A Cell Loss Equalization Protocol for Video Multiplexers

Ioanis Nikolaidis

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332–0280
nikolaid@cc.gatech.edu

Ian F. Akyildiz

School of Electrical & Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332–0250
ian@armani.gatech.edu

Abstract

A serious fairness problem occurs whenever homogeneous, variable bit–rate, video traffic streams are multiplexed at a finite buffer statistical multiplexer. Namely, the cell loss ratio performance varies widely across individual traffic streams. The cause of the problem is the particular random relation of the periodic frame transmission epochs of the video sources that feed the multiplexer. In this paper a protocol is presented which controls the frame transmission epochs, in order to achieve fair distribution of losses among the admitted connections. The protocol operates with the participation of the sources by imposing an additional per–source delay on the traffic entering the network. These additional delays are calculated whenever a connection is accepted or terminated. The overall additional delay imposed by the protocol is guaranteed to be minimal through the use of an appropriate optimization algorithm. The performance of the proposed protocol, operating in conjunction with the optimization algorithm, is simulated and evaluated. A basic framework is given for its implementation on an ATM network. Finally, the issues of buffer overhead and the impact of the protocol on the delay experienced at the multiplexer are also discussed.

1 Introduction

Real–time video sources are characterized by the periodic nature of their frame generation epochs. A frame rate of 30 or 25 frames per second is typical for good quality motion. This periodicity implies that, given the first frame generation instant of a source, all subsequent frame generation instants of the same source are known. In packet networks, e.g., ATM Networks, the packetization and transmission epochs of successive video frames also exhibit the same periodicity. That is, the cells related to successive frames are available for transmission at periodic points in time. In the remaining of this paper, the frame transmission period will be denoted by T. Consequently, any periodic description will be reduced to a time point t in the $[0, T)$ interval with the understanding that it occurs at all $t + kT$ time points ($k = 0, 1, \cdots$). Note also that the periodic nature of real–time video traffic is not influenced by the particular encoding and compression method. However, the same is not true for the number of cells per frame, K, which is variable from frame to frame and dependent on the compression method.

The characterization of K as a random process is a topic that has attracted the attention of many researchers [4, 5, 7, 8, 9, 10]. However, there exists no single conclusive model for all types of video traffic. For this reason, the work we present in this paper does not depend on any specific source model. It only assumes that all sources feeding the multiplexer are homogeneous. For the evaluation of the protocol we have used simulations where the video traffic is produced from a gamma distribution which has been shown to closely match the traffic produced by video–conferencing applications [4, 5, 9]. Moreover, we assume that there exists a maximum, $K_{\max}$, for the number of cells that can be generated per frame corresponding to frames for which the compression scheme does not result in substantial savings. The bit rate, B_{in}, of the link connecting the source to the network is sufficient to ensure that $K_{\max}$ cells can be transmitted in T time units. This assumption is necessary to ensure: (a) that even in the worst case the source has no leftover cells from a previous frame to transmit when the cells from a new frame are generated, and (b) that if the network behaves as an ideal constant delay line, all departing cells related to a frame will arrive at the destination within T time units to be available for playout.

Once the K cells corresponding to a frame have been generated, their transmission in the next T time units can be arranged in a variety of ways. One arrangement is to transmit all K cells back–to–back at the peak link transmission rate in a continuous burst. Hence, every T time units, the source generates a continuous burst of variable length. Another way is to "smooth" the variable length burst over the next T time units and, hence, avoid transmission at the peak rate. The second approach is feasible because the source knows the exact number of cells it has to transmit in the next T time units. Consequently, the cells can be transmitted T/K time units apart. From the two arrangements, the former results in less delay at the cost of more intense congestion, while the latter results in less intense congestion but at the expense of more delay. Other transmission arrangements can also be implemented, as long as we can ensure that, at the end of the T time units, no cells are left for transmission. For the purpose of this paper we opt for the back–to–back peak rate transmission because it represents both a pessimistic scenario with respect to congestion and a more likely candidate for real–time communication because of its low delay.

The problem that we study is the *cell loss fairness* problem which can be stated as follows: *When a number of statistically identical video traffic sources are multiplexed at a finite buffer multiplexer, they suffer different per–connection cell loss ratios dependent on the relative position of the sources' frame transmission epochs*[1]. This problem generates concerns, for example, for providers of compressed video broadcast traffic over packet networks (ATM networks and optical fiber–based distribution in particular) regarding the quality of service of separate channels utilizing the same network. That is, certain channels may consistently receive better performance than others, potentially contrary to contractual agreement.

An investigation and a first solution for this problem was proposed in [4]. The solution is based on a modification of the multiplexer's FIFO scheduling discipline such that each connection is associated with a counter. The counter indicates how many cells of the respective connection have been dropped so far. If, upon arrival, a cell finds the multiplexer buffer full it is not necessarily dropped. Instead, an already queued cell is dropped from the connection with the lowest dropped cells counter. The counter for this connection is subsequently increased. The released buffer space is used to accommodate the new arrival. This modification does indeed improve the performance over the simple FIFO scheduling discipline but it *does not* solve the problem. In particular, even after this scheduling discipline is applied, there exist connections that receive an order of magnitude more losses than others (in Table I of [4]). An intuitive explanation is the following: there can be instances where the connection with the lowest counter value currently has no cells queued. Hence, even though the counter is low, we can not victimize the respective connection. Consequently, we may be forced to victimize another connection with an already higher counter value in order to claim the buffer space to store the incoming cell.

Our approach solves the fairness problem by achieving a particular alignment of the frame transmission epochs of the different multiplexed video sources based on the cooperation of these sources. It does not necessitate a new scheduling discipline for the multiplexer. Our protocol operates using an optimization algorithm where the objective is to reduce the overall delay penalty that the sources must experience in order to achieve fair losses. Thus, the delay vs. fairness tradeoff is accounted explicitly in the protocol. At the same time the protocol operation is triggered by call connection and termination events, thus it can be part of the call admission control which operates at the exact same time points.

An interesting approach to the problem of temporal placement of frame transmission epochs is presented in [6]. However, the objective of [6] is *not* cell loss fairness but the overall cell loss and delay performance. To this extent, control of a source's transmission epoch is performed only once at connection admission, after which it remains fixed. Thus, connection termination events leave the system in an unequalized state. We take a different approach by controlling the transmission epochs after admission with the intention to *always* equi–distribute losses among admitted connections.

The remaining of the paper is organized as follows: In section 2 we present the algorithmic aspects and the operations of the proposed protocol. In section 3 we evaluate the performance of the protocol by presenting its delay overhead and by performing a comparison of the per–connection cell loss ratios for a particular configuration when the protocol is utilized and when it is not. We summarize the findings in section 4 and we indicate future research directions.

2 A Frame Alignment Protocol

As we noted earlier, the fairness problem arises from the particular relative time alignment of the frame transmission epochs (or simply, transmission epochs) of the individual sources. It is therefore beneficial to think in terms of the transmission epochs. We can easily identify three cases regarding the alignment of the transmission epochs:

• *The Worst Alignment*: The transmission epochs of all admitted sources are aligned at the exact same time point. That is, the transmission of the frames from all the sources starts at the same point in time. This alignment is repeated for every single frame. It becomes clear why this is the *worst* case in terms of cell loss ratio by noting that all the sources compete for the multiplexer buffer at exactly the same time. Thus, the likelihood to find the buffer full is greater than if they followed any other time alignment.

• *The Best (Ideal) Alignment*: The transmission epochs of the N different connections are spaced T/N time units apart. Thus, the transmission epochs will be at time points kT/N for $k = 0, 1, \cdots, N-1$. We will call these time points, the *ideal transmission epochs*. That is, all transmission epochs are equi–spaced within the frame period. Assuming that the sources are statistically identical, the multiplexer receives the same (in the statistical sense) load of arrivals in each T/N interval defined by any two consecutive ideal transmission epochs. Thus, the load is equalized and likewise are the cell losses. There can be no better alignment, because if the successive transmission epochs of a pair of sources are moved closer than T/N time units the contention between them for the multiplexer buffer increases. The best alignment does not discriminate favorably for any source.

• *A Random Alignment*: The transmission epochs are positioned randomly in the interval $[0, T)$. For the sake of convenience, and since no indication to the opposite exists, we assume that the transmission epoch s_i of any source i is generated from a uniform distribution in the $[0, T)$ interval. The random alignment represents the arbitrary alignment we expect to find in an actual system that lacks the protocol we propose. Such alignment can suffer in terms of cell loss ratio fairness as well as in terms of overall cell loss ratio compared to the ideal alignment. Note also that a random alignment can not be precluded from being as bad as the worse alignment.

To illustrate the effect of the different alignments consider the scaled down example of Figure 1. We represent time in a slotted fashion and $T = 10$ slot times. Assume that the initial buffer occupancy is zero and that the buffer size is 2 while the output link speed is equal to one input link's speed. Under these assumptions, the worst alignment (Figure

[1]The *frame transmission epochs* of a connection are the periodic time points at which the multiplexer starts receiving the cells of successive video frames from a particular connection. Without lack of generality, we can assume that this is the time point at which the first cell related to a new frame is received by the multiplexer. Note also that we use the terms "source" and "connection" interchangeably since they are related in a one–to–one fashion.

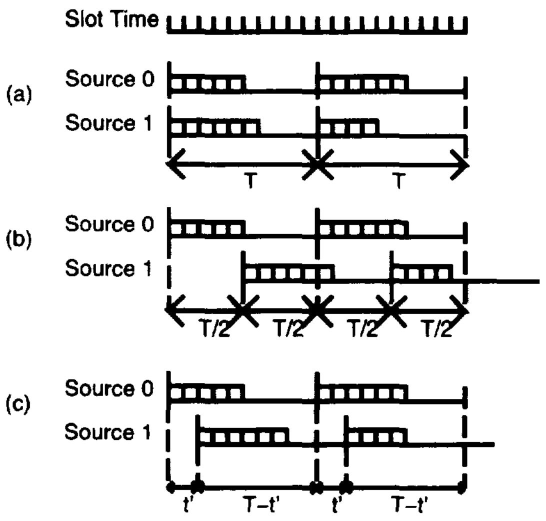


Figure 1: En example of (a) the worst, (b) the best and (c) a random alignment of the frame transmissions epochs for the exact same traffic trace generated by two video sources.

1(a)) receives the worst loss performance, a total of 5 losses. The best alignment (Figure 1(b)) suffers no losses while the random alignment (Figure 1(c)), with $t' = 2$, suffers 3 losses. That is, the exact same frame traffic of two independent video connections will result in markedly different cell loss performance under different frame transmission alignments. This example illustrates the importance that the best alignment has on the overall cell loss statistics. In addition, the best alignment achieves cell loss fairness, as we will see in section 3.

Our objective is to force the random transmission epochs of the sources to be aligned exactly as in the best alignment. In [3] a similar observation was made and it was hinted that additional circuitry is needed to enforce the alignment. However, no details as to how to achieve the alignment, even in circuitry, were given. In our approach, we observe that the alignment depends on the number of admitted sources, N, which is information not necessarily available to the individual source[2] but certainly available to the multiplexer and to the call admission process. Hence, any such operation fits naturally as a function of the Call Admission Control (CAC). Finally, note that the assumption that the transmission epochs are aligned according to the best alignment has been used in the past, e.g., in the case of a multiplexer delay study under different bandwidth sharing schemes in [2]. That is, there seems to be agreement on the importance of the ideal alignment. What we provide in this paper is a mechanism to achieve it.

2.1 The Algorithm

The solution we propose is based on a protocol that notifies each source i to delay its traffic a specific amount of time $\delta_i^* < T$ in order for the transmission epochs of the sources to coincide with the ideal transmission epochs[3].

[2]At least for security concerns, a connection should not be aware of how many other connections are currently admitted.

[3]As a matter of convenience we will use the notation δ^* to refer to all δ_i^*'s collectively, i.e., δ^* is the vector of δ_i^*'s. Similarly, if symbol x represents a vector, x_i will denote its i–th member.

There exists an infinite number of possible delay assignments that map the N random transmission epochs to the N ideal epochs. If we assume that each such delay assignment is represented by an appropriate δ vector, the issue becomes to find the specific δ vector that is optimal, and which we denote by δ^*. The optimality criterion is the minimization of the overall delay $T_{ov}(\delta) = \sum_{i=0}^{N-1} \delta_i$. That is, $T_{ov}^* = T_{ov}(\delta^*) \leq T_{ov}(\delta)$ for any candidate δ. The purpose of the optimization algorithm we present is to calculate the optimal δ^*. The set of candidate δ vectors can be dramatically reduced using the following straightforward lemma:

Lemma 1 *Any delay assignment vector δ which has all its elements, δ_i, greater than zero* can not *be optimal with respect to the minimization of $T_{ov}(\delta)$.*

Proof: Suppose, that we pick a δ which has all its elements not equal to zero. Then we can calculate $\delta_{\min}$, the minimum over all δ_i's. Let us also assume that $\delta_{\min} = \delta_j$ for a certain j. We can now construct a new delay assignment vector δ' where $\delta'_i = \delta_i - \delta_{\min}$ (also, $\delta'_j = 0$). Observe that δ and δ' define the exact same relative frame transmission epochs, and that it is only their absolute position in time that we influenced by subtracting $\delta_{\min}$. Moreover, $T_{ov}(\delta') = T_{ov}(\delta) - N\delta_{\min} \Rightarrow T_{ov}(\delta') \leq T_{ov}(\delta)$. Hence, from a δ with nonzero elements we can directly derive a δ' with at least one zero element which preserves the same relative position of the ideal transmission epochs but results in less overall delay, $T_{ov}(\delta')$. Clearly, δ can not be the optimal.□

Consequently, we limit the search for an optimal δ only between vectors with one or more δ_i elements equal to zero. However, the physical meaning of a zero element, e.g., in position m, in the δ vector is the following: The frame transmission epoch of the m–th connection coincides with an ideal transmission epoch. Following this line of thought, we come to the realization that the optimal δ can be derived by only considering N possible configurations. The i–th configuration is constructed by setting the i–th connection transmission epoch to coincide with an ideal frame transmission epoch. Without harm of generality, we will assume that this ideal transmission epoch is the first ideal transmission epoch, i.e., the one at time 0. For the i–th configuration, all the frame transmission times of the remaining $N-1$ connections are translated relative to the i–th frame transmission epoch and within the $[0, T)$ interval. The problem is thus reduced to finding, for each configuration, the optimal assignment of the remaining $N-1$ connections to the remaining $N-1$ ideal epochs. From the N produced optimal assignments, we select the one that corresponds to the minimum T_{ov} and this is the optimal delay assignment, δ^*.

We will describe the algorithm in depth using the notion of a *reference connection* (ref) which is nothing more but the identification of the configuration being generated and, in a one–to–one fashion, the index of the connection that is aligned to the first ideal transmission epoch. Keeping track of the reference connection is not important for the operation of the protocol, and in this sense, it must not be perceived as having any particular power of significance over the remaining connections. It merely facilitates the description of the control flow of the optimization process.

For each configuration, ref, the first step is to normalize the frame transmission epochs relative to the transmission

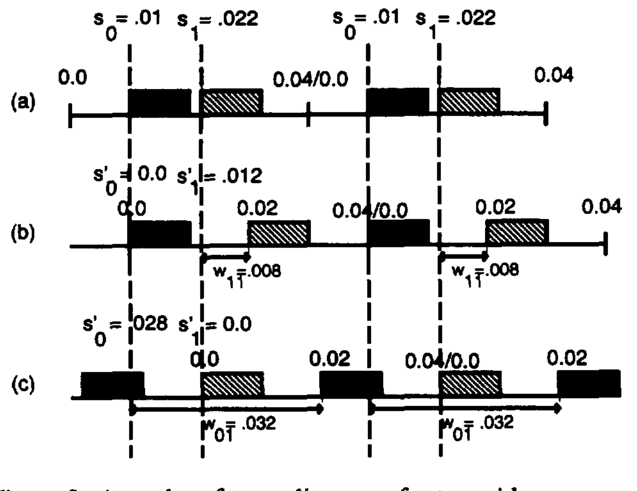


Figure 2: A random frame alignment for two video sources, (a), and the two possible scenarios for the selection of the reference connection: (b) $ref = 0$, and (c) $ref = 1$.

epoch of connection ref in the $[0, T)$ interval. The result of this normalization is the s' vector, with $s'_i \in [0, T)$, which is constructed as follows:

$$s'_i = \begin{cases} s_i - s_{ref}, & s_i \geq s_{ref} \\ s_i - s_{ref} + T, & s_i < s_{ref} \end{cases} \tag{1}$$

$$i \in \{0, \cdots, ref - 1, ref + 1, \cdots, N - 1\}$$

and is calculated in lines 5–8 of the algorithm in Figure 3. Figures 2(b) and 2(c) show, among other information, an example of the derived s' vector values from the s vector of Figure 2(a) for the two possible configurations. The purpose of the s' vector is to retain the relative position of the random transmission epochs while forcing the transmission epoch of ref to coincide with the first ideal transmission epoch.

The *cost* of aligning the transmission of connection i with a normalized random transmission epoch s'_i from the remaining $N - 1$ connections (that is, all except the reference connection) to the j–th ideal epoch at time point jT/N from the remaining $N - 1$ ideal epochs (that is, all except the one at time 0) is the delay necessary to reach the j–th ideal epoch starting at time s'_i. In a simplistic way, the cost, represented by w_{ij}, is the difference of jT/N minus s'_i. However, the cost has to be positive (we can only delay for positive amounts of time!). Hence, if $jT/N < s'_i$, the alignment has to be performed with the same ideal epoch but in the next frame interval. Figure 2(c) depicts exactly this situation, $1T/N = .02 < .028 = s'_0$, hence, the delay w_{01} must extend to the next cycle. Note that the cost w_{ij} can not be more than T since from s'_i we can reach any ideal epoch, either in the current frame interval or in the next within a delay of T time units. Summarizing, the calculation of the costs is performed as follows:

$$w_{ik} = \begin{cases} \frac{kT}{N} - s'_i, & \frac{kT}{N} \geq s'_i \\ T + \frac{kT}{N} - s'_i, & \frac{kT}{N} < s'_i \end{cases} \tag{2}$$

$$i \in \{0, \cdots, ref - 1, ref + 1, \cdots, N - 1\}$$
$$k \in \{1, \cdots, N - 1\}$$

The calculation if w_{ij}'s is performed in lines 9–14 of the algorithm in Figure 3. The next step is the calculation of the optimal assignment for the specific configuration, i.e., for the specific reference connection. It can be stated in the following form:

$$\begin{array}{ll} \textit{minimize} & T_{ov} = \sum_i \sum_k b_{ik} w_{ik} \\ \textit{s.t.} & \sum_k b_{ik} = 1, \quad \forall i \\ & \sum_i b_{ik} = 1, \quad \forall k \end{array} \tag{3}$$

$$b_{ik} \in \{0, 1\},$$
$$i \in \{0, \cdots, ref - 1.ref + 1, \cdots, N - 1\}$$
$$k \in \{1, \cdots, N - 1\}$$

This final step is a typical assignment problem, solvable by well-known polynomial algorithms, e.g., by the $O(N^3)$ "Hungarian" algorithm [1]. We will assume that the solution of the above optimization problem is provided by function ASSIGNMENT(N, ref, w, b^*, T_{ov}) which returns, the array b^* of the optimal b assignments as well as the overall delay T_{ov} according to the optimal assignment[4]. Summarizing, the algorithm for finding the best delay assignment is presented in Figure 3. It initializes the optimal overall delay and delay assignment in lines 2–3. Then it scans through all N possible configurations, constructing the s'_i and the w_{ij} values for each configuration and solving the resulting assignment problem. If (at line 16) the optimal assignment for the current configuration results in a smaller overall delay than all the configurations until now, it is taken to be the optimal and its associated assignment is taken correspond to be the optimal delay assignment δ^* (lines 17–21). At the end of the run, the optimal T^*_{ov} and δ^* are returned.

From the above discussion, it is evident that there is no need to perform any alignment when only one connection is admitted since it can be considered to coincide with the first (and in this case, only) ideal frame start time in each cycle of length T. Suppose now that two connections are accepted as in Figure 2, the resulting example is trivial but easier to follow. Two configurations need to be considered. In the first (Figure 2(b)), $ref = 0$ and the only cost is $w_{11} = .008$. The resulting optimal assignment will be $\delta_0 = 0$ and $\delta_1 = .008$ and thus $T_{ov} = .008$. In the second configuration, the only cost is $w_{01} = .032$. The resulting optimal assignment will be $\delta_0 = .032$ and $\delta_1 = 0$ and thus $T_{ov} = .032$. Obviously, the first configuration results in the least cost and therefore $\delta^*_0 = 0, \delta^*_1 = .008$ and $T^*_{ov} = .008$.

The solution of the optimization problem provides the optimal delay assignment, δ^*_i, for each source i. Source i is informed about the value of δ^*_i and it starts delaying δ^*_i time units each outgoing frame to ensure fair losses. In the next section we describe further details related to the implementation of the scheme on an ATM network.

2.2 Protocol Implementation on an ATM Network

The protocol we describe herein is simple and a more formal description, e.g., by using state diagrams, would be unnecessarily lengthy and less informative. Instead, we describe the functions performed by the protocol. The protocol functions can be distinguished into two groups: (a) multiplexer (MUX) functions, i.e., functions performed by the

[4]We pass ref to ASSIGNMENT to indicate that the row and column of w indexed by ref are to be ignored.

```
begin
$T^*_{ov} = \infty$;
$\delta^* = \emptyset$;
for ref = 0 upto $N - 1$ do
    for i = 0 upto $N - 1$ do
        $s'_i = s_i - s_{ref}$;
        if ($s'_i < 0$) then $s'_i = s'_i + T$;
    endfor
    for i = 0 upto $N - 1$ and i $\neq$ ref do
        for k = 1 upto $N - 1$ do
            $w_{ik} = kT/N - s'_i$;
            if ($w_{ik} < 0$) then $w_{ik} = w_{ik} + T$;
        endfor
    endfor
    ASSIGNMENT($N, ref, w, b^*, T_{ov}$);
    if $T_{ov} < T^*_{ov}$ then
        $T^*_{ov} = T_{ov}$;
        for i = 0 upto $N - 1$ and i $\neq$ ref do
            $\delta^*_i = \sum_{k=1}^{N-1} b^*_{ik} w_{ik}$;
        endfor
        $\delta^*_{ref} = 0$;
    endif
endfor
return($T^*_{ov}, \delta^*$);
end
```

Figure 3: The optimization algorithm.

multiplexer and its accompanying call admission logic and (b) source functions, i.e., performed by the source traffic generator and with the assistance of appropriate control information. The multiplexer functions include the estimation of the s_i time points of new connections, the calculation of the optimal delay vector δ^*, and the transmission of the updated δ^*_i back to source i, for all admitted sources. The source functions include advertising the s_i at connection setup time and the reception of subsequent δ^*_i values that control the delay imposed on the outgoing source traffic. Following is the list of the protocol functions and whether they are part of the source or the multiplexer functions:

• *Source Advertising (Source):* Along with a connection request sent to the CAC entity, the source starts sending to the multiplexer the regular traffic as it would normally. The first cell of the cell burst related of a frame transmission is marked by the ATM Adaptation Layer (AAL) as a Start of Frame (SOF) cell. The transmission of frames continues until the connection terminates.

• *s_i Calculation (MUX):* A continuously running Frame Timer (FT) exists at the multiplexer. Its purpose is to continuously count from 0 to T, cycling back to 0 when it reaches T. The multiplexer also keeps track of the number of already admitted connections, N. Upon reception of an SOF cell, for a connection which is not yet accepted, it records the FT value as s_{N+1}, i.e., the frame start time for the as-yet-not accepted connection. All other incoming cells from the new source are simply ignored (discarded) because the connection is not yet accepted.

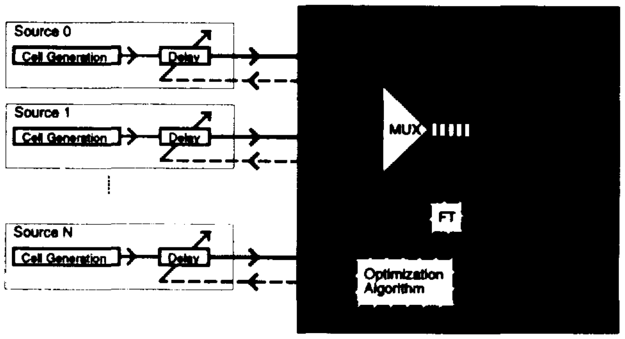


Figure 4: Block diagram of an ATM Network multiplexer and the position of the proposed protocol functions (in dashed lines). The circles around the input links denote the observation of the input stream for SOF cells.

• *Connection Acceptance (MUX):* Upon acceptance of the connection by the CAC, the multiplexer is informed that the connection is accepted. It increases the number of admitted connections from N to $N + 1$ and runs the algorithm that we described in the previous section for the $N + 1$ frame start times. The δ^* vector is determined and its values are sent to the corresponding sources.

• *Connection Acceptance (Source):* Upon reception of the first δ^*_i value, the source assumes that it has been admitted. It starts delaying the departing cells by δ^*_i time units by buffering them locally.

• *Connection Termination (Source):* Source j terminates the connection by sending a request to the CAC entity. It also ceases from transmitting frames.

• *Connection Termination (MUX):* The termination request is processed by the CAC entity and the indication to terminate source j is received by the multiplexer. The multiplexer discards the s_j counter value and runs the optimization algorithm for the remaining sources. It informs the sources about the new delays by sending the updated δ^* vector.

• *δ^* Updates (Source):* If an admitted source, i, receives a new δ^*_i value, it readjusts the delay imposed on each frame's traffic (starting from the next frame to be generated) to the new value δ^*_i.

A block diagram of the placement of the protocol in the network is given in Figure 4. Depending on the sophistication necessary during the acceptance and termination process, the above steps can be modified with more checkpoints and control information. For the sake of simplicity we will assume that at least the above operations are supported. However, as a matter of convenience, we will express the delay value of δ^*_i in terms of number of cell transmission times Δ^*_i (at peak input link rate) that fit in δ^*_i time units. That is, if the bandwidth of the input links is B_{in} bits per second, then:

$$\Delta^*_i = \lceil \frac{\delta^*_i}{(D/B_{in})} \rceil \qquad (4)$$

where D is the size of the cell in bits (53 octets = 424 bits per cell in ATM networks). The reason behind using Δ^*_i instead of δ^*_i is that Δ^*_i represents the additional buffering necessary

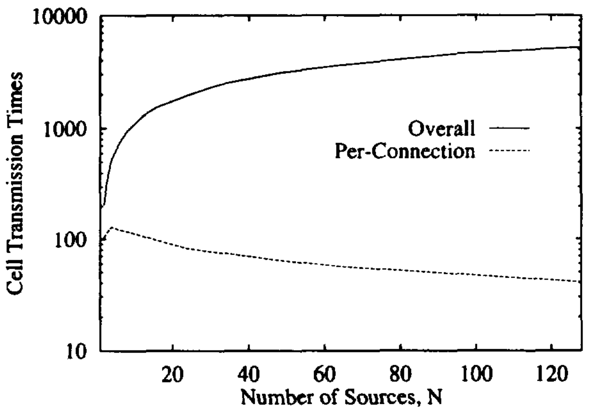


Figure 5: The overall and the per–connection delay imposed on the source traffic due to the protocol, expressed as input link cell transmission times for a variable number of sources, N. Note the opposite trend of the two curves for increasing N.

by the source in order to delay the traffic at the source. Hence, it captures the exact space overhead required at the source in order to support the δ_i^* delay requested by the protocol.

3 Protocol Evaluation

There exist three separate aspects of the protocol that we will examine:

• The extent to which the δ^*, or better Δ^*, values are reasonable for a number of configurations. The optimization algorithm guarantees their overall optimality but that does not give a precise feeling about the average delay and additional buffering that is necessary.

• The effectiveness in terms of cell loss fairness of the best alignment which results from the proposed protocol versus the case where the protocol is not used.

• The queueing delay distribution at the multiplexer under the assumption that the proposed protocol is used. Since the traffic has already been delayed at the source, further delays at the multiplexer would be an extra penalty.

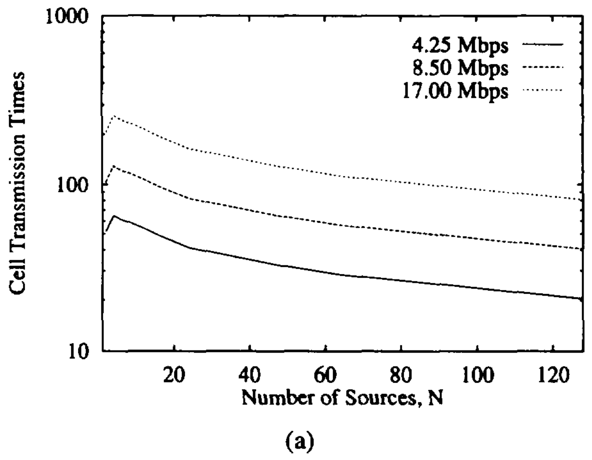
3.1 The Dynamics of Δ^*

We assume, as stated earlier, that the random transmission epochs, represented by the vector s, are produced from a uniform distribution in the $[0, T)$ range and that the input link speed to the multiplexer is B_{in}. Given this, we are interested in two key values:

• The average per–connection delay imposed by the protocol for any of the N sources, expressed in cell transmission times at a rate of B_{in}. That is, the average value of Δ_i^* over all i's for a particular N.

• The average overall delay imposed by the protocol for all N sources expressed in cell transmission times at a rate of B_{in}. That is, the average value of T_{ov}^* for a particular N.

Unfortunately, both Δ_i^*'s and T_{ov}^* are the result of an



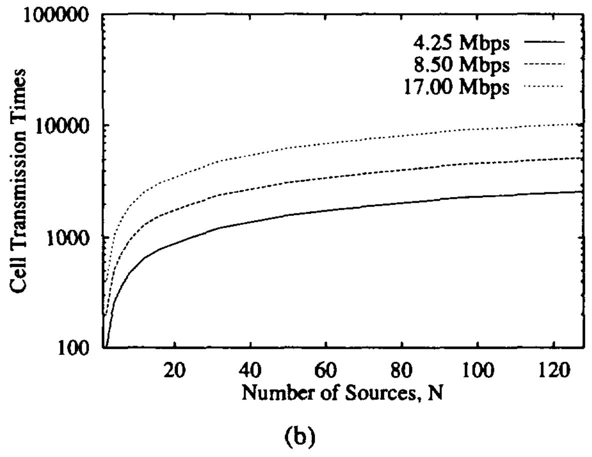


Figure 6: The (a) overall, and (b) the per–connection delay imposed on the source traffic due to the equalization protocol expressed as input link cell transmission times for a variable number of sources and for different input link bit rates.

optimization process and we can not obtain them in any closed form solution in order to derive their averages in an analytical fashion. Instead, we will evaluate their averages as produced through the optimization algorithm over a large set of runs where each run uses a different random vector of transmission epochs, s. We set B_{in} to 8.5 Mbps. With a frame rate of 25 frames per second, $T = .04$. Also 801 cells can be transmitted at 8.5 Mbps during a frame interval. Thus, $K_{\max} \leq 801$. The s vectors are produced by setting each s_i to a time point picked from a uniform distribution in the $[0, .04)$ sec interval.

The results are summarized in Figures 5 and 6. The obvious dynamics are that although the overall delay increases as N increases, the same is not true for the per–connection delay. Indeed, as Figure 5 shows, the average per source delay moves to moderate values of less than a hundred cells. To explain the observed dynamics one has to think of the ideal epochs as splitting the $[0, T)$ in N equal bins. The random points in the $[0, T)$ range, represented by the s vector, are the equivalent of N randomly positioned marbles. For increas-

Source	s_i	CLR			Δ_i^*
	(ms)	Worst ($\times 10^0$)	Best ($\times 10^{-5}$)	Random ($\times 10^{-5}$)	(cells)
0	33.30	0.000	6.917	1.450	85
1	23.21	0.000	5.423	10.447	36
2	17.56	0.000	5.572	4.790	50
3	36.57	0.000	4.881	7.966	69
4	22.50	0.000	4.569	14.495	0
5	29.47	0.179	5.572	9.949	61
6	2.23	0.341	5.265	11.252	56
7	30.32	0.435	4.765	27.182	94
8	26.83	0.497	4.723	22.888	14
9	3.07	0.541	5.509	0.406	89
10	9.95	0.581	6.619	0.235	52
11	7.71	0.608	5.949	2.836	46
12	14.81	0.630	5.436	11.322	4
13	13.52	0.648	5.034	8.533	80
14	2.14	0.664	5.488	23.476	8
15	29.69	0.678	6.117	42.663	7
OVERALL CLR		0.363	5.490	12.501	

Table 1: The cell loss ratio (CLR) experienced by each of the 16 sources under the worst, the best and a random alignment. The random alignment corresponds to the random frame start times under the s_i column. The best alignment is achieved by enforcing the Δ_i^* delays on the random frame start times.

ing N, the interval covered by each bin decreases and so does the probability to find more two or more marbles in each bin. Thus, the more likely to have a one–to–one correspondence between an s_i and a single ideal transmission epoch in any of the N bins. Therefore, the more likely that s_i will not have to be delayed, for the sake of alignment, away from its current bin, i.e., away from the nearest ideal epoch. An interesting observation is that doubling or halfing the input link rate results in a proportional increase or decrease of the average delay times. This behavior is demonstrated in Figure 6. Such behavior would seem to cause problems at higher link speeds by requiring an ever increasing buffer in order to accommodate the delayed traffic at the source. However, there exists a limit which is independent of the input link speed. Namely, $K_{\max}$. Since no frame start time is delayed more than T time units ($\delta_i^* \leq T$), no more than $K_{\max}$ cells (a frame's worth of cells at their maximum) need to be stored.

3.2 A Cell Loss Equalization Example

In order to illustrate the effectiveness of the alignment and thus the effectiveness of the protocol, we implemented a simulation model that closely follows the assumptions in the simulations presented in [4]. Our example consists of 16 sources feeding a multiplexer. Each source sends its traffic without smoothing over a 8.5 Mbps link. The multiplexer has a finite buffer of 300 cells and is serviced by an output link of B_{out} = 45 Mbps, equivalent to a DS3 link. Thus, the worst case delay experienced through buffering at the multiplexer is 2.826 msec. This value is reasonably small for a single node for effective real–time communication. We do not make any particular assumption about the length of the links from the sources to the multiplexer, which can be potentially long without harming our equalization protocol. The traffic generated for each source is generated from a gamma distribution with the same parameters as in [4]. The only difference with respect to source traffic compared to [4] is that we consider a per–cell payload of 48 bytes instead of 64 cells. Finally, we simulate an hour of activity of these 16 sources or, equivalently, 90000 frames per source.

The exact same frame arrivals are replicated for each of the three alignments, for the best (enforced by our protocol), the worst and a random. The random alignment is the one obtained by setting the transmission epochs to the s_i's of the second column of Table 1. The best alignment is achieved through the use of our protocol by delaying the traffic at the sources that originally transmitted at s_i epochs such that they transmit at the ideal epochs (spaced .04/16 secs apart). The per–source delay necessary to achieve this best alignment is depicted in the last column of Table 1 and it is produced by the protocol in the fashion we described in the previous sections. Finally, the worst alignment results are provided just to illustrate the extreme case that a random alignment can reach if it does not implement our protocol.

The results are summarized in Table 1. With CLR we denote the calculated Cell Loss Ratio. The reference connection according to the optimization algorithm is connection 4 ($\Delta_4^* = 0$). The remarkable result is the wide range of the per–connection CLR values for the random configuration. It ranges from 0.235×10^{-5} for source 10 to 42.663×10^{-5} for source 15, a difference of two orders of magnitude. Thus, connection 10 receives an effective QoS CLR of 10^{-6} and connection 15 an effective QoS CLR of 10^{-4}. Under the best alignment that the protocol produces, the cell losses are equalized around the overall value of 5.490×10^{-5}. Note also that at the same time, the overall CLR of the random alignment (12.501×10^{-5}) is worse than that of the equalized frame start times. Notably, not only is the equalized case better for the per–connection performance but also for the overall performance. Finally, consider the worst case CLR, which is of the order of 10^{-1}. Moreover, the losses are not equalized in the worst case. This is an artifact of the simulation due to the fixed way with which ties for arrivals at the same time point were broken (first was considered the arrival from the source with the smallest index). A similar artifact can be present in an actual system when the frame start times of each source differ only slightly (less than the time for a cell transmission time on the input link). Without the proposed protocol there is nothing to prevent the worst case configuration. Thus, a random configuration can deteriorate to the worst CLR of 10^{-1}.

3.3 Multiplexer Queueing Delay

Our results suggest that a small additional delay can severely improve the cell loss fairness. However, the coupling that is put between the delay, the cell losses and the cell loss fairness becomes an issue in the context of the overall Quality of Service (QoS). In particular, for our equalization protocol, having delayed the traffic at the source, we would like to expect that it does not get penalized any further in terms of delay at the FIFO multiplexer. The intuitive argument is that the best alignment "equi–distributes" the load of arrivals over the frame interval. Thus, the delay should be lower than in the random case. This indeed turns out to be true, as Figure 7 suggests. The plot in Figure 7 is for a buffer size of 400 cells. The random configuration that we used in the previous example results in a delay distribution which

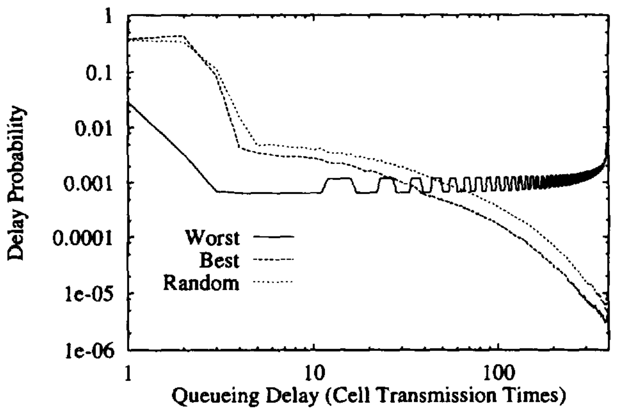


Figure 7: The queueing delay for the worst, the best and the random (with s_i's given in Table 1) alignment in terms of cell transmissions on the output link. The output link speed is 45 Mbps and the cell size 53 bytes, the multiplexer buffer size is 400 cells.

has more mass at its tail. That is, it results in overall longer queueing delays. In contrast, the best alignment has most of its mass in the left side of the distribution, and in particular it has a more pronounced peak at the queueing delay at 2 cell transmission times (measured on the multiplexer's output link). The worst alignment results in a very heavy tail as we expected by the losses. In addition, it demonstrates a remarkably regular shape in the distribution plots. The almost "sawtooth" behavior is justified by the following observation: The arrivals at the beginning of the frame interval can be considered to occur in batches of 16 per cell transmission time of the input link. Until the next batch arrival, only $B_{out}/B_{in} \simeq 5.3$ cells can be serviced. For an initially empty queue, after 16 cells arrive, 5.3 get serviced and the next batch finds the queue with an average of $16 - 5.3 = 10.7$ (where the first sawtooth rises). The next batch will find the queue with an average of $32 - 10.6 = 21.4$ cells (where the second sawtooth rises) and so on for each sawtooth starting at $(16 - 5.3)k = (10.7)k$. Of course, the number of batches of size 16 and the subsequent batches of lesser size during the frame interval are not deterministic, and this results in the overall stochastic behavior that the plots describe. However, the important conclusion is that the additional delay at the sources enforced by our protocol results in an overall *reduced* delay at the multiplexer. Thus, under the protocol the sources do not get penalized twice in terms of delay.

4 Conclusions

In this paper we have proposed a protocol that rectifies the cell loss ratio fairness problems across multiple video connections that feed the same finite buffer multiplexer. The protocol requires the cooperation of the sources in terms of additional delay imposed on the departing cells. The delay requirements, expressed as storage needs at the source, do not exceed the maximum number of cells in a single video frame. Moreover, the protocol needs to operate only as part of the call admission process. Thus, it influences the operation of the sources though changes to the Δ^* values at connection admission and termination points that are relatively sparse. We have shown the dramatic difference that the protocol makes in terms of cell loss ratio fairness, without changing the scheduling discipline and without imposing more delays on the traffic inside the multiplexer.

We are currently considering alternative optimization objectives, e.g., objectives influenced by technological constraints, such as availability and sizes of buffers at the video traffic sources. Finally, we have not discussed the problem of the penalty imposed on non-compliant sources according to the protocol. Non-compliant sources, either due to malicious attempt or failure of equipment, deserve a worse performance than compliant connections. The nature of the protocol suggests that all connections receive the best performance when they all comply to the protocol. Thus, there is no particular attractive incentive for a connection to be non-compliant. However, a failsafe mechanism for policing traffic is still necessary, but it belongs to the more general framework of traffic policing.

References

[1] G. Carpaneto and P. Toth, "Algorithm 548: Solution of the Assignment Problem," *ACM Trans. on Mathematical Software*, Vol. 6, No. 1, pp. 104–111, March 1980.

[2] S. Chowdhury and K. Sohraby, "Alternative Bandwidth Allocation Algorithms for Packet Video in ATM Networks," *Proc. IEEE INFOCOM '92*, pp. 1061–1068, 1992.

[3] D. Cohen and D. P. Heyman, "A Simulation Study of Video Teleconferencing Traffic in ATM Networks," *Proc. IEEE INFOCOM '93*, pp. 894–901, 1993.

[4] D. P. Heyman, A. Tabatabai and T. V. Lakshman, "Statistical Analysis and Simulation Study of Video Teleconference Traffic in ATM Networks," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 2, No. 1, pp. 49–59, March 1992.

[5] D. P. Heyman, A. Tabatabai, T. V. Lakshman and H. Heeke, "Modeling Teleconference Traffic from VBR Video Coders," *Proc. ICC '94*, pp. 1744–1748, 1994.

[6] D.–S. Lee and B. Sengupta, "Policies for Temporal Placement Control of Video Frames in B–ISDN Networks," *to appear in GLOBECOM '94*.

[7] B. Maglaris, D. Anastassiou, P. Sen, G. Karlsson and J.D. Robbins, "Performance Models of Statistical Multiplexing in Packet video Communications," *IEEE Trans. on Communications*, Vol. 36, No. 7, pp. 834–844, July 1988.

[8] M. Nomura, T. Fujii and N. Ohta, "Basic Characteristics of Variable Vit Rate Video Coding in ATM Environments," *IEEE Journal on Selected Areas in Communications*, Vol. 7, pp. 752–760, June 1989.

[9] P. Pancha and M. El Zarki, "A Look at the MPEG Video Coding Standard for Variable Bit Rate Video Transmission," *Proc. IEEE INFOCOM '92*, pp. 85–94, 1992.

[10] P. Skelly, S. Dixit and M. Schwartz, "A Histogram–Based Model for Video Traffic Behavior in an ATM Network Node with an Application to Congestion Control," *Proc. IEEE INFOCOM '92*, pp. 95–104, 1992.