A Control Theoretical Approach to Congestion Control in Packet Networks

Dirceu Cavendish, Mario Gerla, Saverio Mascolo

Computer Science Dept., UCLA
Los Angeles, CA USA
{dirceu, gerla, mascolo}@cs.ucla.edu

Abstract In this paper, we introduce a control theoretical analysis of the closed loop congestion control problem in packet networks. The control theoretical approach is used in a proportional rate controller, where packets are admitted into the network in accordance with network buffer occupancy. A Smith Predictor is used to deal with large propagation delays, common to high speed backbone networks. The analytical approach leads to accurate predictions regarding both transients as well as steady state behavior of buffers and input rates. Moreover, it exposes trade offs regarding buffer dimensioning, packet loss, and throughput.

1 Introduction

With the remarkable growth of the Internet, congestion likelihood has also increased many folds. This fact has attracted the interest of researchers as well as switch manufacturers to refine existing control schemes and develop new ones. At the same time, the introduction of new types of services in the Internet has underlined the importance of Quality of Service (QoS) concepts. These concepts can also be extended to control schemes, in the form of quality guarantees (e.g., in terms of fairness, packet loss control, stability, etc.). Quality guarantees imply *accountability* of a control algorithm, i.e., the ability to analyze and predict its performance in various traffic and network conditions. Most of the control algorithms currently implemented in operational networks are based on heuristic considerations and are not amenable to accurate analysis.

The objective of this paper is to introduce a control theoretical approach to the design and analysis of closed-loop congestion control schemes in packet networks with large delay bandwidth product. As a way to support QoS, we address important issues such as buffer dimensioning, throughput vs packet loss trade-offs, stability, as well as transients. The paper is organized as follows. Section 2 is devoted to a discussion on related work. In Section 3, we present a continuous model of the closed loop congestion control problem. In Section 4, we present an analysis of the transient and steady state behavior of the system, as well as a stability study of the continuous controlled system. Issues such as buffer space, packet loss, throughput and fairness are also addressed. In section 5, we present a discretization of the continuous model, in order to obtain a suitable model of the system for packet networks. Issues regarding system behavior under loss of feedback information are addressed. Section 6 presents a discrete event simulation study of the controlled system. In the section, we exemplify various issues discussed in the analytical part of the work. Conclusions are drawn in the last section. An appendix is provided, with some mathematical derivations used in the stability analysis of the system.

2 Related Work

Most of the control theoretical approaches to closed loop congestion control has appeared in the ATM community, motivated by the Available Bit Rate (ABR) service, whose input rate is supposed to be explicitly controlled by a closed feedback loop. Initially, credit based control algorithms were considered for regulating input traffic entering the network. The concept of credit-based control for ABR traffic was first introduced by Kung [17], which is a link by link back-pressure congestion control scheme where credits are issued from a switch to its immediately preceding switch in a connection's path. The credit based scheme provides max-min fairness [5], it fully utilizes available capacity, and most importantly, it prevents cell loss. However, it requires per VC queueing and scheduling, otherwise back-pressure can no longer be applied selectively to the connections constrained by a bottleneck, generating fairness and throughput problems. Moreover, a link by link control scheme does not allow the flexibility of trading off throughput for buffers.

In rate based control schemes, feedback control is provided on an end to end basis via Resource Management (RM) cells, instead of link by link via credits, as in the credit scheme. In early implementations, congestion was signaled through a single bit, thus providing a binary feedback loop[3]. Based on this congestion information, the source increased its rate according to an additive factor, and decreased it exponentially according to a multiplicative factor. These schemes were named Proportional Rate Control Algorithms (PRCAs), since the amount of rate increase/decrease is somehow proportional to the actual transmission rate. Binary feedback schemes were soon discovered to exhibit poor performance due to their inherent oscillatory behavior. Some rate based schemes attempt to maintain steady state max-min fair rates on all connections without direct concern about transient behavior and stability. Explicit rates (instead of congestion bits) are reported back to the source for rate adjustment. Most of the rate based control algorithms proposed to date are based on distributed explicit rate computation. These algorithms keep track of where the various connections are bottlenecked, and compute the so called max-min fair [5] rates based on available bandwidth measurements ([12], [1])[1]. Stability aspects of such algorithms can also be addressed, as in [25]. The main challenge is the accurate computation, in a max-min fair sense, of rates to be fed back to the sources [9, 10]. The main advantage of using max-min rates is that input rates follow closely the available capacity of bottlenecks, converging to steady-state quickly. However, these schemes tend to overlook transient behavior, hence, they do not address the amount of buffers required to either prevent/bound cell loss, or to trade cell loss with throughput during short lived transients. Another difficulty is the accurate measurement of "available" bandwidth, which may fluctuate quite rapidly if cross traffic exhibits bursty behavior.

In [8], a closed loop proportional control algorithm is attempted to solve the congestion problem. However, although the approach includes feedback delays, system performance (responsiveness, throughput, and stability) degrades quickly as feedback delays increase, preventing its applicability to high speed network environments. [6] presents a successful approach to deal with network delays, although it requires the computation of a large number of control parameters. The number of parameters is proportional to the round trip delays of the system, and must be "tuned" for each set of delays involved. [16] further acknowledges the need to use a separate set of parameters for each set of delays and number of sources in the system by precomputing sets of paramenters, and switching between them dynamically. [4] also includes network delays in the design of controllers, although their approach also suffers from an increasing computation complexity with the number of delays involved in feedback information retrieval. More recently, [14] has proposed a dual controller, aiming at throughput and fairness performance, for high bandwidth-delay product networks. However, their control parameters also need to be adjusted according to the system round trip delays so as to guarantee stability, whereas in our case the

In [18], we have introduced a control scheme that explicitly deals with network delays with a number of parameters that is independent of the delays involved. Moreover, the system parameter can be adjusted independently of the round trip delays [2]. As a consequence, system responsiveness can be better adjusted for arbitrary delays without the danger of driving the system to instability. With that controller, we have been able to reproduce the same simulation experiments of [6], with the same dynamics, without the need to compute a large number of parameters to deal with network feedback delays and a varying number of sources. The work in [18] has prompted other attempts to explicitly incorporate delays into the design of efficient control schemes [21, 20, 11], and has motivated us to publish our complete theoretical approach, first used in [18].

[1]For other bandwidth sharing strategies, see [19]

[2]Strictly speaking, in our case the system parameter needs to be adjusted according to the error of round trip estimates, which is still much better than the actual delays.

3 Closed-loop Congestion Control System

In this section, we introduce a continuous model of the congestion control problem. Although we assume fixed packet sizes throughout the paper, this fact will be of relevance only when we present a discrete model of the system. Thus, the continuous model introduced in this section can be of use for both fixed (ATM) or variable (IP) packet sizes. In case of IP, the approach can easily be used in conjunction with Multiprotocol Label Switching [22] for flow control [11]. Firstly, we state the objectives of the congestion control system developed in this paper, in order to motivate the handling of feedback delays. Next, we describe the network and queue models, as well as the rate control model. We end this section with an extension of the model to multiple intermediate nodes, as well as a fairness study of the system.

3.1 Feedback Control Objectives

We assume that the feedback control is performed over a traffic type that is not sensitive to service rates nor delays, but is sensitive to packet loss. In other words, the throughput of a connection of this type can be decreased as much as necessary, in order to alleviate congestion and control packet loss. We refer to this traffic type as Available Bandwidth Rate (ABR) traffic. The objectives of a congestion control algorithm, for delay insensitive, loss sensitive ABR traffic are:

- Regulate the ABR traffic input rate so that network resources be utilized to their fullest. This implies that the controller must be highly responsive to fluctuations of higher priority cross traffic, which are typically not subjected to flow control.
- Provide fair share of bandwidth (e.g., max-min fairness concept [5]), among various ABR connections.
- Prevent/bound ABR packet loss using appropriate buffer management policies.
- Scale well to large propagation delays and number of connections. Large bandwidth delay products affect responsiveness and stability. Large number of connections increases the complexity of rate computation.

3.2 Network and Queue Models

The network consists of $N = \{1, 2, \ldots, n\}$ nodes and $L = \{1, 2, \ldots, l\}$ links. Each link i is characterized by: transmission capacity $c_i = 1/t_i$ (cells/sec); propagation delay td_i; processing latency tpr_i (sec), which is the time the switch i needs to take a packet from the input and place it on the output queue. We assume fixed packet sizes (cells) [3], as in ATM technology. Herein we include the processing latency of a switch into its outgoing link propagation delay. The network traffic is generated by source/destination pairs (S, D), where $S, D \in N$. To each (S, D) connection,

[3]We use the terms cell and packet interchangeably in this paper.

there is a Virtual Circuit (VC) associated, which has a fixed path $p(S, D)$ over which all packets of a given connection travel. Each source is characterized by its maximum transmission speed, $c_s = 1/t_s$.

For now, we assume that each switch maintains a separate queue for each Virtual Circuit VC passing through each of its outgoing links. Let $x_{i,j}(t)$ be the occupancy at time t of the queue associated with link i and VC_j, and $X^o_{i,j}$ a 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 [15]. The reason for per VC queueing is to ensure, through a round robin service discipline, the fair sharing of the link by each VC, as well as to isolate different flows, by limiting the interference among them to variations on their service rates. Considering the queue associated with the virtual circuit VC_j at link i , if the level of queue occupancy at time t is $x_{i,j}(t)$, its input rate $u_{i,j}(t)$, and service rate $d_{i,j}(t)$, a fluid model of the queue system shown in Fig. 1 is given by:

$$\frac{dx_{i,j}(t)}{dt} = \begin{cases} u_{i,j}(t) - d_{i,j}(t) & \text{if } x_{i,j} > 0 \\ \max(0, u_{i,j}(t) - d_{i,j}(t)) & \text{if } x_{i,j} = 0 \end{cases} \tag{1}$$

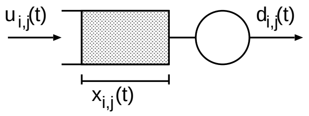


Figure 1: Fluid flow queue model

Notice that per VC queueing implies that no flow injected into the queue modeled by $x_{i,j}$ can be originated by sources other than the one whose input rate is modeled by $u_{i,j}$.

3.3 The Rate Control Model

The rate control scheme proposed is a closed-loop control based on feeding back the network queue occupancy. In order to control the queue level $x(t)$ [4] for a specific VC, we initially use a simple proportional controller. Letting X^o be a target 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 scheme enforces an input rate proportional to the storage space available in the network. The control scheme aims at keeping the bottleneck queue of the controlled VC connection as full as possible, for throughput performance, while being free of packet losses. [5] The calculated input rate $Ke(t)$ at time t affects queue levels only after a round trip delay along the path.

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

[5]The model can also be used in a window control scheme. In this case, the rate $Ke(t)$ would be maintained for $\triangle$ amount of time, which effectively means that $Ke(t) \times \triangle$ packets are allowed to be injected into the network.

Figure 2 depicts the block diagram of this system. T_{fw} is defined as the propagation delay from the flow controlled source to the VC queue at an intermediate congested node, whereas T_{fb} is the propagation delay incurred by packets carrying feedback information from the intermediate node to the destination node, plus the delay incurred from the destination node back to the source node. Therefore, $RTD = T_{fw} + T_{fb}$ is the round trip propagation delay incurred by a packet carrying feedback information. For the moment, we assume that the round trip delay (RTD) of a connection, as well as its components, are invariant with time and known in advance. This is reasonable for single node systems, where the source is connected with its queue via delay lines only. We will address the impact of delay estimation errors on the system shortly, as well as multiple intermediate nodes. In Figure 2, a generic controller $K^*(t)$ is depicted, rather than a simple proportional controller K. The queue service rate is $d(t)$, hence variations in $d(t)$ represent available bandwidth variations, or disturbances, and accounts for cross traffic impact on a flow controlled connection, whereas X^o is a target queue fill level.

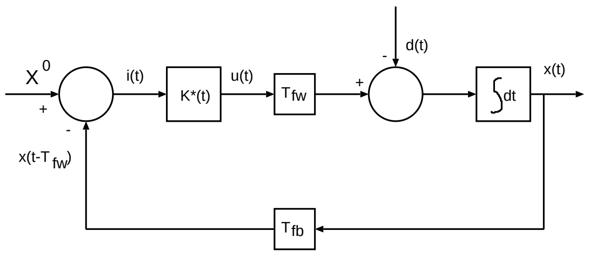


Figure 2: Rate controled flow model with a $K^*(t)$ controller

If we use a simple proportional controller, $K^*(t) = K$ the dynamic behavior of the queue level may exhibit oscillations, and even become unstable under large ledays. 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 up rapidly the queue, resulting in poor link utilization [8]. In order to address these issues adequately, we propose the use of a classical Smith Predictor [23]. Therefore, we substitute the controller $K^*(t) = K$ in Fig. 2 with a controller $K^*(t)$ (see Fig. 3) such that the resultant system has a delay free feedback loop in cascade with pure delays (Fig.4).

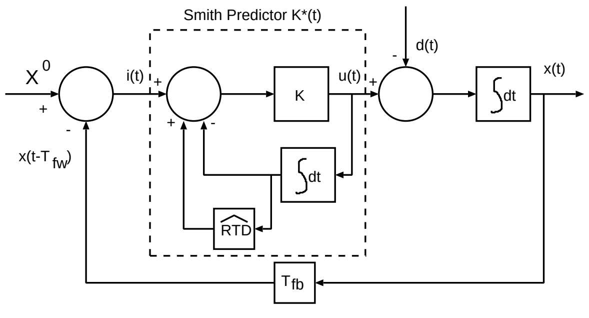


Figure 3: Rate controled flow model using a proportional controller plus a Smith Predictor

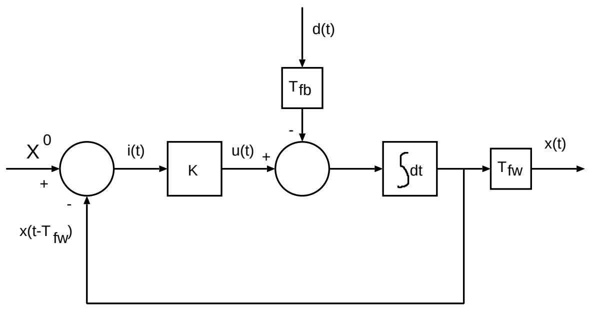


Figure 4: Equivalent model of the system using a Smith Predictor

From Figure 3, the Smith Predictor controller has a Laplace transform $K^*(s) = U(s)/I(s)$ given by:

$$K^* = \frac{U(s)}{I(s)} = \frac{K}{1 + K(\frac{1-e^{-R\hat{T}Ds}}{s})} \tag{2}$$

where s is the Laplace transform complex variable. Other equations of interest are:

$$I(s) = X^o(s) - X(s)e^{-T_{fb}s} \tag{3}$$

$$X(s) = \frac{U(s)e^{-T_{fw}s} - D(s)}{s} \tag{4}$$

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

$$\begin{aligned} u(t) &= K[X^o - x(t - T_{fb}) - \int_0^t u(t^{'})dt^{'} + \int_0^t u(t^{'} - \widehat{RTD})dt^{'}] \\ &= K[X^o - x(t - T_{fb}) - \int_{t-\widehat{RTD}}^t u(t^{'})dt^{'}] \end{aligned} \tag{5}$$

Eq. 5 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. The physical interpretation is that the controller reacts as if all the "in flight" cells were stored at the bottleneck queue, which is a worst case assumption for queue occupancy. Notice that the integral expression in Eq. (5) is known by the traffic sender at all times. The integral computation is over an estimate of the round trip delay $\widehat{RTD}$, which for now we assume accurate. We call this control scheme Enhanced Proportional Rate Control Algorithm (EPRCA), since the scheme is based on a proportional controller enhanced with the Smith Predictor to take care of large propagation delays.

Consider the equivalent system shown in Fig. 4. 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 stable for every positive value of the parameter K. Notice that the parameter K affects the transient behavior only. Namely $1/K$ is the time constant T of the system. Asymptotically, we may assume that after an interval of 4 T the system reaches steady state. Moreover the dynamic response to a step function does not exhibit oscillations in reaching the steady 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, avoiding cell loss;

b) The second part consists of two pure delay blocks that causes a shift in time of the queue level $x(t)$ and disturbance $d(t)$.

Notice that, from a control theory point of view, the use of the Smith Predictor improves system stability because the resulting system, depicted in Fig. 4, has a delay free feedback loop. This allows, for instance, the increase of the control gain K, which regulates the control scheme responsiveness (and thus the length of transients) without the danger of driving the system into unstable behavior. The decoupling of the control gain from stability issues is indeed the main reason for using the Smith Predictor to cope with unavoidable system delays. In its absence, typically system parameters need to be adjusted according to worst case delays [13], affecting system performance [6].

3.4 The case of multiple nodes

So far we have modeled 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 wish to use our controller for multiple node systems, possibly with dynamic change of the bottleneck node during the lifetime of the connection.

We can model the system with multiple nodes as a collection of single node models as previously described. Thus, a n node system, with n nodes between the source and destination nodes, is controlled by n distinct equations (5). Clearly the n controllers have the same RTD, at a given transient interval, although they typically have different T^c_{fw} and T^c_{fb}, $1 \leq c \leq n$. Moreover, we assume uniform control gains K and queue sizes X^o[7]. Each intermediate node of controller c reports a queue level $x^c(t - T_{fb})$.

The source input rate is determined as the lowest rate among all input rates computed. This rate computation is done so as to prevent the source rate exercised from being larger than the one prescribed by one of the single node models. It is easy to see that the prescribed input rate comes from the node that has reported the largest queue level $\max_c x^c(t - T^c_{fb})$, or least storage space. Therefore, all nodes but the bottleneck one may be ignored in the modeling of multiple node systems, as long as the feedback information used by the source is the largest queue level encountered in the forward connection path.

4 Analysis of the Control Scheme

In this section, we present a performance analysis of the controlled system. The analysis includes transient, steady state, and stability of the system.

[6]See [7] results for various delays, as an example of system performance degradation with delays

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

4.1 Queue Transient Analysis

A queue transient analysis is useful to both identify the short term queue reaction to traffic changes and to determine the transient time needed to bring the system back to steady state. The former is important to determine if and how much packet loss might occur due to transient buffer overflow. The latter is useful to determine how fast the control mechanism reacts to transients, which naturally limits its effectiveness to that time scale.

From Fig.3, the queue level can be disturbed by two inputs: the target queue level X^o and the queue service rate $d(t)$. Let the queue level response to input X^o be denoted by $x_X(t)$. Moreover, let $x_d(t)$ denote the queue level response to a disturbance in the queue service rate $d(t)$. In what follows, $d(t)$ is modeled as a step function $a * 1(t)$ [8]. We further assume that $R\hat{T}D = RTD$, leaving the case of delay estimation errors to be addressed later. From Eqs. (2), (3) and (4), by setting $X^o(s) = 0$, and since $D(s) = a/s$, we obtain:

$$\begin{aligned} X_d(s) &= -\frac{s + k(1 - e^{-RTD})}{s(s+k)} D(s) \\ &= \frac{aKe^{-RTDs}}{s^2(s+k)} - \frac{a}{s^2} \end{aligned} \tag{6}$$

Using inverse Laplace transforms [9], we finally find:

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

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:

$$x(t) = x_d(t) + x_X(t) \tag{7}$$

In steady state condition ($t \to \infty$), the queue level is:

$$x(\infty) = X^o - aRTD - \frac{a}{K} \tag{8}$$

Figure 5 shows the transient behavior $x_X(t)$ in response to X^o, the transient behavior $x_d(t)$ in response to $d(t) = 1(t) - 0.5 \times 1(t - T_{off})$ (where 1(t) is the step function), and the overall transient $x(t)$. T_{off} is an arbitrary delay.

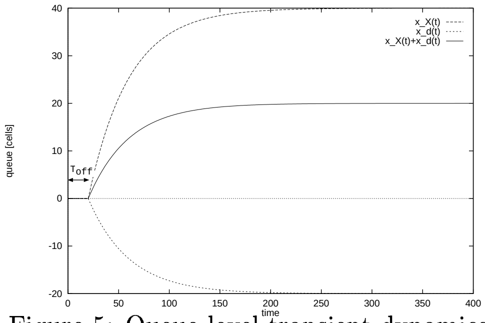


Figure 5: Queue level transient dynamics

[8]It is convenient to model the disturbance as a step function because it is the most basic disturbance function, hence facilitating the study of the system transient behavior, which is the sole purpose of this subsection.

[9]The following inverse transform facts are useful in the derivations: $e^{as}F(s) \leftrightarrow f(t-a)$; $a/(s^2 * (s+a)) \leftrightarrow t1(t) - 1/a1(t) + 1/ae^{-at}$

4.2 Steady State Analysis

The study of how the system behaves in a steady state regime is useful to expose the trade-offs between buffer space, throughput, and packet loss, once transients have vanished. As time approaches infinity, equation (5) gives us the following relation between the steady state rate u_s and steady state queue x_s:

$$u_s = \frac{X^o - x_s}{1/K + RTD} \tag{9}$$

As far as the controller is concerned, any point in the plane u, x satisfying equation (9) is a stable point for the system. This 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 (9) ultimately states that we need buffers proportional to the product of the round trip delay RTD times 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, its queue will eventually empty out, resulting in $x_s = 0$. The buffer requirement is then:

$$X^o = c_b RTD \tag{10}$$

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. In summary, we can say the following. Recalling that $d(t)$ is the service rate of the flow controlled connection, if the connection bottleneck total capacity is c_b, it is indeed desirable that, at steady state, the source rate u_s should achieve: $u_s = d_s$. However, Eq. (9) states that this may not be possible, due to the fact that RTD delay is present, which could cause packet losses if the buffer size X^o is not large enough to withstand in flight packet storage, in the event of a sudden decrease of service rate, $d(t) = 0$ being the worst case. Therefore, Eq. (9) dictates what is the maximum steady state rate achievable by the source, given that a buffer size of X^o is available for the controlled connection, for a system with control gain K and subject to a round trip delay of RTD to withstand an eventual shut off of its service rate.

4.3 Stability analysis

A stability analysis is important for feedback controlled systems, especially in the presence of large feedback delays. Recent literature has examples of feedback systems that are driven to unstable behavior, if careful design of parameters is not exercised (e.g., [8, 24, 13]).

For a generic system, a common stability criterion is to require that the roots of its characteristic equations have all negative real parts. For the system depicted in Fig 4, there are two uncontrolled input variables, X^o and $d(t)$. The stability analysis can be focused either on the directly controlled $u(t)$ input rate variable, or on the indirectly controlled $x(t)$ buffer level variable. Therefore, two transfer functions can be identified: $U(s)/X^o$ and $U(s)/D(s)$. A stability analysis of the former transfer function relates to how and if the system gets back to a stable state when disturbed by a change on the target queue level X^o, whereas a stability analysis of the latter transfer function relates to how and if the system gets back to a stable state when disturbed by a change of its service rate, caused by another connection being served at the same link, and modeled by $d(t)$. For the particular system shown in Figure 4, these transfer functions are: $U(s)/X^o = Ks/(s+K)$ and $U(s)/D(s) = K \times e^{-RTDs}/(s+K)$. Thus, both transfer functions have $s+K$ as their characteristic equation, which is stable for any positive gain K, since for $K > 0$ the polynomial $s+K$ has a single negative real zero. However, the system shown in Figure 4 represents our controlled system only if the delays T_{fw}, T_{fb} are known in advance, so that the $\widehat{RTD}$ estimator box inside the controller (see Fig. 3) matches perfectly the propagation delays present in the feedback loop of the system. In reality, RTD estimation errors, or mismatches, are likely to happen. Therefore, it is important to study how the controlled system behaves in the presence of delay mismatches.

Let the RTD estimated by the source be $\widehat{RTD} = \delta_o$, while the real round trip delay is $RTD = \delta_o + \delta$. We require that $\delta_o + \delta \geq 0$, so that the real round trip delay be non-negative. The system is still represented by Fig. 3, with the following change of variables $T_{fw} + T_{fb} = \delta_o + \delta$ and $\widehat{RTD} = \delta_o$. In this case, the two transfer functions are:

$$\frac{U(s)}{X^o} = [\frac{1}{K} + \frac{1}{s} + \frac{e^{-(\delta_o+\delta)s} - e^{-\delta_o s}}{s}]^{-1} \frac{e^{-T_{fw}s}}{s} \quad (11)$$

$$\frac{U(s)}{D(s)} = [\frac{1}{K} + \frac{1}{s} + \frac{e^{-(\delta_o+\delta)s} - e^{-\delta_o s}}{s}]^{-1} \frac{e^{-T_{fb}s}}{s} \quad (12)$$

Since equation (5) is still valid, the steady state equation (9) 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 at all. We follow a similar approach to Walton and Marshall [26] for stability analysis of time-delay systems. The minor difference is that in the present case the delay parameter δ can be either positive or negative, since it represents a deviation from the estimated delay δ_o, rather than a strictly positive delay. From equations (11) and (12), a single characteristic equation is derived:

$$F(s, \delta) = s + K + Ke^{-\delta_o s}(e^{-\delta s} - 1) \quad (13)$$

In the Appendix, we carry out a study of the roots of $F(s, \delta)$. Based on that study, we can plot few important points of the system root locus, depicted in Figure 6.

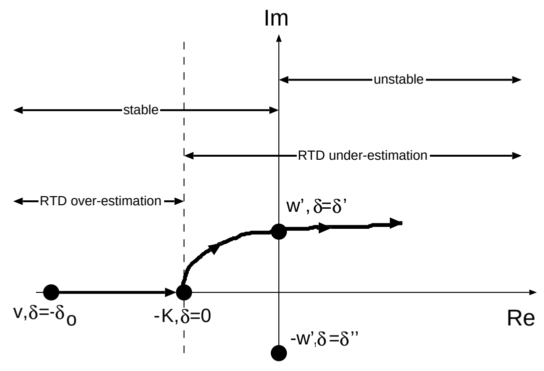


Figure 6: Sketch of the system root locus plot

Figure 6 shows the complex plane, with few points $v+iw$ describing some of the roots that 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 the relevant region for our stability analysis. Each point is described by its coordinates v, w in the plane plus the mismatch value δ at that point. We know that, for $\delta = 0$ (no mismatch), there exists a real negative root at $v = -K$. We have included also the point v, δ_o to stress that points in the real axis to the left of this point are meaningless, since they represent negative real round trip delays. An important observation here is that for $-\delta_o \leq \delta \leq 0$, we prove in the Appendix that the root locus plot can never touch the imaginary axis. Therefore, for $-\delta_o \leq \delta \leq 0$, which is the region at which RTD is overestimated, the system is always stable. This fact makes sense since essentially the control algorithm is using more information about the past than necessary. Moreover, for the region at which RTD is underestimated, there is a limit of δ' in the underestimation error up to which the system is still stable. In the Appendix, we show how to compute this error limit.

4.4 Buffer space and packet loss trade-off

The steady state analysis of the proposed rate control algorithm has revealed the need for buffers sizes proportional to the round trip delay (Eq. (10)). We aim at decreasing this requirement, by trading buffers for packet 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. (10). 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 steady state packet loss rate of:

$$u_{loss} = \frac{X^o - B}{\frac{1}{K} + RTD} \quad (14)$$

Eq. (14) reflects a worst case packet loss rate. This packet loss rate is caused by the fact that the queue level information that comes back to the source reports at most B packets in the queue, which is the actual buffer size available to that connection. If the control algorithm operates with a parameter of X^o (which represents the buffer size available at the bottleneck), it "believes" that there is still $X^o - B$ free buffer space. Notice that the worst case characterization consists of two assumptions: the buffer B remains full indefinitely; an amount of $RTD \times u_s$ remains in transit indefinitely. Under these assumptions, Eq. (14) follows from Eq. (9), and can be interpreted as follows. Under the condition of full buffer $x_s = B$, the control scheme tries to fill up a non-existent buffer space of $X^o - B$, according to Eq. (9), by allowing this number of packets into the network at each $1/K + RTD$ time intervals.

In pseudo RTD, we decrease the buffer requirement dictated by Eq. (10) by operating with a round trip delay smaller than the actual feedback loop round trip delay. In this way, the maximum steady state rate is increased (see Eq. (9)). However, this approach is equivalent to having a mismatch in the round trip delay estimation, addressed previously. The recipe then is to operate with a minimum RTD value (conversely a maximum positive delay mismatch) so that system stability is not jeopardized.

4.5 Fairness and the rate control algorithm

In the congestion control context, fairness is generally defined as the property of exercising 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 context the most common definition is the so called *max-min fairness* [5].

Let r be a rate vector, whose components are connection input rates of a general network, defined previously as links l with finite transmission capacity c_l interconnecting network nodes. Let P be the set of connections p supported by the network. We need the following definitions [5]:

Definition 1 *A feasible rate vector r is a vector $(r_1, r_2, ..., r_n)$ of rates such that $\sum_{i \in U_j} r_i \leq c_j$, where U_j is the set of connections sharing link j.*

Definition 2 *A vector r is max-min fair if it is feasible, and for each $p \in P$ and feasible $\bar{r}$ for which $r_p < \bar{r}_p$, there is some p' with $r_p \geq r_{p'}$ and $r_{p'} > \bar{r}_{p'}$.*

Definition 3 *A steady state rate vector r^s is a rate vector whose components are the steady state rates of the flow controlled connections, u_s (see Eq. (9)).*

Definition 2 means that a max-min fair rate vector r is such that for every rate r_i of vector r, any attempt to increase r_i must result in a decrease of another rate r_j, for which $r_i \geq r_j$, in order to maintain feasibility. The following theorem shows that max-min fairness is achieved by the proposed rate control algorithm.

Theorem 1 *The rate vector r^s is max-min fair.*

Proof of Theorem 1 *The rate vector r^s is feasible, since feasibility is a necessary condition for stability. Moreover, it is not difficult to see that each connection p has a bottleneck link b* [10]*. So we pick a generic connection p, with a bottleneck b associated to it, such that at least one other connection q is bottlenecked at the same link b. If such connection p does not exist, then each flow controlled connection is bottlenecked by a link capacity, hence r^s is a degenerate case of a max-min fair vector, and the Theorem follows. Therefore, let q be another connection bottlenecked at the same link b. At steady state, if we try to increase rate r^s_p, we must decrease the rate r^s_q in order to maintain feasibility. However, we know that $r^s_p = r^s_q$, due to the round-robin service discipline*[11]*. Since p and q are arbitrary connections, vector r complies with Definition 2, and the Theorem follows.*

5 Model Extension to Discrete Time Feedback

So far we have dealt with continuous time models only. However, in packet networks feedback information is relayed in cells or packets, and thus not available in continuous time, but rather in sampled form. We start with the system model shown in Fig. 2. From Nyquist sampling theorem 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 cannot get any better if it is beyond 4 [2]. Indicating by $\triangle$ the sampling time , it follows:

$$\frac{1}{\triangle K} = [2, 4] \tag{15}$$

In order to write the discrete time version of the control equation (5) 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 (5) in its discrete version, we obtain the input rate at time $t_k = k\triangle$ [12]:

$$u(k\triangle) = K[X^o - x(k\triangle - T_{fb}) - u(k\triangle - (m+1)\triangle)\epsilon\triangle - \sum_{i=1}^{m} u((k-i)\triangle)\triangle] \tag{16}$$

ii) $RTD < \triangle$:

$$u(k\triangle) = K[X^o - x(k\triangle - T_{fb}) - u((k-1)\triangle)RTD] \tag{17}$$

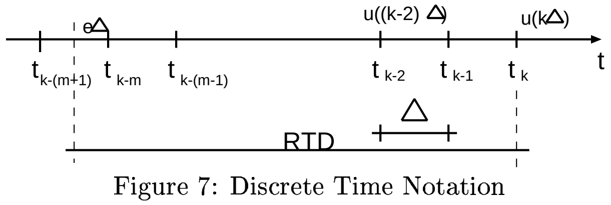


Figure 7: Discrete Time Notation

[10]A bottleneck link always exist, which is the link with highest utilization for a given connection. The queue associated with that connection at the connection's bottleneck link is always served at a rate that matches its arrival rate, so as to keep the queue filling level fixed, since the system is assumed to be in steady state.

[11]A work conserving round-robin service discipline results in the same service rate only among connections sharing the same bottleneck. Therefore, the round-robin service discipline does not imply that the r_i components of vector r have all the same value, unless for the degenerate case in which all connections share the same bottleneck.

[12]Discrete variable k should not be confused with the gain K.

The notation used in the previous equations is illustrated in Figure 7, where $t_k = t_{k-1} + \triangle$. The summation on the right side of the equation (16) can be rewritten as the sum of two parts:

$$u_1(k\triangle) = u(t - T_{fw} - \triangle)\triangle + u(t - T_{fw} - 2\triangle)\triangle + \cdots + u(t - RTD)\triangle$$
$$u_2(k\triangle) = u(t - \triangle)\triangle + u(t - 2\triangle)\triangle \cdots + u(t - T_{fw})\triangle$$

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 traveling from the source to the queue. Therefore the input rate computation at time t can be rewritten as: $u(t) = K[X^o - x(t - T_{fb}) - u_1(k\triangle) - u_2(k\triangle)]$. We can interpret $x(t - T_{fb}) + u_1(k\triangle) + u_2(k\triangle)$ 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. The difference between the queue capacity and the "effective queue level" can be seen as the number of cells that can be transmitted by the source without causing overflow to the bottleneck queue. In this way, the dynamic is delay free, which results in stability and lack of oscillations.

5.1 From periodic to aperiodic feedback

The proposed discrete time control algorithm requires that the controllers located at the sources be furnished with periodic feedback information (every $\triangle$ units of time, with $\triangle$ satisfying equation (15)). This can be realized if the upstream node of a congested link sends feedback information, at every sampling time, to all sources in the upstream direction, as in the Backward Congestion Notification (BCN) scheme of ATM networks. In this type of scheme, called "Periodic Feedback Control", feedback information does not have to compete for network resources. In systems where feedback information has to compete with data traffic in the forward direction, such as Forward Congestion Notification (FCN) scheme, typically the source is responsible for transmitting a special control packet (a resource management (RM) cell in ATM) together with its data traffic (in ATM, one RM cell 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. In order to guarantee the minimum sampling time prescribed by equation (15), we need a control algorithm capable of operating well even if no feedback information is received for a large period of time. If the source receives the feedback information, the control algorithm adjusts the rate accordingly. Otherwise it computes the rate by *estimating* the missing feedback information in a conservative way. We call this type of control "Aperiodic Feedback Control". In a FCN implementation, the feedback information is provided by control cells that collect the maximum buffer level (or minimum storage space) along the path.

The idea is to update the source rate at least once every $\triangle$ 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 $u(t_k)$ and its duration $\triangle_k$, so that $u(t_k)\triangle_k$ becomes one of the terms of the summation in the control equation. The rate updating equation is:

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

where

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

ii) The interval $\triangle$ expires before the source receives feedback information. In this case, the algorithm has to estimate the queue level $x(t_k + \triangle - T_{tb})$. In order to be conservative, preventing cell loss, we propose a "worst case" estimate of the missing queue level, as follows. We 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 becomes the last value $x(t_k - T_{fb})$ plus the number of cells pumped into the network during the interval $[t_k - RTD, t_k - RTD + \triangle]$. The "worst case" estimate of the queue level at time $t_k + \triangle_k$ yields:

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

We call "virtual feedback" to this worst case estimation of the queue level. The approach is equivalent to storing the last received feedback value, $x(t - T_{fb})$, and adding a new term $u(t_k)\triangle$ to the last sum of "in flight" cells SIF, where:

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

The resulting controlled rate becomes

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

In this proposed 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 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.

5.2 Behavior of the System under Lack of Feedback Information

In what follows, we study the dynamic behavior of the controlled rate when the source lacks feedback information. Let $u(0) = K(X^o - x(t - T_{fb}) - \sum_{RTD} u(t_i)\triangle_i)$ be the rate computed when the last feedback cell was received. If no feedback information is further received, the rate is updated every $\triangle$ units of time, using the "worst case" estimate. Thus:

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

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

Therefore, our scheme operates according to a "positive feedback" mechanism, much like a binary scheme [3]. However, 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 (Eq. (15)). Moreover, the rate increasing jumps are related to the queue level and the number of cells released by the source during the last round trip interval.

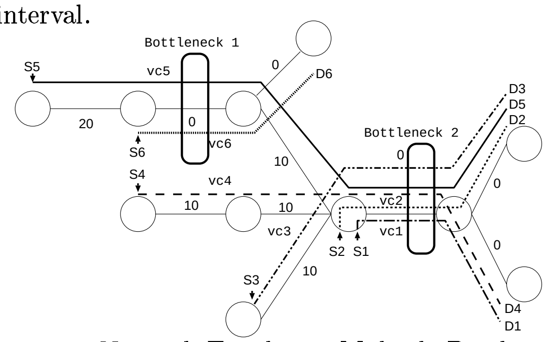


Figure 8: Network Topology - Multiple Bottlenecks

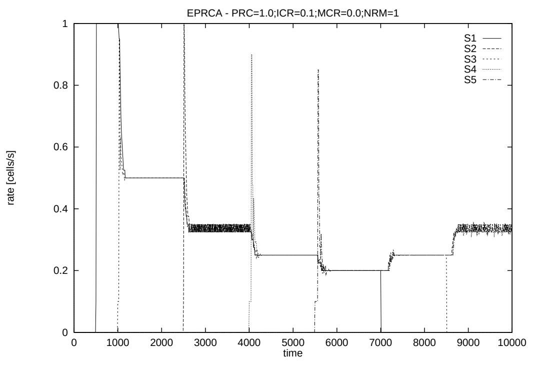
6 Simulation Study

In this section, we present results of several discrete event simulations. The simulation scenario is built so as to exemplify various properties of the control algorithm. We simulate a version of our control algorithm, Enhanced Proportional Rate Control Algorithm (EPRCA), at which we use small buffers in a ATM like framework. We show the performance of EPRCA for the cases when timely feedback information is available and when is not. For comparison, we have included results of a Additive Increase Multiplicative Decrease (AIMD) rate control algorithm, as well. Finally, we discuss scheduling schemes to be used in the implementation of the control algorithm. Although we assume an ATM framework throughout this section, the control architecture can be applied to any packet network.

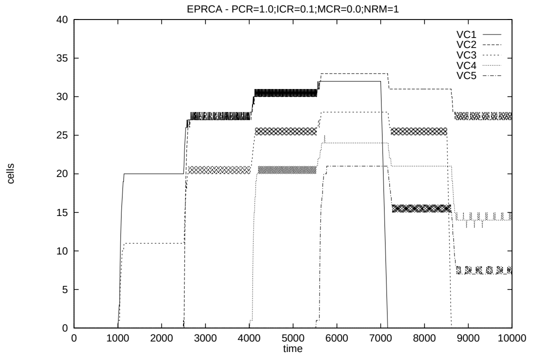
The network topology is shown in Fig. 8. Links have uniform speeds, normalized to 1 cell per unit of time [cell/s]. They are labeled with their respective propagation delays, normalized to the uniform transmission time. In an 150Mbps link speed, 10 units means roughly a 6 mile distance between switches. In this initial setting, five VC connections compete for bandwidth resources of bottleneck #2 link only (connection # 6 is inactive). Queue dynamics are shown for this bottleneck only. VC connection activity (start/end epochs) is as noted in Table 1. We assume infinite backlog at each source. Feedback cells are sent in the reverse direction of the VC traffic over the same links (but sharing VC buffers which are distinct from the ones used in the forward direction). We set a queue level $X^o = 40$ for each queue, so as to have a system sampling time of $40/4 = 10$. Control information, consisting of the maximum of VC queue levels among all nodes traversed by a connection, is transported via special Resource Management (RM) cells, which are especial cells injected once every NRM data cells. RM cells are marked on their way to the destination, and are relayed back to the source via the reverse path. No marking is executed in the reverse direction.

Table 1: VC Connections' Activity

Conn.#	1	2	3	4	5	6
Start t	500	2500	1000	4000	5500	0
End t	7000	10000	8500	10000	10000	0
RTD	0	0	20	40	60	0



(a) Rates

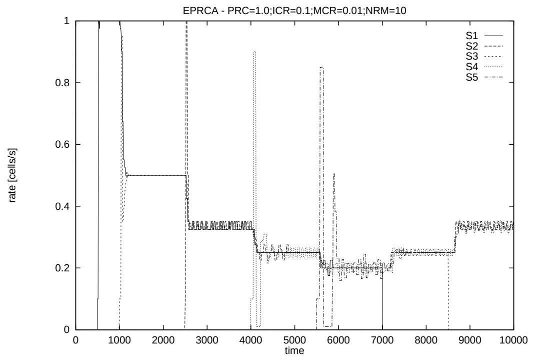


(b) Bottleneck Queues

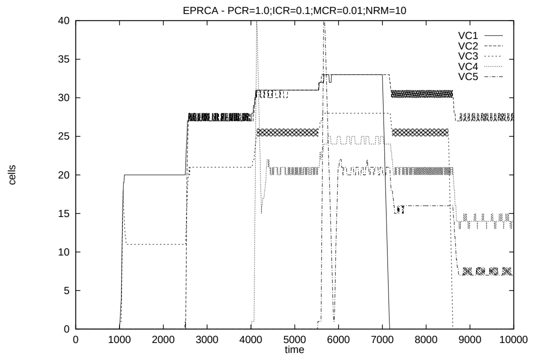
Figure 9: Periodic Feedback

6.1 *Periodic Feedback*

We first show the performance of a periodic sampling version of our control scheme. According to equation (15), we choose a sampling time $\triangle = 10$. Figure 9(a) shows the behavior of the five input rates, corresponding to connections $S1 - S5$, at source nodes. We assume an initial cell rate of 0.1 [cells/s]. After the start/end of a connection, each rate rapidly settles to the new fair steady state value [13]. Figure 9(b) shows the dynamic behavior of the five queues at the bottleneck link, corresponding to $VC1 - VC5$ bottleneck queues. As it can be seen, no queue overflow or cell loss occurs. Moreover, each steady state queue level is in accordance with equation (8).



(a) Rates



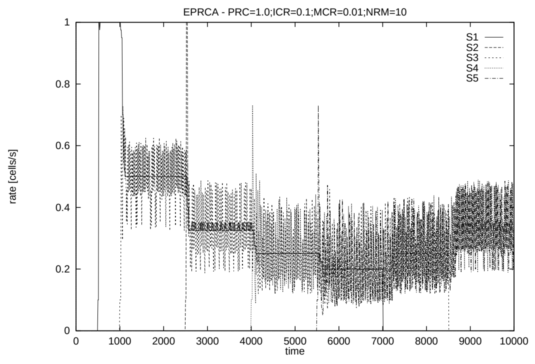
(b) Bottleneck Queues

Figure 10: Aperiodic Feedback

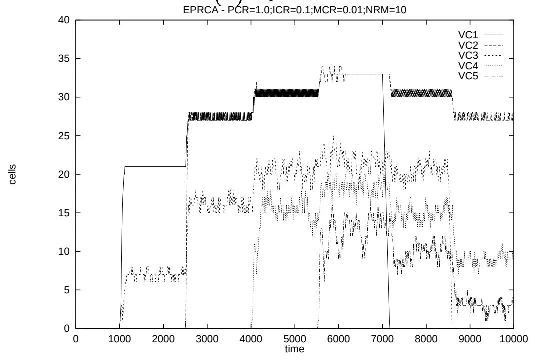
6.2 *Aperiodic Feedback - No Virtual Feedback*

The aperiodic control scheme, with $\triangle = 10$, requires $NRM = 1$ (one control cell every data cell) in order to guarantee the minimum feedback frequency required. The value $NRM = 1$ derives from the fact that, under the heaviest traffic condition (five connections), the feedback cell inter-arrival time is $\triangle = N_{vc}(NRM + 1)$. Since the minimum feedback rate is maintained, simulation results (omitted) are identical to the ones under periodic control. Fig. 10 illustrates performance degradation in case equation (15) is not respected. By setting $NRM = 10$, under the heaviest traffic condition, the feedback inter-arrival time is $55 > 10$. We see from Fig. 10(a) that the rate no longer reaches rapidly the steady state. Moreover, Fig. 10(b) shows that overflow occurs in VC4 and VC5 queues. Other simulation results have verified that the greater the NRM value, the less controlled the queue levels are.

[13]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



(a) Rates

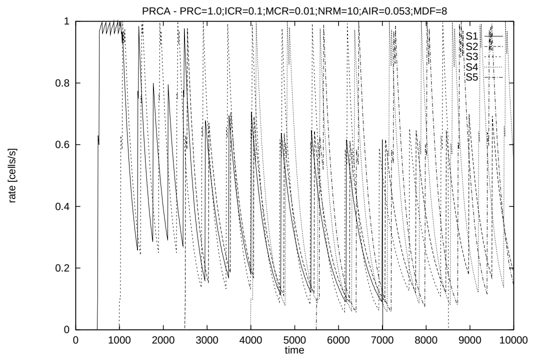


(b) Bottleneck Queues

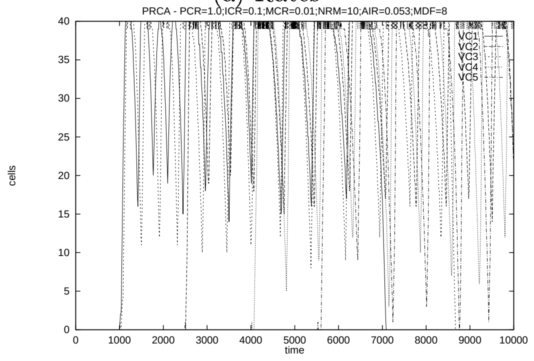
Figure 11: Aperiodic + Virtual Feedback

6.3 *Aperiodic + Virtual Feedback*

Next, we study the performance of the EPRCA under the same conditions and feedback frequency ($NRM = 10$) as previously. Figure 11(a) depicts the oscillatory behavior of the controlled rates, due to the control algorithm "positive feedback" feature, increasing promptly the rate when a feedback cell is received, and decreasing it exponentially otherwise. The oscillations, however, have small amplitude and centered at the fair values of the rates, hence throughput performance is preserved. The high frequency of oscillations is due to the virtual feedback period of $\triangle = 10$, which causes a rate decrease every 10 time units, while the actual feedback inter-arrival time is about 50, causing a rate increase at every 50 time units only. Figure 11(b) shows that queue levels are still bounded, with no cell loss. This verifies that the Virtual Feedback scheme prevents cell loss, due to congestion, even if it is not possible to guarantee the frequency of feedback cells.



(a) Rates



(b) Bottleneck Queues

Figure 12: AIMD PRCA Control Scheme

6.4 *Additive Inc. Multiplicative Dec. PRCA*

For sake of comparison, an additive increase multiplicative decrease (AIMD) PRCA scheme has been simulated under the same traffic conditions as before, with parameters: NRM=10; AIR=0.053; MDF=8. Results are shown in Fig. 12. Fig. 12 a) illustrates that rates never stabilize, as a direct result of the AIMD rate adjustment. More importantly, the AIMD PRCA scheme does not prevent cell loss due to buffer overflow (Fig. 12 b), because it cannot account for the bottleneck queues' levels and the number of "in flight" cells.

6.5 Scheduling Schemes

So far we have assumed a round-roubin scheduling scheme over a per VC queueing architecture. In this subsection, we show that other less costly scheduling schemes are also feasible without giving up much performance. We still assume switch architectures with output queueing. We consider the following alternatives for scheduling schemes :

FIFO scheduling - This is the simplest scheduling scheme, 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.

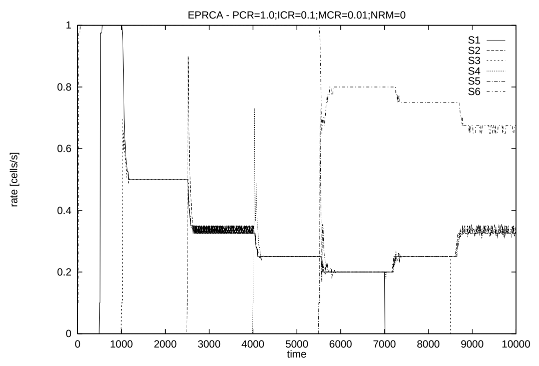
FIFO scheduling with VC counters - In this alternative, cell are served from a single FIFO queue per output port. However, per VC counters are kept in order to measure the number of cells per VC stored in the FIFO queue. This scheduler relieves the switch from the burden of serving a possibly great number of VC queues in round-robin.

VC scheduling - This is the most sophisticated scheduling scheme, where queues per VC are provided for each output port, being served in a (possibly weighted) round-robin service discipline.

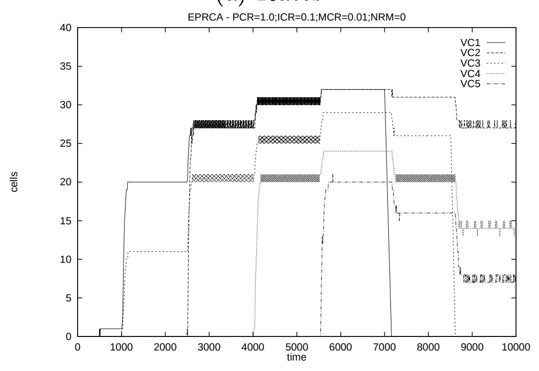
We accrue one more connection to the previous network scenario, with the purpose of giving rise to traffic activity in multiple bottlenecks. Thus, in this setting, connection #6 becomes active, so its "End t" entry in Table 1 becomes 6000. Now connections number 5 and 6 share the first bottleneck, while connections number 1, 2, 3, 4, and 5 share the second bottleneck. The performance results hereafter 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 among the scheluding schemes.

6.5.1 *VC Scheduling*

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, omitted here.

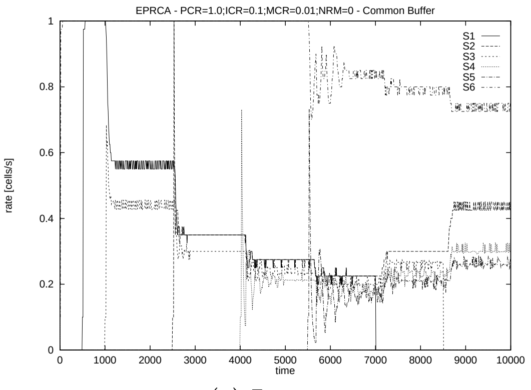


(a) Rates

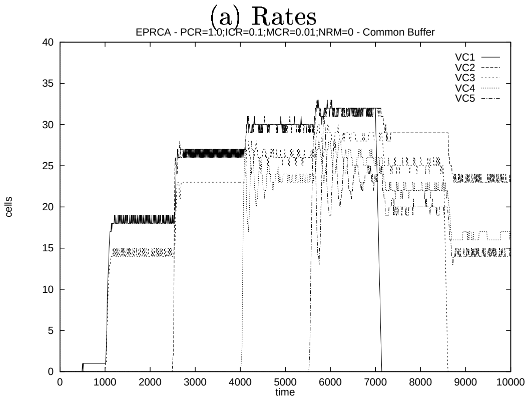


(b) Bottleneck Queues

Figure 13: VC Scheduling



(a) Rates



(b) Bottleneck Queues

Figure 14: FIFO Scheduling with VC Counters

6.5.2 *FIFO Scheduling with VC Counters*

In this case, we have per VC counters that indicate the number of cells currently stored in the FIFO queues. This information is relayed back to the corresponding source for controlling input rate. We have simulated the same network and traffic activity shown in the previous subsection, under a common FIFO queue for all VCs.

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 does not contradict Theorem 1, since the theorem assumes a round robin service discipline. To understand why fairness is not maintained, we recall equation (9). We plot this equation for the various RTDs involved in the simulated system (Fig.15).

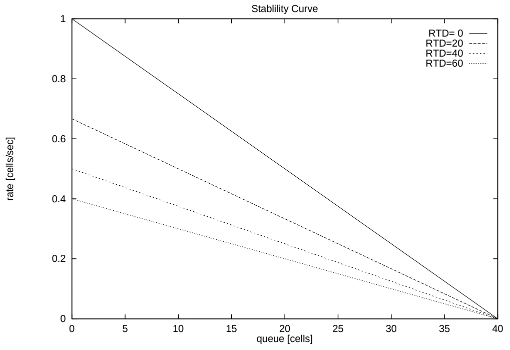
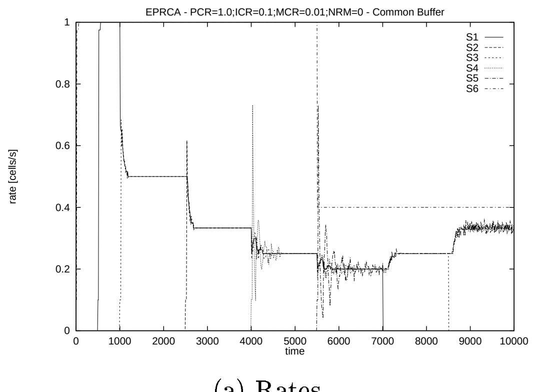


Figure 15: Stability Curves

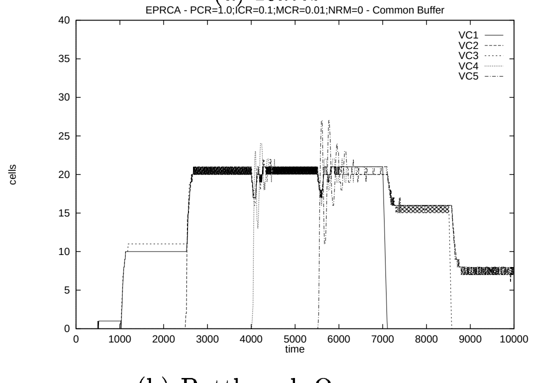
We can interpret the behavior of the control algorithm as seeking a point in each of the lines shown in the figure, so that it can "stabilize", while satisfying $\sum u_s = c_b$ over all connections sharing a bottleneck with capacity c_b. Every time the system is disturbed with the start/stop of a connection, the "operating points" of the remaining connections move down/up accordingly. However, connections with small RTD start earlier their search for a new stable point than the ones with large RTD, and reach another stable point sooner, giving rise to unfairness. Thus, bandwidth re-sharing can be seen as a race between existing VC connections. The connections with longer RTDs start too late and hence grab less bandwidth. We can compensate RTD discrepancies by using a slower gain for VCs with shorter RTD. This is equivalent to matching the curves of Figure 15 by changing their slopes. Hence, we use equation (9) to choose K for each VC_i so that:

$$\frac{1}{K_i} = X^o + RTD_{max} - RTD_i \tag{18}$$

where RTD_{max} is the largest round trip delay among all connections sharing a bottleneck. We call this procedure gain RTD normalization. A fair version of the EPRCA operating under this FIFO scheduling consists in relaying not only queue levels, but also gains K (more conveniently $1/K$, an integer quantity) in RM cells. Whenever VCs join or drop 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 by a common connection. Therefore, gains must be relayed not only from bottlenecks to sources, but also from sources to bottlenecks.



(a) Rates



(b) Bottleneck Queues

Figure 16: Normalizing Gain per RTD

Figure 16 shows 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, EPRCA must be prepared for bandwidth re-sharing at any time on a fair basis, hence making gain RTD normalization necessary. The dwon side of this procedure is that it can prevent some switches to have their available bandwidth fully utilized due to the rate saturation problem. Although this problem also happens with VC scheduling, it is more likely to happen here due to the fact that "pseudo RTDs" are spread around switches traversed by a large RTD connection.

6.5.3 *FIFO Scheduling*

In this case, we do not keep track of the number of cells stored in the switch memory on a per VC basis, but only the total number of cells currently stored in the FIFO queue. The motivation for exploring this strategy is that, if the mechanism provides fair rates, the number of cells stored at intermediate switches should be the same for all VCs using the FIFO queue. 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 almost identical to the ones of Fig. 16, hence they are omitted for space considerations. 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.

It is worth mentioning that throughout our simulation experiments, the flow controlled sources have always traffic to send whenever there is an opportunity. These source are sometimes referred to as "persistent" sources. Other simulation scenarios, with non-persistent sources, have shown that the only effect of not sending traffic at the prescribed rate at a certain time interval, but below it, is that a higher input rate is prescribed at the next time interval by the control algorithm than the rate computed had the source injected traffic at the full prescribed rate.

7 Conclusions

In this paper, we have presented a comprehensive theoretical analysis of the closed-loop congestion problem in packet networks. Through the use of a proportional controller plus a predictor of propagation delay, we have exemplified how important characteristics of the controlled system can be studied. In particular, we have studied the system transients, convergence, stability, as well as exposed trade-offs regarding throughput, packet loss, frequency of feedback information, among other issues. Although more complex control theoretical approaches are possible, the more complex the approach is, the less insights into practical aspects of the flow control problem in packet networks are likely to emerge. The approach used in this paper can be attended to other congestion control algorithms for packet networks. This becomes important as switch vendors move to the design of their own proprietary control algorithms.

References

[1] S. P. Abrahan and A. Kumar, "New Approach for Asynchronous Distributed Rate Control of Elastic Sessions in Integrated Packet Networks, "*IEEE Transactions on Networking*, Vol. 9, No. 1, pp. 15-30, 2001.

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

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

[4] E. Altman, T. Basar, and R. Srikant, "Robust Rate Control for ABR Sources," *Proceedings of INFOCOM98*, Vol. 1, pp. 166-173, San Francisco, 1998.

[5] D. Bertsekas and R. Gallager, "Data Networks," *Prentice-Hall*, 1992.

[6] L. Benmohamed and S. M. Meerkov, "Feedback Control of Congestion in Packet Switching Networks: The case of a Single Congested Node,"*IEEE Transactions on Networking*, Vol. 1, No. 6, pp. 693-707, 1993.

[7] S. Bhatnagar, M. C. Fu, S. I. Marcus, and P. J. Fard, "Optimal Structured Feedback Policies for ABR Flow Control Using Two-Timescale SPSA,"*IEEE/ACM Transactions on Networking*, Vol. 9, No. 4, pp. 479-491, August 2001.

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

[9] A. Charny, K. Ramakrishnan, and A. Lauck, "Scalability Issues for Distributed Explicit Rate Allocation in ATM Networks," *Proceedings of INFOCOM96*, Vol. 2, pp. 1198-1205, San Francisco, USA, 1996.

[10] C. Futon, S. Li, and C. Lim, "An ABR Feedback Control Scheme with Tracking," *Proceedings of INFOCOM97*, Kobe, Japan, 1997.

[11] M. Gerla, R. LoCigno, S. Mascolo, W. Weng, "Generalized Window Advertizing for TCP Congestion Control," *European Transactions on Telecommunications*, No 6, November-December 2002.

[12] R. Jain et al., ERICA Switch Algorithm: A Complete Description," *ATM Forum/96-1172*, August 1996.

[13] R. Johari and D. K. H. Tan, "End-to-end Congestion Control for the Internet: Delays and Stability," *IEEE/ACM Transactions on Networking* , Vol. 9, No. 6, pp. 818-832, December 2001.

[14] D. Katabi, M. Handley, C. Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks," *Proceedings of SIGCOMM 02*, Pittsburgh, Pennsylvania, pp. 89-102, August 2002.

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

[16] A. Kolarov and G. Ramamurthy, "A Control-Theoretic Approach to the Design of an Explicit Rate Controller for ABR Service," *IEEE Transactions on Networking*, Vol. 5, No. 5, pp. 741-753, October 1999.

[17] H. Kung and R. Morris, "Credit-Based Flow Control for ATM Networks," *IEEE Network Magazine*, March/April 1995, pp. 40-48.

[18] S. Mascolo, D. Cavendish, and M. Gerla, "ATM Rate Based Congestion Control Using a Smith Predictor: An EPRCA Implementation," *Proceedings of INFOCOM96*, San Francisco, 1996.

[19] L. Massoulie and J. Roberts, "Bandwidth Sharing: Objectives and Algorithms," *Proceedings of INFOCOM99*, pp. 1395-1403, New York City, USA, 1999.

[20] P. Narvaez and K.-Y. Siu, "New Techniques for Regulating TCP Flow over Heterogeneous Networks," *Proceedings of IEEE LCN*, Boston, Massachusetts, October 1998.

[21] P. Narvaez and K.-Y. Siu, "Optimal Feedback Control for ABR Service in ATM," *Proceedings of IEEE International Conference on Network Protocols*, Atlanta, Georgia, October 1997.

[22] E. C. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture," *Internet Draft*, draft-ietf-mpls-arch-02.txt, July 1998.

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

[24] C-F Su, G. de Veciana, and J. Walrand, "Explicit Rate Control for ABR Services in ATM Networks," *IEEE/ACM Transactions on Networking*, Vol. 8, No. 3, pp. 350-361, June 2000.

[25] G. de Veciana, T-J Lee, and T. Konstantopoulos, "Stability and Performance Analysis of Networks Supporting Elastic Services," *IEEE/ACM Transactions on Networking*, Vol.9, No.1, pp. 2-14, February 2001.

[26] K. Walton and J. E. Marshall, "Direct Method for TDS Stability Analysis," *Proceedings of IEE*, Vol.134, (no.2), March 1987. p.101-107.

Appendix

In this section, we carry on a study of the roots of the following characteristic equation:

$$F(s,\delta) = s + K + Ke^{-\delta_o s}(e^{-\delta s} - 1) \qquad (19)$$

We first examine the roots of $F(s,0)$, which correspond to the case of a perfect estimation of the propagation delays. In this case, we have a single real root at $s = -K$, which is negative for any $K > 0$. Next we consider a infinitesimally small δ, in which case an infinite number of new roots appear, due to the exponentials in equation (13). It is necessary then to study the locations of these new roots in the complex plane. Finally, we need to find values of δ, if any, at which there are roots of Eq. (13) lying on the imaginary axis, i.e., find w, δ values for which $F(iw,\delta) = 0$, and then determine whether the root locus plot merely touches the imaginary axis or crosses from one half-plane to the other with increasing δ. If the root locus plot crosses from left to right, increasing δ destabilize the system, otherwise increasing δ stabilize the system. We start studying the roots of Eq. (13) at the imaginary axis. Namely, $s = \pm wi$ such that:

$$F(wi,\delta) = wi + K + Ke^{-w\delta_o i}(e^{-w\delta i} - 1) = 0$$

which, after some algebraic manipulation, results:

$$e^{-w\delta i} = \frac{K(\cos w\delta_o - 1) + (-K\sin w\delta_o - w)i}{K\cos w\delta_o + (-K\sin w\delta_o)i} \qquad (20)$$

If there exists real δ that satisfies Eq. (20), then by equating the imaginary and real parts of both sides of Eq. (20), there must exist a pair w, δ of real values satisfying the following equations:

$$\sin w\delta = \frac{w}{K}\cos w\delta_o + \sin w\delta_o \qquad (21)$$

$$\cos w\delta = 1 + \frac{w}{K}\sin w\delta_o - \cos w\delta_o \qquad (22)$$

Moreover, if for a particular system described by K, δ_o, we find a pair $w^{'}, \delta'$ such that Eqs. (21,22) are satisfied, then it is easy to see that:

$$\delta = \delta^{'}(w^{'}) + \frac{2\pi q}{w^{'}}; \qquad q = 0,1,2,\cdots \qquad (23)$$

also satisfy Eqs. (21,22). So, once a minimum/maximum [14] δ satisfying Eqs. (21,22) is found, we use Eq. (23) to find the other infinite roots lying on the imaginary axis. Notice that $w^{'}$ is independent of the delay mismatch $\delta^{'}$, which can be seen by removing δ from these equations (using the trigonometric identity: $\sin^2 w\delta + \cos^2 w\delta = 1$). By removing δ, we first find $w^{'}$ from the following resulting equation:

$$2\cos w\delta_o = 1 + \frac{w^2}{K^2} + \frac{2w}{K}\sin w\delta_o \qquad (24)$$

[14] Minimum for stabilizing points, maximum for destabilizing points.

and then use either Eq. (21) or Eq. (22) to determine δ'. Notice that equation (24) 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 ones are destabilizing. We do this by studying the sign of the $\frac{ds}{d\delta}$ derivative at the imaginary roots computed. From Eq. (13), this derivative is given by:

$$\frac{ds}{d\delta} = \frac{Kse^{-(\delta_o+\delta)s}}{1 + K\delta_o e^{-\delta_o s} - K(\delta_o+\delta)e^{-(\delta_o+\delta)s}} \tag{25}$$

If we define S as:

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

after some manipulation we find:

$$S = \text{sgn Re}\left[\frac{m+ni}{o+pi}\right]$$

where

$$\begin{aligned} m &\equiv Kw\sin w(\delta_o+\delta) \\ n &\equiv Kw\cos w(\delta_o+\delta) \\ o &\equiv 1 + K\delta_o\cos w\delta_o - K(\delta_o+\delta)\cos w(\delta_o+\delta) \\ p &\equiv K(\delta_o+\delta)\sin w(\delta_o+\delta) - K\delta_o\sin w\delta_o \end{aligned}$$

We are particularly interested in the positive sign of the previous expression, since for a minimum/maximum δ' found, a positive S gives us the maximum/minimum delay mismatch over which the system becomes unstable. Since we are looking for points that satisfy Eqs. (21, 22), we use these equations, plus some fundamental trigonometric identities to obtain:

$$\begin{aligned} m &= Kw\sin w\delta_o + w^2 \\ n &= Kw(\cos w\delta_o - 1) \\ o &= 1 + K\delta_o + K\delta(1-\cos w\delta_o) \\ p &= w(\delta_o+\delta) + K\delta\sin w\delta_o \end{aligned}$$

The points at which S is positive are the ones such that:

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

or

$$\begin{aligned} (Kw\sin w\delta_o + w^2)[1 + K(\delta_o+\delta) - K\delta\cos w\delta_o] + \\ Kw(\cos w\delta_o - 1)[w(\delta_o+\delta) + K\delta\sin w\delta_o] &> 0 \end{aligned}$$

After some manipulation, the previous relation reduces to:

$$Kw\sin w\delta_o(1+K\delta_o) + w^2(1+K\delta_o\cos w\delta_o) > 0 \tag{26}$$

Some remarks are due. First notice that relation (26) contains no δ parameter, 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 [8] for a simpler time delay system. Notice also that the validity of relation (26) is determined by the particular w' computed from equation (24), which again depends solely on (K, δ_o) parameters, not on δ.

Using equation (24), the above relation becomes:

$$\frac{K}{\delta_o}[1+\delta_o(K+\frac{w^2}{K})]w\delta_o\sin w\delta_o + [\frac{1}{\delta_o^2} + \frac{1}{2\delta_o}(K+\frac{w^2}{K})](w\delta_o)^2 > 0 \tag{27}$$

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 $K, \delta_o > 0$. It is not difficult to prove that relation (27) 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, δ_o) we are dealing with.

Summarizing all observations we have made in this subsection about a system described by (K, δ_o), we can sketch few important points of the system root locus plot, as shown in Figure 6, and computed as follows. First, by solving equation (24), we find $\pm w'$ values. Next we compute the maximum delay mismatch δ' from Eq. (22) or (21), which gives us the first and most important destabilizing point, depicted in Figure 6. Clearly there is a positive range of delay mismatch $[0, \delta')$ in which the system stability is guaranteed despite the delay mismatch.