A Predictive Self-Tuning Fuzzy-Logic Feedback Rate Controller

Rose Qingyang Hu and David W. Petr

***Abstract*— This paper addresses the design and analysis of an end-to-end, rate-based, feedback flow control algorithm motivated by the Available Bit Rate (ABR) service in wide area Asynchronous Transfer Mode (ATM) networks. Recognizing that the explicit feedback rate at time t will not affect the ABR buffer until time $t + D$ for some $D \geq 0$, our approach is to first predict the ABR buffer status at time $t + D$, then base fuzzy-logic rate control decisions on these predicted values, and finally tune the controller parameters using gradient descent methods. Simulations show that this predictive self-tuning fuzzy-logic (PSTF) control scheme is efficient, stable, and outperforms other proposed ABR rate controllers in a variety of network environments. With delays corresponding to a U.S. coast-to-coast connection, the PSTF controller can maintain high link utilization, avoid buffer overflows, and provide fair allocation of resources.**

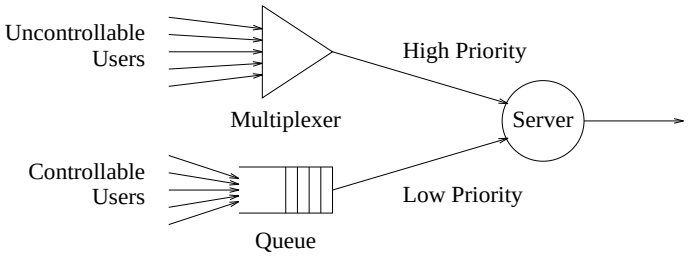

Figure 1: Single node network architecture

1 Introduction

1.1 Problem statement

Communication networks that are based on statistical multiplexing must exercise control over the submitted traffic in order to ensure adequate quality of service (QoS) for all network users. When the traffic characteristics or the QoS requirements or both differ from user to user, as expected in Asynchronous Transfer Mode (ATM) networks, the problem becomes more challenging. A variety of traffic controls can be imposed to address this problem, including call admission control and various flow control mechanisms for the admitted calls.

We consider a flow control problem motivated by ATM networks, but applicable as well in a broader context. A single-node ATM network formulation is shown in Figure 1. Users (sources) are divided into those willing and able to adjust their traffic patterns dynamically in response to network feedback (controllable users), and those that are not (uncontrollable users). At the network node, all controllable users share a single finite queue, and uncontrollable users are multiplexed together (exactly how is not important here). Controllable and uncontrollable users share a single server, with service priority given to the uncontrollable users. Due to fluctuations in the high-priority uncontrollable traffic, the queue for the controllable users experiences a time-varying service rate. The problem is to design a traffic rate controller that will dynamically communicate allowed rates to the controllable users with the goal of maintaining high overall server utilization concurrently with good QoS (low delay and loss) for the controllable users. We apply a combination of self-tuning fuzzy-logic control and minimum variance prediction to this problem.

In an ATM network context, the traffic is data in the form of fixed-size packets known as cells. The shared server is a fixed-rate, high speed communication link. Uncontrollable traffic corresponds to some mixture of Constant Bit Rate (CBR) and Variable Bit Rate (VBR) traffic, controllable traffic corresponds to Available Bit Rate (ABR) traffic, the feedback rates provided by the controller correspond to Explicit Rate (ER) indications from ATM network elements, and the means of feedback communication would be special Resource Management (RM) cells, all of which are defined in detail in [1]. However, this paper does not attempt to follow in detail the current ABR specifications, which include a variety of feedback signals and procedures. We assume merely that the allowed rates are communicated directly to the sources (this corresponds to backward ER control of ABR sources), with a possible delay comprising processing and propagation delays, and that the sources immediately conform themselves (in ways described later) to the allowed rates.

The multiple-node formulation is a generalization that allows each node in a network to function as described above. The generalization would require a means of combining the multiple allowed rates for each source (one from each node through which the source traffic passes) into a single allowed rate, for example by choosing the smallest nodal rate. Our focus in this paper is the controller design and its performance in configurations with a single bottleneck node.

Rose Qingyang Hu is with the Wireless Systems Engineering Group of Nortel Networks. David W. Petr is with the Information and Telecommunication Technology Center, Department of Electrical Engineering and Computer Science, University of Kansas.

1.2 Review of previous research

The ABR service makes it possible for ATM networks to meet the needs of increasing data applications which can take the available bandwidth based on network feedback and tolerate unpredicatable end-to-end delay. Considerable work has been done in this area during the last several years [9, 11, 22, 16, 21, 23, 2, 13]. Many studies have taken a rate-based approach, which was eventually adopted as the ATM forum standard [1]. In [2], the authors proposed a rate-based feedback congestion control approach in a packet switching network. It focuses on the transient behavior of the ABR queue length. An estimation and prediction approach to congestion control in ATM networks is addressed in [28]. The analysis is based on the assumption that the underlying traffic model is a Markov process, which has been proven insufficiently accurate for describing traffic behavior in ATM networks [18]. In recent work described in [4, 11, 22] , the proposed congestion control schemes have good steady state performance and also can deal with large propagation delay, except that they either consider only a specific type of traffic model, e.g., Markov-modulated traffic, or greedy (persistent) ABR sources. These assumptions may not be consistent with real network situations. Some other binary feedback rate-based congestion methods are proposed in [6, 21, 13, 15]. However, in these binary feedback schemes, the ABR traffic must converge gradually to the available bandwidth, whereas ER feedback allows the ABR traffic to immediately utilize available bandwidth. Most importantly, the ER method is more compatible with some other control functions, i.e., ABR UPC [6]. In [24], the author analyzes queue evolution of both binary and ER algorithms, obtaining results for maximum queue length and concluding that ER control performs much better.

The problem remains how to design an efficient congestion controller which can deal with poorly understood ABR and VBR traffic models, large propagation delay and many other uncertainties existing in ATM networks. Recently, neural networks and fuzzy logic control have emerged as viable techniques for dealing with complicated characteristics and environmental changes that cannot be described by an exact mathematical process [8, 26, 19, 5, 8]. Although a number of possible approaches could be taken to this problem, here we propose a self-tuning predictive fuzzy rate control scheme. The smallest round trip propagation delay among all active ABR connections transversing the specified switch is designated D, and both the feedback rate and parameter adaptation for the fuzzy-logic controller are based on the D-ahead predicted future network status, thereby compensating for the effect of round trip delay. This approach is appealing to us because of the intuitive nature of its control (fuzzy logic), its explicit adaptability (self-tuning), and its ability to compensate for the effect of round-trip delay (prediction).

In our simulations we adopt actual movie trace data as VBR traffic. This type of VBR traffic is extremely bursty and no existing traffic models can adequately describe it [18]. Furthermore, in some simulations, we use actual Ethernet trace data from Bellcore as the ABR traffic. The Ethernet trace data differs from commonly-used greedy ABR sources in the sense that sometimes it has less traffic than it is allowed to send.

2 Description of the predictive self-tuning fuzzy rate controller

2.1 Overall control topology

Our approach to the design of a traffic rate controller that will provide high server utilization and good QoS (loss and delay) for the controllable traffic (which we will henceforth refer to as ABR traffic) is based on the following observations. Due to the priority service for the uncontrollable traffic (VBR and CBR), a non-zero ABR buffer length implies full server utilization, thereby minimizing delay. Losses occur only when the buffer overflows its capacity q_c. Hence the controller goals can be satisfied if we maintain the ABR buffer length $q(t)$ close to some non-zero desired length q_d. On the other hand, the large propagation delays in the feedback path existing in Wide Area Networks (WANs) could degrade the controller performance greatly, especially when the VBR traffic is highly unpredictable, unless future knowledge of the network is available. This motivates the design of a predictor to augment the traffic regulator. Although it is impossible to make an exact network status forecast, we still can make the prediction based on some criterion of "goodness", e.g., minimum variance, which is referred to as Minimum Variance Prediction (MVP) [31].

So our overall ABR rate controller consists of two parts: predictor and regulator, as shown in Figure 2. The system to be controlled consists of the ABR source (or sources) and the associated queue. The controller operates with a fixed update/feedback interval T. That is, all controller variables are updated and the rate feedback signal is sent once every T seconds. Thus time is discrete for the controller with all time-related quantities taking on integer values with a fundamental discrete time unit of T seconds. For each interval of T seconds, the set of system measurements used by the regulator is represented by the vector $\hat{X}(t+D)$, which is estimated by the predictor. Unless otherwise noted, both t and D are expressed in fundamental controller time units of T seconds (so that the numerical value 1 in these units represents T seconds). D is the minimum round trip propagation delay (controller to source and back) among all active ABR sources. The regulator takes these predicted values as its input and calculates the per-source allowed feedback rate (in cells per T sec) for the next time interval $r(t+1)$, which is sent to all ABR sources associated with this queue. The function J, described later, is the optimization criterion used by the regulator for tuning its parameters. The predictor takes the system measurements and allowed feedback rate as inputs and uses the MV prediction technique to estimate the fu-

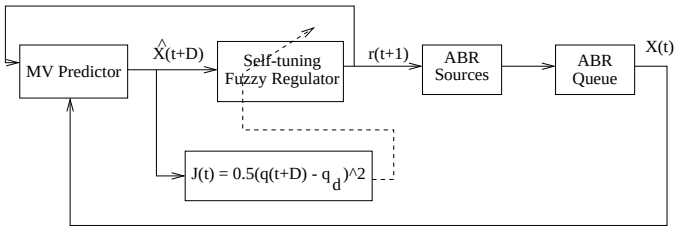

Figure 2: Controller topology in each ATM network node

ture system states.

In the development of the controller (sections 2.2 through 2.4), we assume that there is no lower bound on the allowed feedback rates. In ATM terminology, we assume all ABR sources have a minimum cell rate (MCR) of zero. In section 2.7, we show how the system can be modified easily to achieve fair treatment of all sources even when the sources have differing nonzero MCR values.

2.2 Self-tuning fuzzy logic rate regulator

2.2.1 Fuzzy control theory background

Traditional control engineering uses mathematical models of a system and its inputs to design control actions and/or to analyze their effectiveness. Fuzzy control denotes the field within control engineering in which fuzzy set theory and fuzzy inference are used to derive control laws. It is especially useful for situations in which either the system to be controlled or its inputs cannot be adequately modeled mathematically. For an ABR controller, there is considerable uncertainty about both the input ABR traffic and the time-varying ABR queue service rate, making fuzzy techniques particularly attractive.

The concept of a fuzzy set is an extension of the concept of a traditional, or crisp, set. For a crisp set B, an element either belongs to B or does not. This relationship can also be expressed as a mapping whose domain is some characterization of possible elements of B and whose range is the binary space $\{0, 1\}$. This mapping is called the characteristic function of the crisp set B. A fuzzy set, on the other hand, is defined by a membership function whose range is the closed interval $[0, 1]$. Any value between 0 and 1 can express the degree of membership of a particular element in the fuzzy set. This concept of fuzzy sets makes it possible to use fuzzy inference, in which the knowledge of an expert in a field of application is expressed as a set of "IF-THEN" rules, leading to algorithms describing what action should be taken based on currently observed information, or in our case, on predicted future information.

Fuzzy controllers are the applications of fuzzy sets and fuzzy inference in control theory. Their operation is typically divided into the following three phases.

1. *Fuzzification* is a procedure to define fuzzy sets based on system input measurements. That is, fuzzification defines the membership mappings from the measured values of each input to a set of linguistic values for that input.

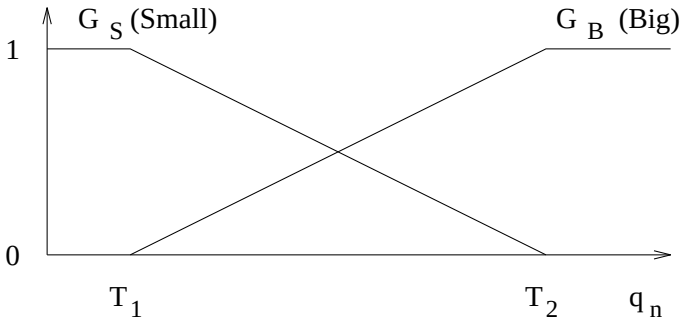

Figure 3: Membership function for queue length $\hat{q}_n$

2. *Inference* is an interface that produces a new fuzzy set from the result of the fuzzification (the linguistic values) using a set of rules. The result of the inference is a fuzzy set that can be called the fuzzy control action.
3. *Defuzzification* is the procedure that produces a crisp control output from the result of the inference.

2.2.2 Regulator input fuzzification

As mentioned earlier, the input vector of the regulator $\hat{X}(t + D)$ comes from the predictor. This estimated input vector consists of the following elements.

1. D-ahead normalized predicted queue length:
$$\hat{q}_n(t + D) = \hat{q}(t + D)/q_c$$
2. D-ahead predicted queue length rate of change:
$$d\hat{q}(t + D)/dt = \hat{q}(t + D) - \hat{q}(t + D - 1)$$
3. Estimated normalized number of lost cells in D-ahead interval:
$$\hat{n}(t + D) = \hat{l}(t + D)/q_c$$
where $\hat{l}(t + D)$ is the estimated number of lost cells.

The fuzzification of the measurement inputs can be described as follows. Let X_1 be the set of all possible values associated with the measurement of $\hat{q}_n(t + D)$; X_2 with $d\hat{q}(t + D)/dt$; X_3 with $\hat{n}(t + D)$.

Let $L(X_i)$ be a set of linguistic values (words) characterizing any measurement over X_i. We define these as:

$$\begin{aligned} L(X_1) &= \{Small\ (S),\ Big\ (B)\}; \\ L(X_2) &= \{Negative\ (N),\ Positive\ (P)\}; \\ L(X_3) &= \{Zero\ (Z),\ Nonzero\ (Nz)\}. \end{aligned}$$

We define $G_L(x)$ as the membership function associated with word L and measurement value x. For the first two measurements, we select complementary triangular shape functions, shown in Figures 3 and 4, as in most industry applications. For example, from Figure 3, if $\hat{q}_n(t + D)$ is slightly less than threshold T_2, $G_S(\hat{q}_n)$ is close to 0 and $G_B(\hat{q}_n)$ is $1 - G_S(\hat{q}_n)$.

For the last measurement input $\hat{n}(t + D)$, X_3 is treated as a crisp set instead of a fuzzy set, but it can be considered as a degenerate fuzzy variable and defined as follows.

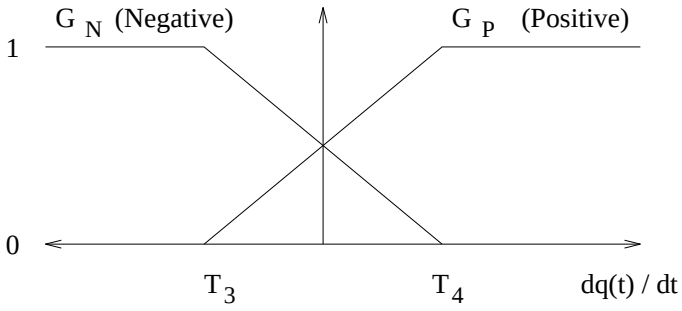

Figure 4: Membership function of queue length change rate $d\hat{q}/dt$

1. If $\hat{n}(t+D)$ is zero, then $G_Z(\hat{n}) = 1$; $G_{Nz}(\hat{n}) = 0$;
2. If $\hat{n}(t+D)$ is not zero, then $G_Z(\hat{n}) = 0$; $G_{Nz}(\hat{n}) = 1$;

The justification for the use of a crisp set in this case is that any cell losses (regardless of the number) represent a serious situation that must be acted upon quickly. Viewed another way, any cell losses are likely to come in bursts, so the number of losses in the burst is of secondary importance; of primary importance is the detection of the burst loss event itself.

The threshold values for the membership functions were selected based on preliminary simulation-based sensitivity studies. These studies involved a system similar to the one described in the simulation comparison section 3.2 but with much smaller ABR queue capacity ($q_c = 30$ cells and $q_d = 10$ cells), and they confirmed the expected trade-offs in threshold value selection. In these sensitivity studies, for example, increasing threshold T_2 between 0.2 and 1.0 increased link utilization (from 0.96 to above 0.995) and mean buffer length (from 4.5 to 14 cells), but also increased cell loss (zero cell loss through $T_2 = 0.4$, then increasing to a cell loss rate of 0.0018 at $T_2 = 1.0$). In this case, we made the tradeoff by choosing the largest value that resulted in no cell losses. Using similar procedures, we chose the following values for the remainder of the work: $T_1 = 0.2$, $T_2 = 0.4$, $T_3 = -0.1$, $T_4 = 0.1$.

2.2.3 Inference, fuzzy rules and defuzzification

We have chosen a Sugeno type fuzzy regulator [20], in which fuzzy sets are involved only in rule premises. Rule consequences are crisp functions of the output variables (usually linear functions). It is robust because few rules are needed for control. It uses a weighted average of individual rule outputs and thus there is no separate defuzzification step. Mathematically speaking, it is easy to tune for optimality, to develop learning algorithms, and to analyze stability and controllability, all of which are desirable for an ABR congestion controller.

Based on our defined measurement input variables and their membership functions, the fuzzy system is described by eight fuzzy IF-THEN rules, each of which locally represents a linear input-output relation for the regulator.

The eight fuzzy rules of our Sugeno regulator correspond to the eight combinations of the linguistic values $L(X_1)$, $L(X_2)$, and $L(X_3)$. The fuzzy rules have the following form, where A_{im} is the word value from $L(X_i)$ used in rule m and F_m is the output function of rule m (for $1 \le m \le 8$):

Rule m: IF $\hat{q}_n(t+D)$ is A_{1m} and $d\hat{q}(t+D)/dt$ is A_{2m} and $\hat{n}(t+D)$ is A_{3m}, THEN ACT $= F_m(t+1)$.

For example, substituting descriptions for the measurement inputs, Rule 8 would state:

Rule 8: IF predicted buffer size is Big and predicted buffer change rate is Positive and predicted cell loss is Nonzero, THEN ACT $= F_8(t+1)$.

The F_m's are defined as:

$$F_m(t+1) = \beta \cdot r(t) + a_m(t+1)$$

The $a_m(t+1)$ are controller parameters (eight of them) that are adapted as described later. The constant β has positive value less than 1.0 to maintain controller stability, as discussed below.

For rule 8 (above), the desired result is clear: the parameters of function F_8 should act to decrease the allowed rate. So a_8 should be negative and its absolute value should be relatively large. The desired result in other cases is not as obvious, but the self-tuning feature (described in section 2.2.5) is used to adjust all $a_m(t+1)$ to near-optimal values.

If all membership functions were crisp, then exactly one rule would have its conditions satisfied at any given time and the output (new allowed rate) would be the appropriate F_m. However, our fuzzy membership functions allow each condition of each rule to be satisfied to some degree, so the output is a weighted combination of the F_m values:

$$\begin{aligned} r(t+1) &= \textstyle\sum_{m=1}^{8} F_m(t+1) w_m(t+1) \\ &= \beta \cdot r(t) + \textstyle\sum_{m=1}^{8} a_m(t+1) w_m(t+1) \end{aligned} \quad (1)$$

where the weights are

$$\begin{aligned} w_m(t+1) &= G_{A_{1m}}(\hat{q}_n(t+D)) \\ &\quad \cdot G_{A_{2m}}(d\hat{q}(t+D)/dt) \\ &\quad \cdot G_{A_{3m}}(\hat{n}(t+D)) \end{aligned} \quad (2)$$

and due to the complementary nature of the membership functions G, the weights must sum to unity: $\sum_{m=1}^{8} w_m(t+1) = 1.0$.

The role of the factor β is now clear: it represents a simple pole in the above rate evolution formula (1). We will also see in section 2.2.5 that the same β is a pole in the recursive rate gradient calculations. Thus we require a value of β strictly less than unity for controller stability.

2.2.4 ABR queue fill model in a multiple node ATM network

We begin our analysis of the ABR queue evolution for a multiple node configuration with a specific example shown in Figure 5, in which we will focus our attention on the control being applied by switch 3.

As introduced previously, D represents the minimum round trip propagation delay (RTPD) among a set of active ABR sources controlled by a particular switch. Due

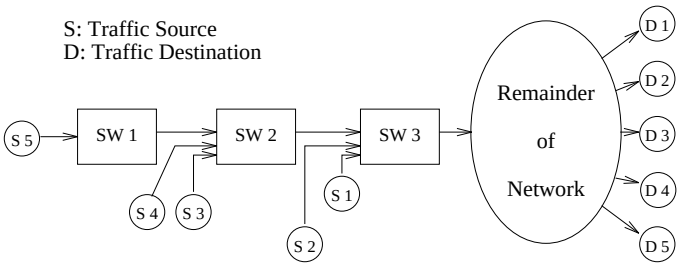

Figure 5: Multiple node ATM network topology

to topology, link distances, etc. we assume that the RTPD K_i from switch 3 to source i and back to switch 3 is as follows, with RTPD expressed in units of controller update period T (one time unit equals T seconds): $K_1 = D$, $K_2 = D+1$, $K_3 = K_4 = D+3$, and $K_5 = D+6$. Hence if switch 3 issues a feedback signal $r(t+1)$ at time t, it takes D time units for cells from source 1 regulated by this new control to reach the ABR buffer inside switch 3, $D+1$ time for cells from source 2, and so forth. $C_a(t+1)$ is the total number of ABR cells served in the T-second interval $(t,t+1]$. Parameter $\alpha_i(t+1)$ (one per source, unitless) is used to describe the transmission demand for a non-greedy ABR source i which has a round trip propagation delay K_i and actual transmission rate (at the ABR queue) $r_{ai}(t+1)$:

$$\alpha_i(t+1) = r_{ai}(t+1)/r(t+1-K_i)$$

Further, let $r_{ABR}(t+1)$ be the number of cells arriving at the ABR buffer in switch 3 during interval $(t,t+1]$, i.e., the aggregate arrival rate to the ABR queue in that interval measured in cells per T sec. With these definitions and keeping in mind the time units, the ABR queue evolution will have the following form:

$$\begin{aligned} q(t+1) &= q(t) + r_{ABR}(t+1) - C_a(t+1) \\ &= q(t) + [r_{a1}(t+1) + r_{a2}(t+1) \\ &\quad + r_{a3}(t+1) + r_{a4}(t+1) \\ &\quad + r_{a5}(t+1)] - C_a(t+1) \\ &= q(t) + [\alpha_1(t+1) + \alpha_2(t+1)z^{-1} \\ &\quad + \alpha_3(t+1)z^{-3} + \alpha_4(t+1)z^{-3} \\ &\quad + \alpha_5(t+1)z^{-6}] \cdot r(t-D+1) \\ &\quad - C_a(t+1) \\ &= q(t) + B(z^{-1}, t+1)r(t-D+1) \\ &\quad - C_a(t+1) \end{aligned} \tag{3}$$

where $B(z^{-1},t)$ is a time-varying polynomial whose coefficients reflect network topology and traffic attributes. However, note from equation (3) that $B(z^{-1},t)$ can be easily calculated as follows.

$$B(z^{-1},t) = r_{ABR}(t)/r(t-D)$$

So we find that the following equation is adequate to describe the general ABR queue behavior.

$$(1-z^{-1})q(t+D) = B(z^{-1},t+D)r(t) - C_a(t+D) \tag{4}$$

Minimum variance prediction of $q(t+D)$ will be discussed in section 2.3, in which we also show how $C_a(t+D)$ can be replaced using simple measurements.

2.2.5 Self-tuning based on gradient descent

We use the gradient descent method to adapt (tune) the parameters of the fuzzy controller to near-optimal values. The performance criterion at time t is defined as:

$$J(t) = 0.5(q(t+D) - q_d)^2 \tag{5}$$

We then define the parameter adaptation as:

$$a_j(t+1) = a_j(t) - \eta \frac{\partial J(t)}{\partial a_j(t)} \tag{6}$$

In this expression, η is an adaptive gain, which must be chosen small enough for stability.

By definition of $J(t)$,

$$\frac{\partial J(t)}{\partial a_j(t)} = (q(t+D) - q_d) \cdot \frac{\partial q(t+D)}{\partial a_j(t)} \tag{7}$$

In equation 4, $C_a(t+D)$ is not a function of $a_j(t)$ (it depends only on link capacity and the uncontrollable traffic) and, to a first-order approximation, neither is $B(z^{-1}, t+D)$, so

$$(1-z^{-1})\frac{\partial q(t+D)}{\partial a_j(t)} \approx B(z^{-1},t+D) \cdot \frac{\partial r(t)}{\partial a_j(t)} \tag{8}$$

From equation 1 for $r(t)$, we have

$$\frac{\partial r(t)}{\partial a_j(t)} = (1-\beta z^{-1})^{-1} w_j(t) \tag{9}$$

completing the adaptation calculations.

Since $q(t+D)$ is future information, we use its predicted value $\hat{q}(t+D)$ here to approximate $q(t+D)$. The prediction process will be introduced in the next section. Future prediction of $B(z^{-1},t)$ is addressed in section 2.4.

2.3 Minimum variance prediction of ABR queue fill

In this section, we will obtain the prediction for the D-ahead queue fill $\hat{q}(t+D)$. We focus on linear prediction for simplicity of computation and adaptation. We begin with our previous expression (equation 4) for queue fill evolution, obtaining an intuitive prediction function. We then show that the same prediction function can be derived more rigorously using the theory of minimum variance prediction [31].

In the governing queue fill equation 4, we can treat $C_a(t+D)$ as a noise with a nonzero mean $\bar{C}_a$, which is the average ABR service rate expressed in cells per update interval T. The buffer equation can then be rewritten as:

$$\begin{aligned} q(t+D) &= q(t+D-1) \\ &\quad + [B(z^{-1},t+D)r(t) - \bar{C}_a] \\ &\quad + [\bar{C}_a - C_a(t+D)] \end{aligned} \tag{10}$$

The control signal is defined as $u(t) = r(t) - \bar{C}_a / B(z^{-1}, t+D)$, and the term $e(t+D) = \bar{C}_a - C_a(t+D)$ can be treated

as noise with zero mean. Using these definitions, $q(t+D)$ can be expressed in terms of $q(t)$, $u(t)$, and $e(t)$ as:

$$q(t+D) = q(t) + \sum_{i=0}^{D-1} [B(z^{-1}, t+D-i)u(t-i)] + \sum_{i=0}^{D-1} e(t+D-i)$$

Since the mean of $e(t)$ is zero, the summation of the error terms $\sum_{i=0}^{D-1} e(t+D-i)$ can be neglected. So the prediction function becomes:

$$\hat{q}(t+D) = q(t) + \sum_{i=0}^{D-1} [B(z^{-1}, t+D-i)r(t-i) - \bar{C}_a] \quad (11)$$

We note that if $\bar{C}_a$ is the average service rate over the entire connection time of the ABR traffic, the above error term may not be neglected all the time, which could introduce some prediction error. Although VBR traffic could be very bursty and so also $C_a(t+D)$, its average rate may remain rather constant during a relatively long period before some dramatic traffic pattern changes happen, which can be verified in the video source traffic in Figure 7. So calculating $\bar{C}_a$ as a window-based average will result in better buffer size estimation. When the window size approaches D, the noise term $e(t)$ tends to be zero mean white noise. Then the above prediction procedure becomes a specific example of an ARMA model-based Minimum Variance k-step prediction [31]. The ABR buffer equation $q(t+D) = q(t+D-1) + B(z^{-1}, t+D) \cdot u(t) + e(t+D)$ is a special case of the general ARMA model $A(z^{-1})y(t) = B(z^{-1})u(t-D) + C(z^{-1})e(t)$, where $A(z^{-1}) = 1 - z^{-1}$, $C(z^{-1}) = 1$. Following the procedures introduced in [31], we obtain exactly the same prediction function as equation 11.

2.4 Controller simplification and summary

During the design process of the self-tuning regulator and MV predictor, we have assumed that we exactly know the term $B(z^{-1}, t)$. Direct construction of $B(z^{-1}, t)$ would require knowledge of the number of active ABR VCs, the round trip propagation delay for each VC, and the transmission demand for each VC. Fortunately this is not necessary since, as already shown, $B(z^{-1}, t)$ can be calculated as $r_{ABR}(t)/r(t-D)$, where $r_{ABR}(t)$ is a measurement of average arrival rate at the ABR queue in the interval $(t-1, t)$. Note that if the set of ABR connections is static over some interval, we would not expect B to change significantly during that interval; in particular, if all ABR sources are greedy (transmission demands α_i are all constant at 1.0), it will not change at all. Based on this observation, we will assume that during the time interval $[t, t+D)$, the polynomial B remains essentially constant. Thus we have $B(z^{-1}, t+D) \approx B(z^{-1}, t+D-1) \approx B(z^{-1}, t) = r_{ABR}(t)/r(t-D)$. Of course, if ABR VCs are added, removed, or significantly change their demand in this interval, this prediction will suffer for a very short time period until sufficient history has been built up for accurate prediction.

The following summarizes the predictive self-tuning, fuzzy-logic (PSTF) control algorithm at time $t+1$.

1. Predict D-ahead PSTF inputs.
   From (11) with the approximations just discussed, the predicted ABR queue size is:

$$\begin{aligned} \hat{q}(t+D) &\approx q(t) + (r(t) + r(t-1) + ... \\ &\quad + r(t-D+1)) \frac{r_{ABR}(t)}{r(t-D)} - D\bar{C}_a \end{aligned} \quad (12)$$

   By definition, the queue size change rate is:

$$d\hat{q}(t+D)/dt = \hat{q}(t+D) - \hat{q}(t+D-1) \quad (13)$$

   Finally, lacking a good model upon which to base a prediction of cell losses, we will assume they remain approximately the same:

$$\hat{n}(t+D) \approx \hat{n}(t) \quad (14)$$

2. Update the fuzzy regulator parameters $a_i(t)$, i=1...8, through the following recursive formulas. From (9):

$$\frac{\partial r(t)}{\partial a_j(t)} = \beta \cdot \frac{\partial r(t-1)}{\partial a_j(t-1)} + w_j(t) \quad (15)$$

   Then, adding a pole γ to (8) that is very close to but less than unity to avoid infinite memory and marginal stability of the gradient evolution and recalling the approximation above for $B(z^{-1}, t+D)$:

$$\begin{aligned} \frac{\partial q(t+D)}{\partial a_j(t)} &\approx \gamma \frac{\partial q(t+D-1)}{\partial a_j(t-1)} \\ &\quad + [r_{ABR}(t)/r(t-D)] \frac{\partial r(t)}{\partial a_j(t)} \end{aligned} \quad (16)$$

   From (7)

$$\frac{\partial J(t)}{\partial a_j(t)} = (\hat{q}(t+D) - q_d) \cdot \frac{\partial q(t+D)}{\partial a_j(t)} \quad (17)$$

   And finally from (6)

$$a_j(t+1) = a_j(t) - \eta \frac{\partial J(t)}{\partial a_j(t)} \quad (18)$$

3. Calculate the feedback rate $r(t+1)$. First calculate the weights from (2), then compute the new feedback rate from (1), repeated below.

$$r(t+1) = \beta \cdot r(t) + \sum_{m=1}^{8} a_m(t+1) w_m(t+1) \quad (19)$$

2.5 Behavior for non-greedy sources

ABR algorithms are oftened evaluated with greedy sources (sources that will take any bandwidth made available to them), but a significant portion of real-world sources can be expected to be non-greedy. This section briefly discusses the PSTF algorithm's behavior with such sources.

If an ABR algorithm is designed with only greedy sources in mind, it may interpret a discrepancy between available ABR bandwidth and used bandwidth as an indication that

the allowed rates need to be increased. In such a case, the ABR algorithm may give ever larger allowed rates to the sources (limited perhaps only by link rate), with potentially undesirable consequences. For example, in a single-source scenario, suppose that the source is relatively inactive (non-greedy) for some time with the allowed rate increasing far beyond some measure of the available ABR bandwidth. If the source then becomes active (greedy), the result could be severe cell loss. This type of situation has, in fact, been observed for other ABR algorithms, as described in section 3.5.

The PSTF algorithm, through its self-tuning procedure described in section 2.2.5, effectively limits the allowed rate $r(t)$ to reasonable values for non-greedy sources. A general analysis would be quite complicated, but we can illustrate the mechanism by considering the simple, extreme case of a single ABR source that is completely idle for an indefinite period of time.

In such a case, it is clear that q_n, dq/dt, and r_{ABR} would all remain at zero, and all but two of the eight fuzzy conditions would be "inactive" (have weight w_m zero). The remaining two conditions would be "active": small queue fill and zero cell loss and queue change rate either positive or negative. We will designate the weights corresponding to these two active conditions as w_+ and w_-, each of which will have value 0.5 (see Figure 4 and equation 2). Following the algorithm summary in section 2.4, $\hat{q}$ will be zero, $\frac{\partial r(t)}{\partial a_j(t)}$ will saturate to zero for the inactive conditions and to $0.5/(1-\beta)$ for the active conditions, $\frac{\partial q(t+D)}{\partial a_j(t)}$ will saturate at 0 for all conditions (active and inactive) since r_{ABR} remains at zero, so $\frac{\partial J(t)}{\partial a_j(t)}$ will also saturate at 0. This in turn will cause all of the a_j values to saturate. Let a_+ and a_- be the coefficients associated with the two active weights w_+ and w_-. We designate the limiting values of a_+ and a_- as $\bar{a}_+$ and $\bar{a}_-$, whose specific values will depend on the history of the self-tuning process. Finally, it can be seen that $r(t)$ will saturate at a value of $\frac{\bar{a}_+ + \bar{a}_-}{2(1-\beta)}$. So there will certainly be an upper bound on the allowed rate given to an idle ABR source. Although this simple analysis does not yield the limiting value of $r(t)$, the simulation results in section 3.5 confirm that the allowed rates are held to reasonable values for non-greedy sources.

2.6 Update interval T

Recall that T is the time interval between two consecutive feedback signals. If T is too long, the feedback signal cannot capture the burstiness of the VBR traffic, which can result in overflow of the ABR queue or under-utilization of link capacity. On the other hand, if T is too small, the network must send feedback signals more frequently than necessary, and the useful link throughput will decrease. The following derives a conservative relationship between T and the ABR buffer parameters.

Let C (in cells per T seconds) be the total server (link) capacity allocated to the CBR/VBR/ABR aggregation, C_{CBR} be the total capacity required by all CBR connections, and C_{VBR} be the total maximum (peak rate) capacity allocated to all VBR connections. Then the maximum capacity available to ABR connections is $C_{a_{max}} = C - C_{CBR}$ and the minimum capacity allocated to ABR traffic is $C_{a_{min}} = C - C_{CBR} - C_{VBR}$.

Recall that the ABR buffer equation is:

$$(1 - z^{-1})q(t + D) = B(z^{-1}, t + D) \cdot r(t) - C_a(t + D) \quad (20)$$

In the worst situation, the ABR VCs may be allowed to send traffic at the rate of maximum available bandwidth,

$$B(z^{-1}, t + D) \cdot r(t) = C_{a_{max}}$$

but the actual service rate for ABR queue when this traffic arrives is the minimum available bandwidth,

$$C_a(t + D) = C_{a_{min}}$$

Assume that the buffer has already reached its desired state, so the current buffer length is approximately q_d. In the above scenario (worst case), in order to avoid overflow of the ABR buffer, the following relationship should be satisfied, where we express time and rates using seconds as the time unit for this expression only.

$$q(t + T \cdot D) = q_d + (C_{a_{max}} - C_{a_{min}}) \cdot T \leq q_c$$

So T (in cell time slots) should be less than

$$T_{max} = \frac{q_c - q_d}{(C_{a_{max}} - C_{a_{min}})/C}$$

From the above equation, T_{max} is larger than $(q_c - q_d)$ time slots, so a conservative policy would be to let $T = q_c - q_d$.

2.7 Achieving fairness with MCR $\neq 0$

According to the fairness definition given by the ATM Forum [1], no set of ABR connections should be arbitrarily discriminated against and no set of connections should be arbitrarily favored. In the previous sections, we have explicitly assumed that all the active ABR connections have zero MCR (Minimum Cell Rate), so the same feedback rate is sent to all active VCs. However, if ABR connections have different non-zero MCR requirements, we have to consider other fairness criteria, one of which is called "MCR plus equal share", defined as follows for VC i:

$$r_i(t) = MCR_i + (R(t) - M)/N \quad (21)$$

where $r_i(t)$ is the feedback rate for VC i, MCR_i is the MCR for VC i, $R(t)$ is the total available bandwidth at time t which is decided by the controller inside the switch, M is the sum of all MCR_i, and N is the number of active VCs traversing the switch in question. This definition is problematic in that it appears to require that the controller have explicit knowledge of the number of ABR VCs that it is controlling and the sum of their MCR values. However,

we can circumvent this problem simply by expressing the "MCR plus equal share" fairness criterion as:

$$r_i(t) = MCR_i + r'(t) \quad (22)$$

where $r'(t)$ is now the per-VC, MCR-adjusted available bandwidth as determined by the controller, to which each source adds its own MCR. We now show that $r'(t)$ can be determined using a controller that is nearly identical to the one that has just been described.

Throughout the previous development in sections 2.2 through 2.6, $r_i(t)$ never appeared explicitly because we had $r_i(t) = r(t)$, which must now be replaced with the feedback rate assignment in equation 22. Following an identical development with this substitution and assuming that each VC transmits with rate at least equal to its MCR, we see that equation 4 becomes

$$(1-z^{-1})q(t+D) = B(z^{-1}, t+D)r'(t)+M-C_a(t+D) \quad (23)$$

where M is again the sum of the MCRs of the ABR VCs being controlled. We recognize $B(z^{-1}, t+D)r'(t) + M$ as $r_{ABR}(t+D)$ and note that all other expressions are valid with the simple substitution of $r'(t)$ for $r(t)$. Thus the only change required from the previous controller algorithm is the expression for $B(z^{-1}, t)$ in terms of $r_{ABR}(t)$, which should now become $B(z^{-1}, t) = \frac{r_{ABR}(t)-M}{r'(t-D)}$. However, this expression still has the drawback that the controller must know the value of M. Noting that M is often likely to be small compared to r_{ABR} and that the adaptive mechanisms in the controller should be able to deal with small errors, we approximate B as $B(z^{-1}, t) \approx \frac{r_{ABR}(t)}{r'(t-D)}$, just as in the zero-MCR case. Hence the modified algorithm is identical to the one summarized in section 2.4 with $r(t)$ universally replaced by $r'(t)$ and the individual feedback rate assignments being $r_i(t) = MCR_i + r'(t)$. As the simulation results in the next section show, this approximation still yields excellent performance.

3 Performance evaluation via simulation

The tradeoffs between mathematical analysis and simulation are well known [7]. However, since it has been shown that both video and LAN measured data traffic cannot be accurately modeled with existing traffic models [18], we believe that simulation is necessary for predicting the operational performance of an ABR rate controller.

3.1 Simulation methodology

Our simulation topology is shown in Figure 6. The simulations assume an ATM environment, so that the basic unit of time is a cell slot, the time required to transmit a single 53-byte ATM cell on the transmission link. Simulation is carried out using the BONeS software package [25].

The uncontrollable traffic (see Figure 1) is modeled as a mixture of Variable Bit Rate (VBR) and Constant Bit Rate (CBR) traffic. We drive the VBR source with multiplexing of actual trace data from the *Star Wars* movie [3]. The luminance of this video signal (monochrome video) is coded using a simplified, DCT-based, variable bit rate coding scheme that does not include differential or motion compensated coding. This type of coding can be expected to produce traffic quite similar in nature to what a JPEG video coder would produce [29]. The resulting number of bits per video "slice" is recorded every 1.39 ms (24 frames/s, 30 slices/frame). These "slices" are segmented into ATM cells for the simulation, so that each VBR source produces a burst of cells every 1.39 ms. For the simulations, we aggregated ten non-overlapping 20-second segments from the Star Wars video trace. The peak rate of this aggregated VBR source is 84.0 Mb/s (including ATM cell overhead) and its average rate is 62.5 Mb/s. The CBR source represents aggregated CBR traffic with an aggregate rate of 46.7 Mb/s and a constant cell emission interval. So that these sources occasionally consume the major part of the transmission link bandwidth, the link rate of the server is set to 140 Mb/s, making one cell slot approximately 3 μs.

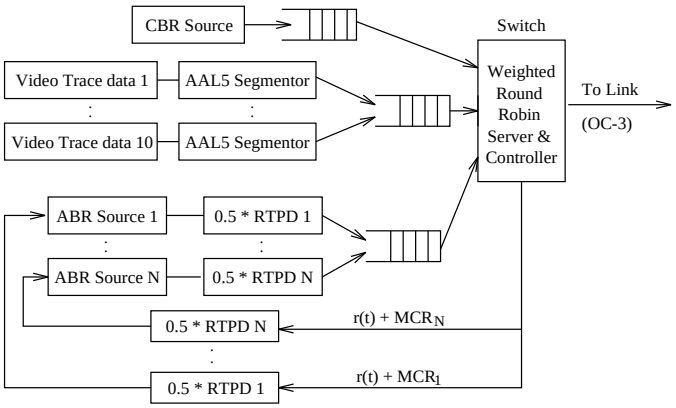

Figure 6: Single bottleneck node, multiple sources simulation topology

The bursty nature of the video source is evident from Figure 7, which shows rate (averaged over 6600 cell slots) as a function of time for the 20 second segment chosen for the simulations. The large and rapid rate fluctuations, on both short and long time scales, provide a challenging input for the rate controller.

Except where noted below, each ABR source is a bursty source derived from a 2-state discrete-time Markov model that transitions once per cell slot. In the ON state, the source emits a cell and transitions to the OFF state with probability p_1. In the OFF state, the source does not emit a cell and transitions to the ON state with probability p_2. For such a source, the mean rate (normalized to the link rate) is $\frac{p_2}{p_1+p_2}$, the mean burst length (geometrically distributed) is $\frac{1}{p_1}$ and the mean silence length (also geometrically distributed) is $\frac{1}{p_2}$. Thus, setting the average rate to the rate allowed by the controller still leaves one degree of freedom for determining the mean burst or silence in-

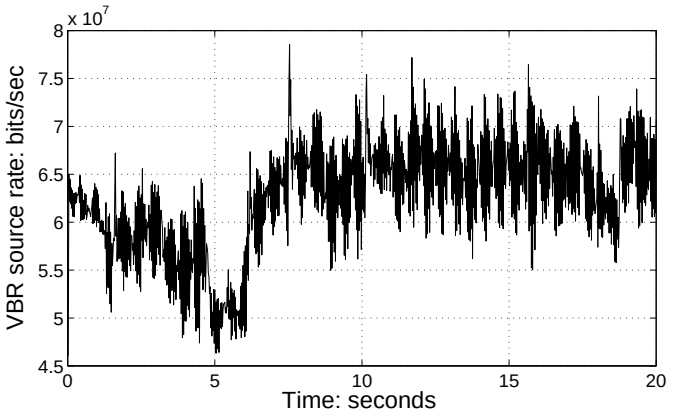

Figure 7: Video source traffic used in the simulations

terval. In the simulations, we choose $p_2 = 0.5$ whenever the normalized mean rate exceeds 0.5 (which limits p_1 to $0 \leq p_1 \leq 0.5$), and $p_1 = 0.5$ otherwise (limiting p_2 to $0 \leq p_2 \leq 0.5$).

The multiplexing and priority service are modeled using a weighted round robin (WRR) service mechanism [14, 30], in which each source (aggregate VBR, aggregate CBR, aggregate ABR) feeds its own first-in-first-out (FIFO) queue. The WRR parameters are adjusted to guarantee sufficient service rate (bandwidth) for the CBR and VBR queues, with only a token amount of bandwidth guaranteed to the ABR queue. The WRR mechanism automatically allows any bandwidth that is not used by the VBR source to be used by the ABR source. Other parameters of the system are ABR buffer capacity (q_c) of 5000 cells, desired buffer size (q_d) of 1500 cells, rate update pole (β) of 0.5, queue partial derivative update pole (γ) of 0.9998, adaptive gain (η) of 0.015, and an update interval (T) of 2000 cell slots ($0.57 * (q_c - q_d)$), which is about 6 ms. This same update interval was used for all of the controllers in the comparison section 3.2. We chose a higher queue capacity than $T + q_d = 3500$ as determined in section 2.6 so that we can observe the largest queue fill with different control algorithms. Actually in the following simulations, after a very short initial period, the queue fill never exceeds 3500 cells under the PSTF control algorithm, which matches our analysis in section 2.6. On the other hand, larger queue capacity can avoid the severe cell loss caused by inefficient initial control situations in some cases, especially when the round trip propagation delay is large.

3.2 Comparison with other controllers in a single source, single bottleneck node model

We first investigate a simple configuration involving a single bottleneck node and a single on-off ABR source as described above whose mean rate is always exactly matched to the feedback rate (we call this a greedy ABR source). For this greedy source, the controller's demand $\alpha(t) = r_a(t)/r(t - D)$ is always close to 1.

Control Algorithm	Minimum link utilization	Buffer mean (cells)	Buffer SD (cells)	Mean delay (ms)
PSTF	1.00	1469	180.4	21.0
RRM	1.00	1867	665.0	26.3
ERICA	0.94	674	366.7	9.3

Table 1: Performance comparison: single ABR source with RTPD = 0

We compare the performance of three different controllers: the predictive self-tuning fuzzy-logic (PSTF) controller described in section 2 of this paper, the relative rate marking (RRM) algorithm [1], and the ERICA algorithm [1]. The RRM algorithm is suggested by the ATM Forum as a potential single bit scheme; its parameters are high queue threshold $q_h = 2000$ cells and low queue threshold $q_l = 1000$ cells. The ERICA algorithm periodically measures the ABR input rate and attempts to direct this rate towards a target rate. The target utilization of ERICA is set to 1. Both RRM and ERICA algorithms conform to the same source behavior as described in ATM Forum [1] with parameters $AIF = 20000$, $RDF = 15/16$. We assume that the single ABR source has a round trip propagation delay denoted as $RTPD$.

3.2.1 Single ABR source with $RTPD = 0$

When $RTPD = 0$, the proposed predictor in the PSTF controller will not be invoked. As shown in Table 1, we found all of the controllers were able to provide very high link utilization (after the first update interval) under these near-ideal conditions. We note the near-perfect utilization for the RRM algorithm and the PSTF algorithm. All three controllers have no cell loss during the entire simulation. Table 1 also lists the mean value and the standard deviation (SD) of the ABR buffer size and the average ABR transmission delay for each control algorithm. Figure 8 shows that our PSTF controller maintains the tightest control over the buffer fill, followed by RRM. The control target of the ERICA algorithm is utilization instead of ABR buffer size, but its utilization is still not as high as PSTF and RRM.

Figure 8 also helps explain the low ABR delay for ERICA, since it tends to produce smaller buffer lengths and lower link utilization than the other two controllers. Of course, the delay for the PSTF algorithm could be reduced by choosing a smaller target buffer fill, with a possible reduction in link utilization (due to the buffer occasionally becoming empty). We conclude that the self-tuning fuzzy controller outperforms the other two controllers in this simple scenario.

3.2.2 Single ABR source with $D = 10T$

With $RTPD > 0$, the PSTF predictor will begin to function. $RTPD$ in these simulations is around 60 ms, close to the U.S. coast-to-coast round trip propagation delay. Fig-

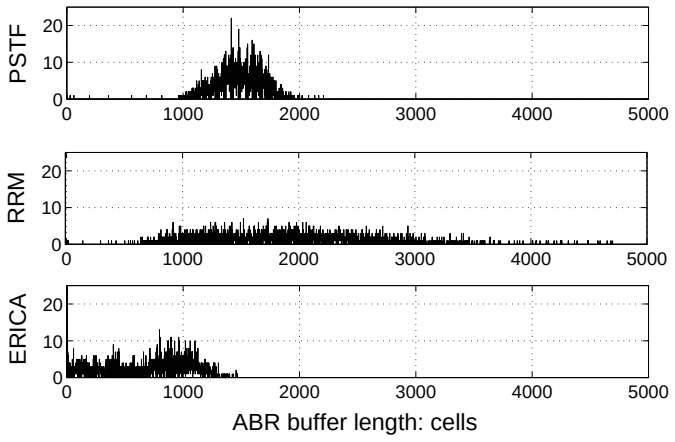

Figure 8: ABR buffer length distribution for three controllers: single ABR source with $RTPD = 0$

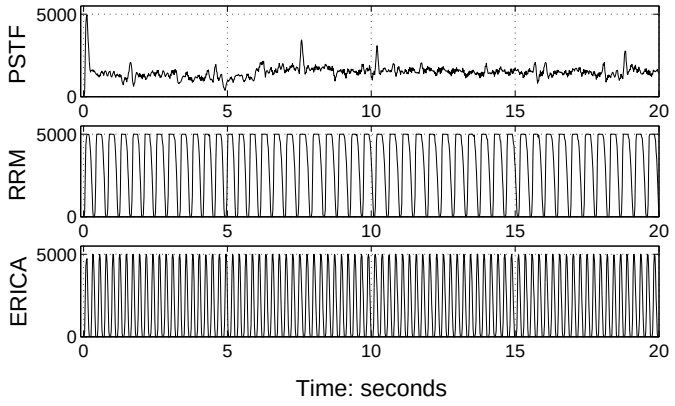

Figure 9: ABR buffer size for three controllers: single ABR source with $RTPD = 60$ ms

ure 9 shows the plots of ABR buffer fill with time for the three controllers. We observe that the buffer size oscillates between 0 and 5000 under both RRM and ERICA control, while the PSTF algorithm shows much more stable behavior. There are 996430 cells losses for the RRM algorithm and 342516 cell losses for the ERICA algorithm, while in PSTF control there are only 136 blocked ABR cells, all of which happened at the beginning of the simulation. Figure 9 shows that at the beginning of the simulation, there are not enough past measurements to be fed into the predictor and hence the predictor does not estimate the future buffer length very precisely at this point. However the buffer size quickly converges around the desired point after a very short initial period. If we were to choose a smaller Initial Cell Rate (ICR) and more conservative initial PSTF parameters, initial cell losses can be avoided. See [10] for a discussion of initial fuzzy controller parameter value selection.

We conclude that the large RTPD causes both RRM and ERICA to fail and that the predictive feature is a valuable addition to the self-tuning fuzzy-logic controller, allowing it to substantially out-perform both RRM and ERICA.

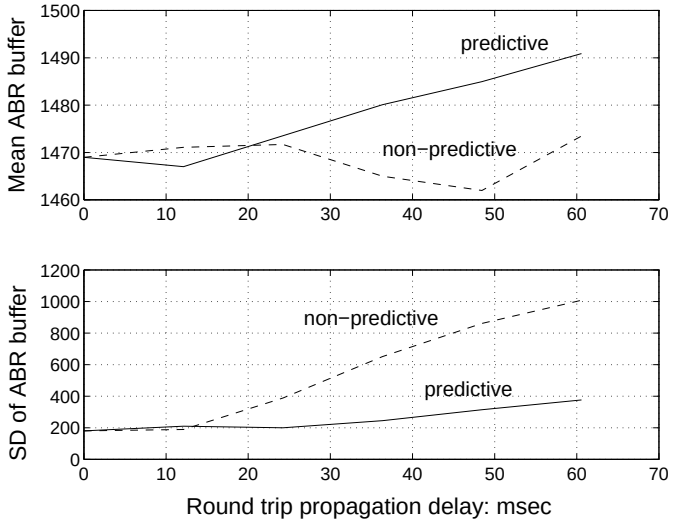

Figure 10: Comparison of PSTF and NPSTF controls: mean and standard deviation of ABR buffer length

3.3 Comparison between predictive and non-predictive self-tuning fuzzy rate control

Absolute round trip propagation delay in any closed-loop control system determines how quickly the system reacts to the control signals. It becomes more and more difficult to achieve tight control as the round trip delay increases. In classical control theory, a predictor will help to solve this large round trip propagation delay problem. Here we show the performance difference between predictive and non-predictive self-tuning fuzzy (NPSTF) rate control as $RTPD$ increases. Figure 10 shows how $RTPD$ affects the mean value and standard deviation of ABR buffer size for both predictive and non-predictive scenarios. For both predictive and non-predictive STF algorithms the mean buffer length is always within 3 percent of the desired value of 1500 cells (note the scale on the figure). However, with the same $RTPD$, the standard deviation of ABR buffer size is much smaller for the controller with the predictor, indicating much tighter control over buffer fill.

3.4 Control of multiple ABR sources

We continue to improve the realism of the simulations in this section by considering the effects of multiple controllable (ABR) sources. It is always desirable to study an algorithm's performance in the simplest topology possible, at least for initial evaluation, so we continue to use a single bottleneck node topology. We use 4 independent on-off ABR sources, each of which has a different $RTPD$ and a distinct Minimum Cell Rate (MCR) requirement. Each source receives the feedback rate calculated by the switch and updates its transmission cell rate to be the sum of the feedback rate and its MCR, as in equation 22. We simulate nonzero minimum $RTPD$ of the active ABR VCs, since in practical systems the ABR VC (or VCs) which has

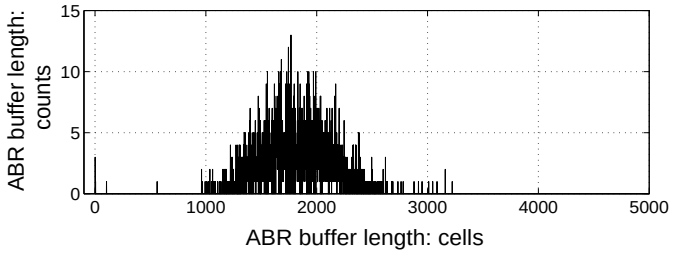

Figure 11: ABR buffer length distribution for multiple ABR sources with $RTPD_{min} = 4T$

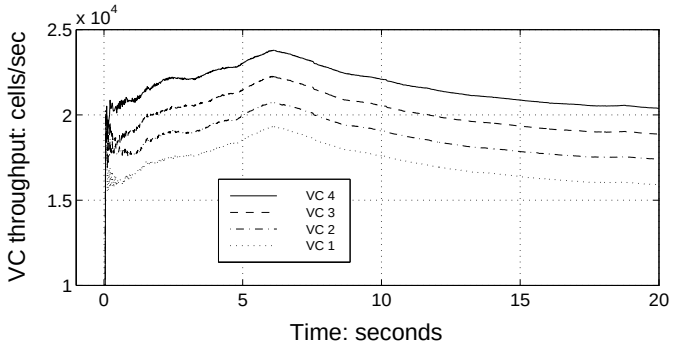

Figure 12: VC throughput for multiple ABR sources with $RTPD_{min} = 4T$

the minimum $RTPD$ does not necessary reside in the same place as the controller.

3.4.1 Multiple ABR sources with $RTPD_{min} > 0$

In this simulation, the $RTPD$ for the 4 active ABR VCs is 4T, 5T, 6T, 7T respectively and the corresponding MCR is 500 cells/s, 2000 cells/s, 3500 cells/s and 5000 cells/s (a 1500 cell/sec increment for each VC). The predictor estimates the 4T ahead buffer size and the controller takes this predicted buffer information as its fuzzy input.

Figure 11 shows that the predictive self-tuning fuzzy controller is able to keep the ABR queue size under tight control. The average queue size is 1790 cells and the standard deviation is 335 cells. The VC throughput plot is shown in Figure 12. As expected, each VC throughput converges to the proper value according to the "MCR plus equal share" fairness criterion. There is no cell loss in this simulation. The prediction alleviates the consequences of large $RTPD$ which otherwise might cause significant problems.

After 10 update intervals, the total link utilization remains above 99.9%. For relatively short simulations such as these (20 seconds of controller time), the initial transient behavior clearly depends on initial parameter values, the selection of which is discussed in [10].

3.4.2 Effect of $RTPD$ spread on performance

Since the PSTF algorithm only requires the minimum $RTPD$ among the controlled ABR sources, we investigate in this section the controller's performance as we increase the spread

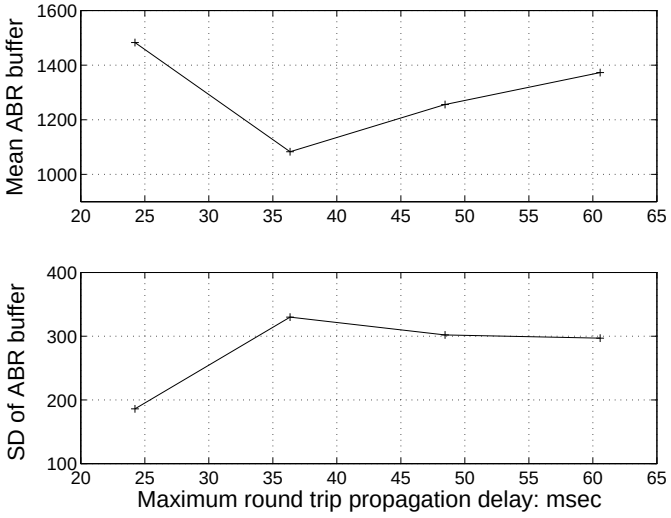

Figure 13: ABR buffer statistics vs. maximum $RTPD$ with minimum $RTPD = 24$ msec

of $RTPD$ values among the controlled sources. We use a single bottleneck node simulation with two ABR sources, each with zero MCR. The $RTPD$ of one source is held at $4T$ (approximately 24 msec), and the $RTPD$ of the other source is varied from $4T$ to $10T$ (approximately 60 ms). In Figure 13, we plot the mean and standard deviation of the ABR buffer fill as a function of the larger $RTPD$ value. As expected, the control is best when both sources have the same $RTPD$. However, the degradation in performance when the $RTPD$s are different is certainly tolerable, especially considering that there are no cell losses for any of the $RTPD$ spread values. We also see the encouraging result that the performance does not continue to degrade with increasing $RTPD$ spread. We conclude that the PSTF controller's performance is robust with respect to $RTPD$ spread.

We also use this set of simulations to illustrate the adaptation of the fuzzy controller parameters. Figure 14 shows the adaptation of parameters a_1, a_3, a_5, and a_7 for the simulation with minimum RTPD of 24 ms and maximum RTPD of 60 ms. The other four parameters correspond to rules for which cell loss is non-zero, which is never true, so these parameters do not actively adapt (see section 2.4). We see that all four parameters in Figure 14 follow similar trajectories, but the magnitudes of the changes are different. This is consistent with the updating equations in section 2.4, in which the partial derivatives of r and q must all be non-negative (since all weights w_j are non-negative), and so the sign of the increment for each a_j will be the same for a given update interval. Comparing this figure with Figure 7, we also note that the general parameter trajectory is roughly the mirror image of the VBR traffic rate trajectory. For example, when the VBR rate decreases (more bandwidth for ABR traffic), the controller parameters a_j all increase, which tends to force the feedback rate r to increase, as it should.

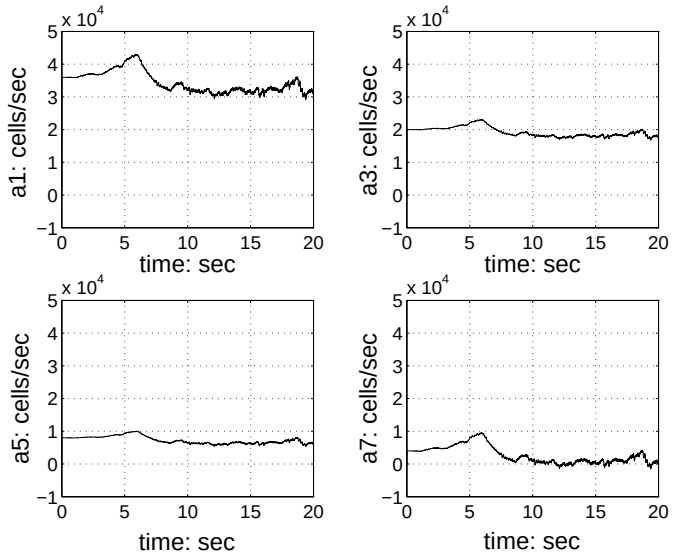

Figure 14: Fuzzy parameter adaptation with RTPDs of 24 msec and 60 msec

3.5 Results using Ethernet trace data for ABR traffic

Our final simulation combines actual video trace data for the VBR source with actual Ethernet trace data for the ABR source [17]. The Ethernet data is given as a sequence of packet size and packet interval time pairs. As with the video "slices", the Ethernet packets are segmented into ATM cells.

We assume that the Ethernet source has $RTPD = 60$ ms. As shown in Figure 15, this Ethernet trace data represents a highly variable source with mean rate 40827 cells/s and peak rate 312524 cells/s. Also shown in Figure 15 is the available ABR bandwidth $C_a(t)$ for all of the simulation experiments with an average value of 73000 cells/s, much larger than the mean Ethernet source rate. Comparing the two plots in this figure, the Ethernet source is a non-greedy source most of the time, but does frequently become greedy. To simulate the pent-up demand that would accumulate while throttling the ABR source during its greedy intervals, we place an infinite FIFO buffer between the source and the ABR buffer as shown in Figure 16. The output rate of the infinite FIFO is adjusted to match the allowed rate.

In section 2.5, we discussed the behavior of ABR algorithms with non-greedy sources, stressing that the allowed rate should be bounded to a reasonable value (ideally, close to the available ABR bandwidth) rather than increasing to the line rate. Figure 17 shows the feedback rate with time under control of the three different algorithms. We see that the feedback rate from PSTF is very well bounded, even though the ABR queue fill is small most of the time. In contrast, both the RRM and ERICA algorithms frequently allow the feedback rate to saturate at the line rate of approximately 330,000 cells/s. Comparing Figure 17 with Figure 15, we observe that the feedback rate from PSTF

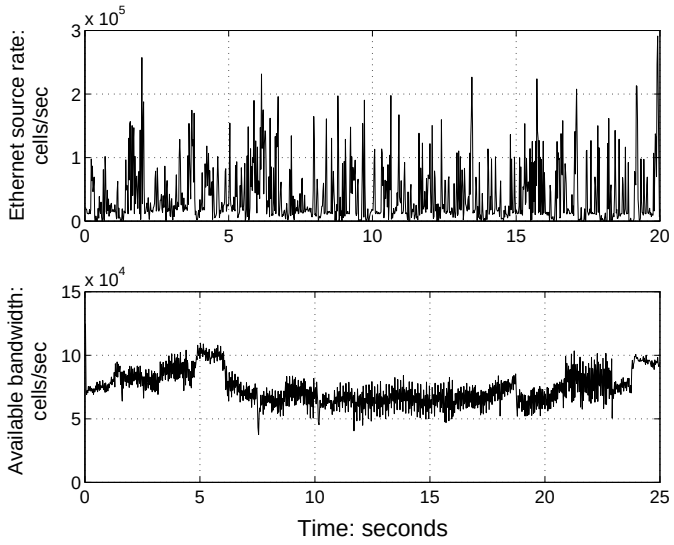

Figure 15: Ethernet trace source rate and available ABR bandwidth

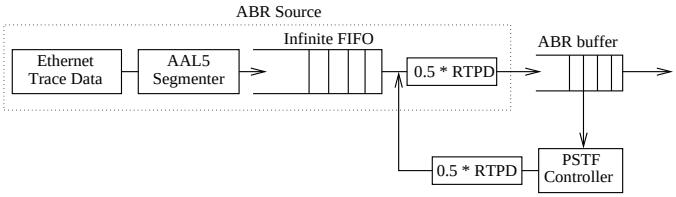

Figure 16: Ethernet ABR Source

is not only bounded, but it is typically slightly larger than available bandwidth, behavior which is nearly ideal.

The importance of this behavior is highlighted dramatically in the cell loss performance for the three algorithms. The maximum ABR queue fill for PSTF is 2500 cells, so there are no cell losses. In contrast, the feedback rate from RRM stays at link capacity most of the time, resulting in extensive cell losses when the Ethernet source suddenly becomes greedy (about 35352 cells in all). The ERICA algorithm exhibits very similar behavior to the RRM algorithm, although slightly better: there are 23343 cell losses under ERICA control. The PSTF algorithm also exhibits a slight edge in delay performance. The mean Ethernet cell transmission delay is 35 ms, 39.3 ms and 39.5 ms for PSTF, RRM and ERICA respectively.

4 Conclusions and Future Work

We have shown that fuzzy-logic principles can be effectively applied to the control of traffic sources in a network, a conventional gradient technique can be used to tune the fuzzy controller's parameters, and Minimum Variance prediction can augment the self-tuning fuzzy logic to handle large round trip propagation delays existing in WANs. The resulting predictive self-tuning fuzzy-logic (PSTF) controller outperforms ATM Forum suggested RRM and ERICA algorithms. The PSTF controller has been shown to be ef-

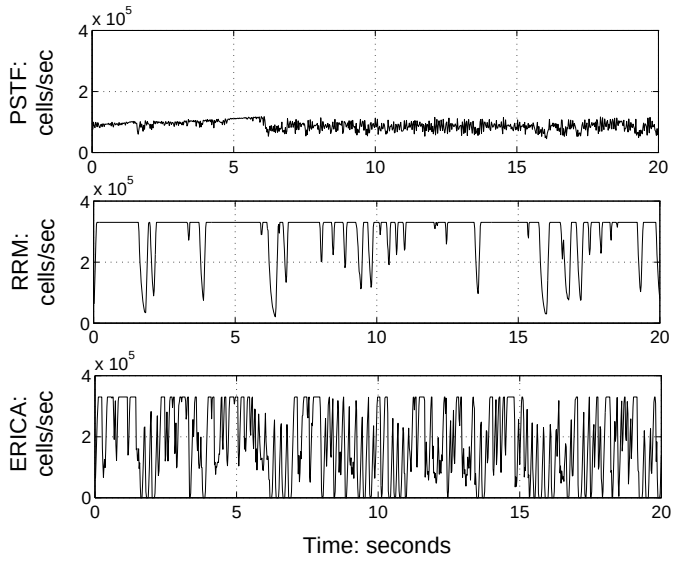

Figure 17: Feedback rate to Ethernet source for three different controllers

fective at maintaining high server utilization simultaneous with high service quality (low delay and cell loss) for the controllable (ABR) traffic under a variety of conditions.

There are several potentially fruitful areas for future work. The focus of one direction would be on generalizing the control from the single-node environment of this paper to a network environment with multiple points of potential congestion. This would involve the coordination of multiple controllers along the lines suggested in section 1.1. Performance and robustness comparisons between our algorithm and other existing ones in a multiple node topology could be made.

Implementation complexity is of great concern to switch vendors. They would like to see an efficient ABR congestion controller with as little knowledge as possible of the network status and traffic attributes. We have shown in this paper that the proposed PSTF algorithm only needs the knowledge of current and past measurements of ABR queue fill, lost ABR cells, available ABR bandwidth, and the minimum $RTPD$ among all active ABR VCs. All the measurements are either already defined as standard inputs to the controllers by ATM Forum or are easily monitored by the switch, with the exception of the minimum $RTPD$. The estimation of the minimum $RTPD$ will be of great importance in the real-time implementation and must be investigated along with other implementation details.

It would also be desirable to combine our proposed minimum variance prediction with some of the ATM Forum suggested ABR flow control algorithms. We would like to see if the MV predictor helps those ATM Forum suggested methods to improve the network performance with long round trip propagation delays.

Acknowledgements

The authors gratefully acknowledge the financial and technical support provided by Sprint Corporation for this research. Earlier results of this paper were presented at the IEEE ICC'96 Conference, Dallas, TX, June 1996.

References

[1] ATM Forum Technical Committee and Traffic Management Working Group, " ATM Forum Traffic Management Specification Version 4.0", April 1996.

[2] L. Benmohamed and S. M. Meerkov, "Feedback control of congestion in packet switching networks: the case of a single congestion node," *IEEE/ACM Transactions on Networking*, vol. 1, no. 6, December 1993.

[3] Jan Beran, Robert Sherman and Walter Willinger, "Long range dependence in variable-bit-rate video traffic," *IEEE Transaction on Communications*, vol. 43, no. 2/3/4, 1995.

[4] Flavio Bonomi, Dedasis Mitra and Judith B. Seery, "Adaptive algorithms for feedback-based flow control in high-speed , wide-area ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1267-1283, September 1995.

[5] C. J. Chang and R. G. Cheng, "Traffic control in an ATM network using fuzzy set theory," *IEEE INFOCOM'94*, vol. 3, pp. 1200-1207, 1994.

[6] Thomas M. Chen, Steve S. liu and Vijay K. Samalam, "The Available Bit Rate service for data in ATM networks," *IEEE Communications Magazine,* vol. 34, no. 5, May 1996.

[7] V. Frost and B. Melamed, "Traffic modeling for telecommunications", *IEEE Communication Magazine*, vol. 32, no. 3, March 1994.

[8] K. Hirota, "Industrial applications of fuzzy technology," *Springer - Verlag*, 1993.

[9] Qingyang Hu, David W. Petr, "Self-tuning fuzzy traffic rate control for ATM networks", *IEEE ICC'96*, vol. 1, pp. 424-428, 1996.

[10] Qingyang Hu, "Development and analysis of ABR congestion control techniques for wide area ATM networks," Ph.D. dissertation, University of Kansas, 1998.

[11] Ilias Iliadis, "A new feedback congestion control policy for long propagation delays ," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1284-1295, September 1995.

[12] Rauf Izmailov, "Adaptive feedback control algorithms for large data transfers in high speed networks," *IEEE Transactions on Automatic Control*, vol. 4, no. 8, pp. 1469-1471, August 1995.

[13] Raj Jain, "Congestion control and traffic management in ATM networks: recent advances and a survey," *Computer Networks and ISDN Systems*, vol. 28, no. 13, pp. 1723-1738, October 1996.

[14] M. Katevenis, S. Sidiropoulos and C. Courcoubetis, "Weighted round-robin cell multiplexing in a general-

purpose ATM switch chip," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 8, pp. 1265-1279, October 1991.

[15] A. Kolarov and G, Ramamurthy, "Comparison of congestion control schemes for ABR service in ATM local area networks," *IEEE GLOBECOM'94*, vol. 2, pp. 913-918, 1994.

[16] Aleksandar Kolarov and G. Ramamurthy, "End-to-end adaptive rate based congestion control scheme for ABR service in wide area ATM networks," *IEEE ICC'95*, vol. 1, pp. 138-143, 1995.

[17] Will E. Leland, Daniel V. Wilson, "High time-resolution measurement and analysis of LAN traffic: implication for LAN interconnection," *IEEE INFOCOM'91*, vol.3, p. 11d.3.1, 1991.

[18] W. E. Leland, M. S. Taqqu, W. Willinger and D. V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, February 1994.

[19] T. D. Ndousse, "Fuzzy neural control of voice cells in ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 12, no. 9, pp. 1488-1494, December 1994.

[20] H. T. Nguyen, M. Sugeno, R. Tong and R. R. Yager, "Theoretical Aspects of Fuzzy Control," *John Wiley & Sons, Inc.*, 1995.

[21] C. M. Ozveren, R. Simcoe and G. Varghese, "Reliable and efficient hop-by-hop flow control" *IEEE Journal on Selected Areas in Communications,* vol. 13, no. 4, pp. 642-650, May 1995.

[22] Rajesh S. Pazhyannur and Rjeev Agrawal, "Feedback-based flow control of B-ISDN/ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1252-1266, September 1995.

[23] Andreas Pitsillides and Jim Lambert, "Adaptive connection admission and flow control: quality of service with high utilization," *IEEE INFOCOM'94*, vol. 3, pp. 1083-1091, 1994.

[24] Michael Ritter, "Network buffer requirements of the rate-based control mechanism for ABR services," *IEEE INFOCOM'96*, pp. 1190-1197, March 1996.

[25] K. Sam Shanmugan, *BONeS Designer - Introductory Overview*, COMDISCO Systems, November 1993.

[26] M. Sugeno, "An Introductory Survey of Fuzzy Control," *Information Sciences*, vol. 36, no. 1-2, pp. 59-83, July-August 1985.

[27] A. A. Tarraf, I. W. Habib, and T. N. Saadawi, "Congestion control mechanism for ATM networks using neural network," *IEEE ICC'95*, pp. 206-210, 1995.

[28] P. Tsingotjidis, J. Hayes, H. S. Kim, "Estimation and prediction approach to congestion control in ATM networks," *IEEE GLOBECOM'94,* vol. 3, pp. 1785-1789, 1994.

[29] W. Verbiest and L. Pinnoo, "A variable bit rate video codec for Asynchronous Transfer Mode networks," *IEEE Journal on Selected Areas in Communications*, pp. 761-770, June 1989.

[30] Y. Wang, T. Lin and K. Gan, "An improved scheduling algorithm for weighted round-robin cell multiplexing in an ATM switch," *Proc. IEEE ICC'94*, pp. 1032-1037, 1994.

[31] P. E. Wellstead and M. B. Zarrop "Self-tuning systems: control and signal processing," *John Wiley & Sons, Inc.*, 1991.

[32] L. Zheng, "A practical computer-aided tuning technique for fuzzy control," *Proceedings of IEEE International Conference on Fuzzy Systems*, pp. 702-707, 1993.

Rose Qingyang Hu (S'95-M'98) received her B.S. degree in Automatic Control from the University of Science and Technology of China in 1992, M.S. degree in Mechanical Engineering from Polytechnic University in Brooklyn, NY in 1995, and Ph.D. degree in Electrical Engineering from the University of Kansas in 1998. Her dissertation addressed the development and analysis of traffic management and congestion control algorithms in wide area ATM networks. Since February of 1998, she has been a Senior Member of Scientific Staff in the Wireless Systems Engineering Group of Nortel Networks. Her recent research interests include performance, provisioning and capacity analysis in 3G wireless networks, as well as traffic management and routing in broadband multimedia satellite networks. Her email address is rosehu@nortelnetworks.com.

David W. Petr (S'75-M'78-S'86-M'89-SM'90) earned a B.S. degree from SMU in 1976, a M.S. degree from Stanford University in 1978, and a Ph.D. degree from the University of Kansas (KU) in 1990, all in electrical engineering. He is currently an Associate Professor in the Electrical Engineering and Computer Science department and the Information and Telecommunication Technology Center at KU. His research interests span various aspects of performance and control in integrated telecommunication networks. Prof. Petr serves as an associate editor for IEEE Communications Letters. He has been named a Sharp Professor of Engineering at KU for his dedication to and innovation in engineering education. His email address is dwp@ittc.ukans.edu.