Efficient Algorithms for Computation of the Burstiness Curve of Video Sources

Christos Tryfonas[†]
Anujan Varma[†]
Subir Varma[‡]

UCSC-CRL-98-13
December 10, 1998

[†] Computer Engineering Department
University of California, Santa Cruz
Santa Cruz, CA 95064

[‡] Hybrid Networks Inc.
6409 Guadalupe Mines Road
San Jose, CA 95120

ABSTRACT

The burstiness of a video source can be characterized by its burstiness curve. The burstiness curve is useful in the optimal allocation of resources to satisfy a desired quality of service for the video stream in a packet network. In this paper, we present deterministic algorithms for exact computation of the burstiness curve of a video source, for both elementary video streams and MPEG-2 Transport Streams. The algorithms exploit the piecewise linearity of the burstiness curve and compute only the points at which the slope of the burstiness curve changes. We also present approximate versions of these algorithms, which save computational effort by considering only a small number of candidate points at which the slope of the burstiness curve may change. The approximate algorithm was able to compute the burstiness curve of a two-hour long elementary video stream in approximately 10 seconds, as compared to over 6 hours for the exact algorithm, with virtually no loss of accuracy in the computation. The efficiency of the proposed algorithms makes them suitable for QoS provisioning not only in off-line environments such as in Video-on-Demand (VoD) servers, but also in real-time applications such as live TV distribution systems.

Keywords: Video, Burstiness Curve, Quality-of-Service, MPEG-2, MPEG-2 Transport Stream.

This research is supported by the NSF Young Investigator Award No. MIP-9257103 and by DARPA under Contract No. F19628-96-C-0038.

1 Introduction

The explosion of the Internet has created demand for new applications traditionally carried over circuit-switched networks. Such applications include audio telephony, video conferencing and video-on-demand (VoD) services. New standards are emerging to support these applications in the context of both connectionless and connection-oriented packet-switched networks.

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.

Significant work has been done in the literature to characterize video sources, so that they can be effectively transported over packet-switched networks. One way to characterize a traffic source is through a *time-invariant traffic constraint* function. This function bounds the maximum number of bits that may be generated by the source over any possible time interval. The least upper bound of this function is also referred to as the *minimum envelope process* [2] or *empirical envelope* [11]. Although the use of the minimum envelope process provides very accurate traffic characterization for a source, its practical significance is diminished by the fact that such a function can only be effectively policed by a large number of leaky buckets [5]. Since current packet-switched networks employ simple leaky-bucket mechanisms for traffic policing, the use of the minimum envelope process does not facilitate traffic policing.

Another method of characterizing a traffic source is by means of its *burstiness curve*, as defined by Low and Varayia [12]. Each (σ, ρ) point in the burstiness curve corresponds to the maximum queue size σ encountered (or the amount of buffering needed), when the traffic source is fed into a server with deterministic service rate ρ. Consequently, if the traffic source is sent to a leaky bucket with parameters (σ, ρ), none of its packets will be tagged as non-conformant. The simplicity of the burstiness curve approach for traffic source characterization makes it very attractive for policing in current packet-switched networks.

The burstiness curve can be obtained through simulation based techniques. However, simulation can be very time consuming, especially when fine rate granularity is needed. In this paper, we present efficient algorithms for *exact* computation of the burstiness curve for both elementary video streams and MPEG-2 Transport Streams. The algorithms exploit the piecewise linearity of the burstiness curves of both elementary and Transport Streams and identify the minimum number of points that need to be computed for an exact computation of the burstiness curve. Therefore, the proposed algorithms are optimal in the number of points needed for an exact computation of the burstiness curve. We also present approximate versions of the algorithms that reduce the computational effort by considering only a smaller number of candidate points. The algorithms exhibit low time- and space-complexity compared to traditional simulation-based approaches. The high efficiency of the proposed algorithms makes them attractive not only to video servers that need to compute the burstiness curve of their video traces and store it as metadata with the trace for QoS control, but also to real-time video distribution systems that need to estimate the burstiness curve of their video programs in real-time.

The rest of this paper is organized as follows: In Section 2 we present the motivation of this work along with examples illustrating the use of the burstiness curve for the selection of traffic parameters under different QoS constraints. In Section 3, we present the algorithm for exact computation of the burstiness curve of an elementary video stream. In Section 4, we extend the algorithm to

MPEG-2 Transport Streams. In Section 5, we show approximate algorithms for the computation of the burstiness curve with much lower running time. In Section 6, we compare the performance of these algorithms on three example video traces. Finally, in Section 7 we conclude the paper with a summary of this work.

2 Motivation

The burstiness of a traffic source is usually characterized by means of a *token-bucket* mechanism. The input of the token-bucket is the traffic generated by the source whereas the output is the corresponding token-bucket constrained traffic. The output of the token bucket is sometimes referred to as a (σ, ρ, r) conformant traffic stream, where σ is the number of tokens in the token bucket, ρ the rate of the incoming tokens, and r the peak rate of the server. If $A(t_1, t_2)$ is the amount of traffic that leaves the token bucket during an interval (t_1, t_2), then the following constraint holds:

$$A(t_1, t_2) \leq \min\{r(t_2 - t_1), \sigma + \rho(t_2 - t_1)\}. \tag{2.1}$$

A traffic source is said to be (σ, ρ, r) conformant if its traffic can go through a token-bucket shaper with bucket size σ and rate ρ, at peak rate r with the queue size never exceeding σ. In the context of ATM networks, (σ, ρ, r) enforcement is done using the *Generalized Cell-Rate Algorithm (GCRA)* [4].

Specifying an appropriate (σ, ρ, r) tuple for the traffic source is critical for estimating the amount of resources (bandwidth, buffer space, etc.) necessary in the network to provide the desired level of service to the traffic stream. However, there is no single tuple that uniquely characterizes a source [8, 12]. For a given peak rate r, it is evident from Eq. (2.1) that, for any value of ρ, there is a corresponding value of σ such that the source is conformant. Hence, the set of conformant (σ, ρ) pairs describe a curve which is referred to as the *burstiness curve* [12]. To completely characterize a source, we need to plot a set of burstiness curves for different values of the peak rate r.

The burstiness curve of a video source is useful in determining the level of resources necessary to achieve a desired QoS level. Both the delay and packet-loss rate in the network are functions of σ and ρ. Thus, knowledge of the burstiness curve of the source enables the admission control process to allocate the minimum amount of resources to achieve a desired QoS level. In the following, we discuss how the delay calculation in the network depends on the burstiness curve of the source.

2.1 Effect of Burstiness on End-to-End Delay

The end-to-end delay of a traffic source can be guaranteed when a packet-switched network utilizes scheduling algorithms able to guarantee a strict upper bound on the delay of a session. In order for a scheduling discipline to guarantee a worst-case delay bound to a session, the burstiness of the source traffic must be bounded. Token-bucket constrained traffic model is used extensively in the literature for the worst-case analysis of delay. Different frameworks have been developed to formalize the characterization of schedulers and obtain the worst-case delay bounds [7, 14]. For the purpose of this section, we consider the LR model [14]. The framework provides a general model for computation of the worst-case delay bound of several schedulers. Schedulers that fall into this classification are called *Latency-Rate* (LR) servers.

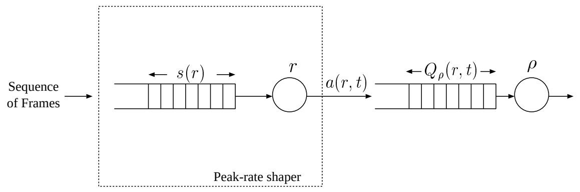

Figure 2.1: Model used for the computation of the burstiness curve of a video stream.

We assume that the bandwidth allocated by the network to the session is equal to the average rate of the source ρ. According to the LR model, the worst-case delay bound d_M of a tandem network consisting of K schedulers belonging to the LR class is given by

$$d_M = \frac{\sigma}{\rho} + \sum_{i=1}^{K} \Theta_i, \tag{2.2}$$

where σ, ρ are the parameters of the source, and Θ_i is a parameter of the scheduler, called *latency.* Solving for σ, we obtain the following constraint that σ and ρ need to satisfy to achieve a specified worst-case delay bound:

$$\sigma = \left(d_M - \sum_{i=1}^{K} \Theta_i \right) \rho. \tag{2.3}$$

Based on this result, the optimum $(\sigma,\ \rho)$ pair can be obtained as the intersection between the line defined by Eq. (2.3) and the burstiness curve.

3 Algorithm for Exact Computation of the Burstiness Curve of Elementary Video Streams

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.

The model that we use for the computation of the burstiness curve in the case of an elementary video stream is shown in Figure 2.1. The traffic source goes through a peak-rate shaper which produces a new time sequence for the bit-stream of the input traffic source. 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 burstiness curve of the traffic source at the output of the peak-rate shaper for a specific value of the peak rate r.

By feeding the traffic source $a(r,t)$ at the output of the peak-rate shaper into a server with deterministic rate ρ in the range $[0,r]$, we can observe the dynamics of its queue size $Q_\rho(r,t)$, and record its maximum queue length denoted by $\sigma(r,\rho)$ which corresponds to the burstiness of the traffic source for rate ρ. The strategy that we will pursue to obtain the burstiness curve is to analytically derive the maximum queue length $\sigma(r,\rho) = \max_t(Q_\rho(r,t))$ for different values of rate ρ. We will show later that we do not need to obtain points for all the possible values of rate ρ for an exact computation of the curve, thus minimizing the time- and space-complexity of the algorithm. The exact curve can be obtained by connecting only the necessary points with linear segments.

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 burstiness 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 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. To capture the dynamics of the bit-stream $a(r,t)$ generated at the output of the peak-rate shaper for a specific peak rate r, we need to compute its 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 i-th 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 immediately added to the shaper queue upon its generation. When the peak rate r satisfies $r \geq d_{max}/f$, it is trivial to compute the active periods of the signal $a(r,t)$.

$$s(r) = d_{max}, \qquad n_a(r) = N, \qquad s_i^r = \frac{i}{f}, \qquad \text{and } t_i^r = s_i^r + \frac{d_i}{r}. \tag{3.1}$$

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 just after the instant when the i-th frame arrives. The maximum queue length will always occur just after an arrival of a frame, and is given by

$$s(r) = \max_{1 \leq i \leq N} q_i(r). \tag{3.2}$$

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 Appendix A. 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 burstiness of the original video stream by observing the queue behavior at the input of the second shaper in Figure 2.1. We now develop an algorithm for calculating the maximum queue size at the input of the second shaper for a given peak rate r and service rate ρ. Since the peak rate r does not change in this algorithm, for simplicity we omit the parameter r from all the notations in this section.

3.1 Computation of the Burstiness 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

$$m(t) = \begin{cases} r, & t \in [s_i, t_i]; \\ 0, & \text{otherwise.} \end{cases} \tag{3.3}$$

Let $I_\rho(t)$ be an indicator function that is defined as follows:

$$I_\rho(t) = \begin{cases} 1, & \text{if } Q_\rho(t) > 0; \\ 0, & \text{otherwise;} \end{cases} \tag{3.4}$$

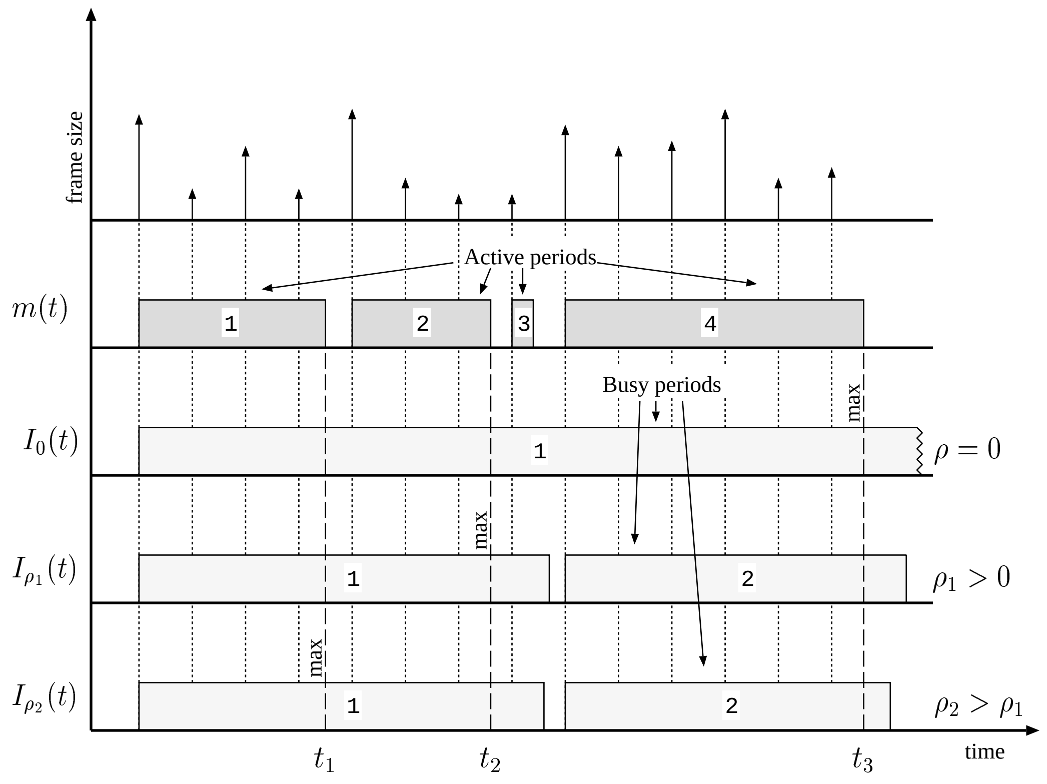

Figure 3.1: Busy periods of an example elementary video stream for various values of the rate ρ.

where $Q_\rho(t)$ denotes the queue size at time t of the server at the output of the peak-rate shaper for rate ρ.

We define *busy period* as a maximal period of time during which the queue of the server at the output of the peak-rate shaper is non-empty, that is, an on-period of the indicator function $I_\rho(t)$. We use the notation α_i, β_i for the starting and the ending times of busy period i. Let $Q^*(\rho)$ denote the maximum queue size reached at this queue over the entire duration of the video stream. Our algorithm for estimating $Q^*(\rho)$ is based on the observation that if the maximum queue size occurs in some busy period i when the rate is ρ, then $Q^*(\rho)$ will vary linearly with the rate ρ until one of the following events occur:

1. The maximum queue size moves to a different instant within the same busy period.
2. The maximum queue size moves to a different busy period.
3. The number of busy periods changes.

These constitute all the possible cases in which the slope of the burstiness curve may change as the rate ρ is increased towards its maximum value r. This can be illustrated with an example.

Figure 3.1 shows the active periods generated when a sequence of frames is shaped by the peak-rate shaper with rate r, and the resulting busy periods at the input of the server operating at rate ρ. When the rate ρ of the server is zero, the queue of the server accumulates all the data arrived during the active periods in one large busy period. Therefore, the maximum queue size occurs at the end of the last active period, i.e., when all data has arrived. As the rate of the server increases to ρ_1, the busy period shrinks until it splits into two busy periods. Let us assume that the maximum queue size occurs at time t_2 for rate ρ_1 which is the end of active period 2. Note that the maximum queue size always occurs at the end of an active period. On increasing the rate further, the busy periods of $I_{\rho_1}(t)$ shrink without breaking and the maximum stays at t_2, until rate ρ_2 is reached. At rate ρ_2, the maximum moves to time instant t_1 while the total number of busy periods stays unchanged.

For a video stream of finite duration, the values of the rate ρ that cause either a break in a busy period or the maximum queue size to be moved to a different location form a *finite* set. We will show that we only need to compute the rates belonging to this set for the exact computation of the burstiness curve. The burstiness curve is piecewise linear between adjacent rate points belonging to this finite set.

When the video stream is sent to a server with zero rate, the resulting maximum queue length $\sigma(0)$ is the total amount of data generated by the video stream. Similarly, when the service rate ρ becomes equal to the peak rate r, $\sigma(\rho) = 0$. For any intermediate rate ρ, $0 < \rho < r$, we need to identify all the busy periods that are generated at the server with service rate ρ, and calculate the local maximum queue lengths for each busy period. The maximum among them corresponds to the global maximum queue length $\sigma(\rho)$.

Fortunately, we do not need to resort to this procedure for all possible rates in the interval $(0, r)$. Instead, we need to perform a fresh calculation of the maximum queue size only at rate points where (i) the instant at which the maximum queue size occurs moves, or (ii) one of the busy periods breaks into two or more smaller busy periods. We will now show that the burstiness curve is linear between such points.

Lemma 1: *The burstiness curve of an elementary video stream is piecewise linear. The slope of the burstiness curve changes only at rate points where one of the following events occurs:*

1. *A change in the time instant at which the maximum queue size $\sigma(\rho)$, or*
2. *a change in the number of busy periods.*

Proof: Consider two distinct rates ρ_1 and ρ_2, with $\rho_1 < \rho_2$, such that (i) the number of busy periods remains the same at ρ_1 and ρ_2; and (ii) the global maximum queue size occurs at the same instant τ_k for ρ_1 and ρ_2.

Assume that τ_k belongs to busy period k. Let α_k be the starting time of this busy period. Then, for any rate ρ in the range $\rho_1 \leq \rho \leq \rho_2$, the queue size at time τ_k is given by

$$Q_\rho(\tau_k) = W(\alpha_k, \tau_k) - \rho(\tau_k - \alpha_k),$$

where $W(\alpha_k, \tau_k)$ is the amount of traffic arriving into the queue during the interval (α_k, τ_k). Thus, for any rate ρ, $\rho_1 \leq \rho \leq \rho_2$, we can write

$$\begin{aligned} Q_\rho(\tau_k) &= W(\alpha_k, \tau_k) - \rho_1(\tau_k - \alpha_k) - (\rho - \rho_1)(\tau_k - \alpha_k) \\ &= Q_{\rho_1}(\tau_k) - (\rho - \rho_1)(\tau_k - \alpha_k). \end{aligned}$$

That is, the plot of $Q_\rho(\tau_k)$ with respect to ρ in the range $\rho_1 \leq \rho \leq \rho_2$ is a straight line with slope $-(\tau_k - \alpha_k)$. This concludes the proof of Lemma 1.

□

Thus, we can characterize the entire burstiness curve of the elementary stream by starting from zero rate and progressively finding rate points at which either the maximum queue size moves to a different time instant, or a break in a busy period occurs. A naive approach to determine the former will need to examine all active periods within each busy period, since the maximum queue size occurs at the end of any active period. However, we can improve upon this by noting that the local maximum queue size within any busy period can move only to an earlier time instant when the rate ρ is increased. The following lemma proves this result formally.

Lemma 2: *Let τ_k and τ'_k be the time instants at which the local maximum queue sizes occur within busy period k for rates ρ and ρ', respectively, with $\rho' > \rho$. If the number of busy periods in the elementary video stream remains the same at rates ρ and ρ', then $\tau'_k \leq \tau_k$.*

Proof: Since two busy periods can never merge into a single busy period when the rate is increased, the starting times of the busy periods are identical at rates ρ and ρ'. We will prove the lemma by contradiction. Assume, if possible, that $\tau'_k > \tau_k$. Let $W(\tau_k, \tau'_k)$ denote the amount of traffic that arrived into the queue during the interval (τ_k, τ'_k). Since the interval (τ_k, τ'_k) belongs to a single busy period, the queue size $Q_\rho(\tau'_k)$ at time τ'_k is given by

$$Q_\rho(\tau'_k) = Q_\rho(\tau_k) + W(\tau_k, \tau'_k) - \rho(\tau'_k - \tau_k). \tag{3.5}$$

Similarly, for the rate ρ' we have

$$Q_{\rho'}(\tau'_k) = Q_{\rho'}(\tau_k) + W(\tau_k, \tau'_k) - \rho'(\tau'_k - \tau_k). \tag{3.6}$$

Since $Q_{\rho'}(\tau'_k)$ is a local maximum for the k-th busy period, we also have

$$Q_{\rho'}(\tau'_k) \geq Q_{\rho'}(\tau_k). \tag{3.7}$$

From Eq. (3.6) and (3.7),

$$\begin{aligned} W(\tau_k, \tau'_k) &\geq \rho'(\tau'_k - \tau_k) \\ &> \rho(\tau'_k - \tau_k), \text{ since } \rho' > \rho. \end{aligned} \tag{3.8}$$

From (3.5) and (3.8), we get

$$Q_\rho(\tau'_k) > Q_\rho(\tau_k),$$

which is a contradiction to the hypothesis that the local maximum queue size at rate ρ occurred at time τ_k. This concludes the proof of Lemma 2.

□

Note that the lemma applies even when there are multiple identical maxima within the same busy period. In such an event we can select the earliest maximum so as to minimize the computational effort.

Lemmas 1 and 2 enable us to design an algorithm for exact computation of the burstiness of the elementary stream by identifying values of the rate ρ at which the slope of the burstiness curve changes. The detailed, hierarchical pseudocode of the algorithm is given in Appendix A. The algorithm starts by setting the current rate ρ to zero. It then computes the busy periods for current rate ρ and the corresponding maximum queue size Q^*. The algorithm then proceeds to compute the next rate point at which the slope of the burstiness curve changes and updates the current rate to this value. This is repeated until the rate ρ reaches the peak rate r. The burstiness curve is obtained by connecting the global maximum queue size at all such points by linear segments.

The function *compute_busy_periods()* in Appendix A is used to compute the busy periods of the stream for a given value of the rate ρ. It takes a sequence of active periods and the current rate ρ as its input. It traverses the active periods and identifies each busy period j by calculating its starting and ending times, denoted by α_j and β_j, respectively. The function checks whether it should start a new busy period by checking the queue size Q at the beginning of the next active period, denoted by s_{i+1}. In case the queue size becomes less than zero, a new busy period is started. The function also records the local maximum queue size Q_j^* for each busy period j, and the global maximum Q^*.

The rate points in the burstiness curve are computed by the function *compute_next_rate()* in Appendix A. The function starts with the calculation of the lowest rate at which any busy period computed in the previous step breaks. It also computes the lowest rate at which the global maximum queue size moves to a different time instant, either within the same busy period or in a different one. In the case that the global maximum queue size stays within the same busy period, it can only move to an earlier time instant. This reduces computational effort. On the other hand, if the global maximum moves to a different busy period, its new position must either coincide with the location where the local maximum queue size occurred at the previous rate point, or at an earlier instant. The function *compute_next_rate()* selects the minimum rate among all these candidates as the rate point for the next iteration of the algorithm.

The worst-case time- and space-complexities of the algorithm can be determined by considering the computations performed at each rate point. It takes $O(n_a)$ steps to compute all the busy periods for a specific rate ρ, where n_a is the number of active periods. The number of candidate rates that need to be considered during each iteration to identify the next rate point is also $O(n_a)$, since we always check for candidate rates at the boundaries of the active periods. Therefore, the worst-case time complexity of the algorithm is $O({n_a}^2)$. The space needed to store the output of the algorithm is proportional to the number of rate points, which cannot exceed the number of active periods n_a. Thus, the space complexity is $O(n_a)$. Note that when simulation techniques are used, the time complexity becomes $O(\frac{r}{\delta}n_a)$, where δ is the chosen rate granularity for the burstiness curve. The corresponding space complexity is $O(\frac{r}{\delta})$ which does not depend on the number of active periods n_a. Note that the burstiness curve obtained by simulation of the source behavior at each rate point is not exact, but only an approximation that depends heavily on the rate granularity δ chosen.

As will be shown in Section 6, the worst-case space complexity of $O(n_a)$ is rather conservative that may occur in extreme cases such as when the frame sizes form an increasing function of time. In the case of MPEG elementary streams, however, the space complexity is mainly governed by the Group-of-Pictures (GOP) structure which gives average complexity much smaller than the worst-case value. Therefore, in such cases, the space complexity becomes $O(\alpha N_{gop})$, where α is constant that depends on the number of scene changes in the video stream, and N_{gop} the number of frames in the GOP structure. The constant α will have values close to 1 when the scenes are fairly static, and higher for streams that exhibit higher rate variability due to scene changes.

We now extend the algorithm to determine the burstiness curve of an MPEG-2 Transport Stream.

4 Algorithm for Exact Computation of the Burstiness Curve of MPEG-2 Transport Streams

The algorithm in the last section can be modified for computation of the burstiness curve of MPEG-2 Transport Streams. In this section we first summarize the MPEG-2 Transport Stream format and then outline the modifications to the algorithm.

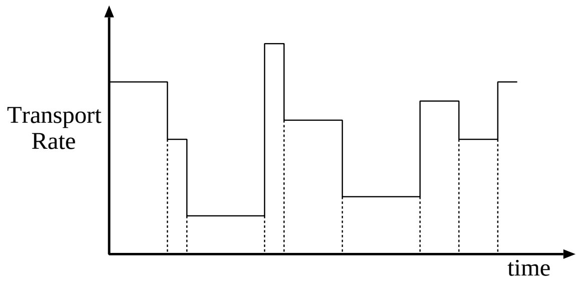

Figure 4.1: The behavior of Transport Rate in an MPEG-2 Transport Stream.

4.1 The MPEG-2 Transport Stream format

The MPEG standard defines a way of multiplexing more than one stream (video or audio) to produce a *program*. A program is considered a single service entity, and consists of one or more elementary streams. Elementary streams are the basic entities of a program. An elementary stream may be an MPEG-encoded audio or video stream, or a non-MPEG stream such as teletext or other information that is offered by a specific service provider.

Two schemes are used in the MPEG-2 standard for the multiplexing process.

Program Stream: This is similar to MPEG-1 Systems layer. It is a grouping of elementary streams that have a common time-base for delivery. Each Program Stream consists of only one program.

Transport Stream: The Transport Stream combines one or more programs into a single stream with fixed-size packets. The programs may or may not have a common time-base.

The MPEG-2 Transport Stream format is the preferred choice in environments where errors are likely to occur, as in the case of transport over a packet-switched network. It is currently used in Digital Video Broadcasting (DVB) systems for digital television and data broadcasting across a broad range of delivery media. The format makes use of explicit timestamps (called Program Clock References or PCRs in MPEG-2 terminology) embedded within the transport packets to facilitate the clock recovery at the receiver.

The transport rate of a Transport Stream may be either fixed or variable. However, it has an important property: it is piecewise constant [10, 15] (see Figure 4.1). The rate of the Transport Stream changes only at the instants when a new PCR value is received at the receiver. During an interval between such rate changes, the transport rate can be computed by

$$\text{transport\_rate(i)} = \frac{\text{(no. of bytes between PCRs)} \times \text{system clock frequency}}{\text{PCR}_{\text{new}} - \text{PCR}_{\text{old}}}. \tag{4.1}$$

Therefore, during the time interval between the reception of two consecutive PCRs, the rate remains constant. We call this time interval as *a rate segment.* We take advantage of this property to design an efficient algorithm for computing its burstiness curve. The algorithm is based on observing the queue behavior at the input of the second shaper in Figure 2.1, as in the case of elementary video streams. Note that, in the case of MPEG-2 Transport Streams, the peak-rate shaping process does not introduce any additional delay if the peak rate is set greater than or equal to the maximum rate found in the stream. We now develop the algorithm for calculating the maximum queue size at the input of the second shaper ($\max_t(Q_\rho(r,t))$) for a peak rate value r greater than or equal to the maximum rate found in the stream, and service rate ρ.

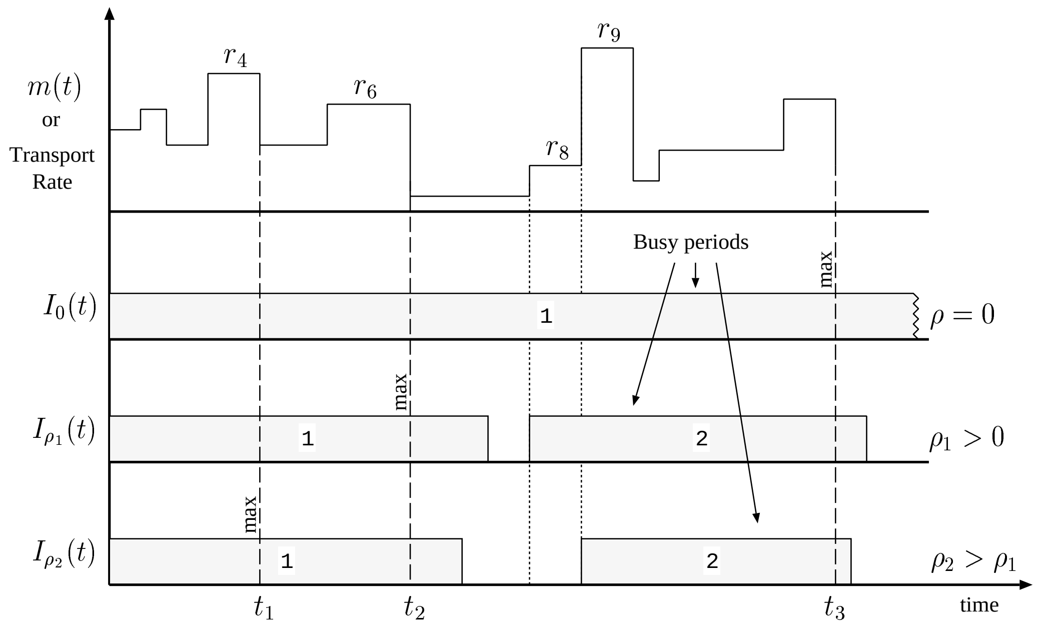

Figure 4.2: Busy periods of an MPEG-2 Transport Stream for different service rates ρ.

4.2 Computation of the Burstiness Curve

Let T be the total duration of the MPEG-2 Transport Stream and $m(t)$ the instantaneous rate at time t, $0 \leq t \leq T$. We assume that the peak rate r is greater than or equal to the maximum rate of the stream. Let n_r denote the number of rate segments, s_i the time instant at which the ith rate segment commences, and t_i the time when it ends. Then, $s_{i+1} = t_i$. Let r_i be the transport rate within the i-th segment, and let r_{min} and r_{max} be the minimum and maximum, respectively, among r_i. That is,

$$r_{min} = \min_{1 \leq i \leq n_r} r_i \quad \text{and} \quad r_{max} = \max_{1 \leq i \leq n_r} r_i.$$

We also use the same indicator function $I_\rho(t)$ defined in Eq. (3.4).

As before, we define *busy period* as a maximal period of time during which the queue of the server at the output of the peak-rate shaper is non-empty. Let $Q^*(\rho)$ denote the maximum queue size reached over the entire duration of the MPEG-2 Transport Stream. Our algorithm for estimating $Q^*(\rho)$ is based on the observation that, if the maximum queue size occurs in some busy period i when the rate is ρ, the variation of the maximum queue size with respect to ρ will be linear between rate points at which one of the following events occurs:

1. The maximum queue size moves to a different instant within the same busy period, or to a different busy period.
2. The number of busy periods changes.
3. The starting time of a busy period moves.

Note that the difference between the Transport Stream and the elementary stream is in (3) above. While an active period in an elementary stream always belongs to a busy period, a rate segment in the Transport Stream may not belong to a busy period if the arrival rate during the segment is less than the service rate ρ. Hence, as the service rate ρ is increased, the starting times of some busy periods may move to the right, as shown in the example of Figure 4.2.

Figure 4.2 shows the transport rate $m(t)$ within each rate segment of the MPEG-2 Transport Stream, and the resulting busy periods at the input of the server operating at rate ρ. When the rate ρ of the server is zero, the queue of the server accumulates all the data arrived during the length of the trace in one large busy period. Therefore, the maximum queue size occurs at the end of the last rate segment. As the rate of the server increases to ρ_1, the busy period shrinks until it splits into two busy periods. Let us assume that the maximum queue size occurs at time t_2 for rate ρ_1 which is the end of rate segment 6. On increasing the rate further, the busy periods of $I_{\rho_1}(t)$ shrink without breaking and the maximum stays at t_2, until rate ρ_2 is reached. Assuming that $\rho_2 > r_8$, the second busy period commences at the beginning of rate segment 9, since no accumulation can occur during rate segment 8. At rate ρ_2, the maximum moves to time instant t_1 while the total number of busy periods stays unchanged. Note that (i) the maximum queue size always occurs at the end of a rate segment, (ii) the maximum queue size can never occur in a rate segment i which has $r_i < \rho$, and (iii) a busy period always commences in a rate segment i which has $r_i > \rho$.

For an MPEG-2 Transport Stream of finite duration, the values of the rate ρ that cause a break in a busy period, the maximum queue size to be moved to a different location, or the beginning of a busy period to move, form a *finite* set. We need to compute only the rates belonging to this set for the exact computation of the burstiness curve. The burstiness curve is piecewise linear between adjacent rate points belonging to this finite set.

Lemma 3: *The burstiness curve of an MPEG-2 Transport Stream is piecewise linear. The slope of the burstiness curve changes only at rate points where one of the following events occurs:*

1. *A change in the time instant at which the maximum queue size $\sigma(\rho)$ occurs,*
2. *a change in the number of busy periods, or*
3. *a change in the starting time of a busy period.*

Proof: The proof is similar to that of Lemma 1 and is therefore omitted.

□

We use the notation α_i, β_i for the starting and the ending times of busy period i. Note that we need to identify the linear segments of the burstiness curve only between rates r_{min} and r_{max}, since the slope cannot change during the intervals $[0, r_{min})$ and $(r_{max}, r]$. Thus, we can characterize the entire burstiness curve of the MPEG-2 Transport Stream by starting from rate r_{min} and progressively finding rate points at which either the maximum queue size moves to a different time instant, a break in a busy period occurs, or the starting time of a busy period moves to a different point. As in the case of elementary video streams, the local maximum queue size within any busy period can move only to an earlier time instant when the rate ρ is increased.

Lemma 4: *Let τ_k and τ'_k be the time instants at which the local maximum queue sizes occur within busy period k for rates ρ and ρ', respectively, with $\rho' > \rho$. If the number of busy periods in the MPEG-2 Transport Stream remains the same at rates ρ and ρ', then $\tau'_k \leq \tau_k$.*

Proof: The proof is similar to that of Lemma 2 and is therefore omitted.

□

Based on the above results, we can design an algorithm that determine the rate points at which the slope of the burstiness curve changes. The detailed pseudocode of the algorithm is given in Appendix B. The algorithm starts by setting the rate ρ to the minimum rate of the Transport Stream segments, that is, r_{min}. It determines the busy periods corresponding to the current rate

ρ. The algorithm then examines the candidate rate points at which the global maximum queue size moves, a busy period breaks into multiple periods, or the starting time of a busy period changes. The minimum among these candidate rates is the next point in the burstiness curve. The entire burstiness curve is obtained by repeating this process iteratively until the maximum rate r_{max} is reached.

The function *compute_busy_periods()* in Appendix B describes how the busy periods are computed for a given value of the rate ρ. The algorithm takes a sequence of rate segments and the current rate ρ as its input. It traverses the rate segments and identifies the starting and ending times of each busy period by maintaining the accumulated queue size over the duration of the stream. The function identifies the start of a rate segment as the start of a new busy period when the accumulated queue is empty at the beginning of the rate segment and the rate of the segment is higher than the current rate ρ. The busy period ends when the accumulated queue size becomes zero.

Determination of the next rate point of the burstiness curve is performed by the function *compute_next_rate()* in Appendix B. For each busy period corresponding to the current rate, the function first calculates the lowest rate at which the busy period breaks into multiple periods. It subsequently computes the lowest rate at which the global maximum queue size moves to a different time instant, either within the same busy period or to a different one. Finally, it also finds the minimum among the rates of the stream at the beginning of each busy period, which marks the next rate point at which the starting point of a busy period moves. The algorithm chooses the minimum among all these candidates as the next rate point.

As in the case of the previous algorithm, the worst-case time complexity of this algorithm is $O(n_r^2)$. Each iteration needs $O(n_r)$ steps, and the number of iterations is also $O(n_r)$, where n_r is the number of rate segments present in the MPEG-2 Transport Stream. The latter arises from the observation that the candidate rates for the next step are always computed at the boundaries of the rate segments. Also, the space needed to store the output is $O(n_r)$ since, in the worst case, one point may be stored for each rate segment. In practice, however, the space complexity of the algorithm is likely to be much less than $O(n_r)$, due to the fact that the rate segments may exhibit some periodicity similar to the GOP structure of the MPEG elementary video streams. In contrast, simulation techniques have a time complexity of $O(\frac{r_{max}}{\delta} n_r)$ and space complexity of $O(\frac{r_{max}}{\delta})$, where δ is the rate granularity.

5 Approximate Algorithms

The algorithms proposed in the previous sections produce the exact burstiness curve with the minimum number of points. The exact computation, however, requires identifying all the candidate rate points where the maximum queue size may move to a different location, a busy period may break, or the starting time of a busy period may change (for Transport Streams). In practice, however, we can approximate the burstiness curve by considering only a subset of these events that are most likely to occur. From our experiments with real video traces, we found that the location of the maximum queue size almost never moved when a break occurred in a different busy period. Similarly, a change in the location of the maximum queue size from one busy period to another almost always occurred as a result of a break in the former busy period. Thus, we can simplify the algorithms in the previous sections by considering only candidate rate points at which one of the following events occur:

1. A break in the busy period where the current global maximum queue size is located,

Trace	**Frame Rate** (Hz)	**Number of Frames**	**Video Length** (mins)	**Alg. Running Time** (h:mm:ss.m)	**Space** (# points)
NTSC trace	30	2335	1.3	0:00:02.7	12
PAL trace	25	21763	14.5	0:03:40.7	13
Garrett's trace	24	174136	120.9	6:19:06.1	63

Table 6.1: Performance results of the exact algorithm on three elementary video streams. The running time is the user time as measured on a Sun Ultra-2 workstation.

2. a change in the location of the maximum queue size within the same busy period, or
3. a change in the starting time of a busy period (for Transport Streams only).

The detailed pseudocodes of the approximate algorithms for both elementary streams and MPEG-2 Transport Streams are given in Appendix C. Our experiments with the algorithms on several video traces indicate that the approximate algorithms can be faster by many orders of magnitude, yet produce burstiness curves that are virtually indistinguishable from those produced by the exact algorithms.

6 Validation

In this section we validate our algorithms with actual traces and compare their performance with results from simulation-based algorithms. For simplicity, we will focus on elementary video streams. Because of the similarity of the algorithms for the elementary and Transport Streams, we can expect the results for the former to be similar.

We consider three separate video traces: The first is an elementary video stream in NTSC format consisting of 2,335 frames, with a total duration of 78 seconds; the second is an elementary video stream in PAL format with 21,763 frames and 14.5 minutes; and the last is a long trace generated by Mark Garrett [6] consisting of 174,136 frames with a frame rate or 24 Hz, corresponding to a duration of approximately 2 hours. The characteristics of the traces are summarized in Table 6.1. In all experiments, we set the peak rate of the shaping mechanism to 155 Mbits/second.

The execution times of the exact algorithm for the three traces are shown in Table 6.1. As expected, the execution time is relatively small for the first two traces, but the fact that the execution time can grow as the square of the number of active periods is evident in the case of the third trace. The number of points in the last column is the number of points in the burstiness curve, and is related to the number of frames in the GOP structure. The relatively large number of points in Garrett's trace is justified by the nature of the trace, which is a mix of several video segments with diverse characteristics, ranging from frame sequences with little motion to high-action sequences [6]. In all traces, the space needed for storing the points of the burstiness curve, for example as metadata in a video file in a video server, is insignificant compared to the actual video trace. This makes the exact algorithms very suitable for Video-on-Demand (VoD) servers, where computation of the burstiness curve can be performed off-line.

An interesting observation is that most of the points computed by the algorithm are concentrated in a very small rate interval (see Figure 6.1). As a result, simulation-based techniques must use extremely fine rate-granularities to construct the burstiness curve, which makes them very inefficient. For example, if we use simulation to compute the burstiness curve of the PAL trace with the same number of output points, the rate granularity must be at least $155/13 = 11.92$ Mbits/second. The

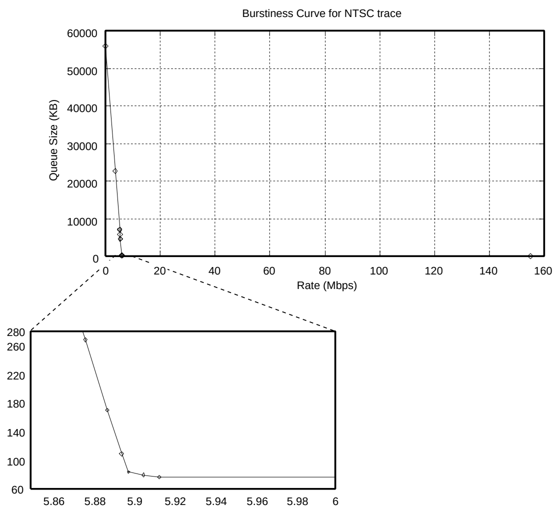

Figure 6.1: Burstiness curve for the NTSC trace as computed by the exact algorithm. Note the concentration of points around 5.9 Mbps.

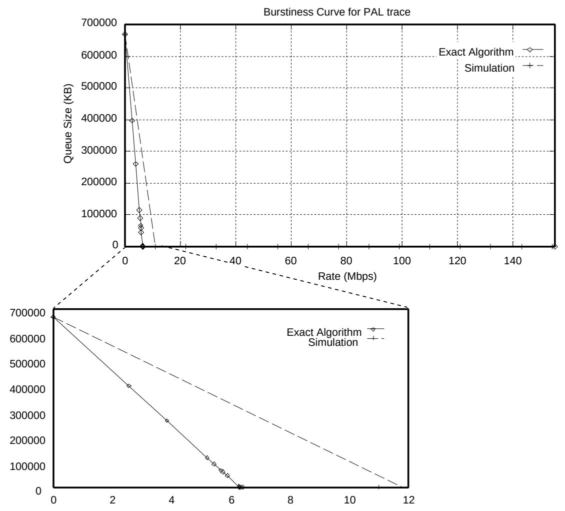

Figure 6.2: Comparison between the burstiness curve computed by the exact algorithm and the burstiness curve computed by simulation with 11.92 Mbps rate granularity for the PAL trace.

resulting burstiness curve is shown in Figure 6.2, where the discrepancy introduced is evident. In all cases, the exact algorithm computed only the necessary points for the burstiness curve.

When the approximate algorithm was used for the computation of the burstiness curve, the results are very close to those from the exact algorithm, as shown in Table 6.2. In the cases of the NTSC and the PAL traces, the approximate algorithm produced the same set of points as the exact algorithm. In the case of Garrett's trace, the approximate algorithm computed one fewer point (62 points instead of 63), and introduced slight discrepancies in the rate values of three other points. However, the discrepancies were minor as indicated in the zoomed version of the burstiness curve shown in Figure 6.3. The difference in the running times of the algorithms, however, is significant (10.6 seconds compared to more than 6 hours for the long trace). This makes the approximate algorithm very suitable for on-line computation of the burstiness curve of real-time video sources such as in video broadcasting. In such cases, the video stream can be segmented to fixed time intervals and the burstiness curve can be obtained for each segment, facilitating per-segment QoS provisioning and call admission control.

To further compare the exact and approximate algorithms, we applied both algorithms on several traces [1] and compared the resulting burstiness curves. The results, shown in Table 6.3, demonstrate that the exact and approximate burstiness curves almost always coincide, except for a limited number of points (up to 5 in our experiments) where there are discrepancies. Column 5 of the table provides the maximum absolute difference in the burstiness values of the two curves at any point. In most cases, the difference is under 10 %. There are a small number of cases where the maximum difference is above 10%. In these cases, however, the large discrepancies occurs at rate points beyond the normal operating range of the video source. For example, in the case of the *MrBean* trace, the maximum difference occurs at a rate close to the average rate of the trace, a region usually avoided in practice (Figures 6.4 and 6.5). In addition, availability of the two algorithms enables a two-step approach

Trace	Exact Algorithm		Approximate Algorithm	
	Time (h:mm:ss.m)	**Space** (# points)	**Time** (h:mm:ss.m)	**Space** (# points)
NTSC trace	0:00:02.7	12	0:00:00.0 (reported)	12
PAL trace	0:03:40.7	13	0:00:00.5	13
Garrett's trace	6:19:06.1	63	0:00:10.6	62

Table 6.2: Comparison between the exact and the approximate burstiness curve algorithms for three elementary video streams. The algorithm running time is the user time as measured on a Sun Ultra-2 workstation.

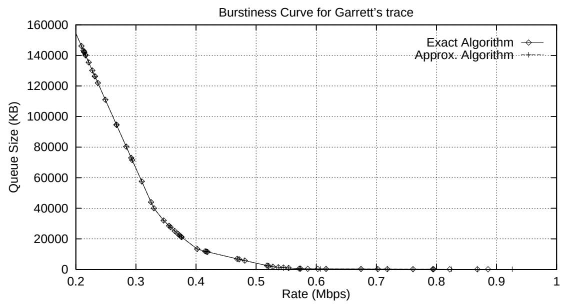

Figure 6.3: Comparison between the burstiness curve computed by the exact and approximate algorithms for Garrett's trace.

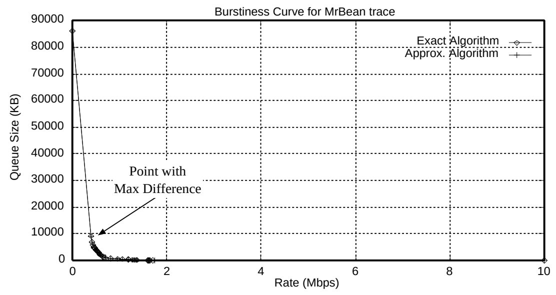

Figure 6.4: Comparison between the burstiness curve computed by the exact and approximate algorithms for the MrBean trace.

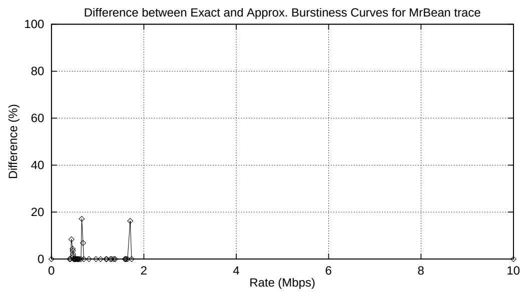

Figure 6.5: Plot of the difference (%) between the burstiness curve computed by the exact and approximate algorithms for the MrBean trace. The average and the peak rate of the trace is 0.441 and 5.727 Mbps respectively.

for the computation of the burstiness curve: the approximate algorithm can be used to establish the operating range of the video source, and the exact algorithm can be applied over this interval to characterize the video stream precisely.

7 Conclusions

In this paper, we developed efficient deterministic algorithms for computation of the burstiness curve of both elementary video streams and MPEG-2 Transport Streams. The algorithms enable the exact computation of the burstiness curve of a video stream, as compared to simulation-based algorithms where the accuracy depends on the rate granularity chosen. We also presented versions

Trace	Exact Alg.	Approx. Alg.	Comparison		
	# Points	# Points	# Points Diff.	Max. Diff. (KB)	Max. Diff. (%)
NTSC trace	12	12	0	0.04	0.05
PAL trace	13	13	0	2.21	1.70
Garrett's	63	62	1	47.47	12.08
MrBean	36	31	5	188.32	17.11
asterix	51	50	1	10.41	6.93
atp	24	24	0	0.14	0.36
bond	43	43	0	7.24	5.48
dino	48	48	0	3.65	8.68
lambs	39	39	0	1.26	8.49
mtv1	48	48	0	1.45	7.04
mtv2	30	30	0	15.46	7.70
news1	34	34	0	5.59	2.62
news2	40	40	0	0.00	0.00
race	32	32	0	5.18	15.02
sbowl	38	38	0	0.74	5.30
simpsons	28	28	0	2.67	7.48
soccer1	33	33	0	384.65	13.62
soccer2	43	43	0	1.53	7.25
star	59	59	0	0.31	2.27
talk1	26	26	0	278.32	14.77
talk2	33	33	0	0.11	0.75
terminator	29	28	1	5.07	8.31

Table 6.3: Comparison between the exact and the approximate burstiness curve algorithms for several elementary video streams taken from [1]. The maximum difference denotes the difference in the burstiness values between the exact and approximate algorithm for the rate point at which the percentage of that difference is maximum.

of the algorithms which trade off computational effort for accuracy.

Our experiments with several video traces suggest that the discrepancies introduced by the approximate algorithm are too small to be noticeable. The key assumption made in the approximation is that the location of the maximum queue size moves rarely from its busy period as the rate is increased, except when the busy period breaks into multiple periods. This avoids the need to examine other busy periods during each iteration of the algorithm.

The accuracy of the algorithms makes them attractive not only to video servers that need to compute the burstiness curve of their video traces for call-admission control (CAC) and QoS provisioning, and store it as metadata with the trace, but also to real-time video distribution systems that need to estimate the burstiness curve of their video programs in real-time. In addition, the algorithms can be used to characterize any bursty ON-OFF source, including voice and data.

All the algorithms presented in this paper have been implemented and can be downloaded from `http://www.cse.ucsc.edu/research/hsnlab`.

References

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

[2] C. S. Chang. Stability, queue length, and delay of deterministic and stochastic networks. *IEEE Transactions on Automatic Control*, 39(5):913–931, May 1994.

[3] S Chong and S-Q Li. Probabilistic burstiness-curve-based connection control for real-time multimedia services in ATM networks. *IEEE Journal on Selected Areas in Communications*, 15(6):1072–1086, August 1997.

[4] The ATM Forum Technical Committee. Traffic Management Specification version 4.0. Technical report, The ATM Forum, 1996.

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

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

[7] P. Goyal, S. S. Lam, and H. M. Vin. Determining end-to-end delay in heterogeneous networks. In *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'95)*, Durham, New Hampshire, April 1995.

[8] M. Graf. Traffic shaping of VBR video in ATM end systems. In *Proceedings of the 4-th Open Workshop on High Speed Networks*, September 1994.

[9] R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE Journal on Selected Areas in Communications*, 9(7):968–981, September 1991.

[10] 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.

[11] E. W. Knightly, D. E. Wrege, J. Liebeherr, and H. Zhang. Fundamental limits and tradeoffs of providing deterministic guarantees to VBR video traffic. In *Proceedings of ACM SIGMETRICS '95*, 1995.

[12] 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.

[13] J. Roberts and M. Hamdi. Video transport in ATM networks. *Journal on Interoperable Communication Networks*, January 1998.

[14] D. Stiliadis and A. Varma. Latency-rate servers: A general model for analysis of traffic scheduling algorithms. Technical Report UCSC-CRL-95-38, University of California at Santa Cruz, Dept. of Computer Engineering, July 1995.

[15] 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`.

A Pseudocode of exact burstiness curve algorithm for elementary video streams

```
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
               of $i$-th frame arrival and index $j$ denotes the $j$-th active period. */
1           /* Perform initialization */
 1.1        $s_1^r \leftarrow 1/f$; $q_1 \leftarrow d_1$; $i \leftarrow j \leftarrow 1$;
2
 2.1        If $\left(\frac{q_i}{r} < \frac{1}{f}\right)$   /* no backlog present at the start of next frame */
 2.1.2          $q_{i+1} \leftarrow d_{i+1}$;
 2.1.3          $t_j^r \leftarrow \frac{i}{f} + \frac{q_i}{r}$;   /* compute end time of current active period */
 2.1.4          $j \leftarrow j + 1$;
 2.1.5          $s_j^r \leftarrow \frac{i+1}{f}$;   /* start time of new active period */
 2.2        else
 2.2.1          $q_{i+1} \leftarrow q_i + d_{i+1} - \frac{r}{f}$;  /* there is backlog carried to current frame */
 2.3        endif
3
 3.1        $i \leftarrow i + 1$;
 3.2        If $(i < N)$
 3.2.1          goto Step 2;
 3.3        endif
4
 4.1        $s(r) \leftarrow \max_{1 \leq k \leq N} q_k$;  /* compute maximum queue length observed */
 4.2        $t_j^r \leftarrow \frac{N}{f} + \frac{q_N}{r}$;            /* compute end time of last active period */
 4.3        $n_a(r) \leftarrow j$;                  /* store the number of active periods */
```

Figure A.1: 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.

Burstiness Curve Algorithm Pseudocode for Elementary Video Streams

```
1       /* Initialization */
 1.1    $\rho \leftarrow 0$; $\alpha_{old} \leftarrow -1$; $\tau_{old} \leftarrow -1$;
2       /* Compute busy periods for current rate $\rho$, the busy period $j$
           in which the global maximum queue size occurs, and the
           corresponding max queue size $Q^*$ */
 2.1    compute_busy_periods($\rho$);
3       /* Compute rate for the next step */
 3.1    $\rho' =$ compute_next_rate($\rho$);
4       /* Output corresponding point of the burstiness curve if necessary */
 4.1    If ($\alpha_j \neq \alpha_{old}$) or ($\tau_j \neq \tau_{old}$)
 4.1.1      Output ($\rho, Q^*$);
 4.2    If ($\rho' = r$)
 4.2.1      Output ($r, 0$);
 4.2.2      STOP;
 4.3    else
 4.3.1      $\rho \leftarrow \rho'$; $\alpha_{old} \leftarrow \alpha_j$; $\tau_{old} \leftarrow \tau_j$;
 4.3.2      goto Step 2;
 4.4    endif
```

Figure A.2: Pseudocode of the algorithm that computes the exact burstiness curve of an elementary video stream.

compute_busy_periods(ρ)

```
        /* For a given sequence of active periods $(s_1,t_1),(s_2,t_2),\ldots,(s_{n_a},t_{n_a})$, and rate $\rho$,
           determines the busy periods $(\alpha_j,\beta_j)$, the local maximum queue size within
           each busy period $Q^*_j$ and the time at which the maximum occurs $\tau_j$. */
        /* Perform initialization */
1.      Define $s_{n_a+1} = \infty$;
2.      $i \leftarrow 1$;    /* index of active period */
3.      $j \leftarrow 1$;    /* index of busy period */
4.      $Q' \leftarrow 0$;    /* $Q'$ is used to compute the current queue size */
5.      $\alpha_j \leftarrow s_i$; $Q^*_j \leftarrow 0$;
6.      **If** $(\rho = 0)$     /* Corner case when rate is zero */
6.1         Busy period $(\alpha_j,\beta_j)$ contains all active periods;
6.2         $Q^*_j \leftarrow$ stream size;
6.3         $\alpha_j \leftarrow s_1$; $\beta_j \leftarrow t_{n_a}$; $\tau_j \leftarrow t_{n_a}$;
7.      **endif**
        /* Compute queue size at the end of current active period */
8.      $Q \leftarrow Q' + (t_i - s_i)r - \rho(t_i - s_i)$;
9.      **If** $(Q^*_j < Q)$    /* update local maximum */
9.1         $Q^*_j \leftarrow Q$; $\tau_j \leftarrow t_i$;
10.     **endif**
        /* Check if current busy period ends before next active period begins */
11.     $Q' \leftarrow Q - \rho(s_{i+1} - t_i)$;
12.     **If** $(Q' < 0)$
            /* current busy period ended before next active period */
12.1        $\beta_j \leftarrow t_i + \frac{Q}{\rho}$;    /* end of busy period */
12.2        $j \leftarrow j + 1$;    /* to next busy period */
12.3        $\alpha_j \leftarrow s_{i+1}$; $Q \leftarrow 0$; $Q^*_j \leftarrow 0$;   /* init variables for next busy period */
13.     **endif**
14.         $i \leftarrow i + 1$;   /* to next active period */
15.     **If** $(i \le n_a)$ **goto** step 8;
16.     **endif**
17.     $Q^* \leftarrow \max_{1\le k\le n_a} Q^*_k$;   /* compute global maximum queue size */
```

Figure A.3: Pseudocode of function compute_busy_period() of the burstiness algorithm that computes the busy periods of an elementary video stream.

```
compute_next_rate($\rho$)

        /* Given current rate $\rho$, determines the next higher rate $\rho'$
           at which the slope of the burstiness curve may change */

1.      $\rho' \leftarrow r$;
2.      For $j = 1$ to $n_b$   /* for each busy period */
 2.1        Determine the lowest rate $\rho_{j,1}$ at which the busy period $j$ breaks (Figure A.5);
 2.2        If ($\rho_{j,1} < \rho'$)
 2.2.1          $\rho' \leftarrow \rho_{j,1}$;
 2.3        endif
3.      endfor
        /* Now determine the rates at which the global maximum moves */
4.      For $j = 1$ to $n_b$
 4.1        Determine the lowest rate $\rho_{j,2}$ at which the position of the global maximum
            queue size changes (Figure A.6);
 4.2        If ($\rho_{j,2} < \rho'$)
 4.2.1          $\rho' \leftarrow \rho_{j,2}$;
 4.3        endif
5.      endfor
6.      return($\rho'$);
```

Figure A.4: Pseudocode of function compute_next_rate() of the burstiness algorithm that computes the rate for the next iteration of the algorithm.

```
        /* Given the busy period $(\alpha_j, \beta_j)$ corresponding to rate $\rho$, determine
           the lowest rate at which the busy period breaks into multiple periods */

        /* Let $(s_k, t_k), (s_{k+1}, t_{k+1}), \ldots, (s_m, t_m)$ be the active periods contained
           within the busy period $(\alpha_j, \beta_j)$ */

1.      If $(m = k)$    /* only one active period within busy period */
 1.1        return($\infty$);
2.      endif
3.      $i \leftarrow k$;   /* start with first active period */
4.      $W \leftarrow 0$;   /* $W$ denotes the accumulated arrivals within busy period */
5.      $\rho_{min} \leftarrow r$;   /* initialize minimum */
        /* Determine the rate at which a break can occur between $t_i$ and $s_{i+1}$ */
6.      $W \leftarrow W + (t_i - s_i)r$;   /* total arrivals up to $t_i$ */
7.      $\rho \leftarrow \frac{W}{(s_{i+1} - s_i)}$;
8.      If $(\rho < \rho_{min})$
 8.1        $\rho_{min} \leftarrow \rho$;   /* update minimum */
9.      endif
10.     $i \leftarrow i + 1$;
11.     If $(i < m)$ goto step 6;
12.     return($\rho_{min}$);
```

Figure A.5: Pseudocode for the computation of $\rho_{j,1}$.

```
        /* Given the busy period $(\alpha_j, \beta_j)$ corresponding to rate $\rho$, determine the
           lowest rate at which the global maximum queue size moves to a
           different time instant within the current busy period. Let $(\alpha_l, \beta_l)$
           be the busy period in which the current global maximum occurs,
           and $\tau_l$ the corresponding time. Let $\tau_j$ be the earliest instant at which
           the local maximum queue size occurs within the current busy period.
           Let $(s_k, t_k), (s_{k+1}, t_{k+1}), \ldots, (s_m, t_m)$ be the active periods contained
           within the interval $(\alpha_j, \tau_j)$. Let $(s_p, t_p), (s_{p+1}, t_{p+1}), \ldots, (s_q, t_q)$ be
           the active periods contained within the interval $(\alpha_l, \tau_l)$. */

        /* compute total arrivals in busy period $l$ up to time $\tau_l$ */
1.      $W_l \leftarrow \sum_{i=p}^{q} r(t_i - s_i)$;

2.      $i \leftarrow k$;   /* start with first active period */
3.      $W \leftarrow 0$;   /* $W$ denotes the arrivals during current busy period */
4.      $\rho_{min} \leftarrow r$;
        /* check if global maximum is in the first active period of current busy period */
5.      **If** $(j = l)$ **and** $(q = p)$     /* maximum cannot move */
 6.1        **return**$(r)$;
7.      **endif**
        /* Compute total arrivals at the end of current active period */
8.      $W \leftarrow W + (t_i - s_i)r$;
        /* Compute the rate at which queue size at $t_i$ becomes equal to that at time $\tau_l$ */
9.      $\rho \leftarrow \frac{W_l - W}{(\tau_l - s_p) - (t_i - s_k)}$;
10.     **If** $(\rho < \rho_{min})$
 10.1       $\rho_{min} \leftarrow \rho$;
11.     **endif**
12.     $i \leftarrow i + 1$;
13.     **If** $(i \leq m)$ **goto** step 5;
14.     **return**$(\rho_{min})$;
```

Figure A.6: Pseudocode for the computation of $\rho_{j,2}$.

B Pseudocode of exact burstiness curve algorithm of MPEG-2 Transport Streams

```
BURSTINESS CURVE ALGORITHM PSEUDOCODE FOR MPEG-2 TRANSPORT STREAMS
1       /* Initialization */
 1.1    $\rho \leftarrow r_{min}$; $\alpha_{old} \leftarrow -1$; $\tau_{old} \leftarrow -1$;
2       /* Compute busy periods for current rate $\rho$, the busy period $j$
           in which the global maximum queue size occurs, and the
           corresponding max queue size $Q^*$ */
 2.1    compute_busy_periods($\rho$);
3       /* Compute rate for the next step */
 3.1    $\rho'$ = compute_next_rate($\rho$);
4       /* Output corresponding point of the burstiness curve if necessary */
 4.1    If ($\alpha_j \neq \alpha_{old}$) or ($\tau_j \neq \tau_{old}$)
 4.1.1      Output ($\rho, Q^*$);
 4.2    If ($\rho' = r_{max}$)
 4.2.1      Output ($r_{max}, 0$);
 4.2.2      STOP;
 4.3    else
 4.3.1      $\rho \leftarrow \rho'$; $\alpha_{old} \leftarrow \alpha_j$; $\tau_{old} \leftarrow \tau_j$;
 4.3.2      goto Step 2;
 4.4    endif
```

Figure B.1: Pseudocode of the algorithm that computes the exact burstiness curve of an MPEG-2 Transport Stream.

```
compute_busy_periods($\rho$)

        /* For a given sequence of rate segments $(s_1, t_1), (s_2, t_2), \ldots, (s_{n_r}, t_{n_r})$, and rate $\rho$,
           determines the busy periods $(\alpha_j, \beta_j)$, the local maximum queue size within
           each busy period $Q_j^*$, the time at which the maximum occurs $\tau_j$,
           and the global maximum queue size $Q^*$. */

        /* Perform initialization */
1.      Define $s_{n_r+1} = \infty$;
2.      $i \leftarrow 1$;   /* index of rate segment */
3.      $j \leftarrow 1$;   /* index of busy period */
4.      $Q \leftarrow 0$;   /* $Q$ denotes the current queue size */
5.      $\alpha_j \leftarrow s_i$; $Q_j^* \leftarrow 0$;
        /* Compute queue size at the end of current rate segment */
6.      $Q \leftarrow Q + (t_i - s_i)r_i - \rho(t_i - s_i)$;
7.      If $(Q_j^* < Q)$   /* update local maximum */
 7.1        $Q_j^* \leftarrow Q$; $\tau_j \leftarrow t_i$;
8.      endif
        /* Check if current busy period ends before
           next rate segment begins */
9.      If $(Q < 0)$
            /* current busy period ended before next rate segment */
 9.1        $\beta_j \leftarrow t_i + \frac{Q}{\rho - r_i}$;   /* end of busy period */
 9.2        $j \leftarrow j + 1$;   /* to next busy period */
            /* Skip all rate segments $i$ which have $r_i < \rho$; */
 9.3        while $(r_i < \rho)$, $i \leftarrow i + 1$;
 9.4        $\alpha_j \leftarrow s_{i+1}$; $Q \leftarrow 0$; $Q_j^* \leftarrow 0$;   /* init variables for next busy period */
10.     endif
11.         $i \leftarrow i + 1$;   /* to next rate segment */
12.     If $(i \leq n_r)$ goto step 6;
13.     endif
14.     $Q^* \leftarrow \max_{1 \leq k \leq n_r} Q_k^*$;   /* compute global maximum queue size */
```

Figure B.2: Pseudocode of function compute_busy_period() of the burstiness algorithm that computes the busy periods.

```
compute_next_rate($\rho$)

          /* Given current rate $\rho$, determines the next higher rate $\rho'$
             at which the slope of the burstiness curve may change */

1.        $\rho' \leftarrow r_{max}$;
2.        For $j = 1$ to $n_b$   /* for each busy period */
 2.1          Determine the lowest rate $\rho_{j,1}$ at which the busy period $j$ breaks (Figure B.4);
 2.2          If ($\rho_{j,1} < \rho'$)
 2.2.1            $\rho' \leftarrow \rho_{j,1}$;
 2.3          endif
3.        endfor
          /* Now determine the rates at which the global maximum moves */
4.        For $j = 1$ to $n_b$
 4.1          Determine the lowest rate $\rho_{j,2}$ at which either the position of the global
              maximum queue size changes, or the starting time of busy period $j$ moves
              to a later time instant (Figure B.5);
 4.2          If ($\rho_{j,2} < \rho'$)
 4.2.1            $\rho' \leftarrow \rho_{j,2}$;
 4.3          endif
5.        endfor
6.        return($\rho'$);
```

Figure B.3: Pseudocode of function compute_next_rate() of the burstiness algorithm that computes the rate for the next iteration of the algorithm.

```
        /* Given the busy period $(\alpha_j, \beta_j)$ corresponding to rate $\rho$, determine
           the lowest rate at which the busy period breaks into multiple periods */

        /* Let $(s_k, t_k), (s_{k+1}, t_{k+1}), \ldots, (s_m, t_m)$ be the rate segments contained
           within the busy period $(\alpha_j, \beta_j)$ */

1.      If $(m = k)$    /* only one rate segment within busy period */
 1.1        return($\infty$);
2.      endif
3.      $i \leftarrow k$;   /* start with first rate segment */
4.      $W \leftarrow 0$;   /* $W$ denotes the accumulated arrivals within busy period */
5.      $\rho_{min} \leftarrow r_{max}$;   /* initialize minimum */
        /* Determine the rate at which a break can occur between $t_i$ and $s_{i+1}$ */
6.      $W \leftarrow W + (t_i - s_i)r_i$;   /* total arrivals up to $t_i$ */
7.      $\rho \leftarrow \frac{W}{(s_{i+1} - s_i)}$;
8.      If $(\rho < \rho_{min})$
 8.1        $\rho_{min} \leftarrow \rho$;   /* update minimum */
9.      endif
10.     $i \leftarrow i + 1$;
11.     If $(i < m)$ goto step 6;
12.     return($\rho_{min}$);
```

Figure B.4: Pseudocode for the computation of $\rho_{j,1}$.

```
        /* Given the busy period $(\alpha_j, \beta_j)$ corresponding to rate $\rho$, determine the
           lowest rate at which the global maximum queue size moves to a
           different time instant within the current busy period. Let $(\alpha_l, \beta_l)$
           be the busy period in which the current global maximum occurs,
           and $\tau_l$ the corresponding time. Let $\tau_j$ be the earliest instant at which
           the local maximum queue size occurs within the current busy period.
           Let $(s_k, t_k), (s_{k+1}, t_{k+1}), \ldots, (s_m, t_m)$ be the rate segments contained
           within the interval $(\alpha_j, \tau_j)$. Let $(s_p, t_p), (s_{p+1}, t_{p+1}), \ldots, (s_q, t_q)$ be
           the rate segments contained within the interval $(\alpha_l, \tau_l)$. */

        /* compute total arrivals in busy period $l$ up to time $\tau_l$ */
1.      $W_l \leftarrow \sum_{i=p}^{q} r_i(t_i - s_i)$;

2.      $i \leftarrow k$;   /* start with first rate segment */
3.      $W \leftarrow 0$;   /* $W$ denotes the arrivals during current busy period */
4.      $\rho_{min} \leftarrow r_k$;
        /* check if global maximum is in the first rate segment of current busy period */
5.      **If** $(j = l)$ **and** $(q = p)$     /* maximum cannot move */
 6.1        **return**($r_{max}$);
7.      **endif**
        /* Compute total arrivals at the end of current rate segment */
8.      $W \leftarrow W + (t_i - s_i) r_i$;
        /* Compute the rate at which queue size at $t_i$ becomes equal to that at time $\tau_l$ */
9.      $\rho \leftarrow \frac{W_l - W}{(\tau_l - s_p) - (t_i - s_k)}$;
10.     **If** $(\rho < \rho_{min})$
 10.1       $\rho_{min} \leftarrow \rho$;
11.     **endif**
12.     $i \leftarrow i + 1$;
13.     **If** $(i \leq m)$ **goto** step 5;
14.     **return**($\rho_{min}$);
```

Figure B.5: Pseudocode for the computation of $\rho_{j,2}$.

C Pseudocode for the approximate algorithms

```
PSEUDOCODE OF STEP 3 OF THE APPROXIMATE BURSTINESS
CURVE ALGORITHM FOR ELEMENTARY VIDEO STREAMS

compute_next_rate($\rho$)

        /* Given current rate $\rho$ and maximum busy period $l$, determines the next
           higher rate $\rho'$ at which the slope of the burstiness curve may change */

1.      $\rho' \leftarrow r$;
2.      Determine the lowest rate $\rho_{l,1}$ at which the busy period $l$ breaks (Figure A.5);
3.      If ($\rho_{l,1} < \rho'$)
 3.1        $\rho' \leftarrow \rho_{l,1}$;
4.      endif
        /* Now determine the rates at which the global maximum moves */
5.      Determine the lowest rate $\rho_{l,2}$ at which the position of the global maximum
        queue size changes (Figure A.6);
6.      If ($\rho_{l,2} < \rho'$)
 6.1        $\rho' \leftarrow \rho_{l,2}$;
7.      endif
8.      return($\rho'$);
```

Figure C.1: Pseudocode of step 3 of the approximate algorithm that computes the burstiness curve of an elementary video stream.

```
PSEUDOCODE OF STEP 3 OF THE APPROXIMATE BURSTINESS
CURVE ALGORITHM FOR MPEG-2 TRANSPORT STREAMS
compute_next_rate($\rho$)

        /* Given current rate $\rho$ and maximum busy period $l$, determines the next
           higher rate $\rho'$ at which the slope of the burstiness curve may change */

1.      $\rho' \leftarrow r_{max}$;
2.      Determine the lowest rate $\rho_{l,1}$ at which the busy period $l$ breaks (Figure B.4);
3.      If ($\rho_{l,1} < \rho'$)
 3.1        $\rho' \leftarrow \rho_{l,1}$;
4.      endif
        /* Now determine the rates at which the global maximum moves */
5.      Determine the lowest rate $\rho_{l,2}$ at which either the position of the global
        maximum queue size changes, or the starting time of busy period $l$ moves
        to a later time instant (Figure B.5);
6.      If ($\rho_{l,2} < \rho'$)
 6.1        $\rho' \leftarrow \rho_{l,2}$;
7.      endif
8.      return($\rho'$);
```

Figure C.2: Pseudocode of step 3 of the approximate algorithm that computes the burstiness curve of an MPEG-2 Transport Stream.