Dynamic Control of the Threshold Policy for MPEG Video Streaming

Christos Panayiotou*, Vasos Vassiliou†, Andreas Pitsillides‡

Abstract

This work investigates techniques for providing statistical quality of service (QoS) guarantees to MPEG video streams and this paper summarizes our work-in-progress. Motivated by the differentiated services approach, we assume that video sources mark I frames as high priority packets (green packets) and P or B frames as lower priority packets (yellow packets). For the work described in this paper, we assume that packets go through a threshold policy which provides preferential treatment to high priority packets. The objective of this work is to dynamically control the parameters of the threshold policy so that certain QoS requirements are met. Towards this end, we investigate various measurement-based techniques from the Perturbation Analysis (PA) framework. The main advantage of such techniques is that they often are non-parametric in the sense that they do not require any knowledge of the underlying stochastic processes that drive the system dynamics.

1 Introduction

The main objective of this work is to investigate various techniques that can be used to provide statistical quality of service (QoS) guarantees to MPEG video streams [1, 2] and compare their performance. This paper summarizes our work-in-progress for a class of measurement-based techniques from the Perturbation Analysis (PA) literature; PA techniques allow us to extract additional information e.g., gradients with respect to a control parameter B, by *simply observing a single sample path of the system* (see [3] and references therein for a detailed analysis of PA techniques).

Motivated by the differentiated services approach [4], we assume that video sources mark I frames as high priority packets (green packets) and P or B frames as lower priority packets (yellow packets); we restrict our analysis in only two classes to improve the exposition but, we point out that the approaches presented here can be extended to multiple classes. For the work presented in this paper, we assume that in the network, packets go through a threshold policy as described in Section 2. The objective of this part of the work is to dynamically control a buffer threshold B (not to be confused with B-frames) so that certain QoS requirements are met. Towards this end, we compare two types of PA techniques. The first is based on a discrete-event model and finite differences. In this approach we investigate the Augmented System Analysis (ASA) and the Modified Lindley Recursion (MLR) as described in [5]. These algorithms allow us to *predict* the network's QoS performance under *any* buffer threshold value B by simply observing the network while operating under some threshold B_0. Furthermore, MLR allows as to predict the system behavior as more sources are admitted. The second approach is based on a Stochastic Fluid approximate Model (SFM). Using this model, it is possible to determine the gradient of some QoS metric of interest with respect to B and then use a stochastic

*C. Panayiotou is with the Dept of Electrical and Computer Engineering, University of Cyprus, Cyprus Email: christosp@ucy.ac.cy

†V. Vassiliou is with the Dept of Computer Science, University of Cyprus, Cyprus Email: vasosv@ucy.ac.cy

‡A. Pitsillides is with the Dept of Computer Science, University of Cyprus, Cyprus Email: andreas.pitsillides@ucy.ac.cy

approximation based algorithm to continually adjust B in order to maintain near optimal performance, as in [6] for two classes or [7] for multiple classes. The results in [5] and [6] are encouraging; they indicate that PA type algorithms are fairly easy to implement, can be evaluated on line and provide accurate performance and/or gradient estimates. This work is a simulation study that compares the performance of the various algorithms for real-time applications such as MPEG video streaming.

The rest of the paper is organized as follows. In Section 2 we describe the threshold policy, present the system model and introduce the QoS metrics of interest. In Section 3 we present the Augmented System Analysis (ASA) algorithm and in Section 4 we present the modified Lindley recursion algorithm. Both of these algorithms can predict the QoS performance of the network under any threshold B. In Section 5 we present an algorithm for estimating the gradient of the QoS performance with respect to the threshold B and then use it together with a stochastic approximation algorithm. In Section 6 we present some preliminary simulation results and conclude with Section 7 where we also describe our future research plans.

2 Threshold Policy and System Model

For video streaming applications it only makes sense to consider policies that preserve the order with which packets are received. Therefore, the model we consider consists of a single server First-In-First-Out (FIFO) queue with a threshold B, as shown in Fig. 1. For simplicity, we assume that only two packet priorities are available, green for I frames and yellow for B or P frames (however, we point out that inclusion of more classes is possible).

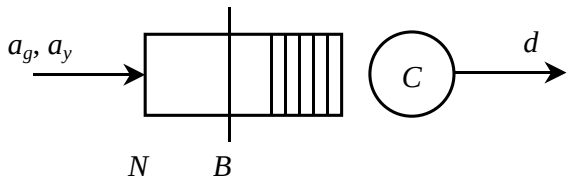

Figure 1: Buffering policy based on a discrete-event model

The threshold policy works as follows. If a green packet arrives, it is accommodated in the buffer of size N, if the buffer queue length $X(t) < N$, otherwise it is dropped. If a yellow packet arrives, then it is accepted in the buffer if $X(t) < B \leq N$, otherwise it is dropped. This policy is selected since [8] has showed that among a number of FIFO policies, this ones better protects the higher priority packets and it is very easy to implement.

In this model, there are three possible events; an arrival of a green packet a_g, an arrival of a yellow packet a_y, and a departure of either green or yellow packet d. Note that there is no distinction in the transmission requirements of different color packets. The state $X(t) \in \{0, 1, \cdots, N\}$, and the state transition diagram of the system is shown in Fig. 2.

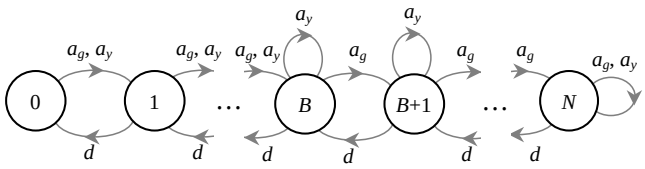

Figure 2: State transition diagram

2.1 Stochastic Fluid Model

Another modeling framework which is often used in the literature is based on Stochastic Fluid Models (SFM). SFMs have recently been shown to be especially useful for the simulation and control of various kinds of high-speed networks [9, 10, 11, 12]. Such models are often inaccurate when it comes to performance evaluation, however, people have shown that such models can be used effectively for optimization purposes (see [6] and references therein). In [6] the authors consider the SFM shown in Fig. 3 which can be viewed as equivalent to the threshold policy. The green and yellow packet streams are represented by two fluid flows $\alpha_g(t)$ and $\alpha_y(t)$ while the server processes fluid at a rate $c(t)$. All random processes $\alpha_g(t)$, $\alpha_y(t)$ and $c(t)$ are assumed piecewise analytic. Given these defining processes, we derive the system state $x(t)$ and the overflow process $\gamma_g(t)$ and $\gamma_y(t)$ for green and yellow flows respectively. In contrast to the discrete event model where $X(T) \in \mathbb{Z}_+$, for the SFM $x(t) \in \mathbb{R}_+$ and the system dynamics are given by

$$\frac{dx(t)}{dt^+} = \begin{cases} 0 & \text{if } x(t) = 0 \text{ and } A(t) < 0 \\ 0 & \text{if } x(t) = b,\ A(t) > 0 \text{ and } \alpha_g(t) < c(t) \\ 0 & \text{if } x(t) = N \text{ and } A(t) > 0 \\ \alpha_g(t) - c(t) & \text{if } b < x(t) < N \\ A(t) & \text{otherwise.} \end{cases} \tag{1}$$

where $A(t) = \alpha_g(t) + \alpha_y(t) - c(t)$. Also, the overflow processes are given by

$$\gamma_g(t) = \begin{cases} \alpha_g(t) - c(t) & \text{if } x(t) = N \\ 0 & \text{otherwise} \end{cases} \qquad \gamma_y(t) = \begin{cases} \alpha_y(t) & \text{if } x(t) > b \\ A(t) & \text{if } x(t) = b \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Note that from (1), during $x(t) = b$ intervals, yellow fluid equal to $c(t) - \alpha_g(t) > 0$ is admitted while, the remaining $\alpha_y(t) - (c(t) - \alpha_g(t)) = A(t)$ is dropped. We also point out that $b \in \mathbb{R}_+$ while $B \in \mathbb{Z}_+$. The reasons why the threshold $b \in \mathbb{R}_+$ will be made clearer later on when we will define sensitivities with respect to b.

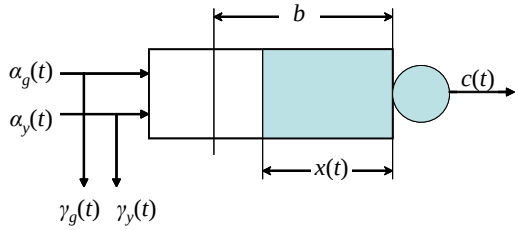

Figure 3: Buffering policy based on a SFM.

At this point we need to emphasize that the SFM model is only used to derive algorithms for determining the optimal threshold B. Once the algorithms are derived, then they are evaluated based on information observed on the *actual* system. Even though it is expected that this approach will introduce some bias, results from [6, 7] indicated that this approach still converges to an optimal or near optimal threshold and thus we are encouraged to use it for this problem. We also note that part of this work involves the investigation of ways for obtaining the minimum necessary information from the real system without affecting converge.

2.2 QoS Objectives

The QoS metrics we consider are the packet loss volume of green and yellow packets $L_T^g(B)$ and $L_T^y(B)$ respectively (number of packets dropped), and the packet delay $Q_T(B)$, over the interval $[0, T]$. These

are defined as follows

$$L_T^q(B) = \sum_{k=1}^{M_T} \mathbf{1}[k\text{th packet dropped}], \quad q \in \{g, y\} \tag{3}$$

$$Q_T(B) = \sum_{k=1}^{M_T} (D_k - A_k) \tag{4}$$

where M_T is the random number of packet arrivals in the interval $[0, T]$, A_k and D_k are the arrival and departure times of the kth packet and $\mathbf{1}[\cdot]$ is the usual indicator function. Note that using these metrics one can obtain additional measures such as the packet loss rate or loss probability.

Equivalently, for the SFM we define the loss volume

$$l_T^q(b) = \int_0^T \gamma_q(t)dt, \ q \in \{g, y\} \tag{5}$$

Note that it is difficult to measure the delay of each particle of the fluid. So, we define the workload

$$q(b) = \int_0^T x(t)dt \tag{6}$$

and point out that one can obtain the delay metric through Little's Law. Furthermore, it is worth while to point out that all quantities defined above are generally random variables and of course one would be interested in their expected value.

In the sequel, we will assume that the buffer size N is fixed and it is required to determine the appropriate value for the threshold parameter B. The main objective is to maintain low drop probability for all green packets. This can be achieved by setting B to a small value. On the other hand, setting B to a very small value, implies that several yellow packets will be lost unnecessarily since they could have been accepted without affecting either the drop probability of green packets or the average delay of either class. Therefore, it is desirable to determine the threshold B such that the drop probability of green packets is always satisfied while the yellow drop probability is minimized. Next, we present the algorithms we study for finding the optimal B. Sections 3 and 4 present algorithms for predicting the performance under *any* threshold. Section 5 presents a gradient descent based algorithm.

3 Augmented System Analysis Algorithm

In this section we present the Augmented System Analysis (ASA) algorithm to be used in the study. For detailed description of the ASA algorithm and its conditions refer to [13, 5]. Table 1 shows the ASA algorithm for constructing the sample path under *any* $B_i \neq B_0$, where B_0 is the threshold for accepting yellow packets in the nominal or observed sample path. In other words, B_0 is the threshold which is used on the actual system. In the algorithm, X_k denotes the state of the system that operates with a threshold equal to B_0 and X'_l denotes the *predicted* system state in the constructed sample path, i.e., the system that operates with a threshold equal to some $B \neq B_0$. Also, t_k corresponds to the occurrence time of the kth event in the observed sample path, while t'_l corresponds to the occurrence time of the lth event in the constructed sample path where in general $l \neq k$.

Remark: The algorithm in Table 1 should be implemented several times, once for each value of the threshold B.

Table 1: ASA Algorithm

1. INITIALIZE: $k = 0$, $l = 0$, $\tau := 0$, $\mathcal{M} := Active$ and $X'_l := X_k$.
2. If kth event is packet arrival at time t_k
   - 2.1. If $\mathcal{M} = Active$ Update state:
     If green packet $X'_{l+1} := \min\{X'_l + 1, N\}$
     If yellow packet $X'_{l+1} := \min\{X'_l + 1, B_i\}$
     $t'_{l+1} = t_k - \tau, \quad l \leftarrow l + 1$
   - 2.2. If $\mathcal{M} = Inactive$
     set: $\mathcal{M} := Active$, $\tau := \tau + (t_k - \hat{t})$
3. If kth event is packet departure at time t_k
   - 3.1. Update state:
     If $X'_l > 0$,
     $X'_{l+1} := X'_l - 1, \quad t'_{l+1} = t_k - \tau, \quad l \leftarrow l + 1$
   - 3.2. If $X_k = 0$ and $X'_l > 0$
     $\hat{t} := t_k, \quad$ set $\mathcal{M} := Inactive$

4 Modified Lindley Recursion Algorithm

In this section we describe another algorithm for predicting the system's performance under any B. This algorithm uses the Lindley recursion [14]

$$D_k = \max\{A_k, D_{k-1}\} + S_k \tag{7}$$

where A_k and D_k denote the arrival and departure times of the kth packet respectively and S_k corresponds to the time it takes to transmit a packet. In general, packet headers contain information about the packet size. Given that the transmission rate of the output link is known, it is easy to determine the time that it will take to transmit the kth packet, i.e., it is easy to determine its service time S_k. Since this information is available at the time of the kth packet arrival A_k, it can be used in (7) to determine the packet's anticipated departure time.

Recall that our objective is to construct the sample paths under $B_1, B_2, \cdots, B_M$ given an observed sample path under B_0 where B_i corresponds to the threshold for yellow packets. For any transmission policy that does not involve packet retransmissions (i.e., this excludes TCP) the arrival event is always feasible and independent of the parameter B, thus the arrival of the kth packet in the observed and constructed sample paths will *always* occur at exactly the same time instance A_k. The only difference between the two sample paths is in the departure times of some packets because the two sample paths exhibit different packet drop behavior.

Dropped packets satisfy the following condition:

$$\begin{array}{ll} X_i(A_k) \geq N & \text{if } k \text{ is a green packet} \\ X_i(A_k) \geq B_i & \text{if } k \text{ is a yellow packet} \end{array} \tag{8}$$

where $X_i(t)$ is the number of packets in the queue at time t under parameter B_i. Using event times, we make the crucial observation that (8) are equivalent to the following conditions:

$$\begin{array}{lcll} X_i(A_k) \geq N & \Leftrightarrow & A_k \leq D_{k-N-l_i} & \text{if } k \text{ is a green packet} \\ X_i(A_k) \geq B_i & \Leftrightarrow & A_k \leq D_{k-B_i-l_i} & \text{if } k \text{ is a yellow packet} \end{array} \tag{9}$$

where l_i is the number of lost (dropped) packets under parameter B_i. In (9), A_k is the same for all sample paths under any B_i, $i = 0, 1, \cdots, M$. The only necessary information then is to determine the departure times D_{k-N-l_i} and $D_{k-B_i-l_i}$ for all sample paths under any B_i. This can be easily obtained through the Lindley recursion if we modify the packet indices as follows (thus the term Modified Lindley Recursion (MLR) construction algorithm). For every constructed sample path under some B_i, we define the index process

$$a_i(k) = k - l_i, \quad i = 0, \cdots, M \tag{10}$$

which corresponds to a sequential indexing of *only* packets that have been accommodated in the queue. Such a sequence is defined for every sample path under any B_i. Using the indexing scheme of (10) we can determine the departure time of each accepted packet using the Lindley recursion

$$D_{a_i(k)} = \max\left\{D_{a_i(k)-1}, A_{a_i(k)}\right\} + S_{a_i(k)} \tag{11}$$

As a result, we can construct any sample path under B_i, $i = 1, \cdots, M$, using the algorithm presented in Table 2. In this algorithm we assume that $D_j = 0$ for all $j \leq 0$. Furthermore, s_k is the size of the kth packet and C is the transmission rate allocated to the specific buffer. From an implementation standpoint the only information required to construct the desired sample path is the departure time of the last N packets that have been accepted in the queue and this can be saved in an N-dimensional array.

Finally, it is important to point out that, unlike ASA, the ML construction algorithm presented in Table 2 does not make *any* assumptions on the distributions of the event lifetimes. These can be derived from any arbitrary distributions.

Table 2: Modified Lindley Recursion (MLR) construction algorithm

1. INITIALIZE: $k := 1$, $a_i(k) := 0$, $l_i := 0$, $i = 1, \cdots, M$
2. When packet k arrives
   - 2.1. IF color green packet GOTO *2.1.1* ELSE GOTO *2.1.2*
   - 2.1.1. IF $A_k \leq D_{a_i(k)-1-N}$, $l_i \leftarrow l_i + 1$, GOTO *4* `% packet dropped`
   - 2.1.2. IF $A_k \leq D_{a_i(k)-1-B_i}$, THEN $l_i \leftarrow l_i + 1$, GOTO *4* `% packet dropped`
3. $S_{a_i(k)} = \frac{s_k}{C}$ `% packet accepted`
   $D_{a_i(k)} = \max\left\{D_{a_i(k)-1}, A_{a_i(k)}\right\} + S_{a_i(k)}$
4. END

Remark: One can also use the MLR algorithm to predict the sample paths under any value of B for systems with different number of sources. This requires some assumptions on the distributions of the packet arrivals and is a topic of future research. The interested reader is referred to [5] for some preliminary results.

5 Gradient-Based Optimization

In this section we present a different approach for determining the threshold B which is based on the SFM introduced earlier and stochastic approximation. For the purpose of *performance analysis* with QoS requirements, the accuracy of SFMs depends on traffic conditions, the structure of the

underlying system, and the nature of the performance metrics of interest. For the purpose of *control and optimization*, on the other hand, as long as a SFM captures the salient features of the underlying real system, it is possible to obtain accurate solutions to problems even if we cannot estimate the corresponding performance with accuracy. In short, an SFM may be too "crude" for some performance analysis purposes, but able to accurately capture sensitivity information. This point of view is taken in various references including [6, 12] and reference therein. In this section we summarize the gradient estimation algorithm of [6] but first we introduce some notation. We define an objective function, for example

$$J(b) = \mathbb{E}\left[q_T(b)\right] + w_g \mathbb{E}\left[l_T^g(b)\right] + w_y \mathbb{E}\left[l_T^y(b)\right] \tag{12}$$

where w_g and w_y are appropriate weight parameters and we are interested in finding b that minimizes it (alternatively one can also introduce some constraints and turn the problem into constraint minimization). Suppose that it is possible to obtain estimates of the sensitivity $dJ(b)/db$ then, one can use a stochastic approximation algorithm to update the threshold b as follows

$$b_{n+1} = b_n - \sigma_n \hat{H}_T(b) \tag{13}$$

where σ_n is an appropriate step size sequence and $\hat{H}_T(b)$ is the estimate of the sensitivity $dJ(b)/db$ obtained over the interval $[0, T]$. Such an approach converges to the optimal threshold as long as the sensitivity estimates are unbiased. The authors in [6, 7] have derived algorithms for obtaining such sensitivity estimates which, if evaluated based on a fluid sample path, are unbiased. In this work we plan to use these sensitivity estimators but evaluate them based on information collected from the actual system (discrete-event based). Evaluating these sensitivities based on information from a discrete-event sample path may introduce bias, however, some preliminary simulation results indicate that such estimates can be used in (13) and still converge to a near optimal threshold. Below, we present the algorithms for obtaining these sensitivities as were derived in [6] where it is assumed that the buffer has infinite capacity ($N = \infty$) and thus the green fluid experiences no loss ($l_T^g(b) = 0$). Algorithms for finite N are available but for the MPEG video streaming we investigate in this work, we expect loss of I frames to be very close to zero.

To describe the algorithm for obtaining the sensitivity $dJ(b)/db$ we define the following quantities. In the interval $[0, T]$, we observe K busy periods[1] denoted by $\mathcal{B}_k$, $k = 1, \ldots, K$, in increasing order. A busy period is further subdivided into intervals $p_{k,i} = [v_{k,i}, v_{k,i+1})$, $k = 0, \cdots, S_k$, where the points $v_{k,0}$ and v_{k,S_k+1} indicate the start and end points of $\mathcal{B}_k$ and the remaining $v_{k,i}$ correspond to points where $x(t)$ either becomes equal to b or stops being equal to b. S_k is the number of times that $x(t)$ either becomes or ceases to be equal to b. Depending on value of $x(t)$, $t \in p_i$ the intervals are classified in three disjoint index sets: *Partial Loss Period Set*: $U_k = \{i \ : \ x(t) = b, t \in p_{k,i}\}$, *Full Loss Period Set*: $V_k = \{i \ : \ x(t) > b, t \in p_{k,i}\}$ and *No Loss Period Set*: $W_k = \{i \ : \ x(t) < b, t \in p_{k,i}\}$. Note that loss only refers to yellow fluid since we assume that $N = \infty$. Also, define the quantities

$$y_{k,i} \equiv \alpha_g(v_{k,i}) + \alpha_y(v_{k,i}) - c(v_{k,i}) \tag{14}$$

$$z_{k,i} \equiv \alpha_g(v_{k,i}) - c(v_{k,i}) \tag{15}$$

These correspond to the arrival and service rates at specific points in time (start and end of a busy period and points where $x(t)$ becomes or stops being equal to b). For the purposes of this paper, we assume that these are known, however we are also investigating various estimators that we can use to estimate these rates. Finally, we define the set of all busy periods that do not contain any partial loss period

$$\Phi = \{k \in \{1, \ldots, K\} \ : \ U_k \neq \emptyset\} \tag{16}$$

[1] A busy period is a maximal interval where $x(t) > 0$

At this point we are ready to present the algorithm for evaluating the derivatives of the yellow fluid loss dl_T^y/db and workload dq_T/db with respect to the threshold parameter b.

$$\frac{dl_T^y}{db} = -\sum_{k=1}^{K} \mathbf{1}[S_k > 0] + \sum_{k=1}^{K} \mathbf{1}[k \notin \Phi] \prod_{i=2,4,\cdots,S_k} \frac{y_{k,i}}{z_{k,i}} \frac{z_{k,i-1}}{y_{k,i-1}} \tag{17}$$

where K is the random number of busy periods in $[0, T]$. The first sum, simply adds all busy periods that experience some loss while the second sum corresponds to certain "correction" terms. These terms require knowledge of the instantaneous arrival and service rates. We are investigating possible ways of evaluating them including the possibility of ignoring them all together; in [6] it is also shown that each fraction is less than one, thus the product might become small enough so that it can be ignored. In such case, the estimate of $\frac{dl_T^y}{db}$ will simply be the number of busy periods where at least a yellow packet is dropped. The workload sensitivity with respect to b is given by

$$\frac{dq_T}{db} = \sum_{k=1}^{K} \left\{ (v_{k,S_k+1} - v_{k,m_k}) + \sum_{i=1}^{m_k-1} (v_{k,i+1} - v_{k_i})\phi_{k,i} \right\} \tag{18}$$

where m_k is the first partial loss period in the $\mathcal{B}_k$, $k = 1, \cdots, K$. Also,

$$\phi_{k,i} = \begin{cases} 1 - z_{k,i} v'_{k,i}, & i \text{ odd} \\ 1 - y_{k,i} v'_{k,i}, & i \text{ even} \end{cases} \tag{19}$$

$$z_{k,1} v'_{k,1} = \frac{z_{k,1}}{y_{k,1}} \tag{20}$$

$$y_{k,2n} v'_{k,2n} = \prod_{i=2,..,2n} \frac{y_{k,i}}{z_{k,i}} \cdot \frac{z_{k,i-1}}{y_{k,i-1}}, \; n > 0 \tag{21}$$

$$z_{k,2n+1} v'_{k,2n+1} = \frac{z_{k,2n+1}}{y_{k,2n+1}} \left\{ \prod_{i=2,..,2n} \frac{y_{k,i}}{z_{k,i}} \cdot \frac{z_{k,i-1}}{y_{k,i-1}} \right\}, \; n > 0 \tag{22}$$

The first term in (18) is simply the interval from the beginning of the first partial loss period v_{k,m_k} until the end of the kth busy period v_{k,S_k+1}. The second term is a little more complicated, consisting of the product of some fractions of rates. As already mentioned, for this paper we assume that these rates are know but we are investigating ways estimating them from observed data and also explore the possibility of ignoring them all together.

6 Preliminary Simulation Results

Currently we are implementing the control algorithms presented in the previous sections on network simulator (ns) [15]. Here we present some preliminary results obtained through custom made simulators that demonstrate the feasibility of using these algorithms.

First we compare the ASA (Table 1) and ML (Table 2) sample path construction algorithms with brute force simulation to determine the quality of their estimates under different operating conditions. We assume a simple one node network with five input sources. Each one transmits packets according to an ON/OFF Markov Modulated Poisson Process (MMPP) During the ON period, each source transmits at a rate 400 packets per second. The length of the ON and OFF periods are both exponentially distributed with mean 0.1 seconds and the size of each packet is exponentially distributed with mean 500 bytes. Each packet is marked as green with probability 0.5 or as yellow with probability 0.5. The transmission capacity of the node is 500Kbytes per second.

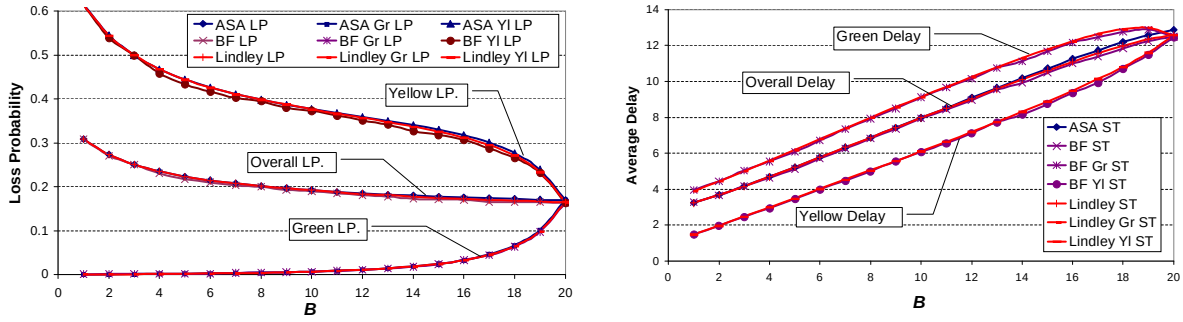

Figure 4: Loss probability and average packet delay vs threshold B

Fig. 4 shows the expected loss probability and average packet delay for the above scenario. For the brute force curve (BF), each point corresponds to a different simulation run. On the other hand, the ASA and Lindley curves are obtained from a *single* simulation run where we assumed that the network was operating with a threshold $B_0 = 10$. As indicated by these simulation results, both algorithms are able to predict the system performance under *any* value of the threshold B; it is worth while mentioning that the network under study violates some of the assumptions of the ASA algorithm, yet ASA still delivers fairly accurate results. In [5] it is also shown that the computation requirements of ASA and MLR are very minimal.

Next, we present some preliminary results for the gradient descent approach. Here we return to the cost minimization problem of (12) where we are trading off the expected loss rate of green and yellow packets with packet rejection penalties w_g and w_y respectively, for the expected queue length (as already mentioned, using Little's law one can obtain the average packet delay). For illustrative purposes, we assume that no green packets will be dropped and that $w_y = 25$. Furthermore, we assume that $\alpha_g(t)$ is piecewise constant with values uniformly distributed over $[0, 12]$, with each constant rate period exponentially distributed with parameter 0.5, $\alpha_y(t)$ is piecewise constant with values uniformly distributed over $[0, 27]$, with each constant rate period exponentially distributed with parameter 0.3, and the service rate is $c(t) = 20$. We also assume the traffic and service rates can be accurately measured and use an estimation period $T = 200K$ time units. In other words, we estimate the cost sensitivity for a period T and then update the threshold using (13) (experiments with significantly shorter observation interval exhibit very similar behavior). In Fig. 5 the cost curve (labelled "cost") is obtained through brute force simulation and it represents an average over 30 sample paths, each T time units long. Next we initialize the network with $B_0 = 2$ and run the optimization algorithm (13). As seen in the figure, the algorithm converges to the optimal cost within a few iterations.

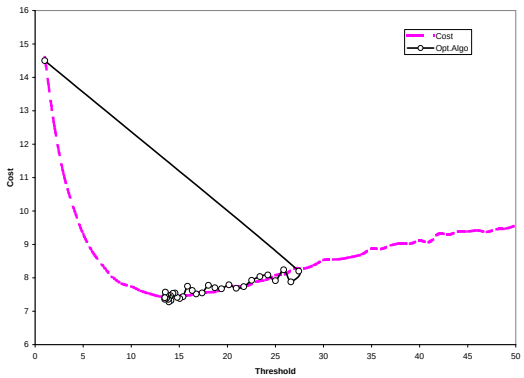

Figure 5: Optimal threshold determination using SFM-based gradient estimators

7 Conclusions and Future Work

This paper is a report on our work-in-progress for MPEG video streaming. We adopt the threshold policy and use it to differentiate between the different type of frames transmitted through the network. Here we describe a number of measurement based control algorithms for adjusting the threshold of the policy and present some preliminary simulation results. These results indicate that for aggregated traffic streams both ASA and MLR can accurately predict the QoS metrics of interest and both have fairly low computational requirements. In the future, we plan to simulate true video streaming applications using ns2 and compare all algorithms. We will report on the convergence properties of each algorithm (number of iterations and length of observation interval) as well as on their computational requirements. Furthermore, we will compare the performance of the dynamic threshold policy against other differentiated services approaches such as active queue management (e.g., see [16]).

References

[1] T. Ahmed, A. Mehaoua, and G. Buridant, "Implementing MPEG-4 video on demand over IP differentiated services," in *Proceedings of IEEE GLOBECOM*, vol. 4, pp. 2489 – 2493, 2001.

[2] Y. Koucheryavy, D. Moltchanov, and J. Harju, "An analytical evaluation of VoD traffic treatment within the EF-enabled diffserv ingress and interior nodes," in *Proceedings of ICT*, pp. 1458 – 1464, Mar. 2003.

[3] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[4] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," *RFC 2475*, Dec 1998. `http://www.ietf.org/rfc/rfc2475.txt`.

[5] C. Panayiotou and C. Cassandras, "On-line predictive techniques for "differentiated services" networks," in *Proceedings IEEE Conference on Decision and Control*, pp. 4529–4534, Dec 2001.

[6] C. Cassandras, G. Sun, C. Panayiotou, and Y. Wardi, "Perturbation analysis and control of two-class stochastic fluid models for communication networks," *IEEE Transactions on Automatic Control*, vol. 48, pp. 770–782, May 2003.

[7] G. Sun, C. Cassandras, and C. Panayiotou, "Perturbation analysis of multiclass stochastic fluid models," *Journal of Discrete Event Dynamic Systems*, vol. 14, no. 3, pp. 267–307, 2004.

[8] I. Cidon, R. Guérin, and A. Khamisy, "On protective buffer policies," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 240–246, Jun 1994.

[9] G. Kesidis, A. Singh, D. Cheung, and W. Kwok, "Feasibility of fluid-driven simulation for ATM network," in *Proc. IEEE Globecom*, vol. 3, pp. 2013–2017, 1996.

[10] K. Kumaran and D. Mitra, "Performance and fluid simulations of a novel shared buffer management system," in *Proceedings of IEEE INFOCOM*, March 1998.

[11] A. Yan and W. Gong, "Fluid simulation for high-speed networks with flow-based routing," *IEEE Transactions on Information Theory*, vol. 45, pp. 1588–1599, 1999.

[12] A. Pitsillides, P. Ioannou, M. Lestas, and L. Rossides, "Adaptive non-linear congestion controller for a differentiated-services framework," *IEEE/ACM Transactions of Networking*, 2005. To Appear.

[13] C. Cassandras and S. Strickland, "Observable augmented systems for sensitivity analysis of Markov and semi-Markov processes," *IEEE Transactions on Automatic Control*, vol. 34, pp. 1026–1037, 1989.

[14] D. V. Lindley, "The theory of queues with a single server," in *Proceedings of Cambridge Philosophical Society*, vol. 48, pp. 277–289, 1952.

[15] *Network Simulator (NS)*. UCB/LBNL/VINT. `http://www.isi.edu/nsnam/ns/`.

[16] C. Chrysostomou, A. Pitsillides, L. Rossides, and A. Sekercioglu, "Fuzy logic controlled RED: Congestion control in TCP/IP differentiated services networks," *Special Issue on "The Management of Uncertainty in Computing Applications" in Soft Computing Journal - A Fusion of Foundations, Methodologies and Applications*, vol. 8, pp. 79–92, 2003.