# SP-EPRCA: an ATM Rate Based Congestion Control Scheme based on a Smith Predictor

D. Cavendish, S. Mascolo, M. Gerla *

dirceu@cs.ucla.edu, mascolo@poliba.it, gerla@cs.ucla.edu

**Abstract**

*This report presents a feedback control algorithm for ATM congestion control in which ABR/UBR source rates are adjusted according to VC queue levels at intermediate nodes along the path. The goal is to fully and promptly utilize the available bandwidth left by QOS constrained traffic (CBR+VBR) for transmitting the less QOS stringent ABR/UBR traffic, without incurring in queue overflow and consequently cell loss. In order to obtain this, we propose a simple and classical proportional controller, plus a Smith Predictor to overcome instabilities due to large propagation delays. In the SP-EPRCA model, each queue ideally behaves as a simple first order dynamic system with a delay in cascade. The delay is out of the feedback loop, and therefore does not affect stabilitity. Moreover, since the system dynamic is a first order one, it is not only stable but it has no damped oscillations as well. The real SP-EPRCA implementation, however, may behave as a higher order system due to innacuracies in estimating propagation delays. We show in which conditions system stability is guaranteed under this more realistic scenario. Finally, we propose an effective SP-EPRCA implementation in which each source computes its input rate based on the maximum queue level along the path, which is compatible with the current EPRCA type UNI specifications. Theoretical and experimental results show max-min fairness of the proposed control scheme, its efficiency under the constraints of the EPRCA implementation, as well as its cell loss free property. Moreover, the relationship between throughput, buffer size, and round trip delay is established. Finally, various queue services disciplines are considered in conjunction with the basic scheme.*

*This work was partially supported by CNPq under Grant 201597/93-4(NV) and by NSF under Grant NCR-9305376

# Contents


**1 Introduction** **3**

**2 Continuous time rate based control** **5**

2.1 Network and Queue Models . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

2.2 The Rate Control Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

2.3 Queue Transient Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9

2.4 Steady State Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

2.5 Stability analysis of the controlled system . . . . . . . . . . . . . . . . . . . . . 11

2.6 The case of multiple nodes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17

2.7 Fairness of the control algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . 18

**3 Discrete time rate based control** **19**

3.1 From periodic to aperiodic feedback . . . . . . . . . . . . . . . . . . . . . . . . 21

3.2 A comparison between the PRCA scheme and the proposed SP-EPRCA . . . . . . . 24

**4 Simulation Results** **25**

**5 Implementation Issues** **31**

5.1 Buffer Requirements . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32

5.2 Queue Configurations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33

**6 Conclusive Remarks** **41**

# 1 Introduction

Asynchronous Transfer Mode (ATM) is a fast packet switching technology which supports a broad range of services with distinct requirements for bandwidth, delay and cell loss. An important issue is the "efficient coexistence" of Constant Bit Rate (CBR) services, Variable Bit Rate (VBR) services and "best effort" services (ABR/UBR). ABR/UBR traffic is typically characterized by unspecified requirements for throughput/delay, and it is used to "fill in" the bandwidth left unused by CBR and VBR traffic. However, available bandwidth must be shared in a fair way among all ABR sources.

To be more precise, we model an ATM network as a set of switching nodes connected by communication links. Queues are provided for temporary storage of incoming cells, to be relayed to the next node in its way to its final destination. If the input cell rate to a link is higher than the correspondent output cell rate, then, after the storage capacity is exhausted, all further incoming cells are lost. A link in this condition is called congested. Clearly, congestion is undesiderable because it jeopardizes the ability of the network to deliver cells, and thus causes a degradation in performace.

Therefore, the amount of ABR traffic entering the network must be carefully and dynamically tuned according to the network congestion situation. Too much ABR traffic may have the side effect of queue overflows when fully utilizing the available bandwidth. On the other hand, too little may cause underutilization of available bandwidth, degrading network efficiency. The ultimate goal is thus to regulate the ABR/UBR input traffic rate of the network such that all entering cells can be completely delivered using the existing network resources (i.e. queues, processing power and link trasmission capacity) without exhausting network resources.

In order to achieve high bandwidth utilization, without incurring in cell loss, closed loop feedback control mechanisms are expected to be used to regulate how much ABR/UBR traffic must enter the network according to the actual congestion state of the network.

Namely, unused network resources are monitored continuously, and this information is fed back to a controller, which adjusts the input traffic rates so that congestion is avoided.

Two classes of controllers are devised: rate-based and credit based control. Rate-based control

[7] aims at controlling the traffic input rate, whereas the credit approach [3] aims at regulating the number of incoming cells per feedback information received. Many rate based algorithms can be found in literature. However none of these is completely satisfactory either for its complexity or for lack of stability properties, as is well reported in the excellent paper by Benmohamed and Meerkov [1]. The main difficulty is that, due to large transmission and propagation delays involved in high speed networks such as ATM, most algorithms exhibit persistent oscillations. Furthermore, they have not been analyzed from the stability point of view, and so cannot guarantee the boundedness of the queues. Considering, for example, the well known additive increase/ multiplicative decrease PRCA [5], it is neither possible to state its stability nor to guarantee cell loss avoidance. To our best knowledge, the Benmohamed and Meerkov's paper [1] is the first attempt to develop an analytic method for the design of congestion controllers which ensure good dynamic performance along with fairness in bandwidth allocation. However, the control law proposed in that paper requires a complex adjustment of control parameters in order to maintain stability and damp oscillations. Moreover, these parameters must be dynamically tuned to the specific input traffic and network condition. Finally, it is difficult to prove global stability, due to the complexity of the control strategy.

This report presents a simple and effective rate based congestion control algorithm capable of "filling in" quickly the unused bandwidth with ABR traffic. The main appeal of the proposed congestion control algorithm consists in the use of a simple, first order dynamic model (for the queue levels) in cascade with a delay. This yields the following properties for the ideal mechanism: a) the queue occupancy never exceeds maximum queue capacity (i. e. no cell loss); b) the queue occupancy dynamic is always stable, thus relaxing the need to dynamically adjust parameters in order to stabilize queues or damp oscillations.

The report is organized as follows: First we present a description of the proposed control algorithm in continuous time, theoretically establishing its properties and requirements, such as max-min fairness, buffer requirements and stability. Then a discrete version of the algorithm is presented, where we keep its implementation as close as possible to the current EPRCA ATM congestion control platform [9]. Next we report several simulations results, contrasting the algorithm with the popular PRCA scheme. We then address algorithm adjustments to be made so that various queue schedule schemes may be used in conjuction with the proposed control algorithm.

Conclusions are addressed in the last section.

# 2 Continuous time rate based control

In this section, we first present the network and queue models for a continuous system. Next we describe the control law used throughout the report, and present an analysis of the transient and steady state behavior of the continuous controlled system.

## 2.1 Network and Queue Models

We mainly follow the notation reported in [1]. The network consists of $N = \{1, ..n\}$ nodes and $L = \{1, ...l\}$ links. Each link $i$ is characterized by: transmission capacity $c_i = 1/t_i$ (cells/sec); propagation delay $td_i$; processing capacity $1/tpr_i$ (cell/sec) where $tpr_i$ is the time the switch $i$ needs to take a packet from the input and place it on the output queue. We assume that the processing capacity of each node is much larger than the total transmission capacity of its incoming links so that congestion is caused by transmission capacity only. The network traffic is contributed by source/destination pairs $(S, D)$, where $S, D \in N$. To each $(S, D)$ connection, there is a Virtual Circuit (VC) associated on the path $p(S, D)$. Each source is characterized by its maximum transmission speed, $c_s = 1/t_s$.

Each link maintains a separate queue for each Virtual Circuit VC passing through it. Let $x_{i,j}(t)$ be the occupancy at time $t$ of the queue associated with link $i$ and $VC_j$, and with $X^o_{i,j}$ the corresponding queue threshold level. The control law computes the source input rate $u(t)$ (cells/sec). The bandwidth delay product $td_i/t_i$ represents the number of "in flight" cells on the transmission link.

For the model of the dynamic behavior of each queue, we assume a deterministic fluid model approximation of cell flow. [2]. Each link maintains a separate queue for each Virtual Circuit (VC) passing through it. The reason for this choice is to ensure, through a round robin service discipline, the fair sharing of the link by each $VC$ (to be discussed later). Considering the queue associated with the virtual circuit $VC_j$ at link $i$ , the level of occupancy $x_{i,j}(t)$ at time $t$, starting at $t = 0$ with

$x_{i,j}(0) = 0$, is the integral over the time $(0, t)$ of the difference between the rate of cells entering the queue (say $u_{i,j}(t)$) and the rate of cells leaving the queue (say $d_{i,j}(t)$):

$$x_{i,j}(t) = \int_0^t [u_{i,j}(t' - T_d) - d_{i,j}(t')]dt' \tag{1}$$

where $T_d$ is the transmission delay from the input source to the $i, j$ queue.

## 2.2 The Rate Control Model

We propose a closed-loop control based on feeding back the network queue occupancy. In order to control the queue level $x(t)$ [1] for a specific VC, we initially use a simple proportional controller. Letting $X^o$ be a set point for the queue level, we compute the difference between the set point and the current queue level $x(t)$. This difference, the error $e(t)$, is multiplied by a positive constant gain $K$, so that $Ke(t)$ is the input rate prescribed to the $VC$ source. The proposed control implements the reasonable idea of enforcing an input rate proportional to the room available in the queue. This mechanism seeks to "fill the queue", thus keeping link utilization high.

The calculated input rate $Ke(t)$ at time $t$ will have effect on queue levels only after the round trip delay along the path, i.e. the time that the feedback information needs to reach the source, change the rate value, and finally returns back to the queue as an inflow rate $Ke(t)$. Figure 1 depicts the block diagram of this system, where $T_{fw}$, $T_{fb}$ are the respective propagation delays involved, and $RTD = T_{fw} + T_{fb}$ is the round trip delay. Notice that in wide area networks, the round trip delay is mostly determined by the propagation delays, so we assume that this quantity is estimated and known in advance. However, for ABR traffic, this assumption may be violated if a higher priority traffic is competing with it for network resources. The impact of delay estimation errors is important in this case, and thus will be dealt with later on. Notice that, in figure 1, a generic controller $K^*(t)$ is depicted, rather than a simple proportional controller $K$, for easy of explanation.

[1] From now on we drop the $i, j$ subscripts, for sake of simplicity

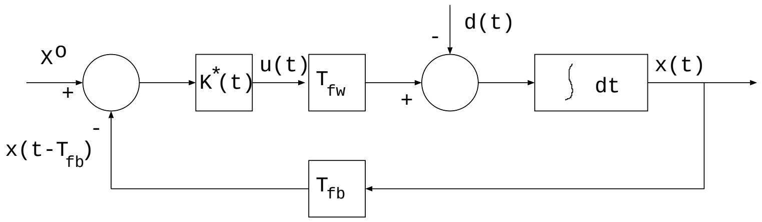



Figure 1: Queue dynamic model with a proportional controller

Lets first use a simple proportional controller, $K^*(t) = K$. Then, due to the large delays, the dynamic behaviour of the queue level might exhibit oscillations, and even become unstable. In order to reduce oscillations, it is necessary to reduce the amplification gain $K$, but this carries the drawback of a very long transient, i.e. the input rate is not able to fill in rapidly the queue, making the outgoing link underutilized [6].

In order to stabilize this system, still preserving the ability of quickly "filling in" the available queue space, we propose a classical Smith Predictor [4]. Following the Smith's principle, we substitute the controller $K^*(t)$ in Fig. 1 with a controller $K^*$ (see Fig. 3) such that the resultant system dynamic is that of a first order system in cascade with a pure delay (Fig.2), in the absence of the input $d(t)$.

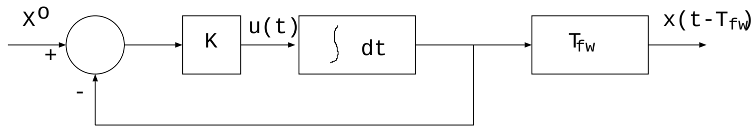



Figure 2: Equivalent model of the system using a Smith Predictor

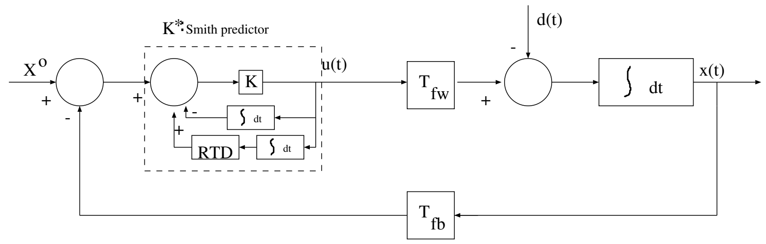



Figure 3: Queue dynamic model using a proportional controller plus a Smith Predictor

Thus, equating the transfer functions of the systems in Figure 3 and in Figure 2 one can verify that the Smith Predictor controller $K^*$ is given by:

$$K^* = \frac{K}{1 + K(\frac{1-e^{-RTDs}}{s})}$$

The Smith Predictor shown in Fig. 3 ($K^*$) gives the following input rate control equation:

$$\begin{aligned} u(t) &= K[X^o - x(t - T_{fb}) - \int_0^t u(t^{'})dt^{'} + \int_0^t u(t^{'} - RTD)dt^{'}] \\ &= K[X^o - x(t - T_{fb}) - \int_{t-RTD}^t u(t^{'})dt^{'}] \end{aligned} \quad (2)$$

Notice that this equation implements a simple proportional control action with the difference that the actual queue level is increased by the number of cells transmitted during the last round trip delay. Thus the physical interpretation is that the controller reacts as if all the "in flight" cells were in the bottleneck queue, which is a worst case assumption for queue occupancy.

In order to describe the dynamic of the system it is helpful to look at the equivalent system shown in Fig. 2. In this figure we can observe two parts:

a)The first one, containing the integrator, the constant gain $K$, and the delay free feedback loop, is a first order system, and thus is stable for every positive value of the parameter $K$. This parameter affects the transient behavior only. Namely $1/K$ is the time constant $T$ of the system (meaning that after 4 $T$ intervals the system reaches stationary condition). Moreover the dynamic response to a step function does not exhibit oscillations in reaching the stationary state. This implies that the queue occupancy never overshoots the set point level $X^o$, and hence the set point can be set equal to the queue capacity without ever incurring cell loss;

b) The second part consists of a pure delay block that causes a shift in time of the queue level x(t).

Concluding, the resulting behavior of the queue occupancy in the absence of $d(t)$, starting at $t = 0$ with an empty queue, is given by the first order system response to a step function, delayed by the round trip RTD, that is: $x_X(t) = X^o[1 - exp(-(t - RDT)/T)]$.

### 2.3 Queue Transient Analysis

The system shown in Fig.3 has a behavior equivalent to the system depicted in Fig.2, in response to the input $X^o$. Now we consider the behavior of the queue level $x_d(t)$ in response to the output rate $d(t)$, where $d(t)$ can be modelled as a step function $a * 1(t)$, and $a$ is the fraction of bandwidth, normalized to one, given to each connection.

Using Laplace transform method, after some calculations, we find:

$$\begin{aligned} x_d(t) &= -a[t * 1(t) - (t - RTD) * 1(t - RTD)] - \\ &\quad \frac{a}{K}[1 - e^{-K(t-RTD)}] * 1(t - RTD) + \\ &\quad x(0) * 1(t) - x(0)[1 - e^{-K(t-RTD)}] * 1(t - RTD) \end{aligned}$$

where $x(0) \geq 0$ is the queue level at $t = 0$. The overall response to $d(t)$ and $X^o$, by the principle of superposition, is therefore given by:

$$x(t) = x_d(t) + x_X(t) \tag{3}$$

In stationary condition ($t \rightarrow \infty$), the queue level is:

$$x(\infty) = X^o - aRTD - \frac{a}{K} \tag{4}$$

Figure 4 shows the transient behavior $x_X(t)$ in response to $X^o$, the transient behavior in response to $d(t) = 1(t) - 0.5 \times 1(t - offset)$ (where 1(t) is the step function), and the overall transient $x(t)$. The initial offset is the time when the bandwidth $d(t)$ drops below the input rate of the corresponding VC queue.

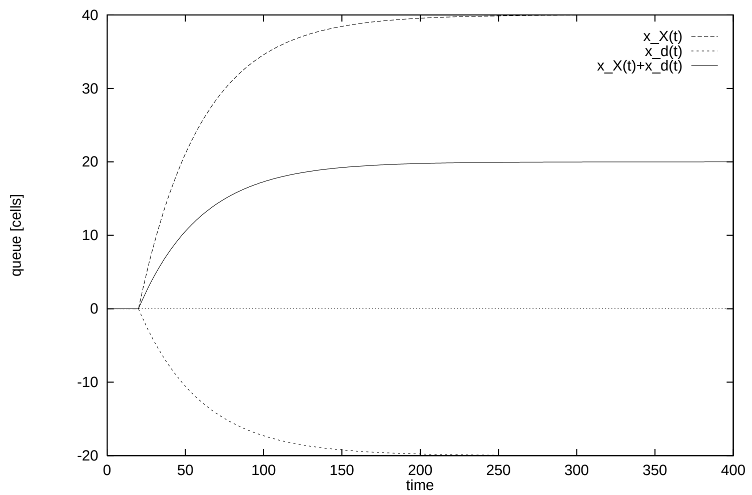



Figure 4: Queue level transient dynamic

### 2.4 Steady State Analysis

As time approaches infinity, equation (2) gives us the following relation between stationary rate $u_s$ and stationary queue $x_s$:

$$u_s = \frac{X^o - x_s}{1/K + RTD} \tag{5}$$

As far as the controller is concerned, any point in the plane $u, x$ satisfying equation (5) is a stable point for the system. This stability plane extends from 0 to $X^o$ in the queue level axis and from 0 to $c_b$ in the rate axis, $c_b$ being the maximum capacity of the bottleneck link. Later we will see that the queue service discipline, initially assumed round robin, may determine which stable point the system is going to rest upon. Notice that if a single connection must be allowed to achieve the maximum bottleneck transmission speed $c_b$ ($c_b = u_s$), equation (5) ultimately states that we need buffers proportional to the product of the round trip delay $RTD$ by the maximum transmission speed $c_b$. To see this clearly we only need to make $K$ go to infinity, thus providing the fastest controller possible in terms of responsiveness, which is the one requiring smallest buffer size. Since $u_s = c_b$ corresponds to having the full bottleneck capacity for a single source/destination pair, so that $x_s = 0$. The buffer requirement is then:

$$X^o = c_b RTD \tag{6}$$

This stringent buffer requirement results from the fact that we insist on designing a cell loss free algorithm. If we are willing to accept some loss, buffer requirements can be reduced. We will address this trade off shortly.

### 2.5 Stability analysis of the controlled system

For general systems, a common stability criterium is to require that the zeroes of their characteristic equation have all negative real parts. For the particular system shown in Figure 2, the characteristic equation is $s+K$, which is stable for any positive gain $K$, since for $K > 0$ this equation has a single negative real zero. However, the system shown in Figure 2 represents our controlled system only if the delays $T_{fw}, T_{fb}$ are known in advance, so that $RTD = T_{fw} + T_{fb}$ box inside the controller (see Fig. 3) matches perfectly the propagation delays present in the feedback loop of the system. In reality, these delays must be estimated, and thus mismatches are likely to happen. Therefore, it is important to study how the controlled system behaves in the presence of delay mismatches. We address this question in this subsection.

We start assuming that the RTD estimated by the source is $RTD = d_o$, while the real round trip delay is $RTD' = d_o + d$. We are not concerned about which among the delays $T_{fw}, T_{fb}$ were misestimated, since as we will shortly see this detail does not affect the characteristic equation. We require that $d_o + d \geq 0$, or $d \geq -d_o$, so that the real round trip delay be non-negative. The system is still represented by Fig. 3, with only $T_{fw} + T_{fb} = d_o + d$ and $RTD = d_o$. After some algebraic manipulation, the transfer function $u(s)/X^o$ is found to be:

$$\frac{u(s)}{X^o} = [\frac{1}{K} + \frac{1}{s} + \frac{e^{-(d_o+d)s} - e^{-d_o s}}{s}]^{-1} \frac{e^{-T_{fw}}}{s} \tag{7}$$

Since equation (2) is still valid, the steady state equation (5) still applies, hence if the system is stable it settles to the same point as if there was no delay mismatch. Our concern here is then whether the system is stable or not. We here follow a similar approach to Walton and Marshall [11] for stability analysis of time-delay systems. The minor difference is that in the present case the delay parameter $d$ can be either positive or negative, since it represents a variation from the estimated delay $d_o$, rather than a strictly positive delay. From Equation (7), the characteristic equation is given by:

$$F(s,d) = s + K + Ke^{-d_o s}(e^{-ds} - 1) \tag{8}$$

We first examine the zeroes of $F(s,0)$, which correspond to the case of a perfect estimation of the propagation delays. Trivially, we have a single real zero at $s = -K$, which is negative for any $K > 0$. Next we consider a infinitesimally small $d$, in which case a infinite number of new zeroes appear, due to the exponentials in Equation (8). It is necessary then to study the locations of these new zeroes in the complex plane. Finally we need to find values of $d$, if any, at which there are zeroes of Eq. (8) lying on the imaginary axis, i.e. find $w, d$ values for which $F(iw,d) = 0$, and then determine whether the root locus plot merely touches the imaginary axis or it crosses from one half-plane to the other with increasing $d$. If the root locus plot crosses from left to right, increasing $d$ destabilize the system, otherwise increasing $d$ stabilize the system.

We start studying the zeroes of Eq. (8) at the imaginary axis. Namely, $s = \pm wi$ such that:

$$F(wi,d) = wi + K + Ke^{-wd_o i}(e^{-wdi} - 1) = 0$$

which, after some algebraic manipulation, results:

$$e^{-wdi} = \frac{K(\cos wd_o - 1) + (-K\sin wd_o - w)i}{K\cos wd_o + (-K\sin wd_o)i} \tag{9}$$

If there exists real $d$ which satisfies Eq. (9), then by equating the imaginary and real parts of both sides of Eq. (9) there must exist a pair $w, d$ of real values satisfying the following equations:

$$\sin wd = \frac{w}{K}\cos wd_o + \sin wd_o \tag{10}$$
$$\cos wd = 1 + \frac{w}{K}\sin wd_o - \cos wd_o \tag{11}$$

Moreover, if for a particular system described by $K, d_o$, we find a pair $w^{'}, d'$ such that Eqs. (10,11) are satisfied, then it is easy to see that:

$$d = d^{'}(w^{'}) + \frac{2\pi q}{w^{'}}, q = 0, 1, 2, \cdots \tag{12}$$

also satisfy Eqs. (10,11). So, once a minimum/maximal [2] $d$ satisfying Eqs. (10,11) is found, we use Eq. (12) to find the other infinite zeroes lying on the imaginary axis. Notice that $w^{'}$ is independent of the delay mismatch $d^{'}$, which can be seen by removing $d$ from these equations using the fact that $\sin^2 wd + \cos^2 wd = 1$. By doing this, we first find $w^{'}$ from the following equation:

$$2\cos wd_o = 1 + \frac{w^2}{K^2} + \frac{2w}{K}\sin wd_o \tag{13}$$

and then use either Eq. (10) or Eq. (11) to determine $d^{'}$. Notice that equation (13) has two solutions, $\pm w^{'}$. Unfortunately, numerical methods are necessary to determine solutions of this equation for non-trivial systems.

We next need to determine which among the roots found in the last computation are stabilizing, and which are destabilizing. We do this by studying the sign of the $\frac{ds}{dd}$ derivative [3] at the imaginary zeroes computed. From Eq. (8), this derivative is given by:

$$\frac{ds}{dd} = \frac{Kse^{-(d_o+d)s}}{1 + Kd_oe^{-d_os} - K(d_o+d)e^{-(d_o+d)s}} \tag{14}$$

[2]Minimum for positive $d$, maximum for negative $d$.

[3]We apologize for the double d notation. The first d stands for derivative, as usual.

If we define $S$ as:

$$S \equiv \text{sgn Re}\frac{ds}{dd}(wi)$$

after some manipulation we find:

$$S = \text{sgn Re}\left[\frac{Kw\sin w(d_o+d) + Kw\cos w(d_o+d)i}{1 + Kd_o\cos wd_o - K(d_o+d)\cos w(d_o+d) + [K(d_o+d)\sin w(d_o+d) - Kd_o\sin wd_o]i}\right]$$

We are particularly interested in the positive sign of the previous expression, since for a minimum/maximum $d'$ found, a positive $S$ gives us the maximum/minimum delay mismatch over which the system becomes unstable. We find it convenient to define:

$$\begin{aligned}
m &\equiv Kw\sin w(d_o+d) \\
n &\equiv Kw\cos w(d_o+d) \\
o &\equiv 1 + Kd_o\cos wd_o - K(d_o+d)\cos w(d_o+d) \\
p &\equiv K(d_o+d)\sin w(d_o+d) - Kd_o\sin wd_o
\end{aligned}$$

Since we are looking for points which satisfy Eqs. (10, 11), we use these equations, plus some fundamental trigonometric identities to obtain:

$$\begin{aligned}
m &= Kw\sin wd_o + w^2 \\
n &= Kw(\cos wd_o - 1) \\
o &= 1 + Kd_o + Kd(1 - \cos wd_o) \\
p &= w(d_o+d) + Kd\sin wd_o
\end{aligned}$$

With the above definitions, the points at which $S$ is positive are the ones such that:

$$\frac{mo + np}{o^2 + p^2} > 0$$

or

$$(Kw \sin wd_o + w^2)[1 + K(d_o + d) - Kd \cos wd_o] + [w(d_o + d) + Kd \sin wd_o]Kw(\cos wd_o - 1) > 0$$

After some manipulation, the previous relation reduces to:

$$Kw \sin wd_o(1 + Kd_o) + w^2(1 + Kd_o \cos wd_o) > 0 \tag{15}$$

Before going any further, some remarks are due here. First notice that relation (15) contains no $d$, which means that all crossing points at the imaginary axis for a given system behave the same way, namely they are either all destabilizing or all stabilizing. So we can conclude that the root locus plot of our system has a similar shape to the one shown in [6] for a simpler time delay system.

Notice also that the validity of relation (15) is determined by the particular $w'$ computed from equation (13), which again depends solely on $(K, d_o)$ parameters, not on $d$.

Using equation (13), the above relation becomes:

$$\frac{K}{d_o}[1 + d_o(K + \frac{w^2}{K})]wd_o \sin wd_o + [\frac{1}{d_o^2} + \frac{1}{2d_o}(K + \frac{w^2}{K})](wd_o)^2 > 0 \tag{16}$$

which is of the form $f(w)w \sin w + g(w)w^2 > 0$, where $f(w), g(w) > 0$ for all $w$, given that $K, d_o > 0$. It is not difficult to prove that relation (16) holds for any real $w$. Therefore, we conclude that, if there are roots of the system characteristic equation lying at the imaginary axis, they are all destabilizing, regardless of which particular system $(K, d_o)$ we are dealing with.

Summarizing all observations we have made in this subsection about a system described by $(K, d_o)$, we can sketch few important points of the system root locus plot, as shown in Figure 5.

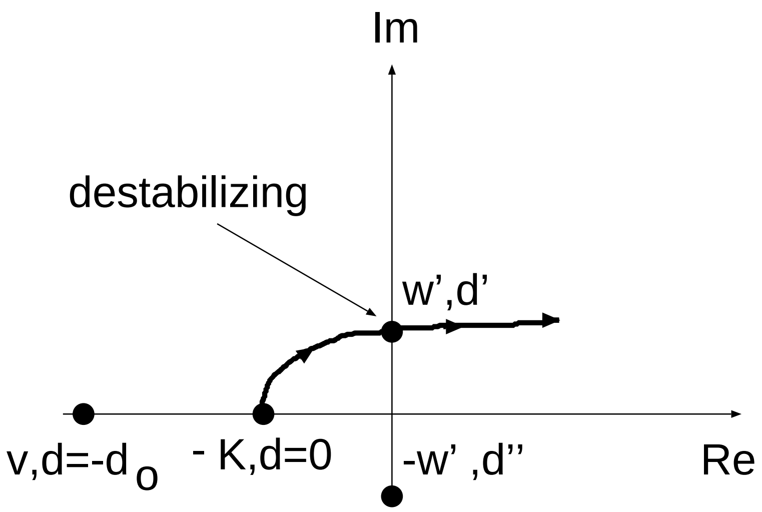



Figure 5: Sketch of the system root locus plot

Figure 5 shows the complex plane, with few points $v + iw$ describing some of the roots which are both easy to compute and sufficient to define the stable region of the system. Notice that this figure does not show all root locus plot of the system, but only zoom at the important part for our stability analysis. Each point is described by its coordinates $v, w$ in the plane plus the mismatch value $d$ at that point. We know that there exists a real negative root at $v = -K, d = 0$ (no mismatch). We have included also the point $v, d_o$ to stress that points in the real axis to the left of this point are not meaningful, since they represent negative real round trip delays. An important observation here is that for $d_o \leq d \leq 0$, the root locus plot can never touch the imaginary axis, for if this was the case the cross point would be stabilizing, which we have shown not to exist. This means that for $d_o \leq d \leq 0$, the system is always stable, which makes sense since essentially the control algorithm is using more information about the past than necessary.

The stability analysis of the controlled system reduces to the following steps. First, by solving equation (13), we find $\pm w'$ values. Next we compute the maximum delay mismatch $d'$ from Eq. (11 or 10), which gives us the first and most important destabilizing point, depicted in Figure 5. Clearly there is a positive range of delay mismatch $[0, d')$ in which the system stability is guaranteed despite the delay mismatch. This concludes the stability analysis of the controlled system.

### 2.6 The case of multiple nodes

So far we have modelled our system as having a single bottleneck node, situated $T_{fw}$ away from the source, and having a propagation delay back to the same source of $T_{fb}$ units of time. We would like to extend the results so far obtained for single node systems for more general systems. Namely, multiple node systems, possibly with dynamic change of the bottleneck node during the lifetime of the connection are more realistic systems, and therefore we would like to rate control these systems also. We first argue that when other less congested nodes are present in the feedback loop, the single node model we have used so far can still be applied. Then we argue that moving bottlenecks does not pose any additional modelling problems either.

We can model the system with multiple nodes as a collection of single node models as previously described. Each single node model is controlled according to equation (2). Clearly all controllers have the same $RTD$, and moreover we assume uniform control gains $K$ and queue sizes $X^{o}$[4]. We also assume that feedback information is not delayed at intermediate nodes, which is similar to our early assumption for single node that propagation delays dominates in the relaying of feedback information over other types of network delays.

The source input rate is determined as the lowest rate among all input rates computed, otherwise violation of one or more prescribed rates may occur.

**Lemma 1** *The prescribed input rate comes from the node with largest queue level.*

**Proof Lemma 1** *Since the integral of equation (2) is the same for all controllers, the Lemma follows immediately. Notice that the age of the feedback information is not important, as far as rate computation is concerned.*

Lemma 1 guarantees that all nodes but the bottleneck one may be ignored in the modelling of multiple node systems. Notice that we can use a similar argument to claim that the control algorithm proposed works equally well when bottlenecks move from node to node.

[4]The extension to various queue sizes is not difficult

**Theorem 1** *Under the same assumptions of Lemma 1, moving bottlenecks do not invalidade any results obtained by single node systems.*

**Proof Theorem 1** *It follows from Lemma 1 and the fact that, as far as its proof is concerned, it does not matter if the age of feedback information changes with time, which happens when the bottleneck moves from one node to another.*

## 2.7 Fairness of the control algorithm

In the congestion control context, fairness is generally defined as the property of exercizing traffic blocking or rate reduction in a "fair" way among all connections. Although fairness can be defined precisely in a number of ways, in the flow control application the most common definition is the so called *max-min fairness* [10]. We here define max-min fairness in the rate control scenario.

Let $r$ be a rate vector, whose components are connections input rates of a general network, defined previously as links $l$ with finite transmission capacity $c_l$ interconnecting nodes. We need the following definitions [10]:

**Definition 1** *A feasible rate vector $r$ is a vector $(r_1, r_2, ..., r_n)$ of rates such that $\sum r_i \leq c_j$ over all connections $i$ sharing link $j$, for all network links.*

**Definition 2** *A feasible max-min rate vector $r$ is a feasible rate vector such that any attempt to increase a given rate $r_i$ must result in a decrease of another rate $r_j, r_i \geq r_j$ in order to maintain feasibility.*

The following theorem shows that max-min fairness is achieved by the proposed rate control algorithm.

**Theorem 2** *The Smith Predictor rate based control algorithm is max-min fair. More precisely, the rate vector of* **stationary rates** *is max-min fair as defined in Definition 2.*

We first need the following lemma:

**Lemma 2** *The rate vector whose components are stationary rates computed by the proposed rate control algorithm is feasible.*

**Proof Lemma 2** *Pick a generic link $i$. If this is not a bottleneck link for any connection, it does not even have to be modelled, as argued in the previous section. So let $i$ be a bottleneck link for some connection $j$. In stationary condition, whatever the stationary rates of bottleneck $b$ connections are, their sum do not exceed capacity $c_i$ for if this was the case queues would overflow. But since the system behaves as a first order one, with the queue size as the set point, queues can never overflow.*

Now for the proof of the Theorem.

**Proof Theorem 2** *The proposed algorithm assumes that each connection $p$ has a bottleneck link $b$ (modelled by the control model). Moreover, feasibility is guaranteed by the previous lemma at stationary regime. So we pick a generic connection $p$, which must have a bottleneck $b$ associated to it shared by $n_b$ connections. At stationary regime, if we try to increase its rate, we must decrease the rate of another connection $q$ sharing the same bottleneck link in order to maintain feasibility, since by definition of bottleneck $\sum r_i = c_b$. Moreover, we know that $r_i = c_b/n_b$, due to the round-robin service discipline, and thus $r_p = r_q$. Thus connections $p, q$ comply with definition 2. Since $p$ is a generic connection, the Theorem follows.*

# 3 Discrete time rate based control

So far we have dealt with continuous time models only. However, in ATM, feedback information is relayed in cells, and thus not available in continuous time, but rather in sampled form. Fortunately, the discrete time implemention of the Smith Predictor is simpler than the continuous one [8].

We start with the system model shown in Fig. 1. From Nyquist sampling theorem and from control theory we know that, in order to have a "continuous like" performance of the system under digitized control, the ratio of the time constant of the system over the sampling time must be at least 2 and can not get any better if it is beyond 4[8]. Indicating by $\triangle$ the sampling time , it follows:

$$\frac{1}{\triangle K} = [2, 4] \tag{17}$$

In order to write the discrete time version of the control equation (2) we must consider two cases:

i) $RTD \geq \triangle$: The ratio $RTD/\triangle = m + \epsilon$ where m is an integer and $\epsilon \in [0, 1)$. Rewriting the continuous time equation 2 in its discrete version, we obtain the input rate at time $t_k = k\triangle$:[5]

$$u(k\triangle) = K[X^o - x(k\triangle - T_{fb}) - u(k\triangle - (m+1)\triangle)\epsilon\triangle - \sum_{i=1}^{m} u((k-i)\triangle)\triangle] \tag{18}$$

ii) $RTD < \triangle$:

$$u(k\triangle) = K[X^o - x(k\triangle - T_{fb}) - u((k-1)\triangle)RTD] \tag{19}$$

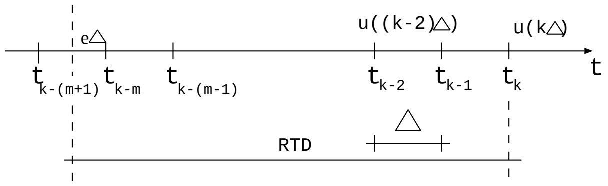



Figure 6: Discrete Time Notation

The notation used in the previous equations is illustrated in Figure 6 and will be followed throughout the paper. It results: $t_k = t_{k-1} + \triangle$.

The summation on the right side of the equation 18 can be rewritten as the sum of two parts:

$$\begin{aligned} I &= u(t - T_{fw} - \triangle)\triangle + u(t - T_{fw} - 2\triangle)\triangle + \cdots + u(t - RTD)\triangle; \\ II &= u(t - \triangle)\triangle + u(t - 2\triangle)\triangle \cdots + u(t - T_{fw})\triangle \end{aligned}$$

[5] Note that k should not be confused with the gain K

The first one represents the number of cells that have already arrived at the bottleneck queue but are not yet know at the source due to the feedback propagation delay $T_{fb}$. The second one represents the number of cells that are travelling from the source to the queue. Therefore the input rate computation at time $t$ can be rewritten as: $u(t) = K[X^o - x(t - T_{fb}) - (I) - (II)]$. We can interpret $x(t - T_{fb}) + (I) + (II)$ as "effective queue level at time $t$". So the calculation of the input rate $u(t)$ is made as if all "in flight" cells were already at the queue. In this way the dynamic is delay free, which results in stability and lack of oscillations.

Finally we would like to interpret the difference between the queue capacity and the "effective queue level" as a number $Q$ of cells that can be transmitted by the source without causing overflow to the bottleneck queue.

### 3.1 From periodic to aperiodic feedback

In order to implement the proposed discrete time control algorithm we need to supply the controllers located at the sources with periodic feedback information (every $\triangle$ units of time, with $\triangle$ satisfying equation (17)). This can be obtained if the upstream node of a congested link sends the feedback information, at every sampling time, to all the sources in the upstream direction, as in the Backward Congestion Notification (BCN) scheme. This is what is assumed in [1]. We call this type of scheme "Periodic Feedback Control". In systems where Forward Congestion Notification (FCN) scheme is used, like in the PRCA scheme, the source is responsible for transmitting a management cell $RM$ every $NRM$ data cells. The control cell itself has to compete for bottleneck link bandwidth, since it has to reach the destination node before being relayed back to the source through either the same or an alternative reverse path. Clearly, under this scheme, it is not possible to guarantee the periodic feedback information used in the discrete-time control equation. Due to the sharing of the congested link, the rate of the feedback cells that can be received by the sources is $B_{av}/[N_{vc} \times (1 + NRM)]$, where $N_{vc}$ is the number of Virtual Circuits sharing the same bottleneck link and $B_{av}$ is the available bandwidth. Thus the interarrival time of the feedback cells increases as $N_{vc} \times (1+NRM)$. Because it is necessary to guarantee the sampling time equation (17) in order to achieve a good performance of the feedback control, we have to increase the time constant of the system (i.e. the queue size per VC) as $N_{vc}$ increases. Thus, it is necessary to use VC queue sizes proportional to the number of

Virtual Circuits sharing the same bottleneck link. This requirement does not derive from the control algorithm itself but rather from the FCN mechanism used for delivering feedback information. To cope with the somewhat irregular delivery of RM cells, we need a control algorithm which must operate well even if no RM cells are received for a while. If the source receives the feedback information, the control algorithm will adjust the rate accordingly. Otherwise it will compute the rate by estimating the missing feedback information in a conservative way. In other words, this algorithm must perform some kind of "virtual feedback". We call this type of control "Aperiodic Feedback Control".

We now propose a version of the previous discrete time control algorithm suitable for the FCN feedback relay scheme. The feedback information is provided by RM cells which collect the maximum buffer level along the path. Note that regardless of the bottleneck location along the VC path, RTD is always the same. The system will still be cell loss free even if we are not able to guarantee the required periodicity of feedback information.

The basic idea is to update the source rate at least after each $\triangle$ sampling interval, regardless whether the source gets the feedback information or not. Let $t_k, t_{k+1}$ be the instants at which the source receives the last and current feedback information, respectively. Two cases need to be considered:

i) $t_{k+1} - t_k \leq \triangle$. The source stores the rate $u(t_k)$, as well as its duration $\triangle_k$, so that $u(t_k)\triangle_k$ becomes one of the terms of the summation in the control equation. Thus the rate updating equation is:

$$u(t_k + \triangle_k) \quad = \quad K[X^o - x(t_k + \triangle_k - T_{fb}) - \sum_{i=0}^{m} u(t_{k-i})\triangle_{k-i} - u(t_{k-m-1})(RTD - \sum_{i=0}^{m} \triangle_{k-i})]$$

where $\sum_{i=0}^{m} \triangle_{k-i} \leq RTD < \sum_{i=0}^{m+1} \triangle_{k-i}, \qquad \triangle_{k-i} \leq \triangle \qquad \forall i, \qquad t_k = t_{k-1} + \triangle_{k-1}.$

ii) The interval $\triangle$ expires before the source receives its control packet. In this case, the algorithm has to estimate the queue level $x(t_k + \triangle - T_{tb})$. In order to be conservative, and to prevent cell loss, we propose the following "worst case" estimate of the missing queue level. We conservatively assume that in the time interval $[t_k, t_k + \triangle]$ (with $\triangle = \triangle_k$) the queue has zero output rate. Thus the "worst case" queue level is the last value $x(t_k - T_{fb})$ plus what has been received in the interval $[t_k, t_k + \triangle]$. The accrued term corresponds to the number of cells pumped into the network during the interval $[t_k - RTD, t_k - RTD + \triangle]$. Therefore, the "worst case" estimate of the queue level at time $t_k + \triangle_k$ is:

$$
\begin{aligned}
x(t_k + \triangle_k - T_{fb}) = x(t_k - T_{fb}) + \\
& u(t_{k-m-2)})(RTD - \sum_{i=1}^{m+1} \triangle_{k-i}) + \\
& u(t_{k-m-1})(\triangle - (RTD - \sum_{i=1}^{m+1} \triangle_{k-i}))
\end{aligned}
$$

We call "virtual feedback" this worst case estimation of the queue level . Note that this is equivalent to storing the last received feedback value, $x(t - T_{fb})$, and adding the new term $u(t_k)\triangle$ to the last sum of "in flight" cells, say $sumF$, i.e.

$$
sumF = \sum_{i=1}^{m+1} u(t_{k-i})\triangle_{k-i} + u(t_{k-m-2})(RTD - \sum_{i=1}^{m+1} \triangle_{k-i})
$$

and the rate is

$$
u(t_k + \triangle_k) = K[X^o - x(t_k - T_{fb}) - u(t_k)\triangle_k - sumF]
$$

In this proposed EPRCA algorithm, the sources at the edge nodes of the network update their input rates at least every $\triangle$ units of time. If they do not get information about the occupancy of the congested queue, they decrease their rates based on a "worst case" estimate of the congested queue level. When they get the next feedback information they will increase their rates because the actual queue level cannot be larger than the conservative estimate. In other words, the algorithm behaves as a "positive feedback ", decreasing the rate when feedback is not available and increasing it when feedback information resumes. Note that this is very important aspect to guarantee stability in any feedback congestion control because, due to congestion, it is not possible to guarantee the rate at which feedback cells are received.

### Correctness of the proposed algorithm

Our control algorithm collects the maximum queue length along the feedback loop of a given source/destination pair. With this procedure we guarantee that the rate control is performed based on the largest queue level. Therefore, given that the discrete system follows its continuous version, by Lemma 1 and Theorem 1, the correctness of the algorithm follows.

## 3.2 A comparison between the PRCA scheme and the proposed SP-EPRCA

In the ATM Forum PRCA proposal [5], an additive increase/multiplicative decrease rate control is exercised at the sources. Binary feedback information ( congested/ not congested ) is received at the sources, and rate increase (additive) is performed in case a "not congested" feedback is received. Failure to receive the "not congested" notification causes multiplicative rate decrease at the source after each time interval $\triangle$, thus making the scheme conservative.

Our proposed EPRCA uses the delayed queue occupancy as the feedback information. Like in the PRCA scheme, if no feedback is received, the source calculates the rate at fixed intervals $\triangle$ related to the time constant of the queue. The calculation is performed using a "worst case" estimate of the queue level.

In the following, we study the dynamic behavior of the rate when the source lacks feedback information. Let $u(0) = \frac{1}{X^o t_s}(X^o - x(t - T_{fb}) - \sum_{RTD} u(t_i)\triangle_i)$ be the rate computed based on the last received feedback cell. If no feedback information is received since then, the rate must be updated every $\triangle$ unit of time, using the "worst case" estimate. It follows :

$$\begin{aligned}
u(1) &= \frac{1}{X^o t_s}[X^o - x(t - T_{fb}) - u(0)\triangle - \sum_{RTD} u(t_i)\triangle_i] \\
&= \frac{u(0)}{X^o t_s}(X^o t_s - \triangle) \\
u(2) &= \frac{u(1)}{X^o t_s}(X^o t_s - \triangle) \\
&\vdots \\
u(k) &= \frac{u(k-1)}{X^o t_s}(X^o t_s - \triangle) = u(0)\left[\frac{(X^o t_s - \triangle)}{X^o t_s}\right]^k
\end{aligned}$$

i.e. the rate decreases exponentially. When the source resumes receiving feedback information, the rate jumps to $(X^o - x(t - T_{fb}) - \sum_{RTD} u(t_i)\triangle_i)$.

Therefore, we note that our EPRCA scheme, developed from a precise and simple mathematical model, operates according to a "positive feedback" mechanism, much like the PRCA scheme. The important difference is that the dynamic behavior of our regulation is related to the network state and parameters. In fact, the rate decreases exponentially with a base related to the sampling time/time constant ratio. More importantly, the increasing jumps are related to the queue level and to the number of cells released from the source during the last round trip interval. As a consequence, our EPRCA scheme does not drop cells, nor does it need a queue size proportional to the RTD to prevent cell loss. In contrast, the conventional PRCA scheme does not use precise information on the queue level and does not take into account the number of cells released during the last round trip delay. Consequently, it cannot perform the correct rate increase so as to prevent congestion and cell loss.

# 4 Simulation Results

In this section, we present results of a discrete event simulation of our control scheme. We first show the performance of the periodic control algorithm under the same scenario considered by [1]. Then we compare the proposed EPRCA scheme with the conventional PRCA scheme [5].

The network topology, shown in Fig. 7, is similar to the one presented in [1]. Links have uniform speeds, normalized to 1 cell per unit of time [cell/s]. They are labeled with their respective propagation delays, normalized to the uniform transmission time. In an 150Mbps link speed, 10 units means roughly a 6 mile distance between switches.

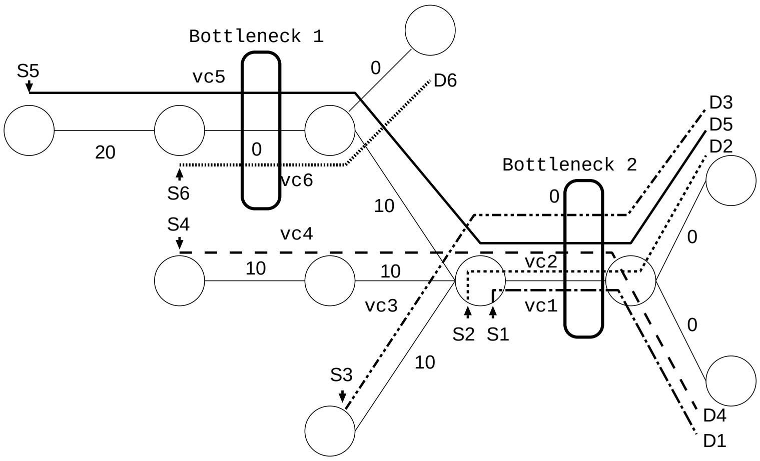



Figure 7: Network Topology - Multiple Bottlenecks

In this initial setting, five VC connections compete for bandwidth resources of the bottleneck #2 link only(connection # 6 is innactive). Thus, queue dynamics are shown for this bottleneck only. VC connection activity (i.e. start and end time) is described in Table 1. We assume infinite backlog at each source. We set a queue level $X^o = 40$ for each queue, in order to have a sampling time of the system of $40/4 = 10$, that is, one fifth of the interarrival time of the feedback cells under the FCN scheme, with $NRM = 10$ and $N_{vc} = 5$.

Table 1: VC Connections Activity

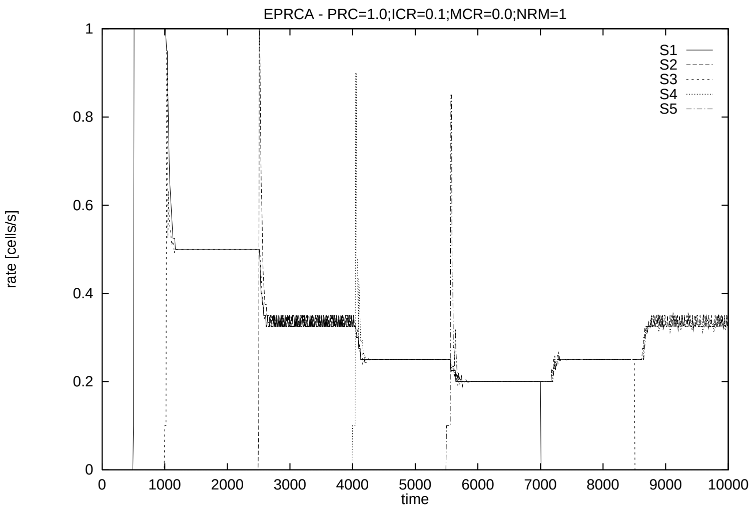
| Connection # | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Start Time | 500 | 2500 | 1000 | 4000 | 5500 | 0 |
| End Time | 7000 | 10000 | 8500 | 10000 | 10000 | 0 |
| RTD | 0 | 0 | 20 | 40 | 60 | 0 |

*Periodic Feedback*

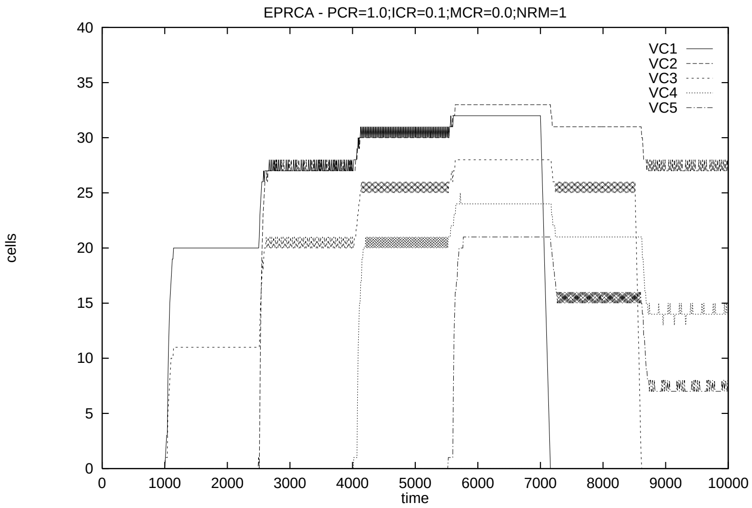
We first show the performance of a periodic sampling version of our control scheme, in conditions similar to [1].

According to equation (17) we choose a sampling time $\triangle = 10$. Figure 8(a) shows the behavior of the five input rates, corresponding to connections $S1 - S5$, at source nodes. For sake of comparison with PRCA, we assume an initial cell rate of 0.1 [cells/s], equal to the PRCA minimum cell rate.

After the start/end of a connection, each rate rapidly settles to the new fair stationary value [6]. Figure 8(b) shows the dynamic behavior of the five queues at the bottleneck link, corresponding to $VC1 - VC5$ bottleneck queues. As can be seen, no queue overflow occurs. Moreover, each stationary level is in accordance with equation ( 4 ). The overall performance is similar to [1]'s periodic control, without having dynamic tuning of control parameters.




(a) Rates




(b) Bottleneck Queues

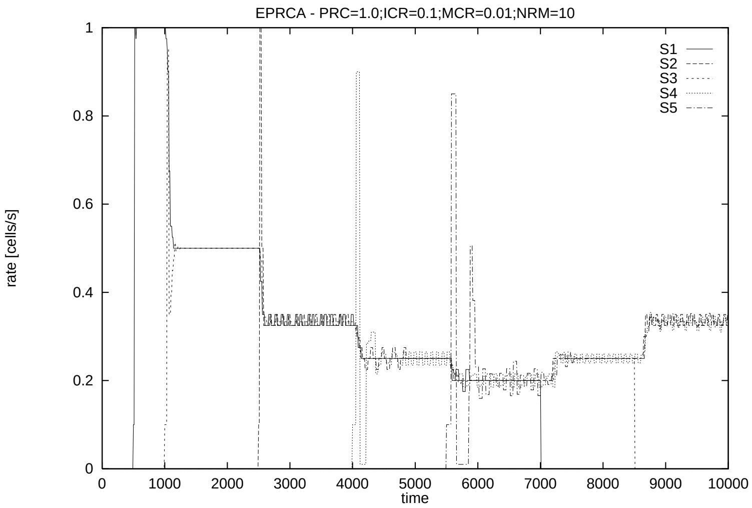
Figure 8: Periodic Feedback

[6]When there are three active connections, the figure shows small oscillations, due to the fact that the control equation tries to regulate the queue occupancy to a value between two integers

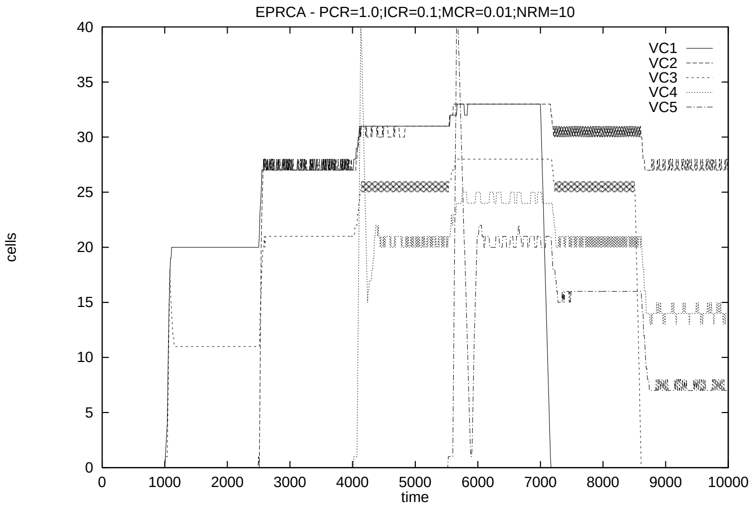
*Aperiodic Feedback*

The aperiodic control scheme, with $\triangle = 10$, requires $NRM = 1$ (i.e. one control cell every data cell) in order to guarantee the minimum feedback frequency rate. The value $NRM = 1$ derives from the fact that, under the heaviest traffic condition (five connections), the feedback cell interarrival time is $\triangle = N_{vc}(NRM + 1)$. Since the minimum feedback rate is maintained, simulation results are identical to the ones under periodic control, as expected, and hence are omitted.

We next show performance degradation in case equation (17) is not respected. By setting $NRM = 10$, under the heaviest traffic condition, the feedback interarrival time is $55 > 10$. We see from Fig. 9(a) that the rate does not reach rapidely the stationary condition anymore. Moreover, Fig. 9(b) shows that overflow occurs in VC4 and VC5 queues. Other simulation results, not reported here, show that the greater the NRM value, the less controlled the queue levels are.




(a) Rates

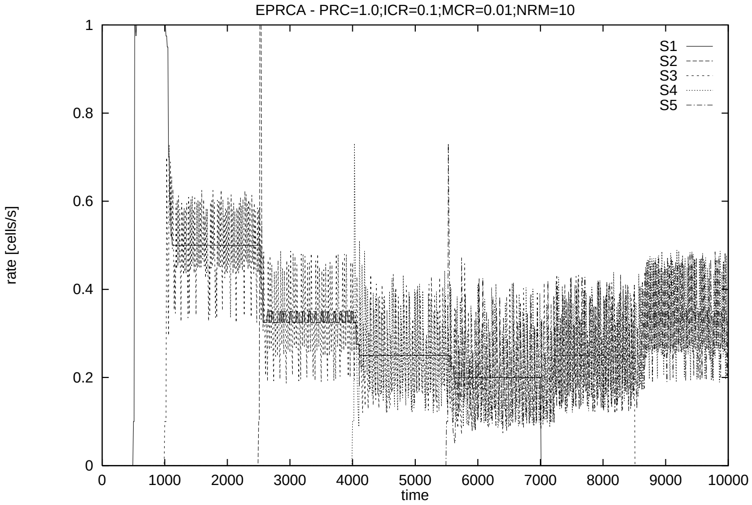



(b) Bottleneck Queues

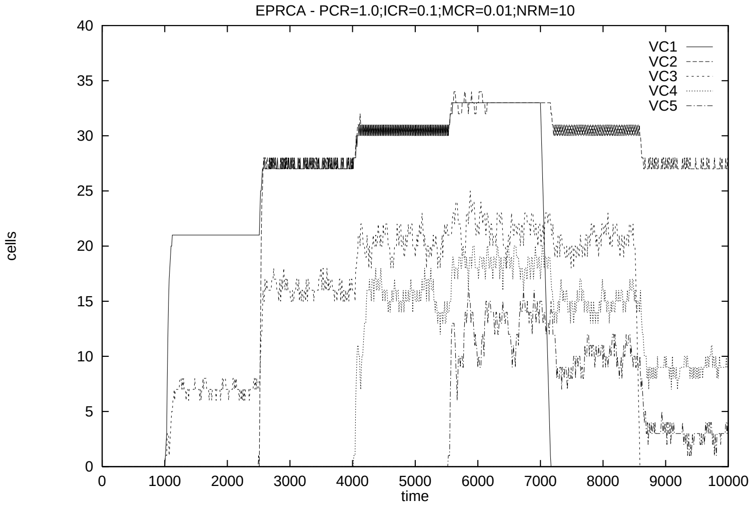
Figure 9: Aperiodic Feedback

*Aperiodic + Virtual Feedback (SP-EPRCA)*

Next, we study the performance of the proposed SP-EPRCA under the same conditions and feedback frequency ($NRM = 10$), as used above. Figure 10(a) shows the oscillatory behavior of the controlled rates. This is so because the control operates in the "positive feedback" mode, i.e. increasing promptly the rate when a feedback cell is received, and decreasing exponentially otherwise. However, the oscillations are constant in amplitude, and centered at the fair value of the rate, so that the throughput performance is preserved. The frequency of oscillations is high because the virtual feedback period is $\triangle = 10$, while the actual feedback interarrival time is about 50. In fact, the control algorithm decreases the rate every $\triangle = 10$, in a conservative way, increasing it promptly, when a feedback cell is received (approx. every 50 units of time). Figure 10(b) shows that the queue levels are still bounded, guaranteeing no cell loss. Thus, the major advantage of the Virtual Feedback scheme is to prevent cell loss, due to congestion, even if it is not possible to guarantee the frequency of feedback cells.




(a) Rates

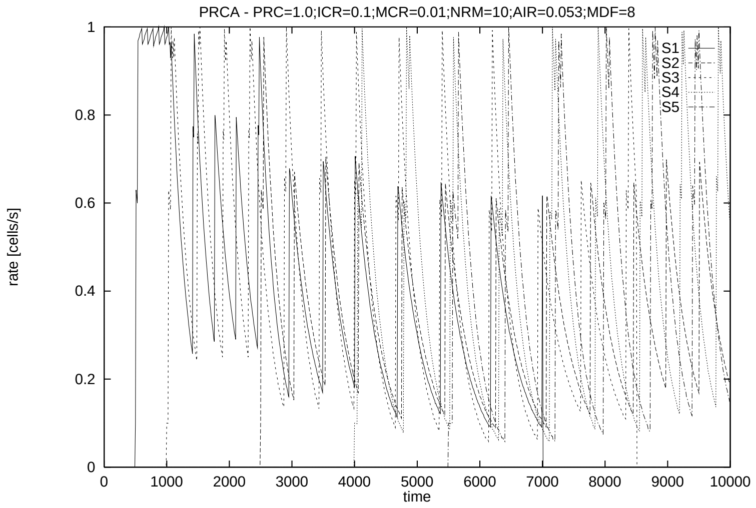



(b) Bottleneck Queues

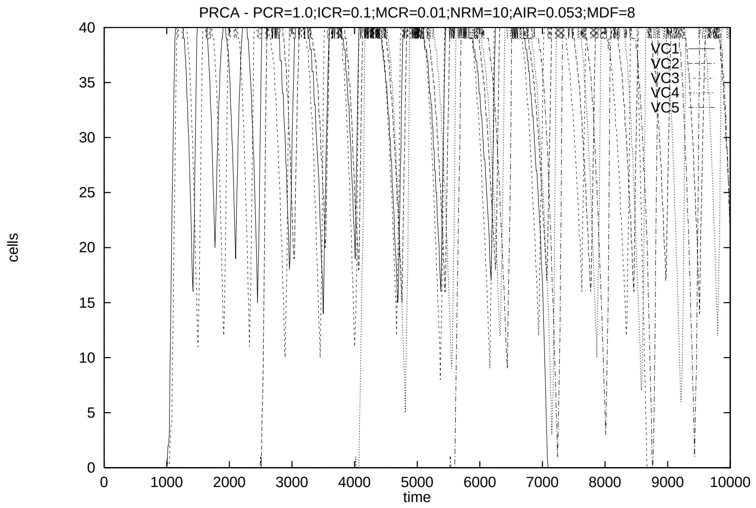
Figure 10: Aperiodic + Virtual Feedback

*Conventional PRCA*

For sake of comparison, the PRCA scheme has been simulated under the same traffic conditions as before, with parameters: NRM=10; AIR=0.053; MDF=8. The results are shown in Fig. 11. As expected, the PRCA scheme does not prevent cell loss, because it cannot account for the bottleneck queue level and the number of cells "in flight".




(a) Rates




(b) Bottleneck Queues

Figure 11: PRCA Control Scheme

# 5 Implementation Issues

In this section, we address possible implementation versions of the basic rate control algorithm described previously. Basically, we have the following parameters available for tuning our algorithm (see Eq.(2)): buffer size $X^o$, gain $K$, and time $RTD$. By choosing these parameters carefully, we first address the algorithm buffer requirements. Then we devise modifications in the basic algorithm in order to have it operating with other schedule disciplines other than round robin.

### 5.1 Buffer Requirements

The steady state analysis of the proposed rate control algorithm has revealed the need for buffers sizes proportional to the round trip delay (Eq. (6)). We aim at decreasing this requirement, by trading buffers for cell loss. Two basic methods can be used: pseudo queues or pseudo $RTD$ delays.

In pseudo queues, the algorithm operates with the parameter $X^o$ prescribed by Eq. (6). However, a smaller buffer size $B$ is allocated for the connection at intermediate switches. It is not difficult to see that a worst case analysis reveals a stationary cell loss rate of:

$$u_{loss} = \frac{X^o - B}{\frac{1}{K} + RTD} \tag{20}$$

In pseudo $RTD$, we decrease the buffer requirement dictated by Eq. (6) by operating with a round trip delay smaller than the actual feedback loop round trip delay. In this way, the maximum stationary rate is increased (see Eq. (5)). However, this approach is equivalent to having a mismatch in the round trip delay estimation, which was addressed previously. The recipe then is to try to operate with a minimum $RTD$ value (conversely a maximum positive delay mismatch) so that system stability is not jeopardized.

Summarizing, the first approach leads to having a permanent flow of lost cells, although stability is guaranteed. The second approach, on the other hand, cells may be lost only at transient time, provided that the system is stable. System stability can be guaranteed, however, through the stability analysis presented previously.

### 5.2 Queue Configurations

So far we have assumed a switch configuration in which separate queues are provided per VC, and are served in round-robin fashion, which is a rather costly mechanism. In this subsection, we relax this requirement, showing that other queue configurations are also feasible without giving up very much of the proposed algorithm performance. We still assume switch architectures with output queueing, which are well known to provide best performance, avoiding problems such as head-of-the-line blocking. We consider the following possible alternatives for queue configurations:

**FIFO per switch port -** This is the simplest configuration, where cells belonging to different VCs and destined to the same output port are stored in a single queue, being served in a FIFO service discipline. No information is kept about individual VCs other than the usual routing tables.

**Pseudo VC queues per switch port -** In this alternative, a single FIFO queue per output port is provided as before. However, per VC counters are kept in order to account for the number of cells per VC stored in the FIFO queue. This alternative relieves the switch from the burden of serving a possibly great number of VC queues in round-robin.

**VC queues per switch port -** This is the most sophisticated queue configuration, where queues per VC are provided for each output port, being served in a (possibly weighted) round-robin service discipline. Switches have to dedicate memory space per VC, and keep track of which among all VCs was last served.

The three queue configurations are shown in Fig. 12.

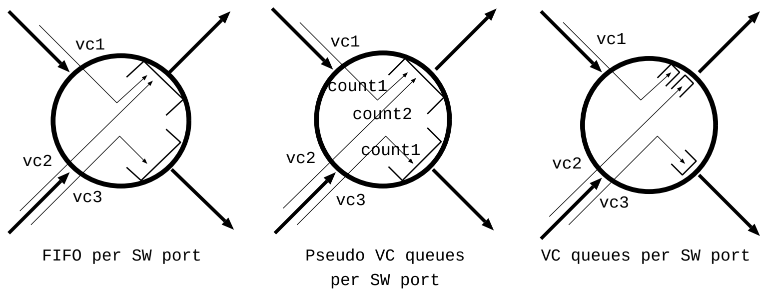



Figure 12: Queue Configurations

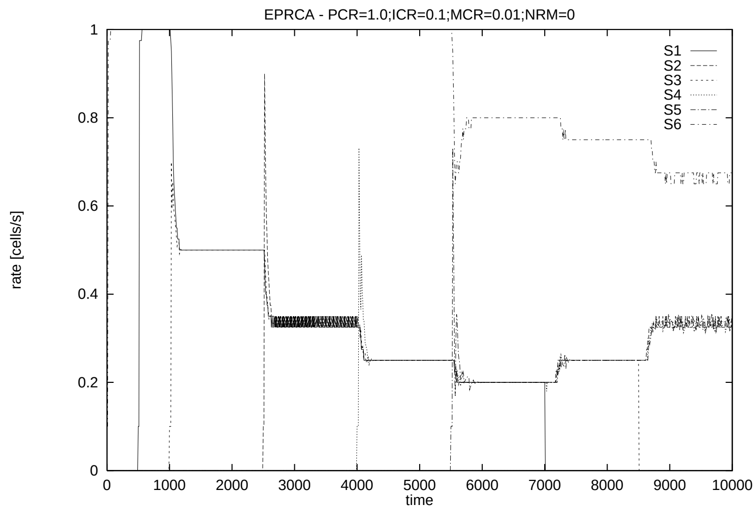
We here accrue one more connection to the previous network scenario, with the purpose of giving rise to traffic activity in multiple bottlenecks. Thus, in this setting, connection #6 becomes active, as shown in Table 2. Connections number 5 and 6 share the first bottleneck, while connections number 1, 2, 3, 4, and 5 share the second bottleneck.

Table 2: VC Connections Activity

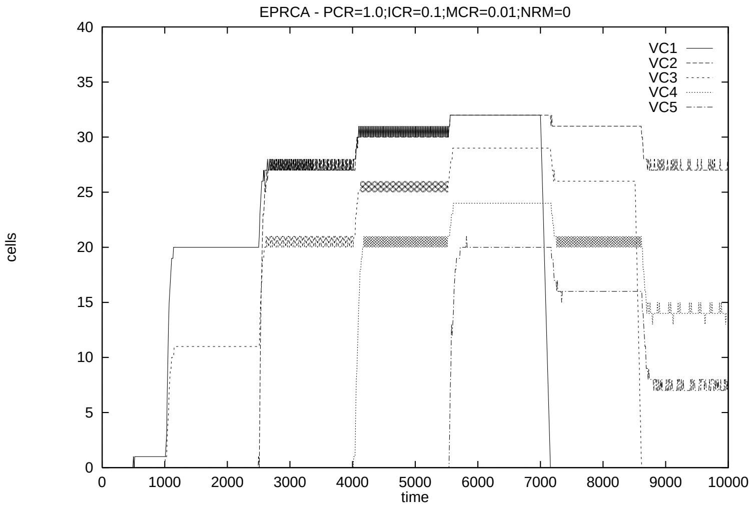
| Connection # | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Start Time | 500 | 2500 | 1000 | 4000 | 5500 | 0 |
| End Time | 7000 | 10000 | 8500 | 10000 | 10000 | 10000 |
| RTD | 0 | 0 | 20 | 40 | 60 | 0 |

*VC queues per switch port*

We now illustrate the behavior of the SP-EPRCA under this queue configuration. The following performance results are based on the control parameters: $X^o = 40\ cells$, $K = 1/X^o$, $\triangle = 10\ units$, and will be used as reference for sake of comparison throughout this section.




(a) Rates



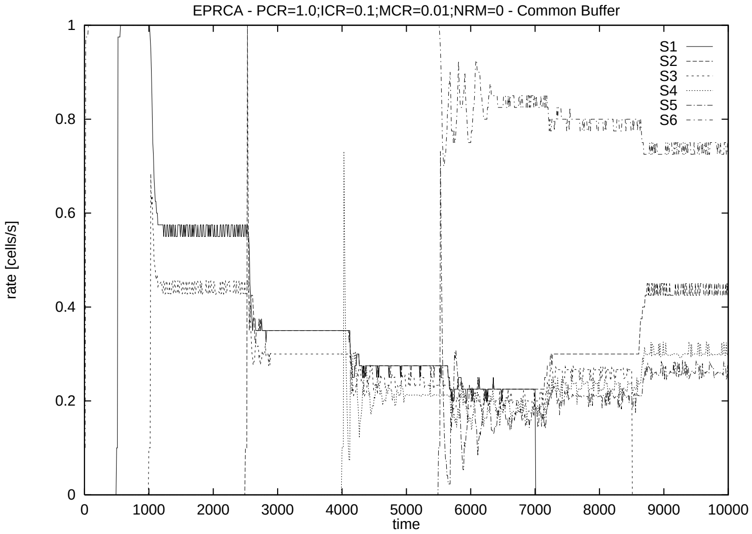

(b) Bottleneck Queues

Figure 13: VC queue architecture

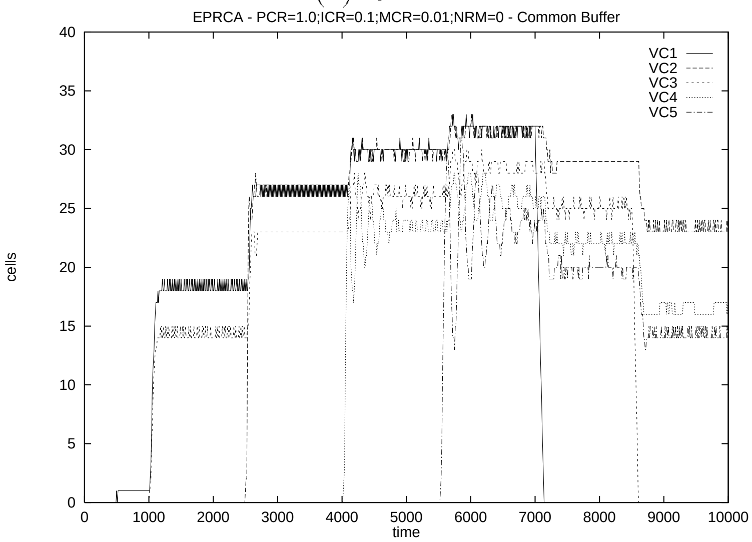
Fig. 13(a) shows the rate adjustment performed by the algorithm when new connections start/stop. The rates are quickly adjusted to their fair share of the available bandwidth every time the system is perturbed. Fig. 13(a) shows that max-min fairness is indeed achieved, by having VC connection number 6 (VC#6) taking most of the bandwidth left by VC#5 on bottleneck 1, since VC#5 rate was constrained by bottleneck # 2 and could not increase its rate any further. Fig. 13(b) shows bottleneck # 2 VC queue levels. Notice that no overflow is experienced. Similar curves can be obtained for queues at bottleneck # 1, ommited here.

*Pseudo VC queues per SW port*

We next investigate a common FIFO queue configuration for ABR traffic at intermediate switches, and its consequences on SP-EPRCA performance. As mentioned earlier, we have "pseudo" VC queues at each switch, i.e. counters per VC that indicate the number of cells currently stored in the common queue. This information is relayed back to the corresponding source for rate control. We have simulated the same network and traffic activity shown in the previous subsection, under a common FIFO queue for all ABR VCs. Results are shown in Fig.14.




(a) Rates




(b) Bottleneck Queues

Figure 14: Pseudo VC queue architecture

Figure 14(a) shows that, although rate control is still performed with cell loss avoidance (Fig.14(b)), the algorithm fails to provide fairness among the connections. Notice that this fact do not contradicts Theorem 2, since the theorem assumes a round robin service discipline. To understand why fairness is not maintained, we recall equation 5. Let us plot this equation for the various RTDs involved in the simulated system (Fig.15).

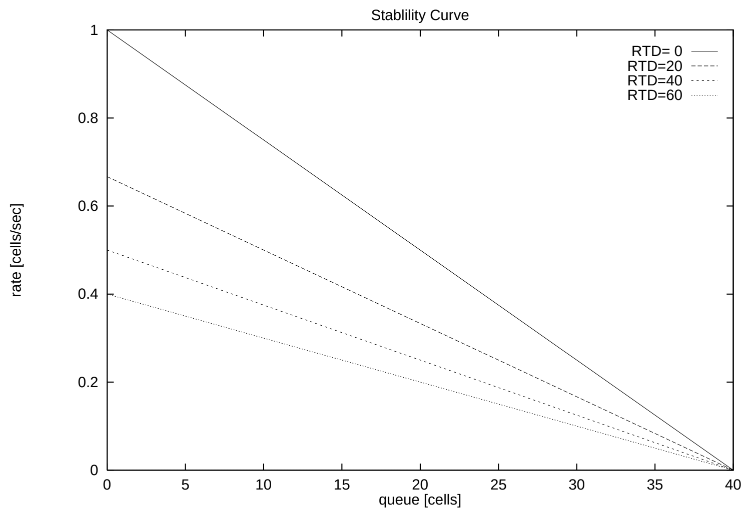



Figure 15: Stability Curve

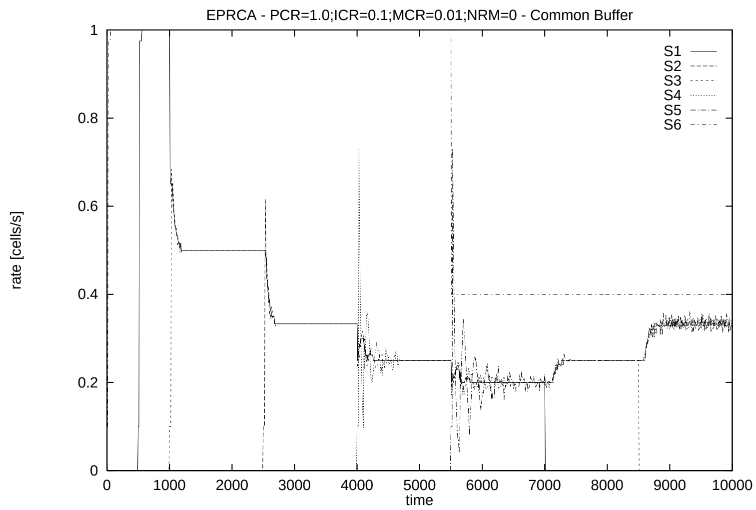
We can interpret the behavior of the control algorithm as seeking a point in each one of the lines shown in the figure, so that it can "stabilize", while satisfying $\sum u_s = c_b$ over all connections sharing a bottleneck with capacity $c_b$. Every time the system is perturbed with the start/stop of a connection, the "operating points" of the remaining connections move down/up accordingly. However, connections with small RTD will start earlier their search for a new stable point than the ones with large RTD, and therefore are likely to end their search first, moving their stable points as little as possible, giving rise to unfairness. Thus, bandwidth resharing can be seen as a race between existing VC connections. The ones with longer RTDs start too late and hence grab less bandwidth. However, we can compensate the long RTD by shortening the time constant of that particular connection. In other words, by using a slower gain for VCs with shorter RTD, we can nullify the discrepancy of differing RTDs. This is equivalent to matching the curves of Figure 15 by changing their slopes. In order to slow down the gain of VCs with short RTD, we use equation 5 to choose $K$ for each $VC_i$ so that:

$$\frac{1}{K_i} = X^o + RTD_{max} - RTD_i \tag{21}$$

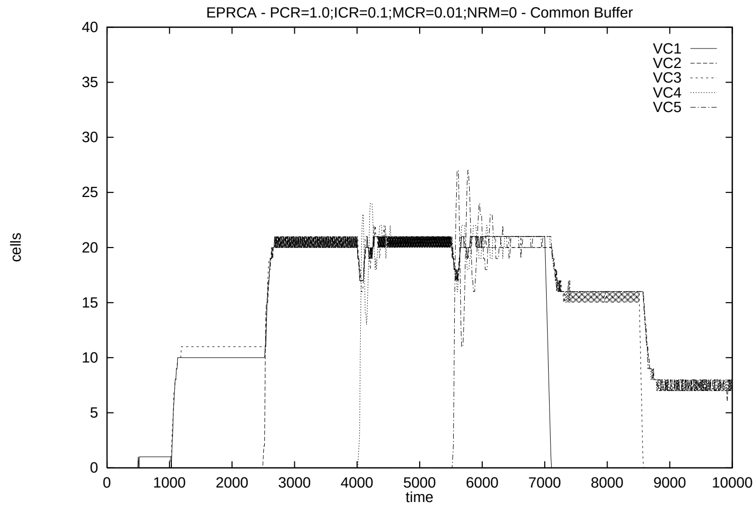
where $RTD_{max}$ is the largest round trip delay among all connections sharing a bottleneck. We call this gain RTD normalization. Thus, a fair version of the SP-EPRCA operating under this queue configuration consists in relaying not only queue levels, but also gains $K$ (more conveniently $1/K$, since it is a integer quantity) in RM cells. Whenever VCs join or drop out from a link, new gains must be computed and relayed back to sources. Since VCs may cross many different bottlenecks, and might have to compete for bandwidth at any of them, the gains should be normalized not only among VCs sharing the same bottleneck, but also among bottlenecks being crossed through by a common connection (since otherwise this connection may not be normalized with respect to one bottleneck, even if it is normalized with respect to another). Therefore, gains must be relayed not only from bottlenecks to sources, but also from sources to bottlenecks.

Figure 16 shows SP-EPRCA performance when such procedure is executed. Note that part (a) of this figure shows a premature saturation rate point for VC # 6 even though its RTD is zero, as if its RTD were in fact $RTD_{max}$, due to the gain RTD normalization procedure. One may argue that, since VC #6 and VC#5 are not competing for bottleneck # 1 bandwidth, because VC#5 is constrained by bottleneck # 2, such normalization is not only unnecessary, but in fact should be avoided in this case. However, the temporary situation in which VC#5 is constrained by bottleneck # 2 may change at any time, reversing the scenario and making VCs # 5 and # 6 start competing for bottleneck # 1 bandwidth. Therefore, SP-EPRCA must be prepared for bandwidth resharing at any time on a fair basis, hence making gain RTD normalization imperative.

This procedure can prevent some switches to have their available bandwidth fully utilized due to the rate saturation problem. As shown earlier, this problem also happens with VC queues per switch port configuration. However, this problem is more likely to happen here due to the fact that "pseudo RTDs" are spread around switches traversed by a large RTD connection.




(a) Rates



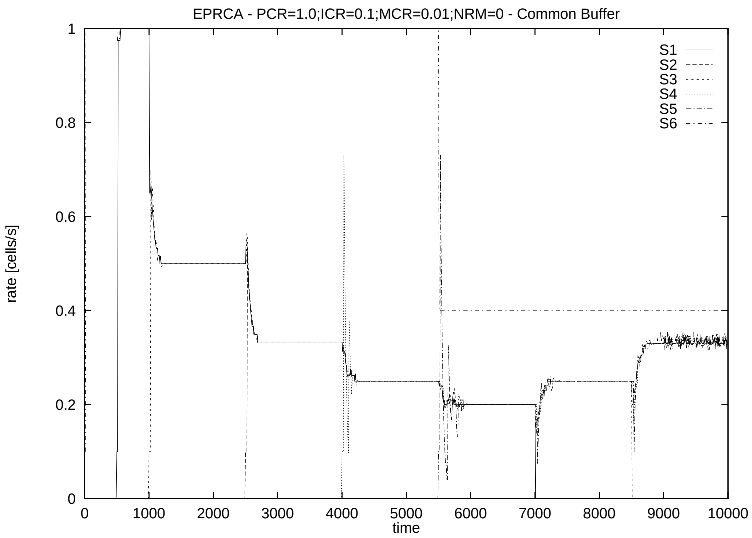

(b) Bottleneck Queues

Figure 16: Normalizing Gain per RTD

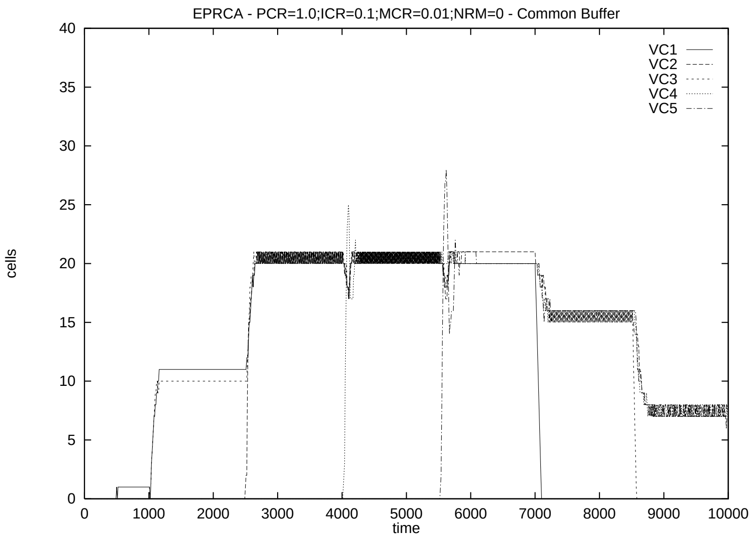
*FIFO per SW port*

We next investigate a truly common FIFO queue architecture for ABR traffic at intermediate switches, and its impact on SP-EPRCA performance. Thus, contrary to the previous section, we do not keep track of the number of cells stored in the switch memory per VC, but only the total number of cells currently stored in the common FIFO queue. The motivation for exploring this seemingly inadequate strategy is that, if the mechanism provides fair rates, the number of cells stored at intermediate switches per VC should be the same. Indeed, the results presented in the last section show that this is the case. Therefore, the switch needs to know only the total number of cells stored in its FIFO queue, plus the number of VC connections currently sharing that queue. It then assumes that stored cells are equally subdivided among VCs, relaying this cell count back to the sources for rate control.

We have simulated the same network and traffic activity shown previously, under a common FIFO queue for all ABR VCs with a single counter to keep track of the total number of cells stored in each queue. Results are shown in Fig.17.




(a) Rates




(b) Bottleneck Queues

Figure 17: Truly FIFO architecture

As expected, the results are identical to the ones shown in the last section. Notice that this simplified version of the algorithm works only because fairness holds. Indeed, if among the competing connections, different number of cells per connection were allowed to be circulating inside the feedback loop, we would certainly have a different number of cells per connection stored at each intermediate switch queues. Dividing the total number of cells evenly among the connections and relaying this information would help only to perpetuate the unfair situation.

# 6 Conclusive Remarks

Theoretical arguments and simulation results have shown that the proposed control algorithm performs an effective congestion control in high speed networks, guaranteeing no cell loss and max-min fairness, provided that round trip delays are known. The control scheme performs well even under the pratical constraints of the EPRCA implementation in an ATM network, which makes it compatible with EPRCA type of UNI interface. We have also shown that simple switch services disciplines, such as FIFO, can operate in conjunction with the proposed rate algorithm. Although buffer requirements to strictly avoid cell loss can be large in WAN scenarios, alternatives have been shown to reduce buffers if one is willing to accept some cell loss.

Some of the assumptions that have been made in the course of this report can be relaxed in a real network scenario. For instance, if queueing delays at intermediate nodes are significant when compared with propagation delays, then a periodic estimation of the ABR/VBR connection round trip delay including such delays can be used in conjunction with the control scheme in order to keep track of such delay variations. In this case, a queue delay bound can be used to avoid cell loss. Also, a priority mechanism can be used to expedite feedback (RM) information, so that the effect of congestion on delivery of control information be minimized.

Finally, the proposed rate control algorithm can also be extended to multicast type of connections. The reader is referred to [12] for further details.

# References

[1] L.Benmohamed, S.M.Meerkov,"Feedback Control of Congestion in Packet Switching Networks: The Case of a Single Congested Node", *IEEE/ACM Trans. on Networking,* vol.1, no.6, pp.693-708, December 1993.

[2] L. Kleinrock, "Queueing Systems. Volume II: Computer Applications," *Wiley*, 1976.

[3] H.T.Kung, R.M.Morris, T. Charuhas, D. Lin, "Use of Link-by-Link Flow Control in Maximizing ATM Network Performance: Simulation Results," *Proc. IEEE Hot Interconnects Symposium, '93*, Palo Alto, California,5-6 August 1993.

[4] O.J.Smith, "A controller to overcome dead time," *ISA J., vol.6, no.2*, Feb. 1959,pp.28-33.

[5] A. W. Barnhart, "Baseline Performance Using PRCA Rate-Control," *ATM Forum/94-0597*, July 1994.

[6] D. Cavendish, Y.Oie, M.Murata, H.Miyahara, "Proportional Rate-Based Congestion Control under Long Propagation Delay," *Int. Journal of Communication Systems*, vol. 8, pp. 79-89, 1995.

[7] F.Bonomi, K.W. Fendick, "The Rate-Based Flow Control Framework for The Availble Bit Rate ATM Service," *IEEE Network*, pp. 25-39, March/April 1995.

[8] K.J.Astrom, B.Wittenmark, Computer Controlled Systems. *Englewood Cliffs, NJ*: Prentice Hall,1990.

[9] Traffic Management Group, "Traffic Management Specifications", *ATM Forum,* Version 4.0, February 1996.

[10] D. Bertsekas, R. Gallager, Data Networks. *Prentice-Hall*, 1992.

[11] K. Walton, and J. E. Marshall, "Direct Method for TDS Stability Analysis," *Proceedings of IEE 134,* pp. 101-107.

[12] D. Cavendish, S. Mascolo, M. Gerla, "Rate Based Congestion Control for Multicast ABR Traffic," *To appear at Globecom96,* November 1996, England.