**Proceedings of the 40th IEEE
Conference on Decision and Control
Orlando, Florida USA, December 2001**

FrM12-3

On-Line Predictive Techniques for "Differentiated Services" Networks[1]

Christos G. Panayiotou and Christos G. Cassandras
Department of Manufacturing Engineering
Boston University, Boston, MA 02215
panayiot@bu.edu, cgc@bu.edu

Abstract

In this paper we propose two on-line predictive algorithms that can provide performance estimates under a set of parameters in the context of differentiated services. These algorithms are based on sample path constructability techniques. The first one (ASA) depends on exponential lifetime distributions, while the second one (ML) is distribution independent. The information provided by these algorithms can be used to solve the admission control problem and to allocate resources to the various traffic streams.

1 Introduction

The introduction of several new Internet applications such as Internet telephony and video-conferencing require networks to be capable of handling a variety of services beyond the current "best effort". The new bandwidth and buffer-hungry applications have Quality of Service (QoS) requirements such as short delay, low packet drop probability and small delay jitter. In order to meet such requirements it is necessary that certain resources such as transmission capacity and buffer space are reserved for each stream according to its QoS requirements. Performing resource reservation for each individual stream is very complex however, and exhibits high resource management cost.

An alternative approach, which is scalable with respect to the number of flows supported, is to *aggregate* microflows with similar QoS requirements in a single stream and perform the resource management functions for the single macro-stream. This is the main objective of *Differentiated Services* (DS) [1]. In a DS model, packets are classified and marked to receive a particular per-hop forwarding behavior on nodes along their path. Sophisticated classification, marking, policing, and shaping operations need *only* be implemented at the network boundaries.

Currently there are two proposals that define the per-hop forwarding behavior (PHB) of packets. These are the Expedited Forwarding (EF) [2] and the Assured Forwarding (AF) [3]. EF defines a PHB behaviour for packets with stringent QoS requirements that is similar to a leased line. AF defines a PHB behaviour for N classes of traffic $AF_1, \cdots, AF_N$, where each class has its own delay and jitter QoS requirements. Furthermore, each AF class is subdivided into M subclasses $AF_{i1}, \cdots, AF_{iM}$, $i = 1, \cdots, N$, where each subclass corresponds to a different packet drop priority. The current AF proposal calls for $N = 4$ and $M = 3$ (see [3]). The three drop priorities are called *green*, *yellow* and *red*, where green has the lowest drop priority while red has the highest. There are two proposals on how these priorities are assigned to incoming packets referred to as single and two-rate, three-color markers [4, 5].

At an ingress node, a new connection will request an AF class based on its delay and jitter requirements and possibly based on the price of each class. In addition, it will request a certain rate and maximum burst (e.g., advertise its leaky bucket parameters). Based on this information, the network must decide whether to accept the new connection (admission control problem [6]). The new connection will be accepted if its acceptance does not imply violation of the QoS requirements of the existing connections. In other words, the network must be able to predict the resulting QoS if the new connection is accepted. Once the new connection is assigned to an AF class (say AF_i), it may be necessary to reallocate resources such that the promised QoS is delivered to all streams.

In this paper, we develop techniques for *predicting* the effect of extra connections on the QoS delivered by the network. In addition, these techniques predict the QoS delivered under different resource (e.g., buffer) allocations, and are measurement-based, therefore they can be used to fine-tune the network's performance on-line. Specifically, the schemes we develop in this paper fall under the class of sample path constructability techniques [7, 8]. These techniques observe the sample path

[1]Supported in part by the National Science Foundation under grants EEC-0088073 and ACI-98-73339, by AFOSR under contract F49620-98-1-0387, by the Air Force Research Laboratory under contract F30602-99-C-0057, by EPRI/ARO under contract WO8333-03 and by Nokia Research Center (Boston).

of a real system under a number of connections (say Q) and under a buffer allocation (say $\mathbf{r}$). Using information from the observed sample path, they construct the sample paths under a different set of parameters (say $Q+1$ connections and buffer allocation $\mathbf{r}'$). Subsequently, using the constructed sample paths, it is easy to determine the network's delivered QoS and therefore the network can use these estimates to solve the *admission control* as well as the *resource allocation* problems.

A dynamic approach for solving the buffer allocation problem in the context of DS is described in [9]. In their approach, the authors use a stochastic approximation scheme based on the observed traffic. A classic problem associated with such techniques is the determination of the "right" step size which is generally a difficult problem. In this paper, we develop a scheme for dynamically adjusting the buffer thresholds that, not only does not require a step size, but it can also determine the best buffer allocation in a single step. This constitutes the first contribution of the paper. In addition, our scheme can be used to solve the admission control problem and make the necessary resource reallocations as sources connect/disconnect from the network.

2 Buffering Policy

One of the requirements in the DS specifications is that same-flow packets are not delivered out-of-order. As a result, when the color marker assigns different colors to same-flow packets, e.g., due to transmission rate violations, if they all make it to the destination, they must be in their original order. Therefore, a First In First Out (FIFO) buffering policy, irrespective of color, is appropriate (see Fig. 1). For FIFO policies, Cidon et. al. [10] have investigated 3 packet accepting policies for preserving the drop requirements of the premium class (green packets). These are *pushout policy*, *limited red policy*, and *threshold policy*. In the analysis, [10] shows that *only* the threshold policy[1] is M-protective implying that only this policy can shield the performance of green traffic from low priority traffic. Furthermore, the threshold polity is the simplest to implement since it only requires knowledge of the number of currently buffered packets. Thus, FIFO with threshold policy makes a good candidate for use in the AF-PHB.

2.1 Model

The model we consider consists of a single server FIFO queue as shown in Fig. 1. For simplicity, we assume that only two packet priorities are available, green (denoted GP) and yellow (denoted YP)(inclusion of red is

[1]When a red packet arrives and the number of packets currently in the buffer is below a threshold B, then it is accepted. Otherwise it is dropped (see Fig. 1).

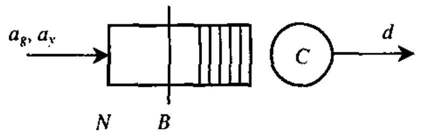

Figure 1: Buffer policy model

a straightforward extension). A green packet is always accepted as long as there is some buffer space available, that is if the buffer queue length $X < N$. A yellow packet is accepted only if $X < B \leq N$. We use the notation $(N|B)$ to denote the buffer size and threshold respectively.

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. As a result, the state transition diagram of the system is shown in Fig. 2. Next we assume that the buffer size N is fixed and it

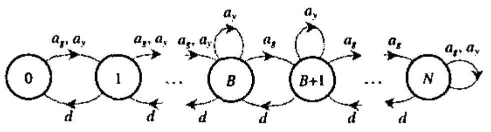

Figure 2: State transition diagram for an $(N|B)$ system

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 B such that the drop probability of green packets is always satisfied while the yellow drop probability is minimized. Our approach is based on sample path constructability where observations of the system sample path under a threshold B_0 are used to construct the sample paths under *any* threshold B_i, $i = 1, 2, \cdots$. Subsequently, it is trivial to select the threshold with the best performance. For an overview on constructability techniques refer to [7].

3 Augmented System Analysis (ASA)

In this section we develop our first technique which is based on Augmented System Analysis (ASA) [11] and is the most efficient and easiest to implement on-line. In general, ASA is applicable when the lifetime distributions of all but one event are memoryless (i.e., exponential). Regardless of this limitation, we chose to explore this technique because the state transitions of this particular system are such that the error due to non-exponential distributions is going to be small, thus the technique will provide good approximations.

In this paper, we adopt the standard Stochastic Automaton Model $(\mathcal{E}, \mathcal{X}, \Gamma, f, x_0, \mathbf{V})$ [7] where $\mathcal{E}$ is the event set, $\mathcal{X}$ is the state space, $\Gamma(x)$ is the feasible event set at state x, f is the state transition mechanism, x_0 is the initial state, and, $\mathbf{V}$ is the clock structure. In addition, we use the notation $\xi(B_i) = \{e_k^i, t_k^i\}$ to indicate a sample path, i.e., a sequence of events e_k^i and their corresponding occurrence time t_k^i, $k = 1, 2, \cdots$, under parameter B_i. Next we present a condition that allows sample path construction.

Observability Condition (OB)[7]: Let $\xi(B_0) = \{e_k^0, t_k^0\}$ be the sample path under parameter B_0 and let $\{X_k(B_0)\}$, $k = 1, 2, \cdots$, be the corresponding state sequence. For any parameter $B_i \neq B_0$, let $\xi(B_i) = \{e_k^i, t_k^i\}$ be the sample path generated by the event sequence observed in $\xi(B_0)$. The sample path $\xi(B_0)$ is observable if

$$\Gamma(X_k(B_i)) \subseteq \Gamma(X_k(B_0)) \text{ for all } k = 1, 2, \cdots$$

3.1 Event-Matching Algorithm

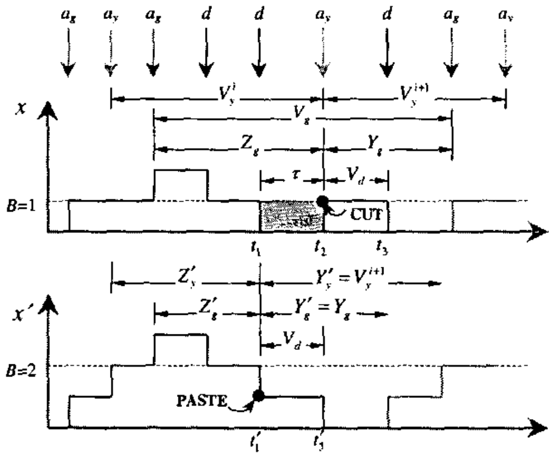

Figure 3: The sample path cut-and-paste technique when the observed system is (3|1) and the constructed system is (3|2)

In ASA once the two sample paths reach states where $\Gamma(X_k(B_i)) \supset \Gamma(X_k(B_0))$, then the sample path construction is suspended until the observed sample path reaches a new state such that $\Gamma(X_k(B_i)) \subseteq \Gamma(X_k(B_0))$ when the sample path construction resumes. As will be pointed out later, this approach maintains the statistical characteristics of the original system as long as the event lifetime distributions are exponential, i.e., they have the memoryless property.

The process of suspending/resuming the sample path construction corresponds to a sample path "cut-and-paste" approach as demonstrated in Fig. 3 where x and x' correspond to the state of the observed and constructed sample paths respectively. At time t_1 we observe a violation of the OB condition since $\Gamma(x = 0) = \{a_g, a_y\} \subset \Gamma(x' = 1) = \{a_g, a_y, d\}$. At this point we suspend the construction of the sample path that corresponds to $B = 2$ until the observed sample path reaches a state where OB is again satisfied (i.e., $x = 1$). This happens at time t_2 when we resume the sample path construction. In effect, this is equivalent to cutting the sample path at time t_2 and pasting it at time $t'_1 = t_1$. This immediately determines the event sequence $\{e'_l\}$, $l = 1, 2, \cdots$. It is also easy to determine the occurrence time of each event and thus construct the sample path $\xi(B = 2) = \{e'_l, t'_l\}$, $l = 1, 2, \cdots$. This is given by

$$t'_l = t_k - \sum_{j=1}^{J(t_k)} \tau_j \qquad (1)$$

where $J(t_k)$ is the number of times the sample path construction has been suspended, due to OB violation, up until time t_k, and, τ_j is the length of the suspended period j. Also notice that because of the suspend/resume process, some of the events observed in the nominal (observed) sample path are not constructed in the perturbed sample path (in this case $B = 2$), therefore, in general $l \neq k$.

Next, we are interested in the statistical characteristics of the constructed sample path. To do this we need to observe the lifetime distributions of all possible events in both sample paths. First we note that for all states where $\Gamma(x') = \Gamma(x)$, the event lifetimes observed in the nominal sample path and the lifetimes used in the constructed sample path are identical, therefore, statistically the two sample paths are equivalent.

A problem arises only at states where $\Gamma(x') \neq \Gamma(x)$. At this point we assume that $\{V_g^i\}$ is a sequence of green packet interarrival times where each V_g^i is an exponentially distributed random variable with mean λ_g^{-1}. Similarly, $\{V_y^i\}$ is a sequence of exponentially distributed yellow packet interarrival times with mean λ_y^{-1}. Finally, $\{V_d^i\}$ is a sequence of transmission (service) times from an arbitrary distribution.

Next, we investigate the effect of the suspend/resume process on the lifetime distributions. In the observed sample path, at time t_2, the residual lifetime for the next yellow arrival $Y_y = V_y^{i+1}$ since a yellow arrival has just occurred. Therefore, this is exponentially distributed with rate λ_y. The residual lifetime of the next green arrival Y_g given that Z_g time has elapsed since the last green arrival is also exponential with rate λ_g due to the memoryless property of the exponential distribution. Finally, the residual lifetime of the next departure

$Y_d = V_d$ since the new packet has just started being transmitted. In the constructed sample path now, the residual lifetimes of the green and yellow arrivals Y'_g and Y'_y given their age Z'_g and Z'_y respectively, are both exponentially distributed with rates λ_g, λ_y due to the memoryless property. Furthermore, the residual lifetime of the next departure Y'_d is always going to be equal to $Y_d = V_d$ since at the cut and paste points a packet transmission is always just starting. Thus, V_d can be from an arbitrary distribution. This suggests that as long as the green and yellow arrivals are exponentially distributed, the two sample paths are statistically equivalent.

3.2 ASA for General Distributions

If the green and yellow arrivals are not exponentially distributed, then every time we invoke the event-matching procedure we introduce an estimation error. This estimation error is small as long as the augmented state $(0, x' \neq 0)$ is not visited frequently. Note that visiting this state is possible only after a yellow packet arrival that causes the two sample paths to have a different number of packets in the queue. Even if this occurs, then it is still possible to avoid visiting this "bad" state, thus reducing the estimation error, when the system loses a green packet. This is very likely for systems with high green traffic.

We also point out that the incoming traffic at any AF class consists of the superposition of $Q_t >> 1$ generally independent sources. As Q_t goes to infinity, the aggregate arrival process at the switch approaches a Poisson process [12]. In practical systems it is reasonable to expect the Q_t is usually large, therefore, from the switch's point of view, the arrival process is well approximated by a Poisson process. As a result, ASA can be applicable even if each individual source is not Poisson.

3.3 ASA Algorithm

Table 1 shows the ASA algorithm for constructing the sample path under any $B_i \neq B_0$ where B_0 is the yellow threshold in the nominal or observed sample path. In the algorithm X_k and X'_l correspond to the system state in the nominal and constructed sample paths respectively. 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 $l \neq k$.

4 "Modified Lindley" Construction Algorithm

In computer communication networks, 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

Table 1: ASA Algorithm

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

time S_k. Since this information is available at the time of the kth packet arrival A_k, it can be used to determine the time of the packet's departure D_k, through a Lindley recursion, even when the OB condition is violated.

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:

$$\begin{array}{ll} X_i(A_k) \geq N & \text{if } k \text{ is a } GP \\ X_i(A_k) \geq B_i & \text{if } k \text{ is a } YP \end{array} \qquad (2)$$

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 (2) are equivalent to the following conditions:

$$\begin{array}{llll} X_i(A_k) \geq N & \Leftrightarrow & A_k \leq D_{k-N-l_i} & \text{if } k\ GP \\ X_i(A_k) \geq B_i & \Leftrightarrow & A_k \leq D_{k-B_i-l_i} & \text{if } k\ YP \end{array} \qquad (3)$$

where l_i is the number of lost (dropped) packets under parameter B_i. In (3), 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 (ML) construction algorithm). For every constructed sample path under some B_i, we define the index process

$$a_i(k) = k - l_i, \quad i = 0, \cdots, M \tag{4}$$

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 (4) we can determine the departure time of each accepted packet using the Lindley recursion

$$D_{a_i(k)} = \max\left\{D_{a_i(k)-1}, A_{a_i(k)}\right\} + S_{a_i(k)}. \tag{5}$$

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 AF class. 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 (ML) 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 $GP$ GOTO 2.1.1 ELSE GOTO 2.1.2
    2.1.1.  IF $A_k \leq D_{a_i(k)-1-N}$, THEN
                $l_i \leftarrow l_i + 1$, GOTO 4   % pkt dropped
    2.1.2.  IF $A_k \leq D_{a_i(k)-1-B_i}$, THEN
                $l_i \leftarrow l_i + 1$, GOTO 4   % pkt dropped
3.  $S_{a_i(k)} = \frac{s_k}{C}$                  % packet accepted
    $D_{a_i(k)} = \max\{D_{a_i(k)-1}, A_{a_i(k)}\} + S_{a_i(k)}$
4.  END
```

4.1 Admission Control

Using the ML construction algorithm above, it is possible to predict the QoS (packet loss or delay) delivered by the network if additional connections become active. Furthermore, this can be done for any threshold B_i, $i = 0, 1, \cdots, M$. For the remaining of this section we assume that there are Q active sources and as a result, the arrival process at the switch is approximated by a Poisson process with rate Λ. Let $\xi_Q(B_0)$ be the observed sample path under Q sources and yellow threshold B_0, and $L[\xi_Q(B_0)]$ be the value of the performance measure of interest (e.g, loss probability). The problem we address in this section is the following. Suppose a new source with rate λ is admitted (λ can be an upper bound), then construct the sample paths $\xi_{Q+1}(B_i)$ and consequently determine $L[\xi_{Q+1}(B_i)]$, for all $i = 0, \cdots, M$.

In our approach, we recognize that the addition of a new source will increase the packet arrival rate and consequently will reduce the packet interarrival times. With Q sources, the kth packet will arrive at time A_k. Let A'_k be the arrival time of the kth packet if there were $Q + 1$ active sources then we can show that

$$A'_k = \frac{\Lambda}{\Lambda + \lambda} A_k. \tag{6}$$

Using A'_k in the modified Lindley algorithm of Table 2 we can construct the expected sample paths and therefore the expected performance measures, under any threshold B_i and under any number of additional sources. Therefore, based on this information, one can easily solve the admission control problem.

5 Simulation Results

In this section 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.

5.1 Adjusting Threshold B

Several scenarios have been investigated, however due to space limitations we only present one where there are 5 MMPP ON/OFF sources. During the ON period, each source transmits at a rate 400 packets per second. The length of the ON and OFF periods are both exponential with mean 0.1 seconds. 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 switch is 500Kbytes per second. Finally, the ASA and ML curves were obtained by simply observing the sample path under $B_0 = 10$.

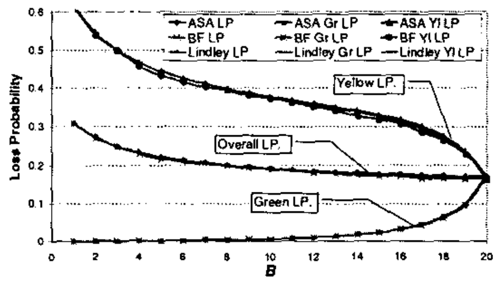

Figure 4: Loss probability

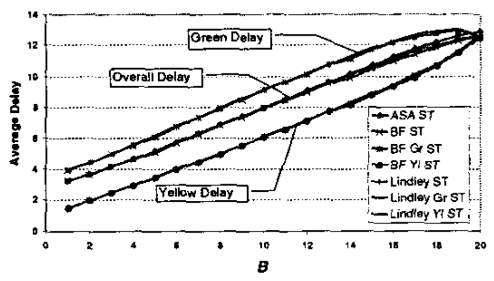

Figure 5: Average packet delay

Figures 4 and 5 demonstrate that both algorithms accurately predict the performance measures of interest, e.g., loss probability and delay.

5.2 Adding New Sources

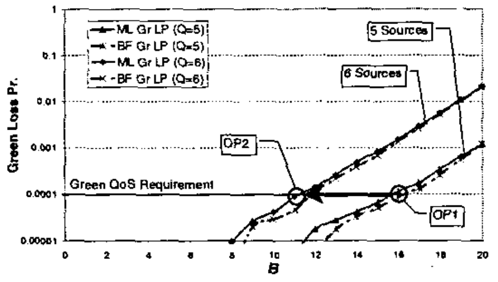

Figure 6: Admission control and resource reallocation

Next we investigate the ability of the ML algorithm to provide estimates of the network's performance under additional sources for any threshold parameter B_i, $i = 1, \cdots, M$. Furthermore, we show how such information can be used for admission control as well as for threshold adjustment.

The scenario we investigate is the following. Five identical sources, each transmitting according to a Poisson process with rate 500 packets per second. The size of each packet is exponentially distributed with mean 500 bytes and the transmission rate of the switch is 1.6Mbytes per second. Again we assume that packets are either green or yellow with equal probability.

In this scenario we further assume that the QoS requirements for green packets is loss probability $p_g \leq 10^{-4}$. To meet this requirement, it is necessary that the threshold $B = 16$ as shown in Fig. 6 (point OP1). Using the ML algorithm, while operating at the point $Q = 5$, $B_0 = 16$, we also get estimates of the green loss probability for either five or six sources and for all thresholds $B = 1, \cdots, 20$. In Fig. 6 we compare these estimates against brute force simulation (BF curves) and show that the two produce comparable results[2].

Subsequently, a sixth source requests to be connected. At this point, the controller can use the information collected by the ML algorithm (i.e., the two ML curves in Fig. 6) to decide whether to admit the new source or not. Fig. 6 shows that even if a sixth source is admitted, *there exists a threshold B that still meets the* green QoS requirements. This is $B = 11$ (OP2), therefore it admits the new source and immediately switches to OP2. Since ML can provide delay estimates, the controller can base the admission control decision on the effect that the sixth source will have on packet delay.

References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," *RFC 2475*, Dec 1998.

[2] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding PHB," *RFC 2598*, Jun 1999.

[3] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding PHB group," *RFC 2597*, Jun 1999.

[4] J. Heinanen and R. Guérin, "A single rate three color marker," *RFC 2697*, Sep 1999.

[5] J. Heinanen and R. Guérin, "A two rate three color marker," *RFC 2698*, Sep 1999.

[6] C. Casetti, J. Kurose, and D. Towsley, "An adaptive algorithm for measurement-based admission control in integrated services packet networks," *Computer Communications*, vol. 23, pp. 1363–1376, 2000.

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

[8] C. G. Cassandras and C. G. Panayiotou, "Concurrent sample path analysis of discrete event systems," *Journal of Discrete Event Dynamic Systems*, vol. 9, pp. 171–195, May 1999.

[9] H. Chaskar, E. Dimitriou, and R. Ravikanth, "Service guarantees in the Internet: Differentiated services approach," in *Eighth International Workshop on Quality of Service (IWQOS)*, pp. 176 -178, 2000.

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

[11] 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.

[12] M. Westcott, "Simple proof of a result on thinned point process," *The Annals of Probability*, vol. 4, no. 1, pp. 89–90, 1976.

[2]The observed differences are due to noise. The four curves are obtained from a single sample path of relatively short length.