounds are $\leq \epsilon$ or if any of the lower bounds are $> \epsilon$. If we compute contention probabilities only for virtual circuits with $p_i \leq 1/3$, we expect that the number of terms that must be computed for these virtual circuits will average no more than about five, implying at most 20 arithmetic operations are needed for each such virtual circuit. Even for 1000 virtual circuits of this type, the time required on a 10 MIPS processor would be no more than a couple milliseconds.

Note that the call acceptance decision does not depend directly on the length of the user's information bursts. This is useful, since the burst length is perhaps the single most difficult parameter for users to quantify. While burst length does not affect the buffer demand, the time duration of bursts does affect the duration of excess demand periods. Because the call acceptance decision is not based on burst length, it ensures only that excess demand periods are rare. It does not limit their length in any way. Fortunately, the length of the excess demand periods is a much less critical performance parameter and so a loose control method is probably sufficient. One simple method is just to limit the time duration of a burst. For example if all bursts are limited to a maximum duration of τ, the probability that an excess demand period exceeds τ is small. Applications that require bursts of duration longer than τ could request an additional virtual circuit to accommodate the end of a burst, while the first part of the burst is sent using a preestablished virtual circuit. If τ is in the range of 1–10 seconds, the added signaling overhead is likely to be acceptable and the length of the excess demand periods should be short enough for the applications.

We can apply the analysis above to evaluate the multiplexing efficiency of virtual circuits with different performance requirements. For the homogeneous case (all virtual circuits have the same requirements) the computation of the contention probability is particularly simple. Define the following variables.

B = average burst size (bytes)
t = burst transfer time requirement (sec)
T = average time between bursts (sec)
R = link rate (bits/sec)
ϵ = bound on contention probability

For ATM networks with 53 byte cells, 46 of which carry user information, the source peak rate $\lambda = 8B(53/46)/t$, the average rate $\mu = 8B(53/46)/T$ and the probability that any given source is active, $p = \mu/\lambda$. We also define m to be the maximum number of *simultaneously active* virtual circuits that the link can support, M to be the number of virtual circuits that can be multiplexed on the link and E to be the effective data rate of the virtual circuit. Then

$$m = \lfloor R/\lambda \rfloor$$
$$M = \text{largest } k \geq m \text{ such that } \epsilon \geq 1 - \sum_{i=0}^{m-1} \binom{k-1}{i} p^i (1-p)^{k-i}$$
$$E = \min\{\lambda, R/M\}$$

Note that in the computation of M, the right hand side of the inequality is the contention probability. We define the multiplexing efficiency to be $M\mu/R$ and the multiplexing gain to be M/m.

For example, if B = 100 KB, t = .1 sec, T = 10 sec, R = 150 Mb/s and ϵ = .01, then λ = 9.2 Mb/s, μ = 92 Kb/s, p = .01, m = 16, M = 822 and E = 183 Kb/s. So the link can support 822 virtual circuits of this type as opposed to 16, which would be the limit if peak rate allocation were used. This yields a multiplexing efficiency of about 50% and a gain of about 51.

As a second example, let B = 1 MB, t = .4 sec, T = 5 sec, R = 150 Mb/s and ϵ = .01. Then, λ = 23 Mb/s, μ = 1.8 Mb/s, p = .08, m = 6, M = 24 and E = 6.5 Mb/s. So the link can support 24 virtual circuits of this type rather than the six it would support using peak rate allocation. The resulting multiplexing efficiency is about 29% and the gain is 4.

Figure 6 shows curves of multiplexing efficiency for different peak rates (expressed as a fraction of the link rate) and different peak-to-average ratios. The upper plot is for ϵ = .01 and the lower one for ϵ = .001. Note that for peak rates (1/4) or less and ϵ = .01, we can achieve efficiencies of better than 20%, even for very large peak-to-average ratios. If ϵ = .001, we need to limit peak rates to about 1/6 to achieve similar efficiencies.

Figure 7 gives a set of similar curves, showing the gain relative to peak rate allocation. The upper plot is for ϵ = .01 and the lower one for ϵ = .001. Here, note that dramatic improvements are possible, even for virtual circuits with large peak rates.

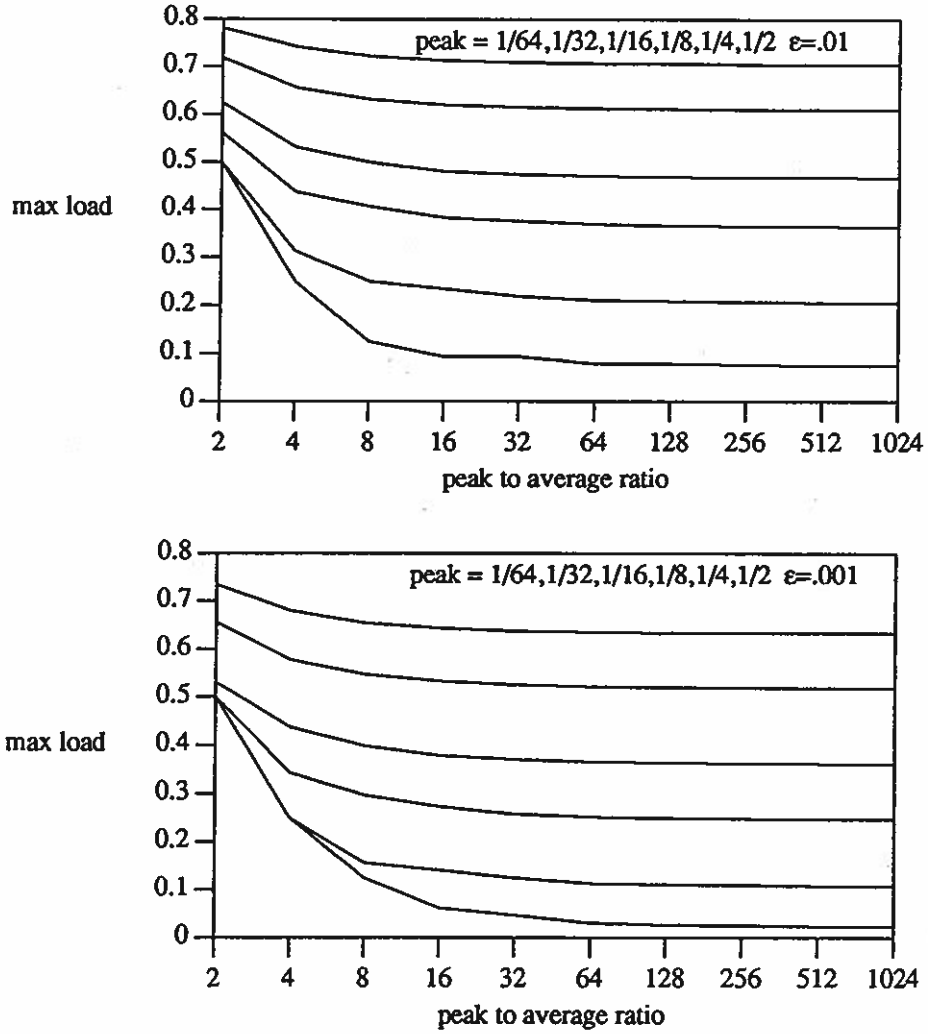


Figure 6: Multiplexing Efficiency for Homogeneous Virtual Circuits

IV. Usage Monitoring

Because call acceptance decisions are based on the network's knowledge of the user's peak and average data rate, the network must monitor actual usage during a call to ensure that the user does not exceed the stated bounds. Note that the buffer reservation mechanism already described effectively monitors the peak rate, since if a user transmits at greater than the peak rate, the excess cells are not certain to pass through the link buffer. To monitor average usage, we need an additional mechanism at the interface between the network and a host or terminal, connected to the network. The token pool mechanism described earlier can be adapted to this purpose. We augment the token pool with a state machine that mimics the state machines at the link buffers. Recall that when the state machine at the link buffer is active, B_i buffer slots are allocated to the virtual circuit, allowing the virtual circuit to transmit at its peak rate. To account for this properly, the state machine at the interface must cause tokens to be consumed at the peak rate, regardless of the user's actual transmission rate. This is illustrated in Figure 8. The operation of the modified token pool is detailed below.

If a start cell is received on virtual circuit i while the virtual circuit is in the idle state:

- If the number of tokens in the token pool, C_i, is ≤ 0 the cell is discarded.
- If $C_i > 0$ the state s_i is changed to active, and a timer for that virtual circuit is set.

So long as the state machine is in the active state, tokens are removed from the token pool at the peak rate λ_i. This may cause the token pool contents to become negative.

If a start or middle cell is received while the virtual circuit is in the active state and $C_i > 0$, it is passed and the timer is reset.

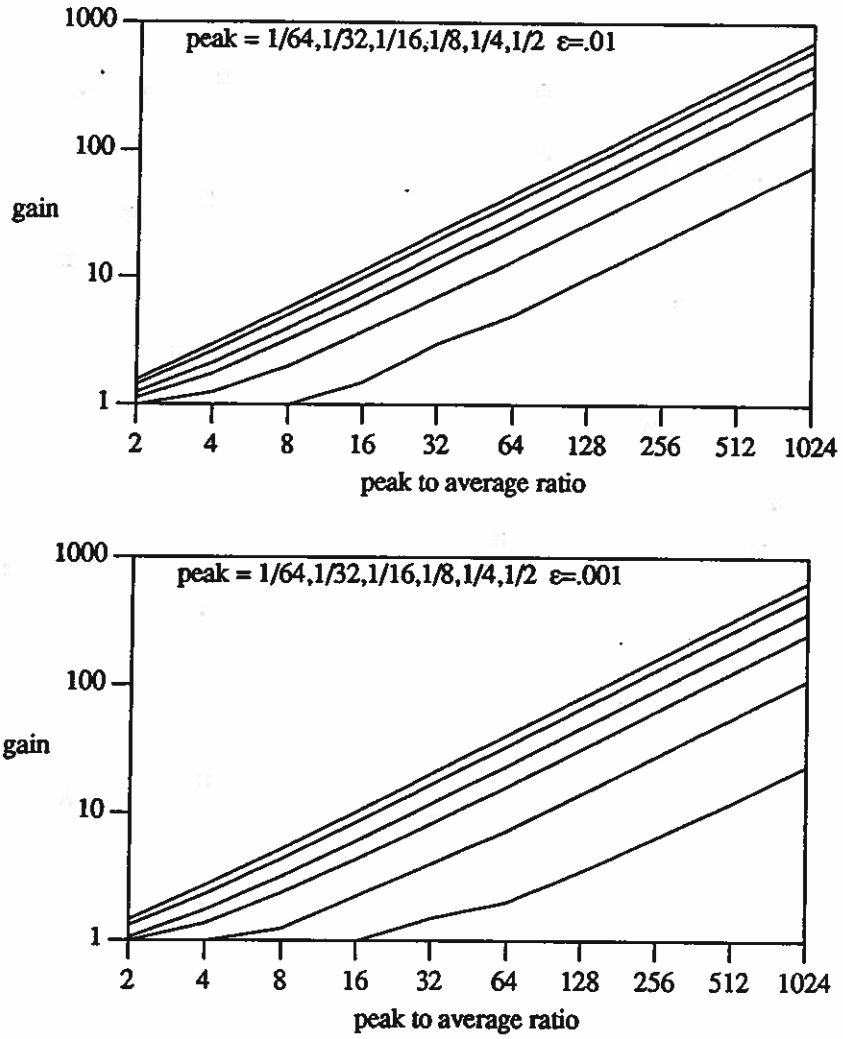


Figure 7: Gain for Homogeneous Virtual Circuits

If an end cell is received while active or if the timer expires, s_i is changed from active to idle.

If a start, middle or end cell is received when in the active state and $C_i \leq 0$, the cell is converted to an end cell and passed on. Also, the state is changed from active to idle.

If a middle or end cell is received while the virtual circuit is in the idle state, it is discarded.

If a loner is received, it is passed on to the network.

The timeout period would be the same as in the switches.

In the typical case, the user transmits a start cell, followed by a sequence of middle cells and an end cell, while the token pool contents remains positive. As soon as the user sends the end cell, tokens are no longer drained from the token pool and the token generator replenishes the supply of tokens in preparation for the next burst. If the user attempts to continue to send tokens after the token pool contents is exhausted, the network forces a return to the idle state by converting the user's next cell to an end cell, which has the effect of releasing the buffers reserved in the switches all along the virtual circuit's path.

An alternative to the approach taken here is for the token generator to send an explicit buffer release cell as soon as the token pool contents reaches 0. This has the advantage that the buffers are released sooner than they would otherwise be, but requires a somewhat more complex implementation. By allowing the token pool contents to drop below zero, we delay the next burst the user can send in direct proportion to the amount of "extra time" that the buffers have been held in the current burst. This ensures that over the long term, the probability that the virtual circuit is active is no more than what was assumed at the time the call acceptance decision was made.

To facilitate use of this mechanism by end user equipment, the token pool mechanism should have

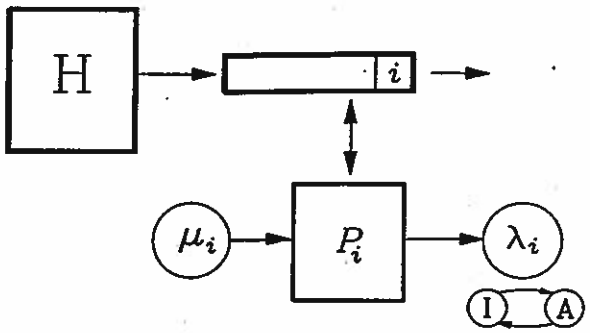


Figure 8: Modified Token Pool

the ability to send a flow control cell to the user, when the token pool contents drops below some threshold. The user's network interface could respond to such a flow control message in one of two ways. First, it could suspend transmission on that virtual circuit temporarily, resuming transmission after enough time has passed to ensure that the token pool is replenished. For most computer-based applications, this would be straightforward to implement. Second, it could continue transmission on the virtual circuit, but switch to loner cells. While delivery could no longer be guaranteed, it may be preferable in some applications to continue to transmit with some losses than to suspend transmission altogether.

V. Closing Remarks

In summary, we have defined a bandwidth management and congestion control scheme for ATM networks that supports both point-to-point and one-to-many multicast virtual circuits. In [14] this scheme is extended to support more general multicast virtual circuits in which there can be multiple transmitters. Reference [14] also details an implementation of the hardware mechanisms required to support the scheme and concludes that the incremental cost of the additional hardware is no more than about 10% of the cost of a typical port controller for an ATM switch. It is possible to extend the buffer reservation mechanism to support burst level priorities. This requires only that the virtual circuit's priority be stored in the buffer allocation table and that the state machine be modified to preempt low priority bursts (by switching them from the active to the idle state) in response to increasing high priority traffic. The key to a simple implementation here is to perform the preemption of the low priority bursts only when processing their cells.

To our knowledge, this proposal is the first really complete approach to handling bandwidth management in ATM networks. As we have shown, the method can handle fully heterogeneous traffic and can be effectively implemented. The algorithm for making virtual circuit acceptance decisions is straightforward and fast, and the hardware mechanisms needed to implement buffer allocation and traffic monitoring at the user-network interface have acceptable complexities. We have shown, through numerical examples that our approach can achieve reasonable link efficiencies even in the presence of very bursty traffic. These efficiencies are directly attributable to the use of explicit buffer allocation. The per cell overhead is acceptably small, two bits being sufficient to encode start, middle, end and loner. Finally, because buffer allocation is done "on the fly," there is no advance reservation required, simplifying the interface between the network and the user and avoiding an initial network round trip delay before data can be transmitted.

Acknowledgements. In completing this work, I have had the benefit of many fruitful (and often spirited) discussions with colleagues at Washington University. Mike Gaddis and Rick Bubenik have been particularly helpful. Neil Barrett, Andreas Bovopoulos, John DeHart, Guru Parulkar and Ellen Witte have also provided useful input.

References

[1] Anick, D., D. Mitra and M. M. Sondhi. "Stochastic Theory of a Data Handling System with Multiple Resources," *Bell System Technical Journal*, 1982.

[2] Boyer, P. "A Congestion Control for the ATM," *International Teletraffic Congress Seminar on Broadband Technologies: Architectures, Applications and Performance*, 10/90.

[3] Cidon, Israel, Inder Gopal and Roch Guérin. "Bandwidth Management and Congestion Control in plaNET," *IEEE Communications Magazine*, 10/87.

[4] Coudreuse, J. P. and M. Servel. "Prelude: An Asynchronous Time-Division Switched Net-

work," *International Communications Conference*, 1987.

[5] Eckberg, A. E. "The Single Server Queue with Periodic Arrival Process and Deterministic Service Time," *IEEE Transactions on Communications*, 3/79.

[6] Fendick, K. W., D. Mitra, I. Mitrani, M. A. Rodrigues, J. B. Seery, and A. Weiss. "An Approach to High-Performance, High-Speed Data Networks," *IEEE Communications Magazine*, 10/87.

[7] Gilbert, Henry, Osama Aboul-Magd and Van Phung. "Developing a Cohesive Traffic Management Strategy for ATM Networks," *IEEE Communications Magazine*, 10/87.

[8] Heffes, H. and D. Lucantoni. "A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexor Performance," *IEEE Journal on Selected Areas in Communications*, 9/86.

[9] Hui, J. Y. "Resource Allocation for Broadband Networks," *IEEE Journal on Selected Areas in Communications*, 12/88.

[10] Melen, Riccardo and Jonathan S. Turner. "Access Arbitration in Tree Structured Communication Channels," *IEEE Transactions on Communications*, 1991.

[11] Roberts, James W. and Jorma T. Virtamo. "The Superposition of Periodic Cell Arrival Streams in an ATM Multiplexor," *IEEE Transactions on Communications*, 2/91.

[12] Sriram, K. and W. Whitt. "Characterizing Superposition Arrival Processes in Packet Multiplexors for Voice and Data," *IEEE Journal on Selected Areas in Communications*, 9/86.

[13] Turner, Jonathan S. "Design of a Broadcast Packet Network," *IEEE Transactions on Communications*, June 1988.

[14] Turner, Jonathan S. "A Proposed Bandwidth Management and Congestion Control Scheme for Multicast ATM Networks." Washington University Computer and Communications Research Center technical report, WUCCRC-91-1.

[15] Woodruff, G. and R. Kositpaiboon. "Multimedia Traffic Management Principles for Guaranteed ATM Network Performance," *IEEE Journal on Selected Areas in Communications*, 4/90.