A Pragmatic Controller for Explicit Rate Congestion Control

Kenneth P. Laberteaux, Charles E. Rohrs, and Panos J. Antsaklis

Abstract – This paper examines congestion control for explicit rate data networks. The Available Bit Rate (ABR) service category of Asynchronous Transfer Mode (ATM) networks serves as an example system for this paper, however the results of this paper are applicable to other explicit rate systems as well. After a plant model is established, an adaptive control strategy is presented and compared to other strategies. Two algorithm enhancements are then introduced. These enhancements reduce convergence time and improve queue depth management. This work differentiates itself from the other contributions in the area of rate-based congestion control in its balanced approach of retaining enough complexity as to afford attractive performance properties, but not so much complexity as to make implementation prohibitively expensive.

I. Introduction

In 1984, the Consultative Committee on International Telecommunications and Telegraph (CCITT), a United Nations organization responsible for telecommunications standards, selected Asynchronous Transfer Mode (ATM) as the paradigm for broadband integrated service digital networks (B-ISDN) [2]. ATM networks provide 6 service categories. Each category of service is customized for a particular type of traffic. Of these 5 categories, only one, Available Bit Rate (ABR), uses a feedback mechanism to create a closed-loop congestion control.

Congestion control is a process by which networks use feedback to adjust the influx of data such that the customer's Quality of Service (QoS) requirements are met while simultaneously attempting to maximize the utilization of the network's resources. The complete ABR congestion control mechanism is described in [1] and [2]. This paper focuses on explicit rate congestion control. The plant description of Section II.A is an approximation to the mechanisms specified in [1]. The present challenge is to devise a controller that resides at the output queue of an ATM switch port and produces a single *explicit rate* to be sent to all ABR sources passing through the queue. The explicit rate must be chosen such that the incoming ABR bandwidth matches the available ABR bandwidth in some appropriate sense. Specifying a single explicit rate at time n for all sources ensures fairness. Matching the incoming ABR bandwidth to the available ABR bandwidth attains efficiency.

Congestion control for ABR traffic utilizes a feedback mechanism, namely *resource management* (RM) cells. An ABR Source periodically inserts RM cells into the stream of data cells. These RM cells pass through each switch along the path to the destination of the *Virtual Connection* (VC). The destination then returns the RM cell to the ABR source along the same path (but in reverse order) used for the forward Virtual Connection from source to destination[1]. RM cells moving from source to destination are called forward RM cells and RM cells returning to the source are called backward RM cells.

A. Related Work

Significant contributions to the understanding of congestion control in ATM ABR networks have been made in the past decade. Contributions include [2]-[18]. Benmohamed and Meerkov made a significant early contribution in plant modeling with [4]-[5]. Benmohamed and Meerkov content themselves to place the closed-loop poles. No effort is made to cancel the plant (and thus closed-loop) zeros. Their calculation is costly, requiring a large matrix inversion and multiplication.

Altman et al. make several contributions; see [6],[7] and references within. Note that in [6],[7] the number of sources and their action delays are assumed to be known. Also note that their models do not include the presence of ABR traffic which is controlled by other switches.

Raj Jain has made the best know contributions to the field of ATM ABR congestion control [9]-[11]. His implementation-friendly Explicit Rate Indication for Congestion Avoidance (ERICA) algorithm [9] works well in a large number of situations and appear to be favored by ATM switch designers. The approach of [11] is very similar to that suggested by Fulton and Li in 1997 [13] and marks an intersection in these two bodies of work. Section II.D includes further comments on [13], [8] and [11].

In addition, there has been significant contributions made in the ATM Forum [1].

This work differentiates itself from the other contributions in the area of rate-based congestion control in its balanced approach of retaining enough complexity as to afford attractive performance properties, but not so much complexity as to make implementation prohibitively expensive.

B. Outline of Paper

The remainder of this paper is as follows: Section II specifies an appropriate plant and controller for the

Kenneth P. Laberteaux (Ken.Laberteaux@tellabs.com) and Charles E. Rohrs (Charlie.Rohrs@tellabs.com) are with the Tellabs Research Center, Tellabs Operations, Inc., 3740 Edison Lakes Parkway, Mishawaka, IN, 46545. Panos J. Antsaklis (antsaklis.1@nd.edu) is with the Department of Electrical Engineering, 275 Fitzpatrick Hall, The University of Notre Dame, Notre Dame, IN 46556.

[1] If for some reason a network does not send backwards RM cells along the reverse path of the data flow, this network can generally use the same congestion control techniques, but with much longer action delays, with the expected performance degradation.

congestion control problem and compares this scheme to other strategies. Section III presents two algorithm enhancements. The first, described in Section III.A, dramatically improves the convergence time of the controller specified in Section II.B. The second, described in Section III.B, extends the purely rate-matching control scheme of Section II.B to provide queue depth management. Conclusions are made in Section IV.

II. The Congestion Control System

A. Plant Definition

Reference [1] defines the mechanism used for congestion control for ATM ABR networks. In this section, the important features of [1] are distilled into a plant model.

Since each switch implements its own, independent controller, one may consider the plant from the perspective of a single switch *SW*. A discrete-time model is used, where sample intervals correspond to control intervals, i.e. a new control action $u(n)$ is calculated for each n. Port j of switch *SW* carries N simultaneous Available Bit Rate (ABR) sessions, and serves as an output port for data cells and an input port for backward resource management (RM) Cells.

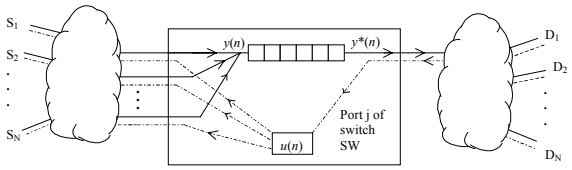


Figure 1 Plant from perspective of Switch Output Port

The present challenge is to devise a controller that resides at port j of switch *SW* and produces a single explicit rate $u(n)$ to be sent to all ABR sources passing through the port. The explicit rate $u(n)$ must be chosen such that the incoming ABR bandwidth $y(n)$ matches the available ABR bandwidth $y^*(n)$ in some appropriate sense. Specifying a single explicit rate at time n for all sources ensures fairness among sources. Matching $y(n)$ to $y^*(n)$ attains efficiency. Rates $u(n)$, $y(n)$, and $y^*(n)$ are in units of cells/second.

With different sources responding to $u(n)$ with different (but no less than d) amounts of delay, and with some unresponsive sources (e.g. they have other bottlenecks), the input rate $y(n)$ can be modeled as

$$y(n) = \mathbf{B}^T \mathbf{u}(n-d) + C,\ \mathbf{B} \equiv \left[b_0, b_1, ..., b_{dB}\right]^T, \quad (1)$$

$$\mathbf{u}(n) \equiv \left[u(n), u(n-1), ..., u(n-dB)\right]^T.$$

It is assumed that C, b_0, b_1,…, b_{dB} remain constant[2] for periods of time long enough for adaptive identification to occur. Faster convergence speed of the adaptive algorithm results in better tracking of these time-varying parameters. Note that (1) models the general application where bandwidth $y^*(n)$ is to be equally shared among customers with non-equal delays. Therefore, the results of this paper could be instructive in the investigation of future resource allocation problems (beyond ATM ABR).

Since the minimum delay in the plant is d, adjustments in $u(n)$ will not be observed until $n+d$. Therefore to generate $u(n)$, it must be decided at time n what the desired value of $y(n+d)$ should be. This desired bandwidth, which is notated as $y^*(n+d \mid n)$, may reflect both bandwidth and buffer measurements made up to time n (this may be generated by a prediction filter as in [7]). By extension, in many cases, the input of the algorithm will be $y^*(n+d+V \mid n)$ (for some non-negative V), i.e. the desired value of $y(n+d+V)$ chosen at time n. The goal of the congestion control mechanism of *SW* is to choose the control signal $u(n)$ at time n so as to minimize

$$E\left[\left(y(n+d+V) - y^*(n+d+V \mid n)\right)^2\right].$$

This plant model was introduced in [12]. It is a direct generalization of the plant models implicit in the work of Fahmy, Jain et al. [11] and Fulton and Li [13], which have been extensively simulated under realistic conditions.

B. Controller Definition

The plant $B(z^{-1})$ is an FIR filter and is thus bounded input/bounded output (BIBO) stable, but may be non-minimum phase. The controller proposed in [16] approximately inverts one Finite Impulse Response (FIR) filter with another FIR filter. This concept is attractive due to its simplicity and its attractive stability properties. The resulting controller, $Q(z^{-1})$, is a causal $(dQ+1)$ tap FIR filter which approximates a delayed (by V) version of the plant $B(z^{-1})$.

$$Q(z^{-1}) = q_0 + q_1 z^{-1} + \ldots + q_{dQ} z^{-dQ} \approx \frac{z^{-V}}{B(z^{-1})}.$$

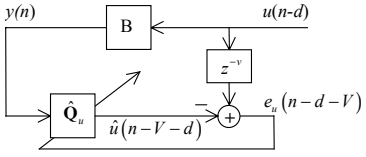


Figure 2 Direct Inverse Plant Modeling

Note that adding delay V is a common characteristic of non-minimum phase plant control, given the large phase lags inherent in non-minimum-phase plants.

[2] The case of non-constant unresponsive sources, where C is replaced by $C(n)$, is explored in [18].

Figure 2 specifies the suggested structure for controller identification. The controller $\hat{\mathbf{Q}}(n)$ can be adaptively determined using the Normalize Least Mean Square algorithm [21]. At time *n*, calculate

$$\hat{u}(n-d-V) = \hat{\mathbf{Q}}(n)^T \mathbf{y}(n) \tag{2}$$

$$\hat{\mathbf{Q}}(n) = \left[\hat{q}_0(n), \hat{q}_1(n), ..., \hat{q}_{dQ}(n), \hat{q}_{DC}(n)\right]^T,$$

$$\mathbf{y}^*(n+d+V \mid n) \equiv \left[y^*(n+d+V \mid n), ..., y^*(n+d+V-dQ \mid n-dQ), y_{DC}\right]^T,$$

$$\mathbf{y}(n) = \left[y(n), y(n-1), ..., y(n-dQ), y_{DC}\right]^T \tag{3}$$

$$e(n) \equiv e_u(n-d-V) \equiv u(n-d-V) - \hat{u}(n-d-V) \tag{4}$$

$$\hat{\mathbf{Q}}(n+1) = \hat{\mathbf{Q}}(n) + \frac{\mu \mathbf{y}(n)}{\mathbf{y}(n)^T \mathbf{y}(n)} e(n),\ 0 < \mu < 2 \tag{5}$$

$$u(n) = \hat{\mathbf{Q}}(n+1)^T \mathbf{y}^*(n+d+V \mid n) \tag{6}$$

The scalar *d* is the minimum plant delay, *V* is an operator chosen (non-negative) inversion polynomial delay, and μ is the adaptive gain chosen such that $0 < \mu < 2$. The constant y_{DC} is operator-chosen, appended to the delay-chain values of $\{y\}$ in (3) so that the final tap of $\hat{\mathbf{Q}}(n)$ becomes a DC tap $\hat{q}_{DC}(n)$ (see [15]), $\hat{\mathbf{Q}}(n) \equiv \left[\hat{\mathbf{Q}}_{lin}(n)^T, \hat{q}_{DC}(n)\right]^T$.

Figure 3 shows the complete control architecture. The Identification section uses NLMS adaptation to determine $\hat{\mathbf{Q}}(n+1)$ (shown with $\hat{q}_{DC}$ separated from the remaining linear taps, $\hat{\mathbf{Q}}_{lin}$, and with $y_{DC} = 1$) by creating estimate $\hat{u}(n-V-d)$ using (2). $\hat{\mathbf{Q}}(n+1)$ is copied into the Controller, which produces $u(n)$ from the set point $y^*(n+V+d \mid n)$. The Plant is represented by (1).

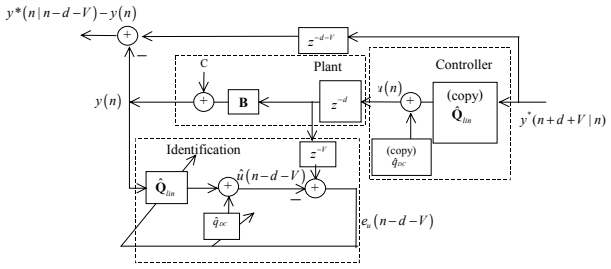


Figure 3 Complete Control Architecture ($y_{DC} = 1$)

The control algorithm (2)-(6) has many attractive features, as discussed in [15], [18]. Specifically, [15] shows that, under a set of reasonable assumptions, the adaptive, Finite Impulse Response controller filter $\hat{\mathbf{Q}}(n)$ converges to its optimal Weiner solution

$$\mathbf{Q}_0 \equiv \left\{E\left[\mathbf{y}(n)\mathbf{y}(n)^T\right]\right\}^{-1} E\left[\mathbf{y}(n)u(n-d-V)\right]$$

in the mean, i.e. $\lim_{n\to\infty} E\left[\hat{\mathbf{Q}}(n) - \mathbf{Q}_0\right] = \mathbf{0}$. Likewise [15],[18] demonstrates that coefficients comprising $\hat{\mathbf{Q}}(n)$ have bounded variance for all *n* and an expression is given for $\lim_{n\to\infty} E\left[\left(\hat{\mathbf{Q}}(n) - \mathbf{Q}_0\right)\left(\hat{\mathbf{Q}}(n) - \mathbf{Q}_0\right)^T\right]$. Further [18] gives conditions which guarantees that $\lim_{n\to\infty} y(n) - y^*(n \mid n-d-V) = 0$.

Despite these results, the algorithm (2)-(6) confronts many practical obstacles. Section III identifies these obstacles and offers solutions, thereby creating an enhanced algorithm with a much-improved practical value.

C. Simulation Framework

To demonstrate the various design issues covered in this paper, a common simulation framework is now defined. These simulations use the Matlab [22] simulation tool. The measurement and control sample time is $T_s = 1$ msec. The set-point y^* is chosen to be a white Gaussian process with mean $E[y^*]$=1 Mcps (million cells per second) and a standard deviation σ_{y^*} of 22 Kcps[3]. The minimum response delay d =10 msec. The distribution of the delays of 22 responsive sources is given by $B(z^{-1}) = z^{-10}\left(2 + 9z^{-1} + 8z^{-2} + 3z^{-3}\right)$. Let C =200 Kcps. The number of taps in the controller is $dQ = 30$, with $V = 10$. The adaptation gain is set at its optimal value $\mu = 1$.[4]

D. Comparisons to Less Complex Schemes

Before proceeding to the main contribution of this paper, the algorithm enhancements, it is appropriate to evaluate the merits of the general control scheme proposed here. Other approaches to congestion control have been outlined in Section I.A. Included in this list are approaches that claim to provide satisfactory performance with a lower computational cost than (2)-(6). In what follows, it is shown that the added computational cost of (2)-(6) provides better performance than less computationally complex schemes, specifically [11], [8], [13]. Also, (2)-(6), in its simplest ($dQ = 0$) case, is essentially equivalent in performance and complexity to these simpler schemes.

[3] These deviations about the mean of the desired ABR rate are determined by the extent that the port measures and re-allocates bandwidth from higher-level service category flows. It is somewhat uncertain how aggressively ports will attempt to re-allocate unused bandwidth. Very small variances are possible.

[4] These values represent a realistic scenario. See discussion in [18].

Consider the proposed controller (2)-(6) with only one adaptive tap, i.e. $dQ = 0$, $V = 0$, $\mu = 1$. Note that in the $dQ = 0$ case, the NLMS adaptation (5) devolves into a single division: $\hat{q}(n+1) = u(n-d)/y(n)$. Compare this to Fulton's identification [13]: $\hat{N}_{eff,\text{Fulton}}(n+1) = y(n)/\overline{u}(n\text{-}1)$, where $\overline{u}(n-1)$ is the time average of a sequence of previous values of u. Fulton does not explicitly estimate d, therefore requires the averaging on u for convergence (unsurprisingly, the recommended time interval for averaging is d samples). Similarly, Imer [8] calculates $\hat{N}_{eff,\text{Imer}} = y/u$ every d' samples, $d' \geq d$, where u is kept constant over the past d' samples. The Fahmy parameter *Effective Number of active VCs*, or $\hat{N}_{eff,\text{Fahmy}}$, is defined similarly [11], albeit in an indirect manner.

Clearly $\hat{N}_{eff,\text{Laberteaux}} = 1/\hat{q}$, $\hat{N}_{eff,\text{Fulton}}$, $\hat{N}_{eff,\text{Imer}}$, and $\hat{N}_{eff,\text{Fahmy}}$ are adaptive estimates that attempt to capture the same information. In each controller, this value is used to divide the amount of available bandwidth $y*$ so that a future y will match a future $y*$ in some appropriate way. Thus the controller (2)-(6), in its simplest version ($dQ = 0$), is essentially equivalent, in performance and complexity, to the suggestions made by Fulton, Imer, and Fahmy.

Consider how these one-tap controllers perform. The controller in each case consists of dividing a future estimate of $y*(n)$ by the associated $\hat{N}_{eff}$. All provide fair and efficient allocation of $y*$ in the long-term. The best such a controller could accomplish is that the incoming ABR bandwidth matches the available ABR traffic in the mean, i.e. $E[\chi(n)] = 0$, $\chi(n) \equiv y*(n) - y(n)$.

While the authors of [11], [8], [13] make a fair and stable allocation their performance goal, here fair and stable allocation is taken to be a minimum acceptable performance objective. This difference in performance objective may be based on a modeling assumption. Clearly for ATM ABR congestion control systems, two quantities change with time: the amount of bandwidth allocated to ABR and the number of competing ABR connections vying for this bandwidth. Since operational experience with ABR is limited, it is difficult to know with certainty the time-scales over which these two quantities change. However, this paper assumes that an *Available* Bit Rate controller is likely to see its available bandwidth change more rapidly than the number of connections.

If the available bandwidth $y*(n)$ remains constant for long periods (e.g. multiples of the maximum round trip time, d), or $\sigma_{y*}^2 \approx 0$, then the single-tap schemes discussed above work effectively. Note that Imer, both in his development and simulations, assumes that $y*(n)$ is constant. Fulton uses $\overline{y}^*(n)$, the sample mean of $y*(n)$, in her calculation of the explicit rate $u(n)$.

Since this paper assumes that $y*(n)$ changes more quickly than changes in the number of ABR connections, $y*(n)$ is modeled as a noise source. Unless, $B(z^{-1}) = b_o$, it is straightforward to show that both the variance of $\chi(n)$, σ_χ^2, and the variance of $queue(n)$, σ_{queue}^2, increase as σ_{y*}^2 increases (this will be validated by simulations below). This, in turn increases the necessary buffer size if overflow is to be avoided. Also, if buffer underflow is to be avoided, a larger average queue size must be targeted as σ_{queue}^2 increases.

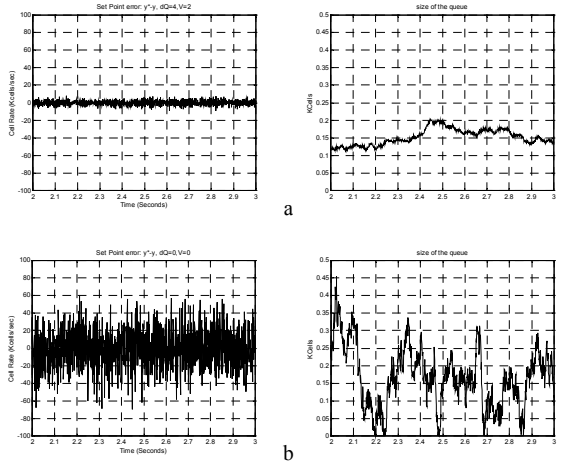


Figure 4 Set-point error, $y*(n) - y(n)$, and size of queue, $queue(n)$, with $\sigma_{y*} = 22$ (a) $dQ = 4, V = 2$, (b) $dQ = 0, V = 0$.

Since larger queue sizes require a larger memory cost and also increases the delay through the switch, both of which are preferably avoided, this paper views minimizing σ_χ^2, and thus σ_{queue}^2, as a desirable performance goal. Using the simulation environment established in Section II.C, Figure 9 shows[5] how effectively σ_χ^2 and σ_{queue}^2 can be minimized by using 31 ($dQ = 30$) taps in the controller (2)-(6). As dQ is decreased to the limiting one-tap ($dQ = 0$) case, performance gracefully degrades to that of the one-tap solutions discussed above. These simulations are identical to those shown in Figure 9-where the desired queue size is 100-200 cells-except for changes in dQ and V as noted.

[5] For the sake of space economy, the simulations of (2)-(6) shown in this section are in fact using the algorithm enhancements of Sections III.A and III.B. However the basic complexity/performance comparisons are essentially the same for the non-enhanced controller of (2)-(6).

Figure 9 (31 taps), Figure 4a (4 taps), Figure 4b (1 tap) show the performance, both in terms of set-point error, $y^*(n)-y(n)$, and the size of the queue, $queue(n)$, gradually degrading as the number of taps decreases. In the limiting one-tap case, shown in Figure 4b, the performance is essentially equal to the performance of Fulton's UT algorithm, shown in Figure 5a. This supports the near-equivalence of performance predicted in the discussion above.[6]

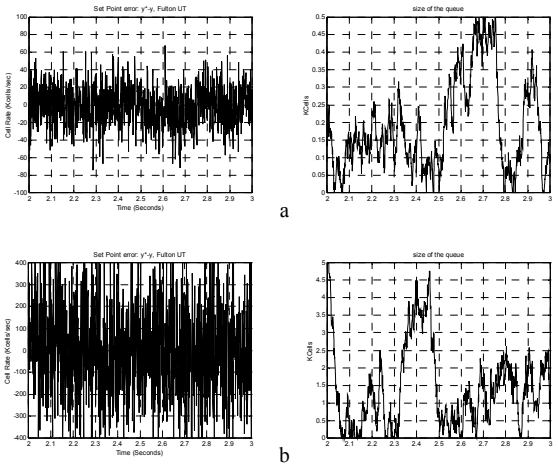


Figure 5 Set-point error, $y^*(n)-y(n)$, and size of queue, $queue(n)$, with Fulton's UT algorithm, (a) $\sigma_{y^*}=22$, (b) $\sigma_{y^*}=223$

Some may be satisfied with the performance of the simple, one-tap controllers shown in Figure 4b and Figure 5a. However, it is important to note that performance of one-tap controllers is highly dependent on the standard deviation of the set-point, σ_{y^*}. When σ_{y^*} is increased an order of magnitude from 22 to 223, it is observed in Figure 5b the performance degrades an order of magnitude.

To summarize, higher-performance-through-higher-complexity solutions such as (2)-(6) have not yet received considerable attention from switch vendors and even many researchers. However, the added complexity of (2)-(6) does indeed provide much improved performance over those of [11], [8], [13]. Further, (2)-(6) can be simplified in implementation (by reducing dQ), thereby gradually reducing its performance and complexity to that of the popular one-tap solutions [11], [8], [13]. For example, if the complexity budget for a specific available bit rate application allows five taps ($dQ=4$), then the added complexity of these five taps appears justified.

[6] It is well known that the convergence time for LMS type algorithms, including NLMS, decreases as the number of taps increases [21]. The plots above begin after 2 seconds, as all cases converge within this time. However, convergence rates of the adaptive estimates increase as the number of taps decreases, revealing a short-term vs. long-term performance trade-off.

Other, even computationally simpler, congestion control schemes have been presented for the Internet. Generally these schemes are one-bit marking approaches. These approaches occupy a very different location on the performance/complexity curve of congestion control. At best, these one-bit schemes will match the arriving bandwidth to the available bandwidth in the mean, with even greater error variance σ_χ^2. The comparisons made above can therefore be extended to the one-bit Internet proposals.

III. Algorithm Enhancements

In this section, two additions to the congestion control mechanism are introduced and discussed. Each addition provides necessary mortar in cementing together theoretical analysis and practical design. These two modifications are singled out for attention here since each addresses a general issue likely to appear in many complex congestion control schemes, not just that of ATM ABR congestion control.

A. Convergence Rate Improvement

Figure 6a shows the results of simulating the system without any modifications to improve the rate of convergence. After 8 seconds, the convergence of the controller is so poor that it appears to be admitting over twice the desired rate of traffic[7]. This is clearly unacceptable performance.

The Least Mean Square (LMS) algorithm has the property that the mean of the coefficient error vector, $E\left[\tilde{\mathbf{Q}}(n)\right]$, converges to zero at a rate inversely proportional to the eigenvalue spread $\lambda_{\max}/\lambda_{\min}$ of $\mathbf{R}=E\left[\mathbf{y}(n)\mathbf{y}(n)^T\right]$ [21]. It is more difficult to specify the convergence trajectory of $E\left[\tilde{\mathbf{Q}}(n)\right]$ for Normalized Least Mean Square (NLMS) adaptation in all but the simplest cases [20], although practical experience shows that speed of convergence is still a strong function of eigenvalue spread.

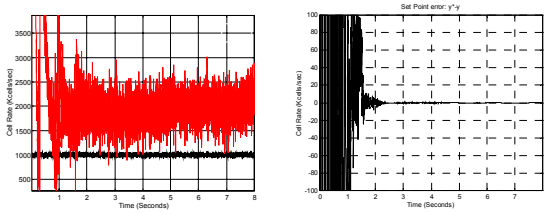


Figure 6 Convergence performance (a) Before modification–Comparing the Set Point (lower curve centered at 1000) and Port Input Rates (higher curve approximately centered at 2000) . (b) After modification–Set-Point error after convergence rate improvement when $Q^*(n)$ is updated twice a second.

[7] Note that the results from [15], [18] ensures that $y(n)$ will eventually coincide with $y^*(n)$.

While we have considered several strategies to improve convergence time of the system defined in Section II, the most promising strategy is as follows: Provide the identification algorithm with zero-mean signals by estimating and removing the signal means (thereby reducing the eigenvalue spread). Then perform "DC correction" in the controller by an additive term q^*.

One obvious method for estimating $E\left[u(n)\right]$ and $E\left[y(n)\right]$ is by directly calculating sample means. The most common method is using a single-pole filter. If the sample means of $u(n)$ and $y(n)$ are notated $u_{SM}(n)$ and $y_{SM}(n)$ respectively, then, with $0<\delta<1$

$$u_{SM}(n)=u_{SM}(n-1)(1-\delta)+u(n)$$

$$y_{SM}(n)=y_{SM}(n-1)(1-\delta)+y(n).$$

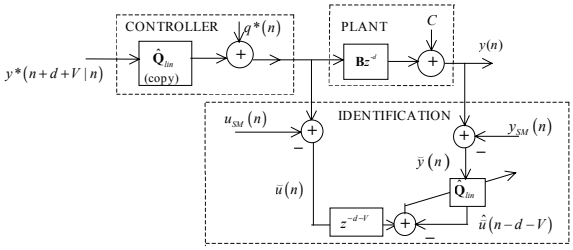


Figure 7 Architecture for subtracting sample mean estimates.

The concept is illustrated by Figure 7. Subtract $u_{SM}(n)$ and $y_{SM}(n)$ from their corresponding signals to perform identification. Estimates $u_{SM}(n)$ and $y_{SM}(n)$ are then used to form $q^*(n)$, which is added to the controller to perform DC correction. Generally no DC tap is needed ($y_{DC}=0$).

The DC correction term $q^*(n)$ uses down-sampled versions of the signal sample means.

$$q^*(n)=u_{SM,q^*}(n) - y_{SM,q^*}(n)\sum_{i=0}^{dQ}\hat{q}_i\left(\left\lfloor\frac{n}{dsInterval}\right\rfloor dsInterval\right) \quad (7)$$

$$u_{SM,q^*}(n)\equiv u_{SM}\left(\left\lfloor\frac{n}{dsInterval}\right\rfloor dsInterval\right)$$

$$y_{SM,q^*}(n)\equiv y_{SM}\left(\left\lfloor\frac{n}{dsInterval}\right\rfloor dsInterval\right)$$

where $\lfloor x\rfloor$ is the integer part of x and *dsInterval* is an integer down-sample interval.

It is easy to show that with such a formulation, when the sample means are accurate, $E\left[y^*\left(n\middle|n-V-d\right)\right]=E\left[y(n)\right]$, i.e. q^* provides proper DC correction [18].

Figure 6b shows the case when $dsInterval=500$, i.e. $q^*(n)$ is updated once per 500 msec. The final measured eigenvalue spread is 6. The convergence rate is satisfactorily fast. If $q^*(n)$ did not use down-sampled versions of $u_{SM}(n)$ and $y_{SM}(n)$, the resulting feedback path would create instability.

B. Control of Queue Size

Congestion Control work done by control theorists, e.g. [4]-[6], [7], often explicitly include queue matching in addition to rate matching in their cost functions, no doubt in part a response to [6]. In contrast, Section II.B presents a pure rate-matching controller. This strategy requires that the bandwidth available for ABR traffic be slightly under-utilized, thus creating extremely short (or zero) queue lengths in steady state. While this has advantages, e.g. shorter end-to-end delay and smaller memory requirements, it may be more desirable to have, on average, longer buffers. Since ABR is not designed for delay-sensitive traffic, it may be preferable to add a small, known delay by targeting a non-zero buffer size in order to ensure network efficiency. The scheme presented thus far does not allow for a desired queue depth greater than zero.

The proposal here is similar to [10], but distinct in that it scales the set point, $y^*(n+d+V\mid n)$, not the explicit rate $u(n)$ directly. Specifically, decide at time *n* the target input rate for time $n+d+V$, but notate this as $\Theta(n+d+V\mid n)$ instead of $y^*(n+d+V\mid n)$. The target input rate $\Theta(n+d+V\mid n)$ is chosen without regard of the queue size. Further, for simplicity of presentation, assume that $\Theta(n+d+V\mid n)$ is the actual service capacity for ABR traffic at $n+d+V$. Define a scalar $\eta(n)$ that is monotonically decreasing function of the queue size $queue(n)$. Control of this queue size is accomplished by

$$y^*(n+d+V\mid n)=\eta(n)\Theta(n+d+V\mid n). \quad (8)$$

To target a non-zero queue-depth, use a $\eta(n)$ function that decreases monotonically with *queue*(*n*). A simple function is shown in Figure 8.

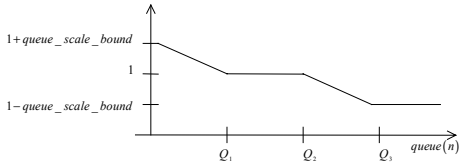
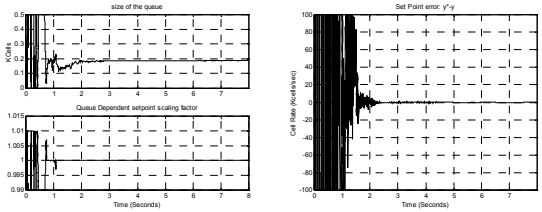


Figure 8 Sample $\eta(n)$ Function

This queue-aware set-point $y^*(n+d+V\mid n)$ is used in exactly the same way as outlined in Section II.B. The plant model now includes the queue-depth $queue(n)$, which progresses as

$$queue(n+1)=queue(n)+y(n)-\Theta(n\mid n-d-V). \quad (9)$$

To demonstrate the effectiveness of this strategy, a representative simulation, with the $\eta(n)$ function shown in Figure 8, *queue_scale_bound* = 0.01, Q_1=100 cells, Q_2 = 200 cells, Q_3 = 300 cells, the target-queue-depth is within the desired range of $\left[Q_1, Q_2\right]$ without perceptibly affecting the convergence rate, as shown in Figure 9. With no queue control, i.e $\eta(n)=1$, simulations show the steady-state queue frequently grows to thousands of cells.



a b

Figure 9 (a) Queue depth (upper plot) and Set-point scaling factor $\eta(n)$ (lower plot) when queue target is 100-200 cells. (b) Set point error when queue depth is actively controlled.

Clearly potentially destabilizing feedback is created by performing queue control with (8). Intuition suggests, and simulations confirm, that stability is only in jeopardy when the scaling of $\eta(n)$ is aggressive, e.g. $queue\_scale\_bound \geq 0.1$. It is intuitive that using a small *queue_scale_bound* can make the impact of $\eta(n)$ on $y^*(n+d+V \mid n)$ nearly negligible, yet still effect the desired behavior.

IV. Summary and Concluding Remarks

This paper takes up the challenge of finding an effective control strategy for the explicit rate congestion controller. The problem is motivated in Section I, where other related work is summarized. The system under study is defined in Section II. The new contributions of this paper are presented in Section III. These contributions consist of algorithm enhancements to the system defined in Section II, and include convergence rate improvements and queue depth management. A method to reduce coefficient bias without compromising convergence or significantly increasing computational complexity has also been found [18] (see Footnote 2), but not reported here due to space limitations.

There are several potential directions for future research. One path would examine real-world protocols and networks in an attempt to improve the fidelity of the plant model. This will almost certainly create a more complex plant model. Modeling the blending effect introduced in [18] is but one possibility. Other modeling extensions include delayed or lost data (e.g. resource management cells), non-linearities due to rate and buffer saturations, bursty sources, and other phenomena.

References

[1] J. Kenney, Editor, *Traffic Management Specification Version 4.1*, available from [19].

[2] R. Jain, "Congestion Control and Traffic Management in ATM Networks: Recent Advances and A Survey," Comp Net ISDN Sys, Vol. 28, No. 13, pp. 1723-1738, October 1996.

[3] C. Rohrs, R. Berry, and S. O'Halek, "Control engineer's look at ATM congestion avoidance," Comp Comm, v 19 n 3, pp. 226-234, Mar 1996.

[4] L. Benmohamed and S. M. Meerkov, "Feedback Control of Congestion in Packet Switching Networks: The Case of a Single Congested Node," IEEE/ACM Trans Netw, Dec 1993.

[5] L. Benmohamed and S. M. Meerkov, "Feedback control of congestion in packet switching networks: The case of multiple congested nodes," Int J Comm Sys, p 227-246, Sep-Oct 1997.

[6] E. Altman, F. Baccelli, J-C. Bolot, "Discrete-time analysis of adaptive rate control mechanisms," Intl. Conf. Data Comm Sys and Performance, pp. 121-140, Raleigh, NC, Oct. 1993.

[7] E. Altman, T. Basar, and R. Srikant, "Robust rate control for ABR sources," Proceedings - IEEE INFOCOM v 1, p 166-173, Mar 29-Apr 2 1998.

[8] O. Imer et al., "ABR Congestion Control in ATM Networks", IEEE Control System Magazine, February 2001.

[9] R. Jain, S. Kalyanaraman, R. Goyal, S. Fahmy, F. Lu, "ERICA+: Extensions to the ERICA Switch Algorithm," AF-TM 95-1145R1, October 1995.

[10] B. Vandalore, R. Jain, R. Goyal, S. Fahmy, "Design and Analysis of Queue Control Functions for Explicit Rate Switch Schemes," IC3N '98, pp. 780-786, October 1998.

[11] S. Fahmy, R. Jain, S. Kalyanaraman, R. Goyal and B. Vandalore, "On Determining the Fair Bandwidth Share for ABR Connections in ATM Networks," ICC, 1998.

[12] K. Laberteaux and C. Rohrs, "Application Of Adaptive Control To ATM ABR Congestion Control," Globecom, 1998.

[13] C. Fulton and S. Q. Li , "UT: ABR Feedback Control with Tracking," Infocom'97, April 1997.

[14] S. Mascolo, "Smith Principle for Congestion Control in High-Speed Data Networks," Trans. Auto. Control, Feb. 2000.

[15] K. Laberteaux and C. Rohrs, "On the Convergence of a Direct Adaptive Controller for ATM ABR Congestion Control," Int Conf Comm 2000, June 2000.

[16] K. Laberteaux and C. Rohrs, "A Direct Adaptive Controller for ATM ABR Congestion Control," Amer Ctrl Conf, June 2000.

[17] K. Laberteaux and C. Rohrs, "A Proof of Convergence for a Direct Adaptive Controller for ATM ABR Congestion Control," http://www.nd.edu/~isis, September 2000.

[18] K. Laberteaux, "Explict Rate Congestion Control for Data Networks," Dissertation, Univ. of Notre Dame, available from http://www.geocities.com/klaberte/dissertation.html, 2000.

[19] ATM Forum web site, http://www.atmforum.org.

[20] M. Tarrab and A. Feuer, "Convergence and Performance Analysis of the Normalized LMS Algorithm with Uncorrelated Gaussian Data," Trans Info Theory, Vol. 34, No. 4, July 1988.

[21] Haykin, Simon S., *Adaptive Filter Theory*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1991.

[22] Mathworks, Inc., *Matlab*, R11, http://www.mathworks.com/.