Efficient Algorithms for Exact Computation of the Loss Curve of Video Sources

Christos Tryfonas
Anujan Varma

Computer Engineering Department
University of California
Santa Cruz, CA 95064
{tryfonas, varma}@cse.ucsc.edu

Subir Varma

Hybrid Networks Inc.
6409 Guadalupe Mines Road
San Jose, CA 95120
subir@hybrid.com

ABSTRACT

The *loss curve* of a video source characterizes the loss rate of the video stream generated by the source as a function of the allocated buffer size for a given transmission rate. The loss curve is useful in the optimal allocation of resources when the video stream is transmitted over a packet network, so that the desired tradeoff can be reached among the loss rate, bandwidth and the buffer space to be allocated in the network. We present a deterministic algorithm for exact computation of the entire loss curve of a video elementary stream. The algorithm exploits the piecewise linearity of the loss curve and computes only the points at which the slope of the loss curve changes. We also present an extension of the algorithm to MPEG-2 Transport Streams. The algorithm was able to compute the entire loss curve of a 2-hour video elementary stream in approximately 11 seconds on a Sun Ultra-2 workstation. The efficiency of the algorithm makes it suitable for both off- and on-line QoS provisioning in networked video environments.

1 Introduction

The explosion of the Internet has spawned video-based services over packet networks, such as streaming video and video-on-demand. Many of these applications require, or can benefit from, the network's ability to provide Quality-of-Service (QoS) guarantees. The QoS guarantees are usually in the form of bandwidth, end-to-end delay, and/or the loss rate experienced by the traffic stream.

The rate variability of video sources has introduced the need for characterizing the traffic so that the amount of resources to be allocated by the network (such as bandwidth, buffer space, etc.) can be estimated during the call admission control (CAC) process. The characterization of the traffic stream is also necessary for efficient policing of the traffic. The two primary resources allocated by the network are the transmission rate ρ and the buffer size B. In an application where no losses are allowed, the video source can be characterized completely by determining the minimum buffer size necessary to avoid losses as a function of the rate ρ. This characterization is referred to as the *burstiness curve* [8]. Efficient algorithms for exact computation of the burstiness curve can be found in [10].

When the application can tolerate some amount of loss, the amount of bandwidth and/or buffer space needed in the network can often be reduced significantly, since the burstiness curve of the source

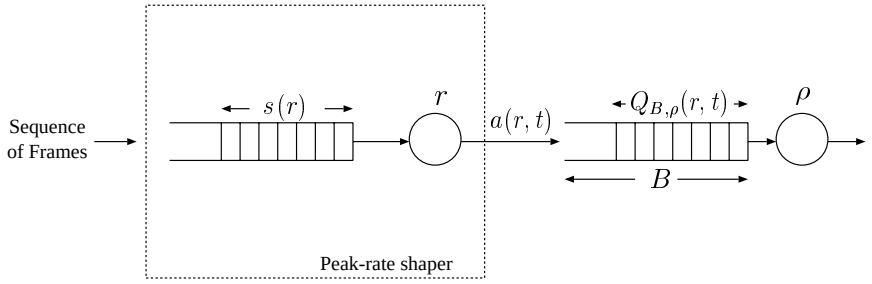

Figure 1: Model used for the computation of the loss curve of a video stream.

typically exhibits a long tail. The problem of determining the necessary network resources then becomes the problem of choosing a specific vector from the three-dimensional space (ρ, B, ϵ). To simplify the problem, either the transmission rate ρ or the buffer size B can be fixed to calculate the corresponding curves. The ϵ versus B curve for a specific transmission rate ρ enables the estimation of the loss rate resulting from a given buffer size to send the video source at a certain transmission rate. We refer to a plot of the loss rate as a function of the buffer size for a given rate ρ as the *loss curve* [2] of the source corresponding to the rate ρ. Our interest in this paper is to investigate efficient algorithms for the computation of such loss curves. In some other cases, it is necessary to obtain the variation of the buffer size B to achieve a given loss rate as a function of the transmission rate ρ. Such a plot can be constructed by computing a series of loss curves for different values of the rate ρ, and reading off the B values for the given loss rate ϵ.

In this paper, we present a deterministic algorithm for the *exact* computation of the loss curve of an elementary video stream. The algorithm exploits the piecewise linearity of the curve and performs calculations only at points where the slope of the loss curve changes, thus using the minimum number of points needed to exactly characterize the curve. The piecewise linear nature of the loss curve was observed by Wong and Varayia [12], but they did not analyze the behavior further. By characterizing only the points at which the slope of the loss curve changes, our algorithm achieves optimality in the number of points needed for the exact computation of the loss curve. The algorithm exhibits low time- and space-complexity and therefore, is attractive for use not only in off-line video systems but also in real-time video distribution systems that need to perform estimation of the loss curve in real time. We also give an extension of the algorithm for computation of the loss curve of MPEG-2 Transport Streams.

The rest of the paper is organized as follows: In Section 2, we first describe the model for characterizing the queue behavior of the corresponding elementary video stream. This model is then used for the exact computation of the loss curve, which is presented subsequently. In Section 3, we consider an extension of the algorithm to MPEG-2 Transport Streams. In Section 4, we evaluate the performance of the algorithm with many video traces. Finally, in Section 5 we conclude the paper with a brief summary of this work.

2 Algorithm for Exact Computation of the Loss Curve of Elementary Video Streams

2.1 Model

The most common case for transporting video over a packet-switched network is by using an elementary video stream. An elementary video stream consists of a sequence of frames generated at a fixed rate (frame period), that may have varying sizes due to scene changes. In this section we present an algorithm for exact computation of the loss curve for such traffic sources.

The model that we use for the computation of the loss curve in the case of an elementary video stream is shown in Figure 1. The traffic from the source goes through a peak-rate shaper which produces a new time sequence for the bit-stream. For different values of the peak rate r, the queue of the peak-rate shaper may have a different maximum length, denoted by $s(r)$. The bit-stream at the output of the peak-rate shaper, denoted by $a(r,t)$ in the figure, is also dependent on the peak rate r. Our interest is in computing the loss curve of the traffic source at the output of the peak-rate shaper for a specific transmission rate ρ and buffer size B.

We can determine the loss rate of the source for a given transmission rate ρ and buffer size B by feeding the bit-stream at the output of the peak-rate shaper in Figure 1 into a second leaky-bucket shaper with rate ρ and buffer size B. By recording the amount of data lost from this buffer for different buffer sizes, we can obtain the loss curve of the source for the given peak rate r and transmission rate ρ. The B versus ρ curve for a given maximum acceptable loss rate ϵ is obtained by constructing a series of loss curves for different choices of the rate ρ and reading off the B values corresponding to the given loss rate.
We first define $L_{B,\rho}(t)$ as the amount of traffic of the video stream lost over the time interval $[0,t]$. Then, the fraction of lost traffic, or the loss rate $\epsilon(B,\rho)$, is given by

$$\epsilon(B,\rho) = \frac{L_{B,\rho}(T)}{M},$$

where M is the total number of bits in the elementary stream and T its duration. For plotting the complete loss curve for a given rate ρ, it is only necessary to consider the range of buffer sizes $0 \leq B \leq \sigma(\rho)$, where $\sigma(\rho)$ is the maximum burstiness of the source at rate ρ. For a complete characterization of the source, a series of loss curves can be constructed for different transmission rates in the range $0 < \rho < r$.

To complete the definition of the loss curve, we must define the loss rate for the boundary values of B and ρ:

$$\epsilon(B,\rho) = 0, \quad B > \sigma(\rho); \tag{1}$$

$$\epsilon(0,\rho) = \frac{r-\rho}{r}, \quad 0 \leq \rho \leq r; \tag{2}$$

$$\epsilon(B,0) = 1, \quad 0 \leq B < M; \tag{3}$$

$$\epsilon(B,0) = 0, \quad B \geq M; \tag{4}$$

To analyze the behavior of the queue at the second shaper in Figure 1, we first need to characterize the on-off signal at the output of the peak-rate shaper. Hence, we first present an algorithm to compute the on-off periods of the bit-stream $a(r,t)$ at the output of the peak-rate shaper when the input source is an elementary video stream. These on-off periods are then used in the computation of the loss curve.

We define *active period* as a maximal period of time during which the peak-rate shaper is continuously transmitting traffic. This corresponds to an on-period of the signal $a(r,t)$. Let $n_a(r)$ denote the number of active periods of the signal for a peak rate of r, s_i^r the time instant at which the ith active period commences, and t_i^r the time when it ends. We need to compute the active periods (s_i^r, t_i^r), for $1 \leq i \leq n_a(r)$.

We assume that the number of frames in the video trace is N, the length of the trace is T, the frame rate is f, and the size of the ith frame is d_i bits. Let $d_{max} = \max_{1 \leq i \leq N} d_i$ be the maximum frame size in the trace. We also assume that a frame is added instantaneously to the queue of the peak-rate shaper at the end of the corresponding frame period. That is, the first frame arrives in the queue at time $1/f$, which marks the beginning of the first active period. When the peak rate r satisfies $r \geq d_{max}/f$, it is trivial to compute the active periods of the signal $a(r,t)$.

$$s(r) = d_{max}, \qquad n_a(r) = N, \qquad s_i^r = \frac{i}{f}, \qquad \text{and } t_i^r = s_i^r + \frac{d_i}{r}. \tag{5}$$

However, in the general case when $r < d_{max}/f$, neighboring frames overlap with each other in the shaper queue, leading to larger maximum queue lengths and a smaller number of active periods. Let $q_i(r)$ be the size of the queue at the input of the peak-rate shaper just after the instant when the ith frame arrives. The maximum queue length will always occur just after an arrival of a frame, and is given by

$$s(r) = \max_{1 \leq i \leq N} q_i(r). \tag{6}$$

The active periods of the elementary stream can be determined by traversing the sequence of frames and computing the queue size at the instant just after each frame arrival. The pseudocode for computing the active periods is given in Figure 2. For a given value of the peak rate r, the algorithm processes the individual frames of the elementary stream in sequence and computes the maximum queue size $s(r)$, the number of active periods $n_a(r)$, and the starting and ending times of each active period $(s_i^r, t_i^r), 1 \leq i \leq n_a(r)$.

We can use the active periods of the signal $a(r,t)$ to compute the loss curve of the original video stream by observing the queue behavior at the input of the second shaper in Figure 1. We now develop an algorithm for calculating the loss curve for a specific service rate ρ. Since the peak rate r remains a constant in the discussion, for simplicity in the rest of this section we omit the parameter r from all the notations.

2.2 Computation of the Loss Curve

The peak-rate shaping procedure produces a sequence of active periods for a given peak-rate r. If we denote by $m(t)$ the output rate of the peak-rate shaper, then

$$m(t) = \begin{cases} r, & t \in [s_i, t_i]; \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

We define *busy period* as a maximal period of time during which the queue of the second shaper in Figure 1 remains non-empty. We use the notation α_i, β_i, respectively, for the starting and the ending times of busy period i. For a given sequence of active periods, the corresponding busy periods are a function of the transmission rate ρ and the buffer size B at the input of the second shaper. Figure 3(a) illustrates the active periods of an elementary video stream and Figure 3(b) the corresponding busy periods for the case of infinite buffer size.

Pseudocode for Computation of Active Periods of Elementary Video Stream

```
        /* Index i denotes the i-th frame, q_i denotes the queue size at the instant
           just after the i-th frame arrival and index j denotes the j-th active period. */
1       /* Perform initialization */
 1.1    s_1^r ← 1/f; q_1 ← d_1; i ← j ← 1;
2
 2.1    If (q_i/r < 1/f)   /* no backlog present at the start of next frame */
 2.1.1      q_{i+1} ← d_{i+1};
 2.1.2      t_j^r ← i/f + q_i/r;   /* compute end time of current active period */
 2.1.3      j ← j + 1;
 2.1.4      s_j^r ← (i+1)/f;   /* start time of new active period */
 2.2    else
 2.2.1      q_{i+1} ← q_i + d_{i+1} − r/f;  /* there is backlog carried to current frame */
 2.3    endif
3
 3.1    i ← i + 1;
 3.2    If (i < N)
 3.2.1      goto Step 2;
 3.3    endif
4
 4.1    s(r) ← max_{1≤k≤N} q_k;  /* compute maximum queue length observed */
 4.2    t_j^r ← N/f + q_N/r;     /* compute end time of last active period */
 4.3    n_a(r) ← j;              /* store the number of active periods */
 4.4    Define s_{n_a(r)+1} = ∞;
```

Figure 2: Algorithm to compute the active periods of the bit-stream at the output of the peak-rate shaper, when the input traffic is an elementary video stream.

If we assume no losses from the buffer, we denote by $Q^*_{B,i}$ the local maximum queue size for busy period i, for a buffer size of B; and by $\tau_{B,i}$ the time at which the local maximum occurs. For simplicity, we will omit the subscript B when the buffer size is obvious from the context. Our algorithm for characterizing the loss curve of the source is based on two key observations:

i. For a given transmission rate ρ, the loss rate ϵ is a *piecewise-linear* function of the buffer size B. As the buffer size is decreased, the slope of the loss curve can change only when (i) a busy period starts to experience losses for the first time, or (ii) a busy period breaks into two or more constituent busy periods. This enables the exact computation of the loss curve by identifying the points at which such events occur.

ii. Within each busy period where a loss occurs, the last time instant at which a loss occurs is the time instant $\tau_{B,i}$ at which the maximum queue size occurs.

The loss curve of the source, for a given transmission rate ρ, can be constructed by considering

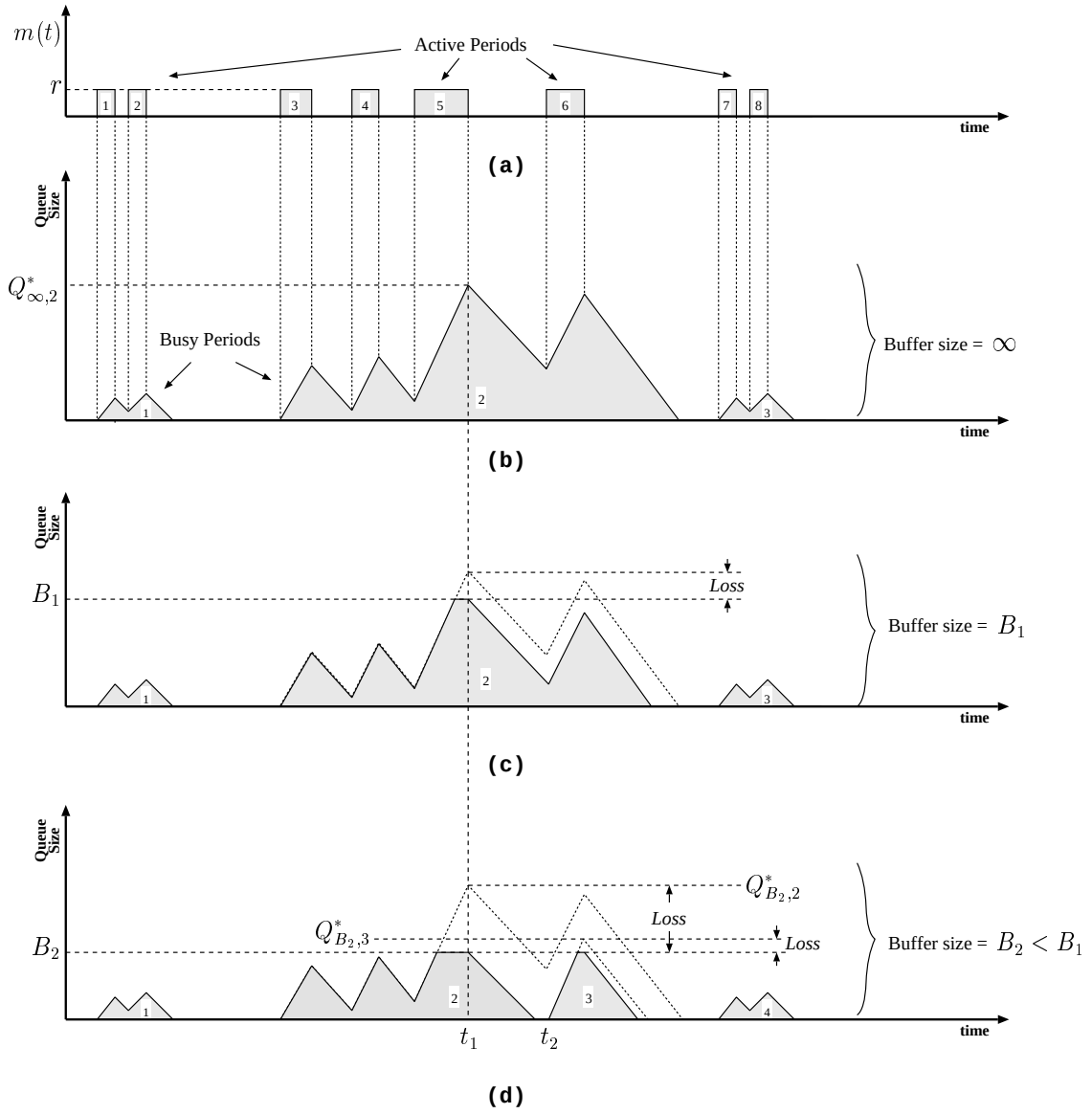

Figure 3: Busy periods of an example elementary video stream for various values of the buffer size B.

various values of the buffer size B in the range $0 < B < \sigma(\rho)$, and computing the loss rate for each case. This procedure, however, provides only an approximation to the actual loss rate between the points considered. Besides, the computation of loss rate requires a simulation of the queue for each buffer size. Our algorithm, on the other hand, exploits the piecewise-linearity of the loss curve and calculates only the slope of the loss curve at each point where the slope changes. In addition, we avoid a total simulation of the queue at each such point by keeping track of only the changes in the busy periods of the queue with a change in buffer size. This results in an efficient algorithm that provides the exact loss curve of the source.

A change in the number of busy periods experiencing losses constitutes the only possible case in which the slope of the loss curve may change as the buffer size B is decreased towards its minimum value

of zero. This can be illustrated with an example. Figure 3(a) shows the active periods of the input signal and Figure 3(b) the busy periods generated when the buffer size is infinite. The global maximum queue size occurs in busy period 2, and is denoted by $Q^*_{\infty,2}$ in the figure. As the buffer size B is decreased, no losses occur until it reaches the global maximum queue size $Q^*_{\infty,2}$. As we decrease the buffer further, to a value $B_1 < Q^*_{\infty,2}$, losses occur in the second busy period. If the peak at time t_1 corresponding to the end of active period 5 is the only peak experiencing losses, the total amount of lost traffic is equal to $(Q^*_{\infty,2} - B)$, which is a linear function of B. We also observe that no losses can occur to the right of t_1 in the same busy period.

On further decreasing the buffer size to $B_2 < B_1$, the second busy period splits into two busy periods at time t_2. Both the resulting busy periods experience losses, and now the total amount of lost data is equal to $(Q^*_{B_2,2} - B) + (Q^*_{B_2,3} - B) = (Q^*_{B_2,2} + Q^*_{B_2,3}) - 2B$, where $Q^*_{B_2,2}$ and $Q^*_{B_2,3}$ are the corresponding local maximum queue sizes of the new busy periods 2 and 3, respectively, as indicated in the figure. Therefore, the amount of lost data is again a linear function of B, and its slope is determined by the number of busy periods experiencing losses.

For a video stream of finite duration, the values of the buffer size B that cause either a loss in a busy period with no prior loss or a break in a busy period that already experiences loss, form a *finite* set. We will show that we only need to compute the values of B belonging to this set for the exact computation of the loss curve. The curve is piecewise linear between adjacent buffer points belonging to this finite set.

Let us denote by $A(t_1, t_2)$ the arrivals into the buffer during the interval $[t_1, t_2]$, and by $L_{B,\rho}(t_1, t_2)$ the number of bits lost in the interval $[t_1, t_2]$ when the buffer size is B and the transmission rate ρ. For simplicity, in the special case when $t_1 = 0$ and $t_2 = t$, we will use the notation $L_{B,\rho}(t)$ instead of $L_{B,\rho}(t_1, t_2)$.

We first show that, for a given busy period experiencing losses, the amount of loss increases linearly with a slope of -1 as the buffer size B is decreased, as long as the busy period does not break into multiple busy periods. Using this result, we can prove that the loss curve is piecewise linear.

Lemma 1 *Let γ_i be the last instant at which a loss occurs within busy period i. Then, the amount of data lost during the busy period is given by*

$$L_{B,\rho}(\alpha_i, \beta_i) = A(\alpha_i, \gamma_i) - \rho(\gamma_i - \alpha_i) - B, \tag{8}$$

where α_i is the starting time of the busy period i.

Proof: The losses from the queue during the interval (α_i, γ_i) must be equal to the arrivals into the queue during the interval minus the total traffic transmitted during the interval, minus the bits remaining in the buffer at the end of the interval. Since the queue does not underflow during the interval (α_i, γ_i), the total traffic transmitted during the interval (α_i, γ_i) is $\rho(\gamma_i - \alpha_i)$. Furthermore, since losses occur at time γ_i, the buffer occupancy at time γ_i is B. Subtracting these two terms from the arrivals gives us the result in Eq. (8). □

Thus, to calculate the losses during the busy period, it is sufficient to determine its starting time and the last instant γ_i at which losses occur during the busy period. Note that γ_i must coincide with the end of an active period of the source. Later, we will show that γ_i coincides with the time instant at which the local maximum queue size would have occurred during the busy period if the buffer size were infinite. We can now use Lemma 1 to show that the loss curve is piecewise linear.

Lemma 2 *For a given transmission rate ρ, the loss curve of an elementary video stream is piecewise linear. The slope of the curve changes only at values of the buffer size B where one of the following events occurs:*

i. A change in the number of busy periods in which losses occur.

ii. A change in γ_i, the last instant at which a loss occurs in a busy period i, for any busy period i.

Proof: Consider two distinct values of B, B_1 and B_2, with $B_1 < B_2$, such that (i) the number of busy periods undergoing losses remains the same at B_1 and B_2; and (ii) the last time at which a loss occurs in each of these busy periods, γ_i, also remains the same. Let S_l denote the set of busy periods in which losses occur. Then, according to Lemma 1 the total amount of lost data over the entire duration T of the video stream is given by

$$L_{B,\rho}(T) = \sum_{i \in S_l} (A(\alpha_i, \gamma_i) - \rho(\gamma_i - \alpha_i)) - n_l B, \quad B_1 \leq B \leq B_2, \tag{9}$$

where n_l is the number of busy periods in the set S_l.

Thus, the plot of $L_{B,\rho}(T)$ and therefore that of $\epsilon(B, \rho)$ with respect to B in the range $B_1 \leq B \leq B_2$ is a straight line with slope $-n_l$. This concludes the proof of Lemma 2. □

We can obtain the entire loss curve of the elementary video stream for a given transmission rate ρ by starting from a buffer size equal to the corresponding burstiness value $\sigma(\rho)$ (which is equal to the global maximum queue size when the buffer size is infinity) and progressively finding buffer sizes at which either (i) a busy period with no prior loss starts to experience losses, or (ii) a busy period experiencing loss breaks into smaller busy periods. The time instant at which the last loss occurs within a given busy period is the time at which the queue size reaches its local maximum within the busy period, when no losses occur from the buffer. This makes it easy to determine the parameter γ_i in Eq. (9) that is required for computation of the loss rate. The interested reader is referred to [11] for a formal proof of the latter statement.

The buffer size at which the first loss occurs for a busy period can easily be identified by computing the local maximum queue size within the busy period ignoring any losses. The buffer size at which a break occurs in a busy period, however, is more difficult to identify. We discuss this problem next.

Since the queue size reaches a minimum at the start of an active period, the starting instants of active periods within the busy period are the points at which a break could potentially occur. Thus, we can determine the maximum buffer size at which a break occurs in the busy period by computing the buffer size that causes the queue size to be zero at each of these points and taking the maximum among all the points. This procedure is cumbersome, however, because the effect of losses must be accumulated over multiple active periods to determine the buffer size that causes the queue size to reach zero exactly at the start of a given active period. Instead, we use a more efficient scheme to identify the buffer size that causes a break in the busy period.

Our approach can be best illustrated by the example in Figure 4, where a single busy period is shown, consisting of seven active periods. The peaks and valleys for the queue size correspond to the ending and starting times, respectively, of the active periods. With no losses in the busy period, the maximum queue size within the busy period occurs at time t_6, at the end of active period 6. As the buffer size is decreased from this value, losses start to occur first during active period 6. This causes a corresponding dip in the valleys following the peak at t_6, and a break results in the busy period if the queue size drops below zero at any of the valleys.

As the buffer size is decreased, losses start to occur progressively from the highest peak, to the next highest, and so on. In addition, if a loss occurs from one of the peaks within the busy period, no losses

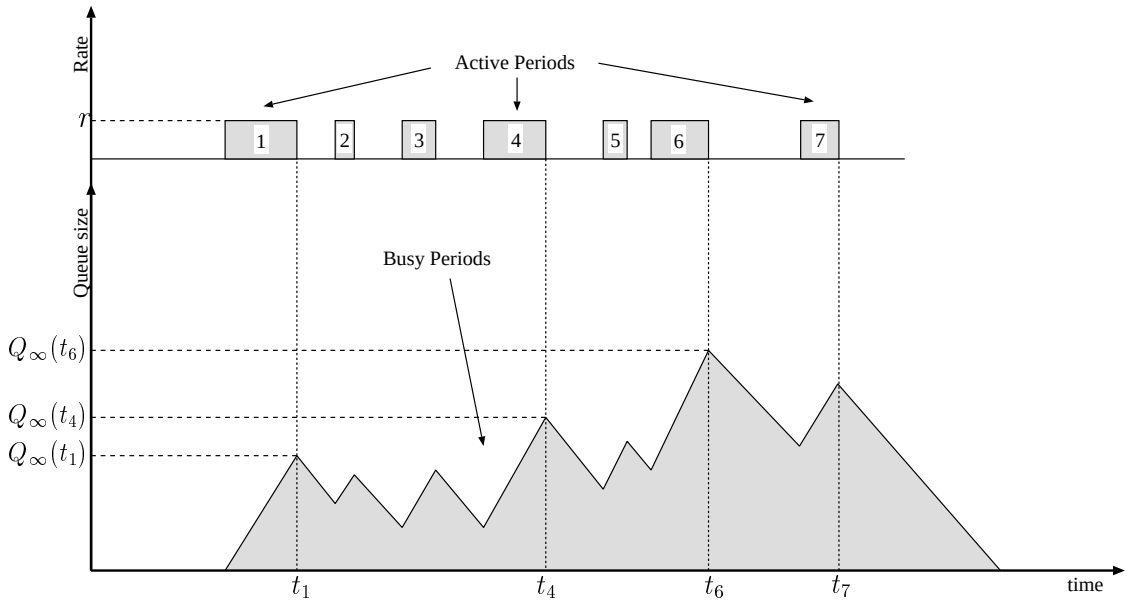

Figure 4: Example of how to construct the set $\mathcal{S}$ that contains the active periods with increasing queue sizes, and the last active period of a given busy period.

can occur from a following peak unless the latter is larger than the former, or the busy period breaks. For example, in Figure 4 no losses can occur between the peaks at t_1 and t_4 without causing a break in the busy period first. This enables us to process only the active periods corresponding to monotonically increasing queue sizes within the busy period for identifying the buffer size at which a break occurs.

The algorithm for identifying the buffer size that causes a break in the busy period works as follows: We first construct a sequence of active periods $\mathcal{S}$ within the busy period with monotonically increasing queue sizes. In addition, we also require the last active period to be part of the sequence. For example, in Figure 4 such a sequence consists of the active periods 1, 4, 6 and 7.

We can now look for potential points for a break in the busy period, starting from the end of the sequence. For each active period in the sequence, starting from the end, we can identify the buffer size at which the queue size becomes zero within the interval between the current active period and the previous active period of the sequence. For example, in Figure 4, the algorithm first calculates the buffer size for the queue size to reach zero at the valley between t_6 and t_7 (this is the difference between the queue sizes at t_6 and at the valley following it). If this buffer size is larger than the next peak in the sequence (the peak at t_4 in the example), a break will occur between t_6 and t_7 before any losses occur at time t_4, as the buffer size is reduced. No further processing of the sequence is then needed. If the peak at time t_4 is smaller than the buffer size identified so far, however, the algorithm calculates the buffer size that causes a break between t_4 and t_6. This continues until a peak that is larger than the currently identified buffer size is reached, or the sequence of active periods in $\mathcal{S}$ is exhausted. The algorithm then chooses the buffer size identified last as the point at which the break occurs.

A high-level pseudocode of the entire algorithm for computation of the loss curve is shown in Figure 5. The algorithm starts by computing the busy periods when the buffer size B is infinite. This is done by function *process_active_periods()* which processes the active periods 1 to n_a to determine the busy periods.

After computing the busy periods, the algorithm inserts into the heap all the buffer points at which each busy period starts experiencing loss, and the cause associated with each buffer point (LOSS). These buffer points correspond to the maximum queue size for each busy period. The algorithm also computes the maximum buffer sizes at which each busy period breaks, and inserts them into the heap for further processing.

In the main iteration (Step 2), the algorithm proceeds to process the maximum buffer point extracted from the heap. Each buffer point is associated with a busy period b and a cause. The cause takes two values: LOSS, if the buffer point causes a loss in the busy period b, and BREAK, if it results into a break. In case that the cause is a loss in the busy period, the algorithm updates the loss variables (L_B and n_l variables) that are used in the computation of the total loss for the current value of the buffer size. The total amount of loss is given by $(L_B - n_l B)$. In the case of a break cause, the algorithm updates the loss variables again and then processes the break by calling the function *process_break()*. First, the function computes the new busy periods generated after the break of busy period b, and checks them for possible losses. For each busy period computed that experiences loss, the function updates the loss variables. For the remaining new busy periods, it inserts into the heap the maximum buffer sizes at which loss starts occurring in the busy periods. Finally, it computes the buffer sizes for the first break of all the generated busy periods, and inserts them into the heap. To avoid multiple output points with the same buffer size value, the algorithm outputs a (B, ϵ) pair only when a new buffer size is processed by calling the function *output_point()*.

The procedure described above repeats for the new maximum buffer point extracted from the heap and the associated busy period. The procedure ends when no buffer point is left. The loss curve is obtained by connecting the loss ratio (ϵ) values at all such buffer points by linear segments.

The worst-case time and space complexities of the algorithm can be determined by considering the computations performed at each buffer point. The number of buffer points is $O(n_a)$ since a buffer point corresponds to either a break or a loss occurring in a busy period. It takes $O(n_a)$ steps to process a busy period for a specific buffer size B, where n_a is the number of active periods. This includes the time to compute any new busy periods and the buffer size values at which either a loss or a break occurs in each of these busy periods. Therefore, the total worst-case time complexity for the execution of the algorithm is $O(n_a^2)$. The space needed to store the output of the algorithm is proportional to the number of buffer points processed by the algorithm. Thus, the space complexity is $O(n_a)$.

The loss curve can be used for the computation of the B versus ρ curve, i.e., the plot of the minimum buffer size B versus the service rate ρ for a specific loss rate ϵ. To compute the B versus ρ curve for a specific value of ϵ, we need to compute the loss curve for every ρ, and then read off the values of B for the specified value of ϵ. This procedure can be done using some granularity of the rate ρ, or using the rate points of the burstiness curve.

3 Application of the Algorithm to MPEG-2 Transport Streams

The algorithm presented in the last section can be modified for computation of the loss curve of MPEG-2 Transport Streams. In this section, we present how the algorithm can be applied to MPEG-2 Transport Streams.

The MPEG-2 Transport Stream format is a grouping of one or more programs into a single stream, with a *program* defined as a grouping of elementary streams (audio, video, teletext, etc.) that have a common time-base for delivery. The MPEG-2 Transport Stream is the preferred choice for error-prone environments

as in the case of packet-switched networks. A basic characteristic of this format is that the rate throughout the stream is piecewise constant [7]. Therefore, an MPEG-2 Transport Stream consists of a sequence of constant-rate segments (see Figure 6). Detailed description of the MPEG-2 Systems Layer can be found in [7, 9].

Using the same definition for the busy period, a busy period in the MPEG-2 Transport Stream case always starts at the beginning of a rate segment i at which $r_i > \rho$, with r_i being the rate of the segment. An example busy period is shown in Figure 7 which consists of rate segments 2–12.

As in the case of elementary video streams, the amount of data that is lost in a busy period is again given by Eq. (8), where $m(t)$ corresponds to the transport rate of the MPEG-2 Transport Stream. Similarly, the total losses throughout the whole stream is again given by Eq. (9). Therefore, to compute the exact loss curve in this case, we need to consider only the buffer points at which either (i) a change in the index of the last rate segment experiencing loss in a busy period occurs, or (ii) a change in the number of busy periods experiencing loss occurs. A difference between the cases of the elementary video stream and the MPEG-2 Transport Stream is in the computation of the next buffer size at which a break occurs for a certain busy period. The set of rate segments with increasing peaks of the queue size (occurring at the ending time instants of the rate segments) is again computed first. The time instants within the busy period that need to be checked for a possible break are only the ones corresponding to the valleys inside the busy period and not all the time instants corresponding to the end of a decreasing line segment of the busy period. As an example, the rate segments 3, 5 and 8 of Figure 7, are inserted into the set of increasing queue peaks, and it is sufficient to examine the valley points at the end of rate segments 4, 7 and 10 for determination of the buffer size that will result in the first break of the busy period.

4 Validation and Results

To evaluate the time- and space-complexities of the algorithm, in this section we show results from applying the algorithm to several actual video traces available over the Internet. For simplicity, we focus on elementary video streams. The performance on MPEG-2 Transport Streams can be expected to be similar.

We applied the algorithm to a number of elementary video stream traces taken from [1, 6]. All the traces except Garrett's are approximately 30 minutes in duration, while Garrett's trace is approximately 2 hours long. The execution times of the algorithm for the various traces are shown in Table 1. The execution time is relatively small for all the traces, typically within a few seconds on a Sun Ultra-2 workstation. The number of points on the loss curve varied from 9,632 for "lambs" to 89,489 for Garrett's trace, comparable to the total number of frames in the respective traces. An example loss curve and typical B versus ρ curves for the "lambs" trace are shown in Figures 8 and 9, respectively.

The algorithm for the computation of the loss curve along with the algorithms for the computation of the burstiness curve presented in [10] have been implemented in a JAVA tool that can be used for the derivation of the characteristic curves of video traces. The tool provides the ability to analyze video traces and view the resulting curves graphically. It can be downloaded from
`http://www.cse.ucsc.edu/~tryfonas/mpeg-neat.htm`.

Trace	**Frame Rate** (Hz)	**Number of Frames**	**Video Length** (mins)	**Running Time** (secs)	**Space** (# points)
MrBean	25	40000	26.7	2.3	18812
asterix	25	40000	26.7	2.8	24345
atp	25	40000	26.7	2.5	22641
bond	25	40000	26.7	2.9	27169
dino	25	40000	26.7	1.9	14293
lambs	25	40000	26.7	1.6	9632
movie2	25	40000	26.7	2.1	17634
mtv1	25	40000	26.7	3.3	28262
mtv2	25	40000	26.7	5.2	22525
news1	25	31515	21.0	2.1	20036
news2	25	40000	26.7	2.1	16888
race	25	40000	26.7	4.2	33317
sbowl	25	40000	26.7	2.5	24044
simpsons	25	40000	26.7	2.3	20527
soccer1	25	40000	26.7	3.2	30265
soccer2	25	40000	26.7	3.1	27129
star	25	40000	26.7	1.8	12189
talk1	25	40000	26.7	1.9	14110
talk2	25	40000	26.7	2.0	15683
terminator	25	40000	26.7	1.7	11396
Garrett's trace [6]	24	174136	120.9	11.2	89489

Table 1: Performance results of the exact algorithm on several elementary video streams taken from [1]. The peak rate is set to 10 Mbps whereas the service rate ρ of the second server is set to 1 Mbps. The running time is the user time as measured on a Sun Ultra-2 workstation.

5 Conclusions

In this paper, we developed efficient deterministic algorithms for exact computation of the loss curve of both elementary video streams and MPEG-2 Transport Streams. The algorithms enable the exact computation of the loss curve of a video stream or any bursty ON-OFF source such as voice or data. In addition, they facilitate the computation of the B versus ρ curve for a fixed loss rate ϵ of a video stream.

Our experiments with several video traces suggest that the proposed algorithms can be used not only in off-line environments in which the video stream is stored (e.g. video servers), but also in on-line systems such as in real-time TV broadcasting. In the latter case, the video stream can be segmented to fixed time intervals and the loss curve can be constructed for each segment, facilitating per-segment QoS provisioning and call admission control.

References

[1] Mpeg traces. ftp://ftp-info3.informatik.uni-wuerzburg.de/pub/MPEG.

[2] D. D. Botvich and N. G. Duffield. Large deviations, the shape of the loss curve, and economies of scale in large multiplexers. *Queueing Systems, Theory and Applications*, 20(3–4):293–320, 1995.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. McGraw Hill, 1994.

[4] R. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, January 1991.

[5] R. Cruz. A calculus for network delay, part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, January 1991.

[6] M. W. Garrett. *Contributions toward real-time services on packet-switched networks*. PhD thesis, Columbia University, May 1993.

[7] International Organization for Standardization. *Information Technology — Generic Coding of Moving Pictures and Associated Audio: Systems, Recommendation H.222.0, ISO/IEC 13818-1*, draft international standard edition, November 1994.

[8] S. Low and P. Varaiya. A simple theory of traffic and resource allocation in ATM. In *Proceedings of GLOBECOM '91*, volume 3, pages 1633–1637, December 1991.

[9] C. Tryfonas. MPEG-2 transport over ATM networks. Master's thesis, University of California at Santa Cruz, September 1996. Available at `http://www.cse.ucsc.edu/research/hsnlab`.

[10] C. Tryfonas, A. Varma, and S. Varma. Efficient algorithms for computation of the burstiness curve of video sources. to appear in Proceedings of International Teletraffic Congress (ITC) '99, June 1999.

[11] C. Tryfonas, A. Varma, and S. Varma. Efficient algorithms for exact computation of the loss curve of video sources. Technical Report UCSC-CRL-99-2, University of California at Santa Cruz, Dept. of Computer Engineering, January 1999.

[12] M. K. Wong and P. P. Varayia. A deterministic fluid model for cell loss in ATM networks. In *Proceedings of IEEE INFOCOM '93*, volume 2, pages 395–400, San Francisco, USA, March–April 1993.

Loss Curve Algorithm Pseudocode for Elementary Video Streams

1 /* INIT. B holds the available buffer, L_B holds the sum of the unrestricted queue sizes of the busy periods with loss at the time instants at which the maximum queue occurs, n_l holds the number of busy periods with loss, n_b the number of busy periods, and n_a the number of active periods.*/

1.1 $B \leftarrow \infty$; $L_B \leftarrow 0$; $n_l \leftarrow 0$; $M \leftarrow$ stream_size;

/* Compute busy periods for rate ρ, and buffer size B. */

1.2 **process_active_periods**(1, n_a, B);

/* Insert into the heap the buffer point at which a loss occurs for each busy period, compute the buffer size for the first break for each busy period and insert it into the heap as well. */

1.3 **For** $b = 1$ **to** n_b /* for each busy period b */

1.3.1 **heap_insert**(Q_b^*, LOSS, b);

1.3.2 $B' \leftarrow$ **compute_next_break**(b);

1.3.3 **heap_insert**(B', BREAK, b);

1.4 **endfor**

2 /* Extract the maximum buffer from the heap, and process the corresponding busy period until the heap becomes empty. */

2.1 (b, B, cause) $\leftarrow$ **heap_extract_max**();

/* Let $(s_p, t_p), (s_{p+1}, t_{p+1}), \ldots, (s_q, t_q)$ be the active periods contained within the busy period b. Also, let $(s_p, t_p), \ldots, (s_j, t_j)$ be the active periods contained within the interval (α_b, τ_b). */

2.2 **If** (cause = LOSS)

/* update n_l and L_B variables */

2.2.1 $L_B \leftarrow L_B + \left(\sum_{i=p}^{j} r(t_i - s_i) - \rho(t_j - s_p)\right)$;

2.2.2 $n_l \leftarrow n_l + 1$;

2.3 **else** /* cause = BREAK */

/* update n_l and L_B variables */

2.3.1 $L_B \leftarrow L_B - \left(\sum_{i=p}^{j} r(t_i - s_i) - \rho(t_j - s_p)\right)$;

2.3.2 $n_l \leftarrow n_l - 1$;

/* Process the break in busy period b: Compute the new busy periods, and update n_l and L_B when a busy period already experiences loss. For the remaining new busy periods, insert into the heap the buffer points at which the first losses occur. Finally, for all the new busy periods, compute the buffer sizes for the first break and insert them into the heap. */

2.3.3 **process_break**(b, B);

2.4 **endif**

2.5 **output_point**(B, $\frac{L_B - n_l B}{M}$);

2.6 **If** (heap not empty)

2.6.1 **goto** Step 2;

2.7 **endif**

/* Output the last point of the loss curve */

2.8 **Output** $(0, \frac{r-\rho}{r})$;

2.9 **STOP**;

Figure 5: Top-level of the algorithm that computes the exact loss curve of an elementary video stream. A representative description of the function *heap_extract_max()* can be found in [3].

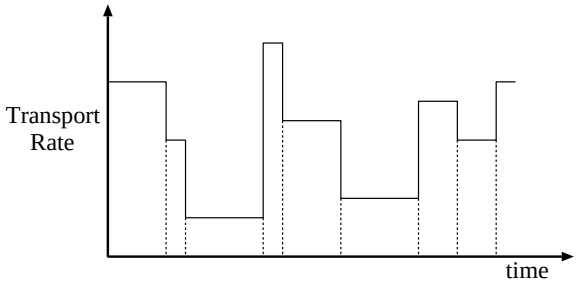

Figure 6: The behavior of Transport Rate in an MPEG-2 Transport Stream.

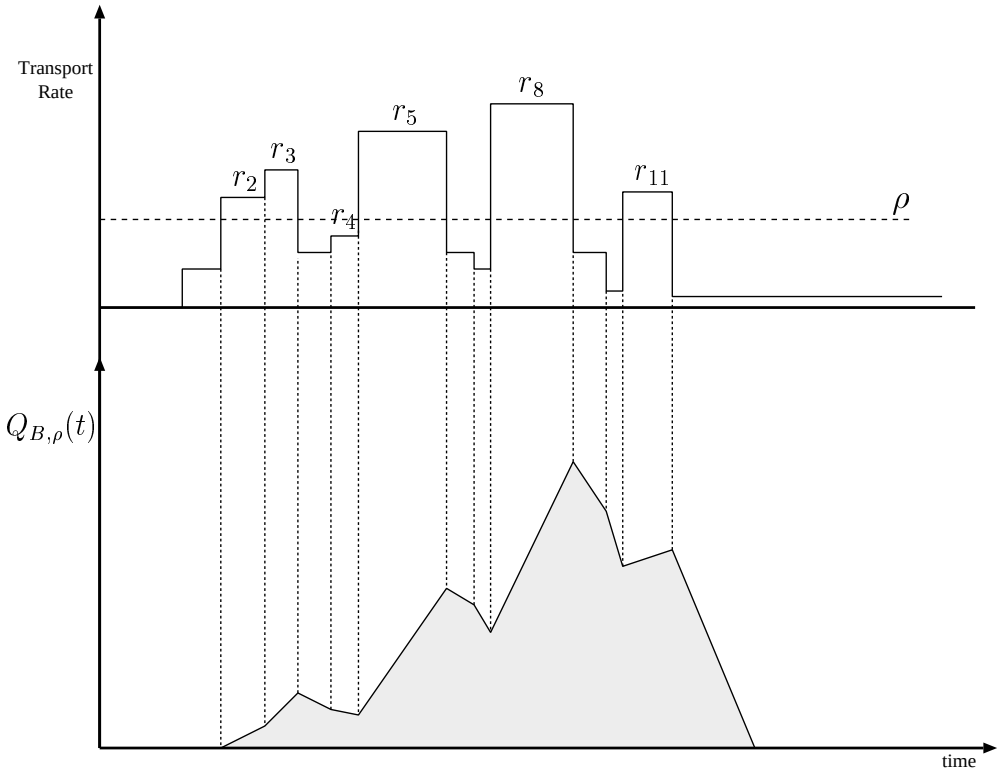

Figure 7: Busy period of an example MPEG-2 Transport Stream.

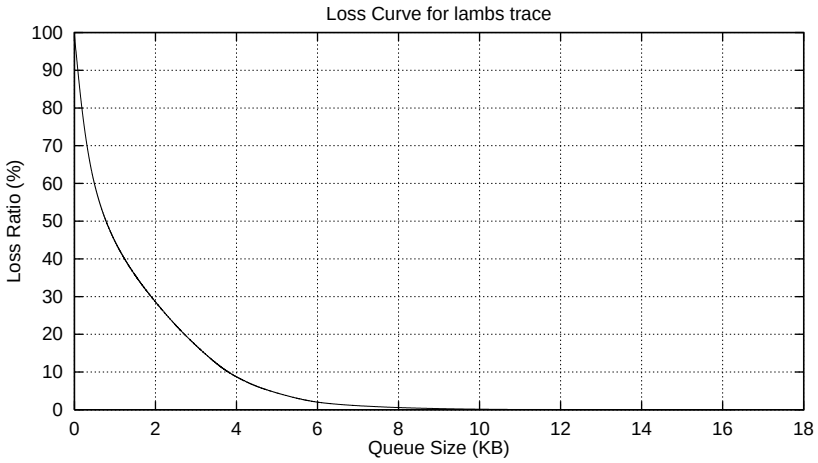

Figure 8: Example loss curve for lambs trace as computed by the algorithm.

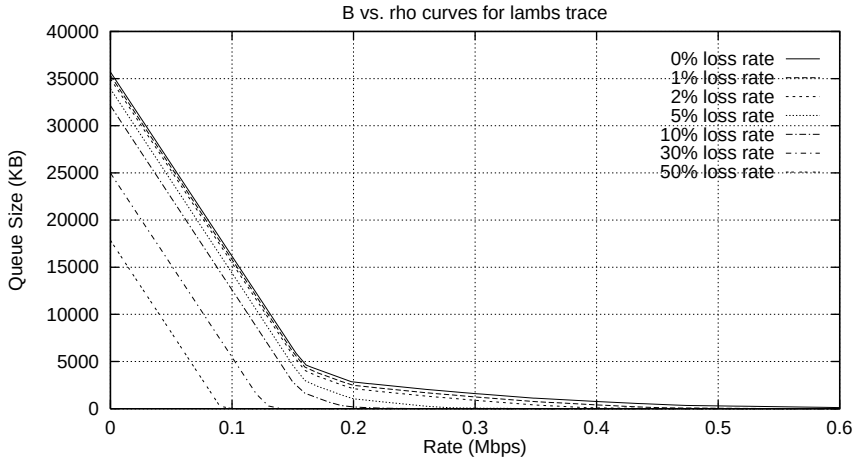

Figure 9: Example B versus ρ curves for lambs trace calculated using the algorithm for the computation of the loss curve.