Optimal Cell Loss Equalization for Video Multiplexers

Ioanis Nikolaidis
Ian F. Akyildiz

GIT–CC–94/51

October 14, 1994

Abstract

Video traffic multiplexers in high speed ATM networks are prone to fairness problems with respect to the per-connection cell loss ratio experienced by multiplexed video sources. The problem is a result of the random, but fixed over time, relation of the frame transmission epochs that feed a multiplexer. This paper presents a solution to this fairness problem which is based on the enforcement of controlled per–connection delays. The amount of delay imposed on each source is calculated by an optimization process at connection admission and termination instants. Two different optimization objectives, one minimax and one minisum, are considered. Their performance and their relation to buffer space constraints is examined. The loss and delay performance of the scheme is also evaluated through simulations. In particular, very low per–connection delay variance is observed, indicating reduced jitter. Finally, two implementation alternatives of the scheme on an ATM network are presented: (a) as a protocol between multiplexer and sources and (b) as a non–work conserving service discipline for multiplexers. The engineering aspects and, in particular, the buffer demands of the two alternative implementations are discussed in detail.

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332–0280

Optimal Cell Loss Equalization 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 and
Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332–0250
ian@armani.mirc.gatech.edu

Abstract

Video traffic multiplexers in high speed ATM networks are prone to fairness problems with respect to the per-connection cell loss ratio experienced by multiplexed video sources. The problem is a result of the random, but fixed over time, relation of the frame transmission epochs that feed a multiplexer. This paper presents a solution to this fairness problem which is based on the enforcement of controlled per–connection delays. The amount of delay imposed on each source is calculated by an optimization process at connection admission and termination instants. Two different optimization objectives, one minimax and one minisum, are considered. Their performance and their relation to buffer space constraints is examined. The loss and delay performance of the scheme is also evaluated through simulations. In particular, very low per–connection delay variance is observed, indicating reduced jitter. Finally, two implementation alternatives of the scheme on an ATM network are presented: (a) as a protocol between multiplexer and sources and (b) as a non–work conserving service discipline for multiplexers. The engineering aspects and, in particular, the buffer demands of the two alternative implementations are discussed in detail.

Keywords: Video Distribution, Video Traffic, Statistical Multiplexing, ATM Networks, Connection Admission Control.

*A significantly shorter version of this paper, incorporating earlier results, appears in the Proceedings of the ACM Multimedia 94 Conference. Techniques presented herein may be subject to patents pending.

1 Introduction

Video sources are characterized by the periodic nature of their frame generation epochs. Frame rates in the range of 24 to 30 frames per second are typical for good quality motion. In packet–based networks, e.g., ATM networks, the packetization and transmission epochs of successive frames also exhibit the same periodicity. That is, the cells related to successive frames are available for transmission at periodic points in time. In 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 ($k = 0, 1, \cdots$).

The focus of this paper 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 regarding the quality of service (QoS) of separate channels utilizing the same network. That is, certain channels may consistently receive better performance than others, potentially contrary to contractual agreement.

A first solution to the cell loss fairness problem is proposed in [5] and it is based on a modification of the multiplexer's FCFS scheduling discipline. Each connection is associated with a counter indicating how many cells of this connection have been dropped so far. If, upon its arrival, a cell finds the buffer of the multiplexer full, it is not necessarily dropped. Instead, an already queued cell is dropped from the connection with the lowest counter value of dropped cells. The counter for this connection is subsequently increased and the released buffer space is used to accommodate a new arrival. This modification ameliorates the fairness problem but it *does not* solve it. For example, even when the modified scheduling discipline is applied, there exist connections that receive an order of magnitude more losses than others (in Table I of [5]). The intuitive explanation is that there exist instances when the connection with the lowest counter value has no cells queued. Hence, even though its counter is low, we can not victimize this connection. Consequently, we are forced to victimize a connection with a higher counter to salvage the buffer space necessary to store the incoming cell. Moreover, when we consider the call–level dynamics, we realize that long–lived connections gain a performance advantage over short–lived ones since short–lived connections are always associated with small counter values.

Three alternative solutions to the problem of temporal placement of frame transmission epochs are presented in [8]. However, the objective in [8] is *not* cell loss fairness but the overall cell loss and delay performance. Moreover, control of a source's transmission epoch is performed only once at connection admission, after which it remains fixed. In addition, the best of the three alternatives presented in [8] needs extensive information about the source traffic (stationary distribution of each source's traffic intensity throughout the frame period). We take a different approach by controlling the transmission epochs even after admission with the intention to *always* equi–distribute losses among admitted connections. We do not require knowledge about the traffic model but we assume homogeneous sources and the same frame rate for all sources.

The scheme we present can be implemented either as a protocol between the sources and the multiplexer or as a non–work conserving service discipline at the multiplexer. In

[1]The *frame transmission epochs* of a connection are the periodic time points at which the multiplexer receives the first cell of a new frame. Note also that we use the terms "source" and "connection" interchangeably since they are related in a one–to–one fashion.

the first case, space overhead is necessary at the sources while in the second the space overhead is located at the multiplexer. The objective is to enforce a particular alignment of the frame transmission epochs of the multiplexed sources. An underlying optimization process minimizes the necessary delays to achieve the alignment. The optimization and alignment operations are triggered by call connection and termination events, thus they can be part of the call admission control which operates at the exact same time points. The infrequent nature of call admission and termination events (at least in comparison to cell arrival events) implies that a very low, amortized, computational cost is needed to support the proposed scheme.

Before we introduce the scheme in more detail, a few clarifications are necessary about the context of the present study. First, we consider two example applications for the subsequent simulation and evaluation sections. One is a multiplexer of video–conferencing connections while the second is a multiplexer for a video distribution service of entertainment video. Notably, the nature of the first application is real–time and delays are more crucial than in the second application.

Secondly, the characterization of the number of cells per frame, K, as a random process is a topic that has attracted the attention of many researchers during recent years [4, 5, 6, 10, 11, 12, 13]. Since there exists no single conclusive model for all types of video traffic, our evaluation section is based on simulations conducted using (a) a Gamma distribution–based traffic and (b) an actual compressed video trace. The Gamma distribution closely matches the traffic produced by videoconference applications [5, 6, 12]. The trace, on the other hand, corresponds to the traffic from an actual entertainment video [4] according to the MPEG compression standard [9].

Futhermore, we assume that there exists a maximum, $K_{\max}$, for the number of cells that can be produced per frame. This maximum typically corresponds to frames for which the compression scheme does not result in substantial savings. We also assume that 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 ensures: (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 the next T time units to be available for playout.

Finally, once the K cells representing a frame have been generated, there exists a variety of ways to arrange their transmission in the next T time units. One arrangement is to place the transmission of all K cells back–to–back at the peak link transmission rate in a continuous burst. This arrangement results in more intense congestion at the multiplexer but it experiences better delay. A second arrangement is to "smooth" the variable length burst over the next T time units and, hence, to avoid transmission at the peak rate. Consequently, the cells can be transmitted T/K time units apart. This second arrangement results in less intense congestion at the multiplexer but at the expense of additional (as much as T) delay. For the purpose of this paper we opt for the back–to–back peak rate transmission because it both represents a pessimistic scenario with respect to multiplexer congestion and it does not cause extra delays that can harm real–time communication.

The remaining of the paper is organized as follows: in section 2 we present the algorithmic aspects and the operations necessary to implement the proposed scheme. In section 3 we evaluate the scheme in terms of attained cell loss fairness, and in terms of delay and delay

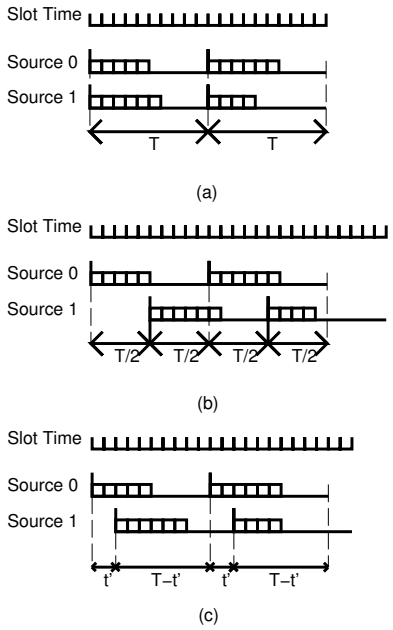

Figure 1: En example of (a) the worst, (b) the best and (c) a random alignment of the frame transmissions epochs for the exact same traffic trace generated by two video sources.

jitter while we also provide the engineering guidelines for its implementation on an ATM network. We conclude in section 4 with a summary of the contributions of the proposed scheme and we point out future directions for related research.

2 The Frame Alignment Technique

In the previous section we pointed out that the cell loss fairness problem is caused by the particular relative time alignment of the frame transmission epochs of multiplexed video sources. In order to solve the problem, we focus on ways to enforce a certain "best" alignment with respect to perceived cell loss ratio performance. Without any formal proof[2], we appeal to intuition and distinguish three cases regarding the transmission epoch alignment of multiplexed sources:

- *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*.

[2]Any such proof would necessitate an analytical model for the video traffic which is both beyond the scope of this paper and is still an open research question.

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 a control mechanism such as the one 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 close to the worst alignment.

To illustrate the effect of the different alignments consider a scaled down example such as that 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. Thus, whenever two arrivals occur in the same input time slot, the one can be serviced while the second can be stored in the buffer, provided there exists space in the buffer, otherwise it is lost. Under these assumptions, the worst alignment (Figure 1(a)) receives the worst loss performance, a total of 5 losses, 3 in the first cycle and 2 in the second. The best alignment (Figure 1(b)) suffers no losses while the random alignment (Figure 1(c)), with $t' = 2$, suffers 3 losses, 1 in the first cycle and 2 in the second. 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 [2], 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[3] 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 [1]. That is, there seems to be an agreement on the importance of the ideal alignment. What we provide in this paper is a mechanism to achieve it.

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

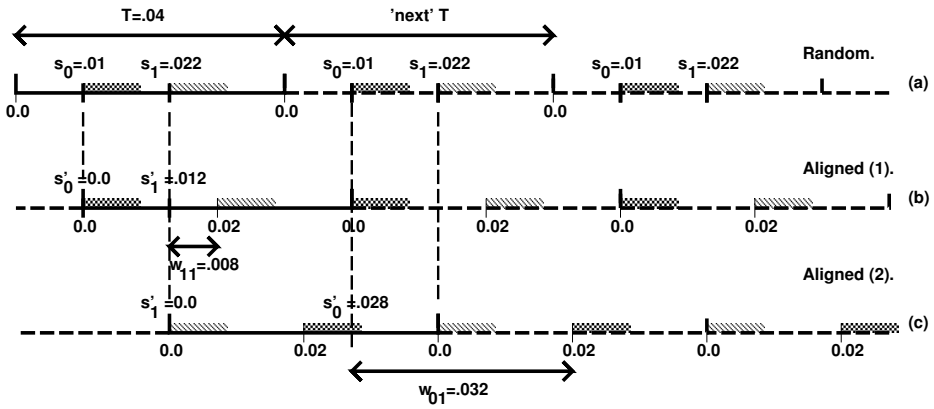

Figure 2: A random frame alignment for two video sources, (a), and the two possible scenarios for the selection of the reference connection: (b) $ref = 0$, and (c) $ref = 1$.

2.1 The Optimization Algorithm

The solution we propose operates by enforcing a vector of delays δ^* on the admitted connections. That is, connection i is delayed δ_i^* time units where $\delta_i^* < T$. The delays cause the transmission epochs of the sources to coincide with the ideal transmission epochs. 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^*. We consider two optimality criteria:

- Minimization of the aggregate delay over all connections: $T_{\texttt{MS}}(\delta) = \sum_{i=0}^{N-1} \delta_i$, i.e., a minisum (MS) formulation.
- Minimization of the maximum delay over all connections: $T_{\texttt{MM}}(\delta) = \max_{i=0}^{N-1}\{\delta_i\}$, i.e., a minimax (MM) formulation.

The MS criterion attempts to minimize the overall delay enforced on all connections but does not consider inherent limitations on the acceptable maximum per–connection delay. In this sense, it may cause excessive delays on a few connections, thus jeopardizing their real–time nature. This case is handled by the MM formulation which, however, may inflate the delay values for more connections than necessary, compared to the delay values produced by the MS formulation. The optimal δ for MS and MM will be denoted by $\delta^*_{\texttt{MS}}$ and $\delta^*_{\texttt{MM}}$ respectively[4]. The purpose of the optimization algorithm we present is to calculate the optimal $\delta^*_{\texttt{MS}}$ and the optimal $\delta^*_{\texttt{MM}}$. However, whichever of the two optimization criteria we pick, the set of candidate δ vectors can be dramatically reduced using the following straightforward lemma:

Lemma 1 *Any delay assignment vector δ where $\delta_i > 0$ $(\forall i)$* can not *be optimal with respect to the minimization of either $T_{\texttt{MS}}(\delta)$ or $T_{\texttt{MM}}(\delta)$.*

Proof: The proof is by contradiction. Suppose, that we pick a δ vector which has all its elements not equal to zero and it is optimal under either MS or MM optimization. We

[4]In the following, for the sake of brevity, the MS and MM subscripts will be omitted for statements that are valid regardless of whether MS or MM is assumed.

can calculate δ_{min}, the minimum over all δ_i's. Let us also assume that $\delta_{\text{min}} = \delta_j$ for a certain j. We can now construct a new delay assignment vector δ' where $\delta'_i = \delta_i - \delta_{\text{min}}$ (hence, $\delta'_j = 0$). 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 δ_{min}. Moreover, $T_{\texttt{MS}}(\delta') = T_{\texttt{MS}}(\delta) - N\delta_{\text{min}} \Rightarrow T_{\texttt{MS}}(\delta') \leq T_{\texttt{MS}}(\delta)$ and, at the same time, $T_{\texttt{MM}}(\delta') = T_{\texttt{MM}}(\delta) - \delta_{\text{min}} \Rightarrow T_{\texttt{MM}}(\delta') \leq T_{\texttt{MM}}(\delta)$. 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, $T_{\texttt{MS}}(\delta')$, and less maximum delay, $T_{\texttt{MM}}(\delta')$. Clearly, δ can not be the optimal choice. □

Consequently, we limit the search for an optimal δ among vectors with one or more δ_i elements equal to zero. However, the physical meaning of a zero element, e.g., in position j, in the δ vector is the following: The frame transmission epoch of the j–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 of the $[0, T)$ interval.

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. Therefore, the problem is 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 $T_{\texttt{MS}}$ ($T_{\texttt{MM}}$) and this will be the optimal assignment, $\delta^*_{\texttt{MS}}$ ($\delta^*_{\texttt{MM}}$).

We will describe the algorithm in detail using the notion of a *reference connection* (ref) to identify 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. The concept of a reference connection does not give particular significance or power to any connection. It merely facilitates the description of the control flow of the optimization process. For each of the N configurations, (corresponding to assigning ref from 0 up to $N - 1$), the following steps are performed:

1. The frame transmission epochs are normalized relative to the transmission epoch of connection ref in the $[0, T)$ interval. The transformation preserves the relative position of the frame transmission epochs *within* a cycle of period T and, at the same time, it fixes the transmission epoch of the ref–th connection to coincide with the beginning of the frame transmission cycle (time zero in the $[0, T)$ interval). The result is a vector, s', which preserves the transformed position of the random transmission within the $[0, T)$ cycle ($s'_i \in [0, T)$). The s'_i values are calculated as follows:

$$s'_i = \begin{cases} s_i - s_{ref}, & s_i \geq s_{ref} \\ \\ s_i - s_{ref} + T, & s_i < s_{ref} \end{cases} \qquad i = 0, \cdots, ref - 1, ref + 1, \cdots, N - 1 \quad (1)$$

   The calculation is performed in the 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.

2. The costs, i.e., delays, of aligning any of the remaining (that is, except ref) $N-1$ frame transmission epoch to any of the $N-1$ remaining ideal transmission epochs are calculated. The *cost* of aligning connection i to the j–th ideal epoch is the delay necessary to reach the jT/N time point (either in this $[0,T)$ cycle, or in the next), starting from s'_i. That is, the cost, represented by w_{ij}, is the difference between jT/N and s'_i. If $jT/N < s'_i$, and since the delay has to be positive, 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:

$$w_{ij} = \begin{cases} \frac{jT}{N} - s'_i, & \frac{jT}{N} \geq s'_i \\ T + \frac{jT}{N} - s'_i, & \frac{jT}{N} < s'_i \end{cases} \quad i = 0, \cdots, ref-1, ref+1, \cdots, N-1 \;\; 1 \leq j \leq N-1 \tag{2}$$

The calculation of w_{ij}'s is performed in lines 9–14 of the algorithm in Figure 3.

3. The next step is the calculation of the optimal assignment for the specific configuration, i.e., for the specific reference connection. We will use b_{ij} as the binary variables expressing the selection (if set to 1) or not (if set to 0) of the respective w_{ij}. Therefore, the optimization problem can be stated as follows:

$$\textit{(MS): minimize} \qquad T_{\texttt{MS}} = \sum_{\substack{i=0 \\ i \neq ref}}^{N-1} \sum_{j=1}^{N-1} b_{ij} w_{ij} \tag{3}$$

OR

$$\textit{(MM): minimize} \qquad T_{\texttt{MM}} = \max_{\substack{i=0,\ldots,N-1 \\ i \neq ref \\ j=1,\ldots,N-1}} \{b_{ij} w_{ij}\} \tag{4}$$

$$\textit{s.t.} \qquad \sum_{j=1}^{N-1} b_{ij} = 1, \qquad \sum_{\substack{i=0 \\ i \neq ref}}^{N-1} b_{ij} = 1,$$

$$b_{ij} \in \{0,1\}, \quad i = 0, 1, \cdots, ref-1, ref+1, \cdots, N-1, \quad 1 \leq j \leq N-1$$

The MS optimization version is a typical assignment problem, solvable by polynomial time algorithms, e.g., by the "Hungarian" algorithm [7] and similarly is the MM optimization version, solvable by "bottleneck assignment" algorithms, e.g., [3]. We will assume that the solution of the MS formulation is provided by the ASSIGNMENT_MS function and the solution for the MM formulation by the ASSIGNMENT_MM function. In both cases, the function returns an array b^* of the optimal b assignments as well as the optimal value, T^* of the optimization objective[5].

[5]We pass ref to the respective optimization function to indicate that the ref–th row and column of w indexed by ref are to be ignored.

```
1.  begin
2.  T*_overall := ∞;
3.  δ* := ∅;
4.  for ref := 0 upto N − 1 do
5.      for i := 0 upto N − 1 do
6.          s'_i := s_i − s_ref;
7.          if (s'_i < 0) then s'_i := s'_i + T;
8.      endfor
9.      for i := 0 upto N − 1 and i ≠ ref do
10.         for k := 1 upto N − 1 do
11.             w_ik := kT/N − s'_i;
12.             if (w_ik < 0) then w_ik := w_ik + T;
13.         endfor
14.     endfor
15.     ASSIGNMENT_MS(N, ref, w, b*, T*); (ASSIGNMENT_MM(N, ref, w, b*, T*);)
16.     if T* < T*_overall then
17.        T*_overall := T*;
18.        for i := 0 upto N − 1 and i ≠ ref do
19.            δ*_i := Σ_{k=1}^{N−1} b*_ik w_ik;
20.        endfor
21.        δ*_ref := 0;
22.     endif
23. endfor
24. return(T*_overall, δ*);
25. end
```

Figure 3: Overview of the optimization algorithm.

The algorithm for finding the optimal 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 $T^*_{overall}$ and δ^* are returned. Whether this is the optimal for MS ($T^*_{\texttt{MS}}$) or for MM ($T^*_{\texttt{MM}}$) is only influenced by the selection of the core optimization function at line 15.

Evidently, that there is no need to perform any alignment when only one connection is admitted. Suppose now that two connections are accepted as in Figure 2, the resulting example is trivial but easy 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

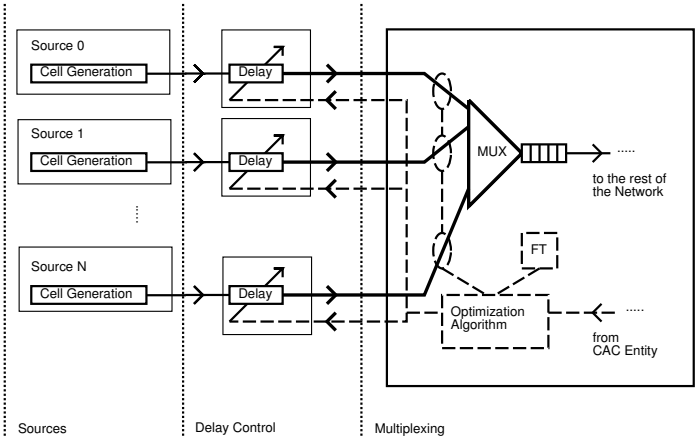

Figure 4: Block diagram of an ATM Network multiplexer and the position of the proposed protocol functions (in dashed lines). The circles around the input links denote the observation of the input stream for SOF cells.

will be $\delta_0 = 0$ and $\delta_1 = .008$ and thus $T^* = .008$. 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 $T^* = .032$. Obviously, for both MS and MM, the first configuration results in the least cost and therefore $\delta^*_{\mathtt{MS},0} = \delta^*_{\mathtt{MM},0} = \delta^*_0 = 0$, $\delta^*_{\mathtt{MS},1} = \delta^*_{\mathtt{MM},1} = \delta^*_1 = .008$ and $T^*_{\mathtt{MS}} = T^*_{\mathtt{MM}} = T^*_{overall} = .008$.

The optimization process provides the optimal delay assignment, δ^*_i, for each source i. What remains, is a mechanism that enforces these delays on each source for cell loss equalization. The implementation of such a mechanism is addressed in the next section. The common assumption for the proposed implementations is the existence of an underlying ATM network. It is therefore convenient to consider the delays in terms of cell transmission times at the respective link speeds. This allows us to directly gain insight into the space (buffer) overheads necessary to hold the cells in order to enforce the delays. For this reason, we will express the delay values of $\delta^*_{\mathtt{MS},i}$ ($\delta^*_{\mathtt{MM},i}$) in terms of integer cell transmission times $\Delta^*_{\mathtt{MS},i}$ ($\Delta^*_{\mathtt{MM},i}$) at peak input link rate. That is, if the input link speed is B_{in} and D is the size of the cell (53 octets for ATM networks) then:

$$\mathtt{MS}: \Delta^*_{\mathtt{MS},i} = \lceil \frac{\delta^*_{\mathtt{MS},i}}{(D/B_{in})} \rceil \qquad \mathtt{MM}: \Delta^*_{\mathtt{MM},i} = \lceil \frac{\delta^*_{\mathtt{MM},i}}{(D/B_{in})} \rceil \tag{5}$$

2.2 Implementation on an ATM Network

The block diagram of Figure 4 presents the operations of the proposed scheme on an ATM network. The implementation depends on the location of the "Delay Control" part. One approach is to integrate the "Delay Control" with the sources because the delay control for any source can be separated from all other sources. Consequently, a protocol must be established between the multiplexer and the sources to ensure the equalized operation, whereby the multiplexer, at connection acceptance and termination points, updates the delays enforced on each source.

A second approach is to integrate the "Delay Control" with the multiplexing operation. Therefore, we can essentially modify the multiplexer scheduling discipline to ensure that the

per–source additional delay is enforced. In this case, the sources are unaware of the necessary alignment operations, and there is no need for the definition of any protocol between sources and multiplexer. However, the multiplexer is burdened with the task of allocating additional buffers in order to delay the incoming traffic.

2.2.1 Protocol–based Implementation

We will describe the functions performed by the protocol while distinguishing them into two groups: (a) multiplexer (MUX) functions, i.e., functions performed by the 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 incoming connections, the calculation of the best 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 of the source 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.

- *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^*.

Refinements to the above framework can be applied, including, for example, compensation for drift of the clocks between the multiplexer and the sources. Note that the destination must also be informed whenever the δ^* assignment is changed in order to calculate the delay necessary to absorb the resulting jitter[6]. If the network behaves as an ideal delay, the necessary delay at the destination is never more than the frame period interval. Unfortunately, this is rarely the case in actual networks where the variance of delays inside the network may require large jitter–absorption delays at the destinations, in the order of several frame intervals.

2.2.2 Scheduling–based Implementation

If the "Delay Control" part of the scheme is incorporated into the multiplexer, then the following *non–work conserving* scheduling discipline is the result: whenever a cell from connection i arrives and Δ_i^* is not zero, it is not directly forwarded to the shared multiplexer buffer but, instead, it enters a pipeline of size Δ_i^* cells (which is functionally equivalent to a shift–register where the shifted unit is a cell). The cell progresses through the pipeline, clocked by the input link cell rate. When the cell reaches the end of the pipeline, it enters the common multiplexer buffer. Hence, effectively, a delay of Δ_i^* cell times is enforced on each incoming cell.

The scheduling discipline is non work–conserving because cells may be in their respective pipelines although they have arrived at the multiplexer, and at the same time the server remains idle because no cell has exited the pipeline to enter the common multiplexer buffer. The operation of sizing the pipeline is controlled by the successive Δ_i^* values. In every other aspect, the operations are as the ones described in the protocol–based implementation with the exception that there is no need to communicate information to the sources.

In the protocol–based implementation, we must determine the size of the necessary buffer at each source, while in the scheduling–based implementation, we must determine the cumulative size of all the pipeline buffer sizes[7]. These engineering problems correspond, respectively, to the determination of the maximum anticipated per–connection delay and the maximum anticipated aggregate, over all connections, delay. Engineering guidelines for dimensioning these buffers are given in the next section along with a delay and loss evaluation of the scheme.

3 Experimental Evaluation

In this section, we present the results of simulations we conducted in order to evaluate four different aspects of the proposed control mechanism:

[6]Unless we allow occasionally the picture to jump or freeze (but never for more than one frame interval) whenever the delay assignments change, which is not a frequent operation anyway.

[7]We assume that in the scheduling–based implementation, all connections utilize a common pool of buffer space.

1. *The per–connection experienced cell loss ratio under the best alignment versus the cell loss ratio under the worst and a random alignment.* The random alignment is constructed by picking time points according to a uniform random distribution in the $[0, T)$ interval. The random alignment is converted to the best alignment through the use of the delays produced by the proposed optimization process. Two sets of examples are considered. The first set represents traffic originating from videoconferencing applications while the second represents a video distribution service based on an actual video traffic trace.

2. *The extent to which the δ^* and Δ^* values are reasonable for a number of configurations.* The optimization process guarantees their optimality but does not give a direct idea about their magnitudes. Therefore, we observe the average and maximum delay assignment and the subsequent additional buffering that is necessary. In addition, we present the relation of the results produced by the `MS` and the `MM` formulations of the optimization problem.

3. *The buffer sizing demands for both the protocol–based and the scheduling–based implementation.* The selection is influenced by the maximum expected value of Δ_i and $\sum_i \Delta_i$ for, respectively, the protocol–based and the scheduling–based implementation.

4. *The extent to which the queueing delay distribution of the connections is influenced by enforcing additional delay according to the proposed scheme.* In particular, the squared coefficient of variation of the queueing delay is used as an indication of the experienced jitter. Since the traffic is delayed for the sake of alignment, further delays at the multiplexer or more variable delays would be an extra penalty that we wish to avoid.

3.1 Cell Loss Equalization Examples

In order to illustrate the effectiveness of the best alignment and thus the effectiveness of the proposed control scheme, we performed a number of simulations. One set of simulations closely followed the assumptions in the simulations presented in [5]. Our example consists of 16 sources feeding a multiplexer. Each source sends its traffic, without smoothing, over a 8.5 Mbps link at a frame rate of 25 frames per second. 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 2.83 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 scheme. The traffic of each source is generated by a Gamma distribution with the same parameters as in [5]. The only difference with respect to the Gamma–based traffic model in [5] is that we consider a cells 53 octets instead of 64 octets. Bursts of more than 801 cells per frame are truncated to satisfy the assumption that the per–frame burst of cells can be transmitted in a single frame period at maximum input link rate. We simulate two hours of source activity or, equivalently, 180000 frames per source and we report the results in Table 1. Less than 10 frames per simulated

Source	s_i (ms)	MS		MM		CLR		
		$\Delta^*_{\mathtt{MS},i}$ (cell times)	$\delta^*_{\mathtt{MS},i}$ (ms)	$\Delta^*_{\mathtt{MM},i}$ (cell times)	$\delta^*_{\mathtt{MM},i}$ (ms)	Best ($\times 10^{-6}$)	Random ($\times 10^{-6}$)	Worst ($\times 10^{-1}$)
0	0.08	199	9.93	199	9.93	4.45	98.47	0.00
1	18.87	23	1.15	23	1.15	3.11	910.03	0.00
2	8.32	84	4.19	84	4.19	4.37	31.90	0.00
3	13.12	38	1.89	38	1.90	3.95	4.89	0.00
4	20.13	48	2.39	48	2.39	3.46	214.62	0.00
5	32.99	241	12.02	191	9.53	4.09	340.95	1.69
6	34.59	109	5.44	109	5.44	6.33	1021.10	3.27
7	36.09	229	11.42	179	8.93	4.18	945.94	4.19
8	35.01	0	0.00	51	2.54	4.59	556.65	4.82
9	25.24	146	7.28	46	2.29	5.17	441.06	5.26
10	17.50	1	0.05	0	0.00	3.71	35.33	5.66
11	23.91	22	1.10	173	8.63	4.30	780.16	5.93
12	39.62	58	2.89	159	7.93	4.76	29.34	6.14
13	26.65	218	10.87	168	8.38	7.00	19.80	6.33
14	21.95	162	8.08	62	3.09	4.73	113.52	6.50
15	23.93	72	3.59	122	6.09	4.48	4.13	6.63
					Overall CLR	4.54	347.00	3.52

Table 1: The cell loss ratio (CLR) experienced by each of the sources under the worst, the best and a random alignment for a Gamma distribution traffic model. The random alignment corresponds to the random frame start times under the s_i column. The best alignment is achieved using the equalization protocol on the random frame start times by generating the respective delays under the $\Delta^*_{\mathtt{MS},i}$ or the $\Delta^*_{\mathtt{MM},i}$ column, depending on selected optimization objective.

source were truncated during the simulations. Summarizing, this set of simulations captures the operation of a multiplexer fed by traffic generated by videoconferencing applications.

However, the diversity of traffic models for video traffic and the need for an experiment for video distribution services motivated a second set of simulations, where we used actual traffic traces and in particular the almost 2 hour long "Star Wars" MPEG sequence [4]. Each of the 16 sources used in this set was sending out the "Star Wars" sequence starting at a random frame of the movie (for the sake of completeness, we provide the starting frame indices for each source on Table 3) and cycling through the entire trace of frames for the duration of the movie. The input link rate is set to 6 Mbps and the output link rate to 15 Mbps. The buffer size is set to 350 cells, representing a maximum queueing delay of 9.92 msec. The traffic generated per frame is fragmented into ATM cells assuming a payload of 44 octets per cell. The frame rate is 24 frames per second. The results of a simulation spanning the duration of the trace are presented in Table 2.

In both sets, three runs of the simulation were performed: for the best, the worst and a random alignment. The sequence of arrivals for a particular source was identical across the three different runs, ensuring the replication of the exact same traffic demands for each run.

Source	s_i (ms)	MS $\Delta^*_{\mathtt{MS},i}$ (cell times)	MS $\delta^*_{\mathtt{MS},i}$ (ms)	MM $\Delta^*_{\mathtt{MM},i}$ (cell times)	MM $\delta^*_{\mathtt{MM},i}$ (ms)	CLR Best ($\times 10^{-6}$)	CLR Random ($\times 10^{-6}$)	CLR Worst ($\times 10^{-1}$)
0	0.08	110	7.77	147	10.39	0.76	79.89	0.00
1	19.66	17	1.20	17	1.20	5.97	50.94	0.00
2	8.67	25	1.77	62	4.38	4.70	97.03	0.02
3	13.67	28	1.98	28	1.98	1.27	1.14	0.06
4	20.97	36	2.54	36	2.54	15.61	0.76	0.16
5	34.36	30	2.12	141	9.96	0.25	181.48	0.32
6	36.03	80	5.65	80	5.65	5.10	32.79	0.54
7	37.60	21	1.48	132	9.33	9.92	1051.91	1.08
8	36.46	258	18.23	37	2.61	0.00	262.00	1.86
9	26.30	108	7.63	34	2.40	0.00	9.63	0.56
10	18.23	0	0.00	0	0.00	6.36	2.67	1.12
11	24.91	311	21.98	127	8.97	0.00	7.15	1.27
12	41.27	43	3.04	117	8.27	1.65	47.87	1.23
13	27.76	50	3.53	124	8.76	6.10	89.88	1.18
14	22.87	82	5.80	46	3.25	2.74	169.13	1.98
15	24.93	16	1.13	90	6.36	8.79	12.99	2.56
					Overall CLR	5.87	131.01	0.87

Table 2: The cell loss ratio (CLR) experienced by each of the sources under the worst, the best and a random alignment for the "Star Wars" trace traffic. The random alignment corresponds to the random frame start times under the s_i column. The best alignment is achieved using the equalization protocol on the random frame start times by generating the respective delays under the $\Delta^*_{\mathtt{MS},i}$ or the $\Delta^*_{\mathtt{MM},i}$ column, depending on selected optimization objective.

Source	Starting Frame
0	116516
1	128719
2	147252
3	89226
4	126688
5	115295
6	158192
7	132923
8	135107
9	44055
10	4874
11	157641
12	69588
13	125959
14	170511
15	150755

Table 3: The starting frame index for each of the 16 sources as used in the trace–driven simulations.

The random alignment was obtained by setting the transmission epochs to the s_i's of the second column of Tables 1 and 2. The s_i's were taken from a uniform distribution in the respective $[0, T)$ interval. The best alignment was achieved through the loss equalization scheme by delaying the traffic that is generated at the s_i epochs. The per–source delay necessary to achieve the best alignment is produced by either the MS or the MM optimization algorithm. The values of $\Delta^*_{\texttt{MS},i}$, $\Delta^*_{\texttt{MM},i}$, $\delta^*_{\texttt{MS},i}$ and $\delta^*_{\texttt{MM},i}$ are presented under the respective columns of Tables 1 and 2. Finally, the worst alignment results are provided just to illustrate the extreme case that a random alignment can reach if no frame alignment control is enforced.

First, consider Table 1. The reference connection according to MS is number 8, while according to MM it is connection 10. Note how the MM optimization avoids the excessive delays that MS produced for connections 5, 7, 9, 10, 13 and 14 while at the same time it inflates the delays of 8, 11, 12 and 15. Consequently, the maximum delay under MM is 9.93 msec (source 0) compared to 12.02 msec of MS (source 5). The remarkable result is the difference of the CLR values for the random configurations. It ranges from 4.13×10^{-6} for source 15 to 1021.10×10^{-6} for source 6, a difference of almost three orders of magnitude. Thus, connection 6 receives an effective QoS CLR of 10^{-3} and connection 15 an effective QoS CLR of 10^{-6}. Under the best alignment that the protocol produces, the cell losses are equalized around the overall value of 4.54×10^{-6}. Note also that at the same time, the overall CLR of the random alignment (347×10^{-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 transmission epochs of each source differ only slightly (less than the time for a cell transmission time on the input link). Without the operation of the proposed scheme there is nothing to prevent the worst case configuration. Thus, a random configuration can deteriorate to a worst–case CLR of 10^{-1}.

Table 2 summarizes the results for the "Star Wars" sequence. In the random alignment case, certain connections are fortunate to have no losses at all. This is a consequence of the random point at which the respective connection started playing out the video trace (Table 3). If the frame transmission epoch of a connection is sufficiently spaced apart from other transmission epochs and if the source is sending high activity frames when neighboring sources (in the sense of neighboring transmission epochs) are sending low activity frames, then it is possible for such a connection to experience zero losses. For this reason, under the best alignment, sources 8, 9 and 11 suffer no losses at all while source 4 suffers the most, 15.61×10^{-6}. Even in this case though, the benefits of using the equalization scheme are obvious. The CLR of the random alignment ranges from 0.76×10^{-6} to 1051.91×10^{-6}, while the overall average CLR is 131.01×10^{-6}. In sharp contrast, the best alignment brings the loss ratio of all sources closer together and around the value of 5.87×10^{-6}. Finally, note that the reference connection is number 10 for both the `MS` and the `MM` configuration. From our experimentation with the two optimization criteria, we have concluded that in the majority of cases, the optimal delay assignment under both `MS` and `MM` is characterized by the same reference connection. Moreover, in both the "Star Wars" and the videoconference experiment, the delay imposed on the traffic is lower than the delay necessary for smoothing which is in the range of 40 msec. A detailed discussion of the dynamics of Δ^* follows in the next subsection.

3.2 The Dynamics of the Optimal Delay Assignment

Since Δ_i^*'s are the result of an optimization process, we can not obtain them in a closed form. For this reason, we report on the results generated by solving a large number (tens of thousands) of optimization instances for uniformly generated s_i's in the $[0, T)$ range, where $T = 40$ msec (frame rate of 25 frames per second). Moreover, we vary the input link rate, B_{in}, from 4.25 to 8.5 to 17.0 Mbps and the number of multiplexed sources from 2 to 128.

The two different optimization formulations accentuate two different aspects of the alignment delay assignment. The `MS` optimization aims towards maintaining an overall low delay overhead with the hope that no connection is delayed more than absolutely necessary. The `MM` optimization tries to avoid victimizing particular connections in terms of delay more than all other connections. Intuition suggests that `MM` may tend to inflate the overall delay. Surprisingly this is *not* the case. As Figure 5 indicates, the average value of the overall delay ($\mathrm{E}[\sum_i \Delta_i^*]$) for both `MS` and `MM` is almost identical for different link rates and for variable number of sources. The same holds true for the average per–connection delay ($\mathrm{E}[\Delta_i^*]$), as Figure 6 suggests.

Increasing number of sources brings an anticipated increase in the overall delay (more sources, more delays to sum up). But it also brings the interesting observation that the per–source delay, depicted in Figure 6, is reduced dramatically (note that in Figure 6 the y–axis is logarithmic). An explanation is the following urn–like example: one can think of the sources as partitions of a box. A number of marbles equal to the number of partitions is dropped

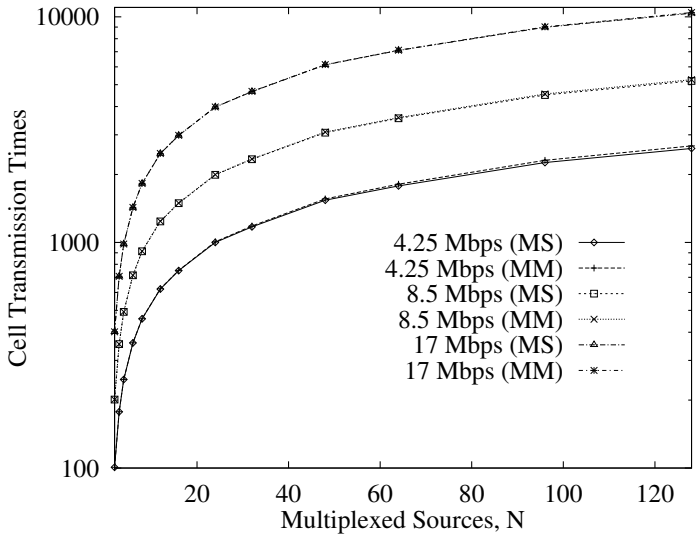

Figure 5: Average value of the sum of alignment delays, E[$\sum_i \Delta_i^*$], for both `MS` and `MM` optimization at different input link rates, B_{in}. The delay is expressed in cell transmission times at the input link rate. Note that the lines for `MS` and `MM` are almost identical.

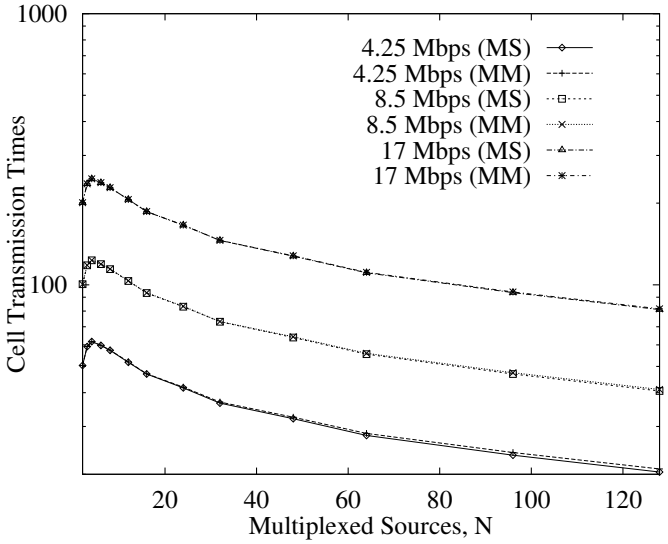

Figure 6: Average value of the per–connection alignment delay, E[Δ_i^*], for both `MS` and `MM` optimization at different input link rates, B_{in}. The delay is expressed in cell transmission times at the input link rate. Note that the lines for `MS` and `MM` are almost identical.

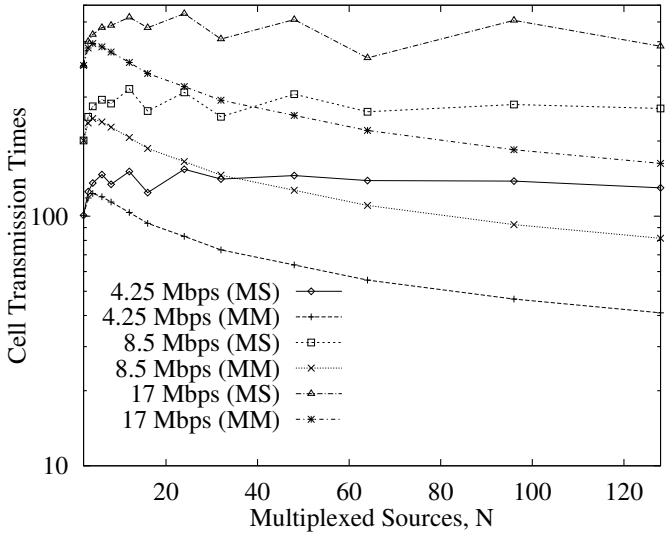

Figure 7: Average value of the maximum per–configuration delay, E[max$_i$ Δ_i^*], for both `MS` and `MM` optimization at different input link rates, B_{in}. The delay is expressed in cell transmission times at the input link rate.

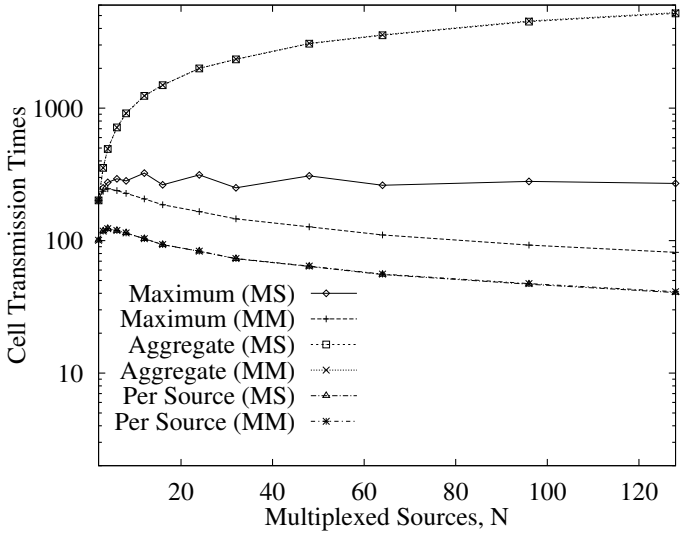

Figure 8: Comparison of the averages of (a) the maximum per–configuration delay, E[max$_i$ Δ_i^*], (b) the aggregate per–configuration delay, E[$\sum_i \Delta_i^*$], and (c) the per–source alignment delay, E[Δ_i^*], for both `MS` and `MM` optimization at an input link rate of 8.5 Mbps. Note that the only marked difference is in the average value of the maximum alignment delay.

randomly into the box. The more the partitions, the more likely it is that no more than one marble falls within each partition. In our example, the marbles are the s_i's and the number of partitions is equal to the number of sources. Hence, the more the sources, the smaller the average per–source delay alignment time.

Another aspect of the behavior of Δ_i^*'s, supported by the results in Figures 5 and 6, is that doubling or halfing the input link rate results in a proportional increase or decrease of the delay times. Such behavior would seem to cause problems at higher link speeds by requiring an ever increasing buffer in order to accommodate the delayed traffic. However, there always 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. Unfortunately, the average delay assignment can not be used for engineering the buffers necessary to delay traffic. Instead, we are interested in the maximum, i.e., worst–case, buffer size necessary to accommodate the assigned delays.

Let us first focus on the expected value of the per–configuration maximum delay over all simulated configurations, E[$\max_i \Delta_i^*$]. Figure 7 shows that, indeed, the `MM` algorithm plays a crucial role in restricting the maximum delay, and hence the maximum necessary buffer size. That is, as Figure 7 indicates, the expected maximum for the `MS` optimization remains almost constant for increasing number of sources while, for `MM` optimization, it is reduced drastically for increasing number of sources. One would therefore anticipate that the `MM` optimization is the obvious choice. This is not necessarily true, as we will see in the next subsection. Finally, Figure 8 summarizes the dynamics of the delay assignments for both `MS` and `MM` for variable number of sources at a fixed input link rate of B_{in} = 8.5 Mbps.

3.3 Engineering Guidelines

The average value of the maximum anticipated delay is not sufficient for engineering the buffer sizes for either the protocol–based or the scheduling–based implementations. Instead, we would like to know whether, for a certain fixed buffer size, Q, we are able to support the delays produced by the optimization process. Alternatively, we would like to select the minimum buffer size, $Q_{\min}$, that ensures that the necessary delays produced by the scheme can be satisfied in100 % of the cases. In the protocol–based implementation, the fixed size buffer is assigned to each individual source. In the scheduling–based implementation, it is assigned for all sources and it is located at the multiplexer. For this study, we introduce a probability quantity that we will call *probability of coverage*. In the protocol–based implementation it is interpreted as Pr[$\max_i \Delta_i^* \leq Q$] while in the scheduling–based implementation it is interpreted as Pr[$\sum_i \Delta_i^* \leq Q$].

To this extent, we present the results in Figures 9 and 10 for `MS` and `MM` optimization respectively, for the protocol–based implementation assuming an input link rate of 8.5 Mbps. Note that the results are influenced *only* by the input link rate, B_{in}, and the assumption that the s_i's are uniformly distributed in the $[0, T)$ interval. The surface depicting the `MM` optimization "flattens" at 100 % faster than the surface for the `MS` curve. In fact, most of the "flat" area of the `MS` surface is not exactly at 100 %, but at lower values.

A better look at the same set of experiments is presented in Figures 11 (32 sources but variable input link rate) and 12 (8.5 Mbps input link rate but variable number of sources) comparing the `MS` and the `MM` results. As we can see, choosing `MM` makes sense for moderate to

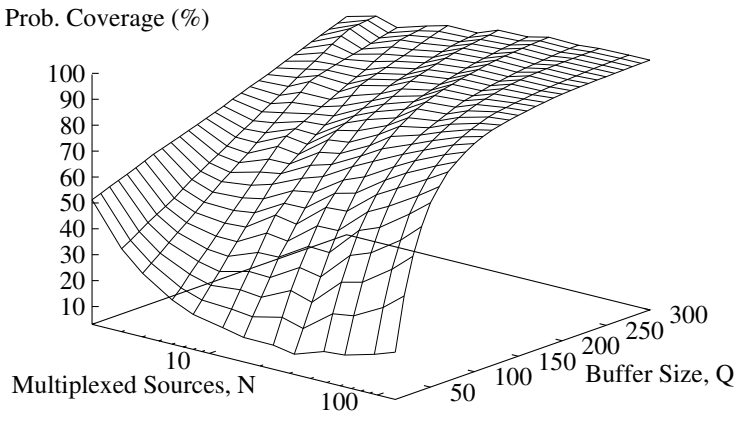

Figure 9: Probability of coverage for the protocol–based implementation, Pr[$\max_i \Delta_i^* \leq Q$], (displayed as a percentage) under the MS optimization for variable number of multiplexed sources, N, and variable buffer size, Q. The input link rate is 8.5 Mbps.

large buffer sizes (for larger buffer values than the ones of the crossover points at a1, a2, a3 and b1, b2, b3) while for smaller buffer sizes it is preferable to use the MS optimization. Hence, if, because of engineering constraints, we have to minimize the buffer sizes allocated to delaying traffic, a good choice for assigning delays is the MS optimization. If we are relatively flexible in selecting buffer sizes, then the MM optimization is preferable[8].

The results in Figures 12 imply that for decreasing number of admitted sources, the necessary buffer size increases. For example, if we maintain 32 Permanent Virtual Circuits (PVC) conveying video traffic, and we use MM optimization, then any other non–permanent video connection established through the same multiplexer does not need more than 160 cells allocated for the equalization scheme. However, if there are no PVCs and the number of admitted sources varies through time, then the necessary per–source buffer can reach its maximum value when the number of admitted sources[9] is 2. The calculation of the per–source buffer necessary for the 100 % coverage assuming only two sources is trivial because only one of the two sources is delayed for a maximum of $T/2$ time units. Hence, the minimum per–source buffer that can successfully support the scheme is:

$$Q_{\min} = \lceil \frac{(T/2)}{(D/B_{in})} \rceil \tag{6}$$

For B_{in} = 8.5 Mbps, $Q_{\min}$ is a very modest 402 cells.

Different steps are necessary for dimensioning the pipeline buffer in the scheduling–based implementation. As Figure 13 suggests, the number of multiplexed sources plays a crucial role. However, this time, the transition to the 100 % coverage is very steep and it does not differ significantly for the MS and the MM formulation (therefore, we only present the graph for the MS optimization in Figure 13). The main feature is that increasing the sources implies

[8]If a source receives a request to delay its traffic more than what is possible with its available buffer size, we assume that it delays it for the maximum it can.

[9]Recall that for only one source there is no need for alignment delays.

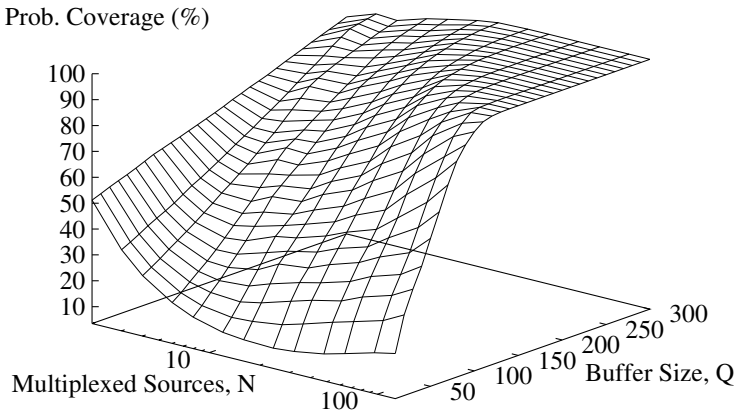

Figure 10: Probability of coverage for the protocol–based implementation, Pr[$\max_i \Delta_i^* \leq Q$], (displayed as a percentage) under the `MM` optimization for variable number of multiplexed sources, N, and variable buffer size, Q. The input link rate is 8.5 Mbps.

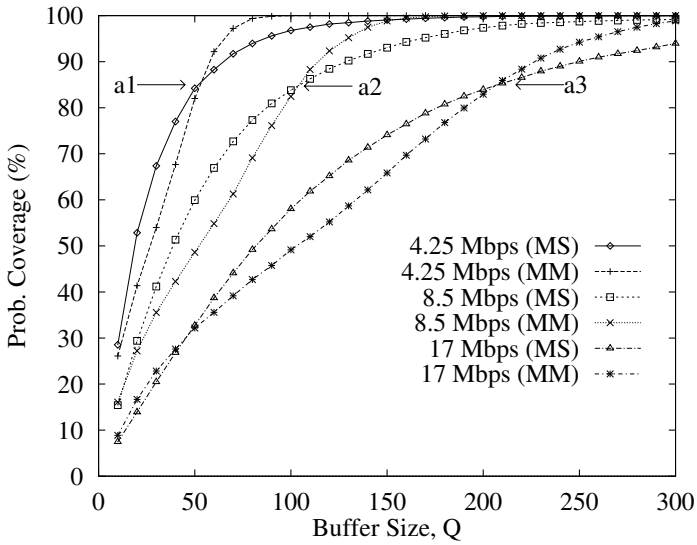

Figure 11: Probability of coverage for the protocol–based implementation, Pr[$\max_i \Delta_i^* \leq Q$], (displayed as a percentage) under both `MS` and `MM` optimization for different input link rates, B_{in}, 32 sources and variable buffer size, Q.

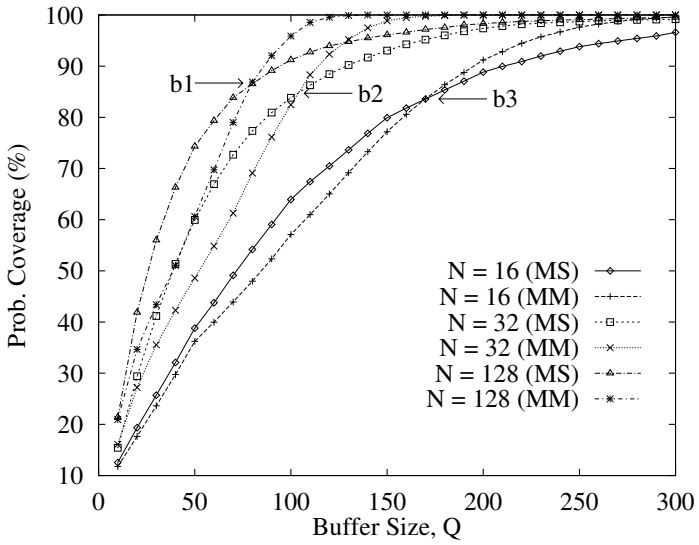

Figure 12: Probability of coverage for the protocol–based implementation, Pr[$\max_i \Delta_i^* \leq Q$], (displayed as a percentage) under both `MS` and `MM` optimization for different number of sources, input link rate of 8.5 Mbps and variable buffer size, Q.

increase of the necessary maximum pipeline buffer. Consequently, we must dimension the pipeline buffer based on the maximum number of anticipated connections. Fortunately, other concerns, e.g., cell loss ratio, already place a cap on the number of connections that can be supported. For example, by inspecting Figure 15, we can conclude that if a maximum of 16 connections can be admitted, a pipeline buffer of around $Q_{\min}$ = 3500 cells is sufficient to support the scheduling–based implementation of the scheme. Finally, for completeness, consider Figure 14, which displays the dependency of probability of coverage (and hence that of Q_{min}) on the input link rate, B_{in}.

3.4 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, our equalization scheme delays the traffic for the sake of an ideal alignment. We would anticipate that the traffic is not penalized any further in terms of delay at the 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. To illustrate this, we return to the simulation experiments using the Gamma and the "Star Wars" simulations and we evaluate the delay performance of the aggregate traffic as well as that of an individual connection. (We pick source 9 to be the individually observed source in all our simulation experiments.)

Figures 16 (for the aggregate traffic) and 18 (for the traffic of connection 9) present the sample distribution of the delay at the multiplexer, i.e., not including alignment delays, for the worst, a random and the best alignment for the Gamma–based simulation. Evidently, the

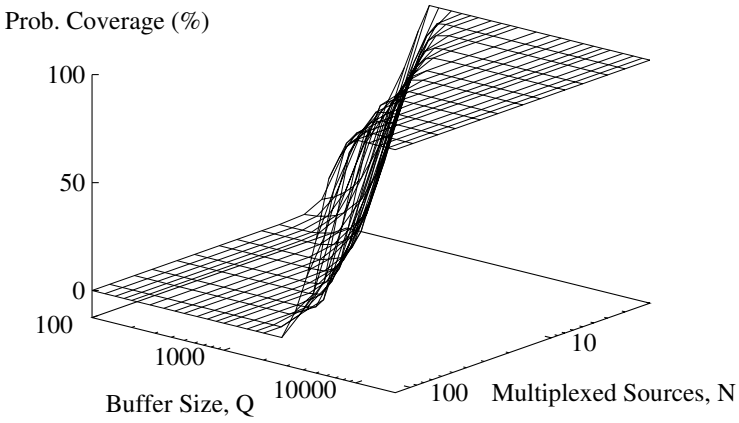

Figure 13: Probability of coverage for the scheduling–based implementation, Pr[$\sum_i \Delta_i^* \leq Q$], (displayed as a percentage) under the `MS` optimization for variable number of multiplexed sources, N, and variable buffer size, Q. The input link rate is 8.5 Mbps. Note that the corresponding graph for the `MM` optimzation is very similar to this one.

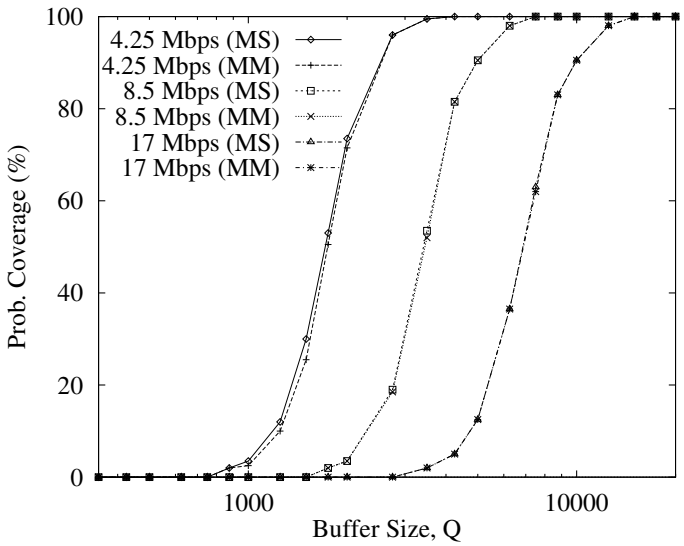

Figure 14: Probability of coverage for the scheduling–based implementation, Pr[$\sum_i \Delta_i^* \leq Q$], (displayed as a percentage) under both `MS` and `MM` optimization for different input link rates, B_{in}, 32 sources and variable buffer size, Q.

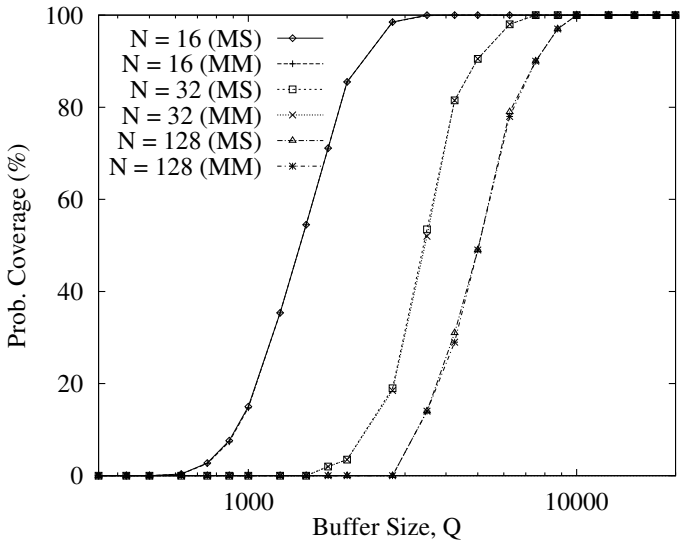

Figure 15: Probability of coverage for the scheduling–based implementation, Pr[$\sum_i \Delta_i^* \leq Q$], (displayed as a percentage) under both `MS` and `MM` optimization for different number of sources, input link rate of 8.5 Mbps and variable buffer size, Q.

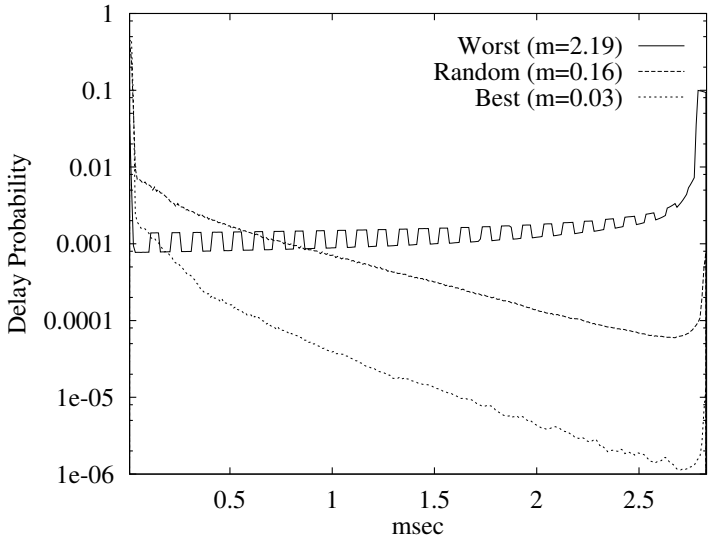

Figure 16: Sample distribution of the multiplexer delay under the best, the worst and the random alignment of Table 1 for Gamma distribution traffic.

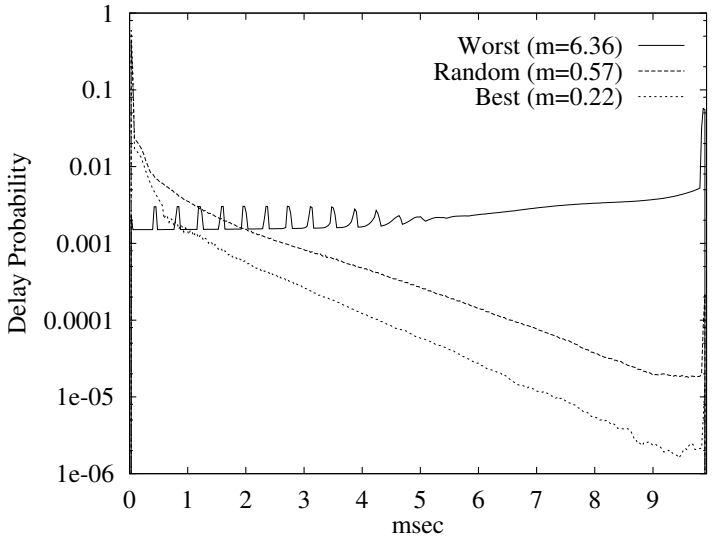

Figure 17: Sample distribution of the multiplexer delay under the best, the worst and the random alignment of Table 2 for the ”Star Wars” trace traffic.

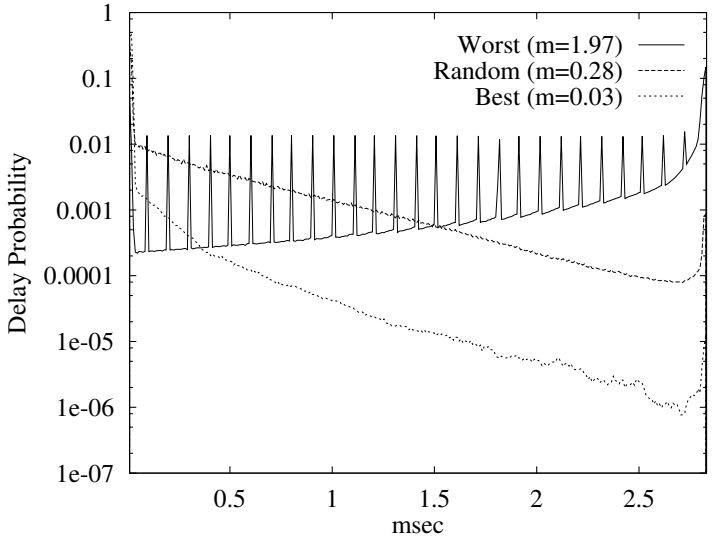

Figure 18: Sample distribution of the multiplexer delay under the best, the worst and the random alignment of Table 1 for Gamma distribution traffic, as experienced by connection number 9.

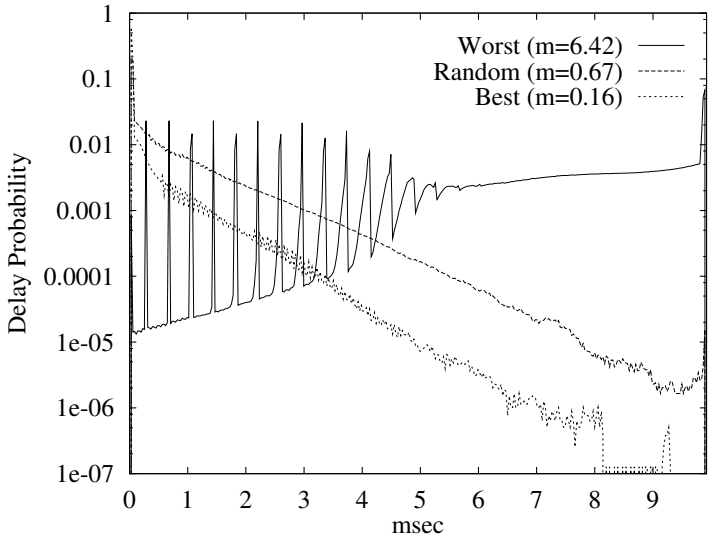

Figure 19: Sample distribution of the multiplexer delay under the best, the worst and the random alignment of Table 2 for the ”Star Wars” trace traffic, as experienced by connection number 9.

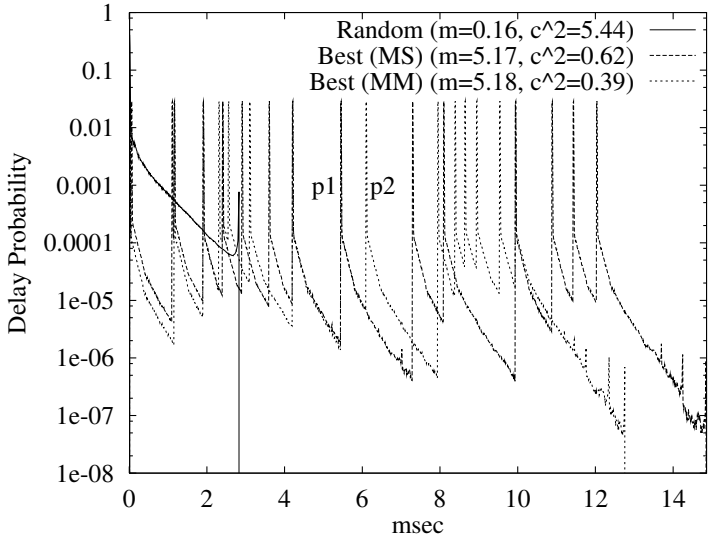

Figure 20: Sample distribution of the total delay (inclusive of alignment delays) under the best, the worst and the random alignment of Table 1 for Gamma distribution traffic.

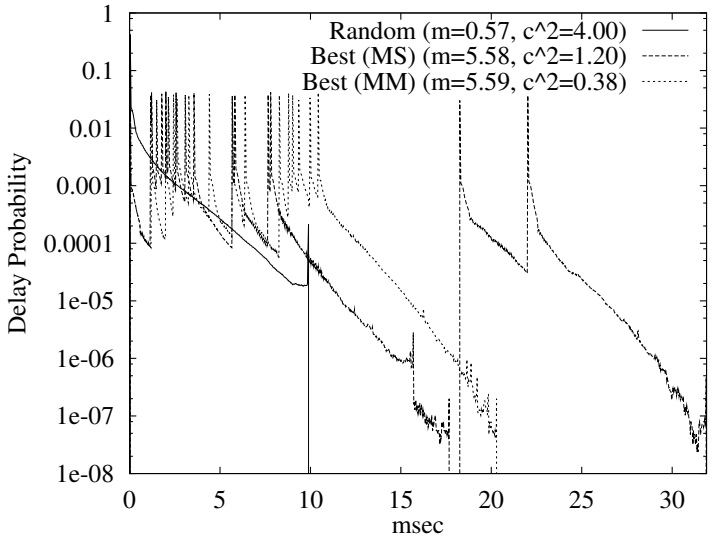

Figure 21: Sample distribution of the total delay (inclusive of alignment delays) under the best, the worst and the random alignment of Table 2 for the "Star Wars" trace traffic.

average delay (denoted with "m" in the plots) of the aggregate traffic greatly improves from the worst (2.19 msec) to the random (0.16 msec) to the best alignment (0.03 msec) and so does the average delay of the isolated connections (from 1.97 to 0.28 to 0.03 msec). Similar observations hold for the "Star Wars" simulation, presented in Figures 17 (aggregate) and 19 (individual).

The worst alignment results in a very heavy tail as we would expect from the heavy losses. In addition, it demonstrates a regular shape in the sample 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 a fixed size equal to the number of sources. Therefore, for our examples, the batch size is 16. Until the next batch arrival, only B_{out}/B_{in} cells can be serviced. For B_{in} = 8.5 Mbps and B_{out} = 45 Mbps, only $B_{out}/B_{in} \simeq 5.3$ cells can be serviced on average. For an initially empty queue, after 16 cells arrive, 5.3 get serviced and the next batch finds the queue with an average of $16 - 5.3 = 10.7$ cells, representing an average delay of around 0.1 msec for the next incoming batch. The 0.1 msec is the point where the first sawtooth rises in Figure 16. The next batch will very likely find the queue with an average of $32 - 10.6 = 21.4$ cells or almost 0.2 msec of delay time (where the second sawtooth rises) and so on for each sawtooth starting at $0.1 \times k$ msec for $k = 1, 2, ...$ etc. 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 depicted by the plots. However, the important conclusion from these figures is that the additional alignment delay results in an overall *reduced* delay at the multiplexer. Thus, the sources do not get penalized twice in terms of delay, i.e., both for the alignment and at the multiplexer.

The investigation of the delay performance is not complete without a look at the jitter performance of the connections. If the alignment inflates the delays for the sake of cell losses, it should not do so at the cost of the jitter performance. For the study of jitter, we consider

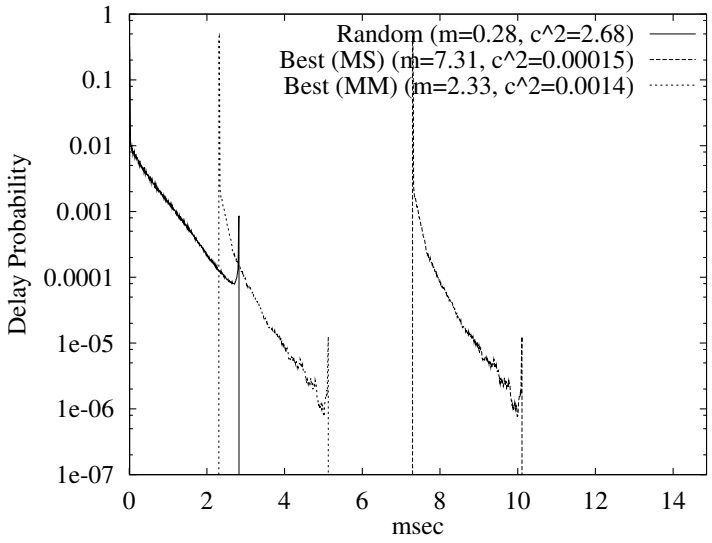

Figure 22: Sample distribution of the total delay (inclusive of alignment delays) under the best, the worst and the random alignment of Table 1 for Gamma distribution traffic, as experienced by connection number 9.

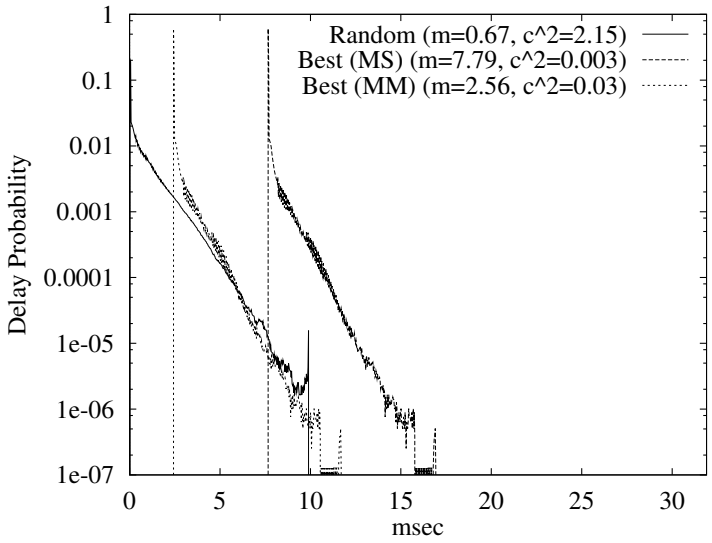

Figure 23: Sample distribution of the total delay (inclusive of alignment delays) under the best, the worst and the random alignment of Table 2 for the ”Star Wars” trace traffic, as experienced by connection number 9.

the delay distributions that include the alignment delays in addition to the multiplexer delays. This setting is more representative of the delay experienced by connections from the traffic generation point and up to the departure from the multiplexer. In particular we will consider the squared coefficient of variation (denoted in the plots by "c^2") as a convenient way to quantify jitter. Removing from consideration the worst case, we focus on a comparison between the results for the best alignment under `MS` and `MM` against the random alignment. Figures 20 (aggregate traffic) and 22 (individual traffic from connection 9) present the results for the Gamma–based simulation, while Figures 21 and 23 present the corresponding results for the "Star Wars" trace traffic simulation. Note that the peaks of the aggregate distribution for the best alignment under `MS` and `MM` correspond to the alignment delays under the respective optimization. For example, the peak to the right of p1 in Figure 20 corresponds to the delay (5.44 msec) imposed on source 6 under `MS` as indicated on Table 1. Similarly the peak to the left of p2 in Figure 20 corresponds to the delay (6.09 msec) imposed on source 15 under `MM` of the same table. Similarly, the two peaks on the extreme right side of Figure 21 correspond to the alignment delays for sources 8 (18.23 msec) and 11 (21.98 msec).

The important result is that the s.c.v. of delay is reduced for the best alignment (for either `MS` or `MM`) compared to the random case. In fact, if we focus on the delay distribution of one particular connection (as we do in Figures 22 and 23), the s.c.v. is essentially zeroed. Since the jitter absorption is likely performed on a per–connection basis, the s.c.v. values for the individual connection are more representative of the actual jitter than the values for the aggregate traffic. Hence, the presented scheme achieves not only fair losses but low jitter as well.

4 Conclusions

In this paper we have proposed a scheme that rectifies the cell loss ratio fairness problem across multiple video connections that feed the same finite buffer multiplexer. The scheme can be implemented either as a protocol between sources and multiplexer or as a particular scheduling discipline.

Through simulation experiments, using both traffic models and actual traffic traces, we have verified the effectiveness of the scheme in terms of cell loss ratio fairness. We also have demonstrated that the scheme results in reduced delay variability that directly benefits real–time video connections. Moreover, we have investigated the anticipated buffer overheads necessary to support the scheme. Finally, we have presented two underlying optimization choices, that capture the sensitivity to increased delays and we have illustrated through examples the tradeoff of selecting one over the other as it relates to buffer overheads.

Open problems, in the same context, that are worth particular attention include the generalization to heterogeneous sources and different frame rates and the generalization to a cluster of networks elements instead of an individual multiplexer. One can also view the optimization of frame transmission epochs as a burst–level scheduling scheme that strives to minimize delays inside the network by enforcing controlled delays at the network access points. As such, it can have ramifications, through suitable extensions, to the scheduling of other types of bursty traffic apart from video.

Acknowledgements

We thank Professors James F. Kurose and Don Towsley of the Univ. of Massachusetts at Amherst for their constructive comments on earlier versions of this paper. We are also grateful to Dr. Mark W. Garrett of Bellcore and Professor Martin Vetterli of Univ. California at Berkeley for making available the ”Star Wars” video trace.

References

[1] S. Chowdhury and K. Sohraby, "Bandwidth Allocation Algorithms for Packet Video in ATM Networks," *Comp. Networks and ISDN Systems*, vol. 26, no. 9, May 1994.

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

[3] R. S. Garfinkel, "An Improved Algorithm for the Bottleneck Assignment Problem," *Operations Research*, vol. 19, pp. 1747–1751.

[4] M. W. Garrett and W. Willinger, "Analysis, Modeling and Generation of Self-Similar VBR Video Traffic," to appear *Proc. ACM SIGCOMM '94*, pp. 269–280, 1994.

[5] 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.

[6] 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.

[7] H. W. Kuhn, "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, vol. 2, pp 83–97, 1955.

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

[9] D. LeGall, "MPEG: A Video Compression Standard for Multimedia Applications," *Commun. of the ACM*, pp. 47–58, April 1991.

[10] 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.

[11] 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.

[12] 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.

[13] 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.