# A Control Theory Approach to Cell Scheduling in ATM Switches

**Srinivasan Ramaswamy**
**AcceLight Networks**
**80 Colonnade Road**
**Nepean, Ontario, CANADA K2E 7L2**
**srini@accelight.com**

**Pawel Gburzynski**
**University of Alberta**
**Department of Computing Science**
**Edmonton, Alberta, CANADA T6G 2H1**
**pawel@cs.ualberta.ca**

**Keywords:** ATM networks, bandwidth allocation, feedback control

**Abstract**

We present a control theory based approach to scheduling departing cells in an ATM switch. With our approach, queue service rates fluctuate within some limits, maintaining the QoS at the desired level. The feedback mechanism prescribing the service rate is based entirely on local information. Consequently, our method is indifferent to propagation delays and it can be deployed in large and/or high speed networks.

## INTRODUCTION

Consider a server of fixed rate handling cells from two queues: VBR and ABR. The service rate for each queue is fixed by CAC (call admission control). Any remaining bandwidth is utilized by ABR traffic. Within a queue, cells are served in FIFO order.

Several scheduling policies have been proposed to guarantee that each queue receives its allocated share of the bandwidth, e.g., PGPS [6], virtual clock [9], and WFQ [4]. The problem with WFQ and other schemes is that they rely on CAC to indicate the correct bandwidth allocation to each queue. If a VBR source does not fully utilize the reserved bandwidth, its QoS is better than that requested. However, it would be preferable for ABR traffic to utilize the excess bandwidth. Also, bit rate fluctuations in the aggregate VBR traffic can result in the actual QoS being lower than desired, resulting in QoS violation.

The goal of our proposed scheme is to modulate the bandwidth allocated to a queue in such a way that the QoS delivered to the aggregate traffic remains at its desired level. CAC specifies the **mean level** about which the service rate is to fluctuate, as well as the **bounds** for the service rate modulation. Our model assumes that nothing is known about the distribution of aggregate traffic. The only information used by the scheduler is the average rate of the aggregate traffic and the time scale information. Like WFQ, we assume the availability of a CAC algorithm that indicates the amount of bandwidth to be allocated to a queue receiving traffic from a group of statistically multiplexed sources.

A similar feedback control approach has been proposed in [7]. One problem with that scheme is that its performance is adversely affected by the distance between the source and the queue. Moreover, flow control information must be sent through the network to each source. Another disadvantage is that the selection of a sampling time for the controller is dictated by the source that is farthest away, leading to unnecessarily slow control of nearby sources. We avoid all these problems by modulating the service rate at the queue itself.

## THE PLANT

Consider the cell scheduling problem in an ATM switch. The system to be controlled (the *plant*) consists of the queue for the VBR traffic [3] and the virtual server for that queue. The queue has a finite number of cell buffers. The virtual server transmits cells at the constant rate indicated by CAC, which specifies upper and lower constraints on the service rate.

We assume that the traffic descriptor of a source consists of [3]: Peak Cell Rate (PCR), Sustainable Cell Rate (SCR), and Burst Tolerance (BT). CAC passes to the virtual scheduler the peak and average rate of the aggregate traffic.

The ATM Forum has specified the following QoS parameters for real-time VBR traffic [3]: Cell Loss Ratio (CLR), Maximum Cell Transfer Delay (maxCTD), and Peak-to-peak Cell Delay Variation (ppCDV). In addition, we allow a source to specify an average delay requirement.

We notice that the sum of ppCDV and the fixed delay gives the $\alpha$ percentile of CTD. Since our system consists of one queue, we assume the fixed delay to be zero. We also use [3] $\alpha = 100(1 - \mathrm{CLR})$. The QoS requirement is: "$1 - \mathrm{CLR}\,\%$ of the cells

should suffer delay less than ppCDV, and the average cell delay should not exceed avDEL." Consequently, the QoS requirements can be specified as a 3-tuple: $< \mathrm{CLR}_{qos}, \mathrm{ppCDV}_{qos}, \mathrm{avDEL}_{qos} >$. Our results indicate that the average delay is a function of the service rate allocation made by CAC. For this reason, *the cell scheduler is only responsible for maintaining the cell loss and maximum delay at the desired levels.* Notably, this can be accomplished by controlling a single output signal. If we can control our queuing system in such a way that the $\alpha$ percentile of the buffer occupancy is maintained at a desired level $K_{ref}$, then by choosing $\alpha = 100(1 - \mathrm{CLR}_{desired})$ and $K_{ref} = \mathrm{ppCDV}$, we can ensure that the QoS requirements are met.

Consider a snapshot of the buffer occupancy histogram during a hypothetical run. The number of cells, $N_i$ that occupy position $i, 1 \leq i \leq K_{max}$ in the queue are recorded, where $K_{max}$ is the highest position occupied at any time during the run. Define the frequency of buffer position $i$ as $f_i = N_i/(\sum_{i=1}^{K_{max}} N_i)$ and the cumulative frequency as $c_i = (\sum_{j=1}^{i} N_j)/(\sum_{i=1}^{K_{max}} N_i)$. The cumulative frequencies satisfy $c_i > c_{i-1}, \forall i = 1, \ldots, K_{max}$ and $c_{K_{max}} = 1$. Let $j$ be the buffer position such that $c_j \geq \alpha$ and $c_{j-1} \leq \alpha$; then the $\alpha$ percentile, $K_{alpha}$, is obtained by interpolation as $j - (c_j - \alpha)/(c_j - c_{j-1})$.

## SYSTEM IDENTIFICATION

We assume that very little *a priori* information about the plant is available, i.e., its model can be built using only sampled plant input and output data. One way to build such a model is to perturb the plant by a carefully chosen input signal. The corresponding plant outputs are recorded at the sampling instants. The collection of input and output data is then analyzed off-line to obtain an open-loop model.

Alternatively, the plant input and output data can be obtained while the system is under feedback control. Typically, in such a system, an independent dither signal is introduced in the feedback loop to provide sufficient excitation. Also in this case the collected data is analyzed at the end of the experiment, resulting in the *batch identification* procedure.

The last possibility is *online* identification. When the plant starts up, it is left to run for some time in open-loop mode, during which time it is excited by a random signal. When sufficient data has been recorded, a model of a pre-specified order is identified. From then on, at every sampling instant, the new sample of the plant input and output is used to update the model calculated at the previous sampling instant. Such a procedure of *recursive identification* is especially important in adaptive control, applied to plants whose characteristics change dynamically. As our ATM plant fits into this category, we use recursive identification in our approach.

Because of disturbances, the system output is related in a non-deterministic manner to its input. Therefore, the system input and output are treated as stochastic processes; they are also assumed to be discrete-time. Our model is of the form

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})} u(t-d) + e(t), \tag{1}$$

where $q^{-1}$ is the backward shift operator defined by $q^{-1}y(t) = y(t-1)$, $q^{-2}y(t) = y(t-2)$, and so on. $A(q^{-1})$ and $B(q^{-1})$ are polynomials in $q^{-1}$ of orders $n$ and $m$, respectively. $e$ is a random disturbance signal with zero mean and unit variance. The system delay is $d$ time samples. The system output at time $t$ is a function of $m$ past inputs, $n$ past outputs and a random noise component, $e(t)$. This structure represents an autoregressive, exogenous input (ARX) model of the input-output relation.

Let us start with batch identification, which by itself is insufficient for our plant. Assume that $N$ input-output values of plant data are available. We would like to find the ARX model of some order $n$ that satisfactorily explains the input-output relationship. Equation 1 can be rewritten in the following manner:

$$y(t) = \mathbf{x}^T(t)\,\theta + e(t), \tag{2}$$

where $\theta$ is the vector of unknown coefficients of the $A$ and $B$ polynomials, $\theta^T = [-a_1, \ldots, -a_n, b_0, \ldots, b_m]$, and $\mathbf{x}$(t) is a regression vector consisting of past measured input and output values $\mathbf{x}^T(t) = [y(t-1), \ldots, y(t-n), u(t-1), \ldots, u(t-m-1)]$.
Assume that equation 2 is an exact representation of the system, and that we wish to determine from available data the vector $\theta$ of true system parameters. For this, we further assume a model of the system of the correct structure:

$$y(t) = \mathbf{x}^T(t)\,\hat{\theta} + \hat{e}(t), \tag{3}$$

where $\hat{\theta}$ is a vector of adjustable model parameters and $\hat{e}(t)$ is the corresponding modeling error at time t. Our aim is to select $\hat{\theta}$ such that the modeling error over all the data points is minimized in some sense. Since we have $N$ input-output values,

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(N) \end{bmatrix} = \begin{bmatrix} \mathbf{x}^T(1) \\ \mathbf{x}^T(2) \\ \vdots \\ \mathbf{x}^T(N) \end{bmatrix} \theta + \begin{bmatrix} \hat{e}(1) \\ \hat{e}(2) \\ \vdots \\ \hat{e}(N) \end{bmatrix} .$$

To estimate the $m+n$ unknowns uniquely, we need $N \geq m+n$. In the presence of noise, we must have $N >> m+n$ and employ an error reduction technique (*linear least squares*). Let us rewrite equation 3 as $\mathbf{y} = \mathbf{X}\hat{\theta} + \hat{\mathbf{e}}$, where $\mathbf{y}^T = [y(1), \ldots, y(N)]$, $\hat{\mathbf{e}}^T = [\hat{e}(1), \ldots, \hat{e}(N)]$, and

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^T(1) \\ \mathbf{x}^T(2) \\ \vdots \\ \mathbf{x}^T(N) \end{bmatrix} ,$$

which can be written as $\hat{\mathbf{e}} = \mathbf{y} - \mathbf{X}\,\hat{\theta}$. With the least squares minimization, we can choose an estimate $\hat{\theta}$ that minimizes $J$, the sum of squares of errors $J = \sum_{i=1}^{N} \hat{e}^2(t) = \hat{\mathbf{e}}^T\,\hat{\mathbf{e}}$, that is,

$$\begin{aligned} J &= (\mathbf{y} - \mathbf{X}\,\hat{\theta})^T\,(\mathbf{y} - \mathbf{X}\,\hat{\theta}) \\ &= \mathbf{y}^T\,\mathbf{y} - \hat{\theta}^T\,\mathbf{X}^T\mathbf{y} - \mathbf{y}^T\,\mathbf{X}\,\hat{\theta} + \hat{\theta}^T\,\mathbf{X}^T\,\mathbf{X}\,\hat{\theta}. \end{aligned}$$

To find the minimum value of $J$, we set

$$\frac{\partial J}{\partial \hat{\theta}} = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\theta = 0. \qquad (4)$$

If the second derivative matrix

$$\frac{\partial J^2}{\partial \hat{\theta}^2} = 2(\mathbf{X}^T\mathbf{X})$$

is positive definite, the least squares estimator for the parameter vector from equation 4 is $\hat{\theta} = [\mathbf{X}^T\mathbf{X}]^{-1}[\mathbf{X}^T\mathbf{y}]$. Therefore, given the data matrix $\mathbf{X}$ consisting of past input and output samples, and the vector $\mathbf{y}$ of current output samples, we can find the unknown coefficients of the ARX model relating the plant input-output.

The batch identification technique must be modified, to identify an up-to-date evolving model at each sampling instant. We use *recursive identification* in which the results for $N$ observations (the *control horizon*) are used to get the estimates for $N+1$ observations [8]. At any time $t$, the model obtained at the previous sampling instant, $\hat{\theta}(t-1)$, summarizes the past input and output of the plant. This model is used to obtain an estimate $\hat{y}(t)$ of the current output. This is compared with the actual output $y(t)$ to generate an error $e(t)$. This in turn generates an update to the model, which corrects $\hat{\theta}(t-1)$ to the new value $\hat{\theta}(t)$. This recursive "predictive-corrector" form eliminates the need to save all the past input and output data. The algorithm for recursive least squares identification (RLS) is as follows. At time step $t+1$:

1. Form $\mathbf{x}(t+1)$ using the new data sample
2. Form $e(t+1)$ using
   $e(t+1) = y(t+1) - \mathbf{x}^T(t+1)\hat{\theta}(t)$
3. Form $\mathbf{P}(t+1)$ using

   $$\mathbf{P}(t+1) = \mathbf{P}(t)\left[\mathbf{I}_m - \frac{\mathbf{x}(t+1)\mathbf{x}^T(t+1)\mathbf{P}(t)}{1 + \mathbf{x}^T(t+1)\mathbf{P}(t)\mathbf{x}(t+1)}\right],$$

   where $\mathbf{I}_m$ is the identity matrix.
4. Update $\hat{\theta}(t+1) = \hat{\theta}(t) + \mathbf{P}(t)\mathbf{x}(t+1)e(t+1)$
5. At the end of the sample period, go to step 1.

Before the estimator can be started up, the data vector $\mathbf{x}(t)$ must be full. For a model with $A$ and $B$ polynomials of degree $n$ and $m$, respectively, the sampling process must run for $\tau = max(n, m+1)$ steps before the estimator can be started.

One way of getting the recursive estimator to adapt to time-varying true parameters is to use a *forgetting factor*, i.e., a number $\epsilon$ between 0 and 1, to progressively reduce the importance of past information. While the normal least squares minimizes the cost function

$$J = \sum_{i=1}^{N} \hat{e}^2(i) ,$$

in which all values of $\hat{e}^2(i)$, $i = 1,\ldots,N$, are equally important, the forgetting factor approach uses the function $J' = \sum_{i=1}^{N} \epsilon^{N-i}\hat{e}^2(i)$ to assign progressive lower ranks to older values. The only change necessary to the RLS algorithm is in step 3 which now becomes

$$\mathbf{P}(t+1) = \epsilon^{-1}\mathbf{P}(t)\left[\mathbf{I}_m - \frac{\mathbf{x}(t+1)\mathbf{x}^T(t+1)\mathbf{P}(t)}{\epsilon + \mathbf{x}^T(t+1)\mathbf{P}(t)\mathbf{x}(t+1)}\right] ,$$

with $\epsilon$ typically lying between 0.9 and 1.

As presented so far, the RLS still suffers from two disadvantages: the results depend on the initial choice of the covariance matrix and they are not the same as those obtained by the batch least squares method. To alleviate this problem, we modify RLS into RRLS

(rigorous RLS) according to [5]. The process is modeled by an ARX equation of the form $y_k = A_1\, y_{k-1} + \cdots + A_n\, y_{k-n} + B_1\, u_{k-1} + \cdots + B_m\, u_{k-m} + c + e_k$. For a single-input-single-output (SISO) system, $y_k$ is the process output sampled at time $k$, $u_k$ is the process input at time $k$ and $\{e_k\}$ is the noise with zero mean. Estimation of a constant offset $c$ means that the sampled $y$ and $u$ values need not be mean-centered and auto-scaled. Define $\mathbf{b}_k = \mathbf{P}_{k-1}\mathbf{x}_k, c_k = \epsilon + \mathbf{x}_k^T\mathbf{b}_k, \mathbf{a}_k = \mathbf{Q}_{k-1}\mathbf{x}_k$ and let the initial conditions be $\hat{\theta}_0 = 0, \mathbf{P}_0 = r\mathbf{I}, \mathbf{Q}_0 = \mathbf{I}$. The algorithm is defined in the following way:

$$\begin{cases} \mathbf{g}_k = \mathbf{a}_k(\mathbf{a}_k^T\mathbf{a}_k)^{-1} \\ \hat{\theta}_k = \hat{\theta}_{k-1} + \mathbf{g}_k(y_k - \mathbf{x}_k^T\hat{\theta}_{k-1}) \\ \mathbf{P}_k = (\mathbf{P}_{k-1} - \mathbf{g}_k\mathbf{b}_k^T - \mathbf{b}_k\mathbf{g}_k^T + c_k\mathbf{g}_k\mathbf{g}_k^T)/\epsilon \\ \mathbf{Q}_k = (\mathbf{Q}_{k-1} - \mathbf{g}_k\mathbf{a}_k^T) \end{cases} \quad \text{if } \mathbf{a}_k \neq 0$$

$$\begin{cases} \mathbf{g}_k = \mathbf{b}_k c_k^{-1} \\ \hat{\theta}_k = \hat{\theta}_{k-1} + \mathbf{g}_k(y_k - \mathbf{x}_k^T\hat{\theta}_{k-1}) \\ \mathbf{P}_k = (\mathbf{P}_{k-1} - \mathbf{g}_k\mathbf{b}_k^T)/\epsilon \\ \mathbf{Q}_k = (\mathbf{Q}_{k-1}) \end{cases} \quad \text{if } \mathbf{a}_k = 0$$

## CONTROLLER DESIGN

The plant output $y_k$ at time $k$ is a function of $< y_{k-1}, \ldots, y_{k-n}, u_{k-1}, \ldots, u_{k-m}, c >$. Define two sets $S_k^{fr} = \{y_k, y_{k-1}, \ldots, y_{k+1-n}, u_{k-1}, \ldots, u_{k+1-m}, c\}$ and $S_k^{fo} = \{u_k, u_{k+1}, \ldots, u_{k+i-1} : i \geq 1\}$. At time $k$, the predicted model output $\hat{y}_{k+i} = \hat{y}_{k+i}^{fr} + \hat{y}_{k+i}^{fo}$, where $\hat{y}_{k+i}^{fr}$ is the predicted free response, and $\hat{y}_{k+i}^{fo}$ is the predicted forced response. For some control horizon $N \geq 1$, define

$$\hat{y}_{k,N}^{fr} = \begin{bmatrix} \hat{y}_{k+d}^{fr} \\ \hat{y}_{k+d+1}^{fr} \\ \vdots \\ \hat{y}_{k+d+N-1}^{fr} \end{bmatrix}, \hat{y}_{k,N} = \begin{bmatrix} \hat{y}_{k+d} \\ \hat{y}_{k+d+1} \\ \vdots \\ \hat{y}_{k+d+N-1} \end{bmatrix}$$

and

$$u_{k,N} = \begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+N-1} \end{bmatrix},$$

where $d$ is the system delay (equal 1 in our case). Then, the plant output predicted $N$ steps into the future is $\hat{y}_{k,N} = \hat{y}_{k,N}^{fr} + C_{k,N}u_{k,N}$, where $C_{k,N}$ is determined by the parameter vector $\hat{\theta}_k$.

The problem of finding the minimum-bias control can be expressed as the following multi-objective optimization problem: "minimize $J_{MB} = \|\hat{y}_{k,N}^{fr} + C_{k,N}u_{k,N} - y_{k,t}^*\|$, subject to $u_{k,t} \in \mathcal{U}_{k,t}$," where $\mathcal{U}_{k,t}$ is the set of admissible control values, specified as an upper and lower bound on $u_{k,t}$. The vector $y_{k,t}^*$ consists of values of the reference signal at the $N$ future sampling instants.

The output signal of our SISO system is the percentile of buffer occupancy, which can be easily measured on-line. Besides measuring the actual percentile of buffer occupancy, we must also be able to calculate it in the model—for prediction.

A method for approximating the probability distributions of stationary statistics in FIFO single server queues is described in [1]. It can be applied when the *net input process* forms a stationary, ergodic, Gaussian discrete-time process. Since the service rate in ATM networks is deterministic, the service process has zero variance and is specified by its mean alone. The Gaussian process can be used to model any traffic encountered in practice, since its mean, variance, and auto-covariance function can be matched to any process. Moreover, the aggregate traffic resulting from the multiplexing of a number of independent streams is likely to be Gaussian. The predicted $\alpha$-percentile is a function of the mean, variance, and auto-covariance sum of the net arrival process defined as $Y_n = A_n - B_n$, where $A_n$ is the amount of work entering the queue during the $n^{th}$ sampling interval (of duration $T_{sample}$), and $B_n$ is the amount of work processed by the server during the same interval. For a deterministic server, $B_n = T_{sample}/\tau_{cell\ time}$.

The following two quantities are defined in [1]:

$$s^* = \frac{2m}{\sigma^2 + 2S}, \quad \tilde{c} = \frac{erfc\left(\frac{-m}{\sigma\sqrt{2}}\right) - e^{(u-m)s^*}\, erfc\left(\frac{2u-m}{\sigma\sqrt{2}}\right)}{s^*\left(\frac{\sigma\sqrt{2}}{\sqrt{\pi}}e^{-\left(\frac{u^2}{2\sigma^2}\right)} - u\, erfc\frac{u}{\sigma\sqrt{2}}\right)},$$

where $u = \sigma^2 s^*/2$, $m$ is the mean value of $Y_n$, i.e., $m = E\{Y_n\} = E\{A_n\} - E\{B_n\}$, $\sigma^2$ is the variance of $Y_n$, i.e., $\sigma^2 = Var\{Y_n\} = Var\{A_n\}$ (since $B_n$ is deterministic), $S$ is the auto-covariance sum of $Y_n$ defined as the sum of the covariances of $Y_n$ for all lags $> 1$, i.e., $S \triangleq \sum_{k=1}^{\infty} Cov(Y_n, Y_{n+k})$. These quantities are used to compute the $\alpha$-percentile of the buffer occupancy as

$$K_{alpha} = \frac{1}{s^*}\, ln\left(\frac{1 - \frac{\alpha}{100}}{\tilde{c}}\right). \qquad (5)$$

As mentioned in [2], computing the auto-covariance sum ($S$) on-line is not practical. For this reason, we compute an approximate value for the $p^{th}$ percentile using only the mean and variance of the net arrival process (assuming $S = 0$). With this approximation,

the actual percentile may be quite different from the computed value. Therefore, if $\mid y(t) - y_{act}(t) \mid > \omega * y_{act}(t)$, we use $\delta * y(t) + (1-\delta) * y_{act}(t)$ instead of $y(t)$. In our experiments, the threshold $\omega$ was set to 0.2 and the bias $\delta$ was 0.1.

In [7], the calculated percentile alone is used as the plant output signal. The control system in that case guarantees that the long term QoS will approach the desired value. However, during the transient period, the estimated percentile (and the actual percentile) may greatly exceed the reference level. The use of the weighted average has the advantage that the QoS requirements are not violated at any time, even if the call duration is short.[1] Also, as our approach is not based on flow control, we can use much shorter sampling intervals (independent of the round-trip propagation delay). This way we are able to bring the QoS to the desired level in a much shorter time.

The buffer occupancy distribution can be altered by varying either the arrival process or the service process. One of the guidelines in choosing the input signal is that the output must respond quickly to a change in the input. To avoid the influence of propagation delays (and other problems with the approach taken in [7]), we choose to modulate the service rate. The mean service rate as well as upper and lower limits on the service rate are specified by CAC.

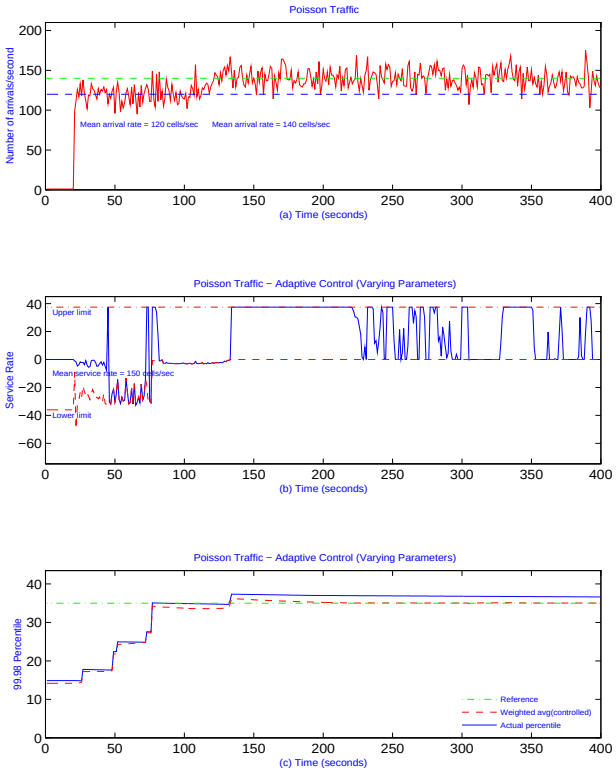


Figure 1: Adaptive control (effect of varying parameters)

## RESULTS AND CONCLUSIONS

We have introduced a cell scheduler for a queue fed with aggregate traffic from a group of multiplexed VBR calls. The scheduler uses adaptive feedback control to maintain the QoS at a desired level—in spite of traffic fluctuations and/or variations in the aggregate traffic characteristics.

Our scheduler modulates the service rate in order to control the QoS. This local control is indifferent to propagation delays; in that respect, it is superior to schemes controlling the individual distant sources. Consequently, it can be deployed without any extra tuning or adaptation in very large and/or fast networks.

Consider the scenario illustrated in figures 1 and 2. In this experiment, the arrival rate is changed from 120 cps (cells per second) to 140 cps.[2] In the open loop test (figure 2) the buffer percentile jumps to 55 cells as a result of the increased queue utilization. However, the adaptive controller restricts the increase to within 15% of the reference level (figure 1).

[1] The method presented in [7] is useless for calls lasting less than the length of the adaptation period.

[2] Of course, the *second* is an abstract time unit. Our model, unlike e.g. [7], is invariant with respect to the time scale.

In order to investigate the effect of long-range dependence on the controller performance, we fed the queue with self-similar traffic. The reference level for buffer percentile was set at 250 cells. The mean arrival rate was set to 120 cps, with the variance of the arrival rate being equal to 2 $\mu$ (i.e., 240). The Hurst parameter was set to 0.7.

Figure 3 illustrates three different experiments (for different values of the random seed) with the self-similar traffic. Although the controller needs some time to bring the buffer percentile to the reference level, the final error turns out to be acceptable in all cases.

Our numerous experiments (the limited size of this paper makes it impossible to report them all) demonstrate that the scheme is robust in its ability to control widely varying traffic types—without any modification whatsoever to the controller—using Poisson, batch Poisson and self-similar traffic models. This is an improvement over the scheme in [7], where the control effort weighting parameter $\lambda$ must be re-tuned whenever the traffic characteristics change. The improvement is obtained by the use of minimum-bias control [5].

Our controller is able to maintain the QoS at the

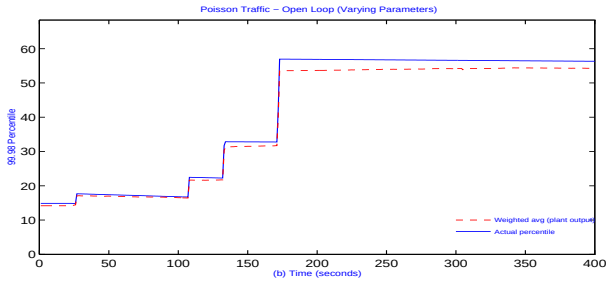


Figure 2: Open loop (effect of varying parameters)

desired level in spite of bit rate variations. Good control is achieved even at high utilizations of 80%. The controller is also able to adapt when the mean arrival rate deviates from its declared value.

The use of the weighted average of the measured buffer percentile and the calculated buffer percentile removes the restriction of large call duration present in [7]. In our case, the QoS requirements are met, even if the connection is very short. The calculated percentile is obtained using the mean and variance of the net arrival process (the auto-covariance sum is not calculated), which reduces the amount of computation that must be performed at each sampling instant.

The use of control methodology makes it possible to utilize the plethora of online process monitoring techniques. The model of the plant estimated at each sampling instant can be examined for stability. Local congestion can therefore be detected, diagnosed, and corrected by the network operator. Large deviations of the plant output from the reference value—especially when the QoS exceeds the reference value by large amounts—can be used to generate alarms, requesting manual intervention.

The initial bandwidth allocated by CAC is used as the mean level about which the instantaneous service rate is varied to achieve the desired QoS. The choice of the initial bandwidth allows the switch to control the average delay, while the maximum delay, delay jitter and cell loss ratio are controlled by the scheduler.

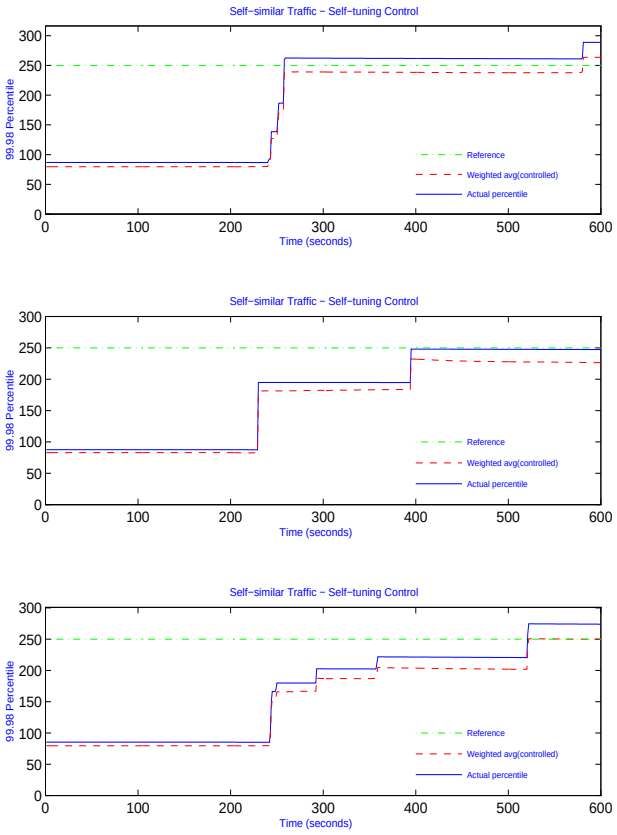


Figure 3: Self-tuning control (self-similar traffic)

# References


[1] R. Addie and M. Zukerman. An approximation for performance evaluation of stationary single server queues. *IEEE Transactions on Communications*, 42(12):3150–3160, 1994.

[2] R. Addie and M. Zukerman. A Gaussian characterization of correlated ATM multiplexed traffic and related queueing studies. *Proceedings of ICC '93*, Geneva, May 1993.

[3] ATM Forum Technical Committee. Traffic management specification v. 4.0. Technical report, Available from ftp://ftp.atmforum.com/pub/approved-specs/af-tm-0056.000ps, 1996.

[4] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. *Journal of Internetworking Research and Experience*, pages 3–26, October 1990.

[5] D. Y. Liu, S. L. Shah, and D. G. Fisher. Multimodel-based minimum bias control of a benchmark paper machine process. *Canadian Journal of Chemical Engineering*, 75:143–151, 1997.

[6] S. Parekh. *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks*. PhD thesis, Massachusetts Institute of Technology, February 1992.

[7] A Pitsillides and J. Lambert. Adaptive connection admission and flow control : quality of service with high utilisation. *INFOCOM '94*, pages 1083–1091, 1994.

[8] P. E Wellstead and M. B Zarrop. *Self-Tuning Systems : Control and Signal Processing*. John Wiley and Sons, Inc. Sussex, England, 1991.

[9] L. Zhang. Virtual clock : a new traffic control algorithm for packet switching networks. *Proceedings of ACM SIGCOMM '90*, pages 19–29, September 1990.