Dynamic Bandwidth Allocation for Efficient Transport of Real-Time VBR Video over ATM

Song Chong, San-qi Li and Joydeep Ghosh
Department of Electrical and Computer Engineering
The University of Texas at Austin, TX 78712

Abstract

This paper[1] presents a novel approach to dynamic transmission bandwidth allocation for transport of real-time variable-bit-rate video in ATM networks. Describe video traffic in the frequency domain: the low frequency signal captures the slow time-variation of consecutive scene changes; the high frequency signal exhibits the feature of strong frame autocorrelation. Our study indicates that the video transmission bandwidth in a finite-buffer system is essentially characterized by the low frequency signal. Since the time scale of scene changes is usually in the range of a second or longer, the low frequency video signal is defined in a well-founded low frequency band. Hence, it is feasible to implement dynamic allocation of video transmission bandwidth using on-line observation and prediction of scene changes. Two prediction schemes are examined: recursive least square method vs. time delay neural network method. A time delay neural network with low-complexity high-order architecture, called "Pi-Sigma Network", is successfully used to predict scene changes. The proposed dynamic bandwidth allocation scheme is shown to be promising and practically feasible in obtaining efficient transmission of real-time video traffic with guaranteed quality of services.

1 Introduction

ATM technology offers a great flexibility of transmission bandwidth allocation to accommodate diverse demands of individual connections. One major application in ATM networks is to provide real-time loss-free transmission of variable-bit-rate (VBR) video. A key issue in video transmission design is to find an effective transmission bandwidth for guaranteed quality of services. For simplicity, we consider a single-link finite-buffer system, shown in Fig. 1. By stochastic modeling, video traffic is represented by a stationary random process. The notion of *effective bandwidth*, measured in cells per unit time, is equivalent to the minimum transmission bandwidth allocated to the input traffic subject to quality of service requirements. Limited analytical solutions are available on transmission bandwidth evaluation, usually with simplified input traffic models and under the asymptotic assumption of large buffer size and small loss rate [1, 2]. Finding *effective bandwidth* for video is especially difficult for the following two reasons. First, a real VBR video signal exhibits highly bursty and nonstationary properties which greatly complicates the queueing analysis (if feasible). Second, the *effective bandwidth* must be designed to handle the worst-case input scenario in order to avoid buffer blocking and excessive delay. This extreme case is difficult to predict due to its infrequent occurrence and its dependence on individual sources.

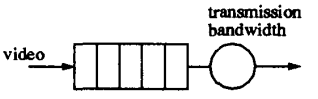

Figure 1: A single-link finite-buffer system

In practice, the transmission bandwidth requirement needs to be assessed using on-line traffic measurement. The study in [3] indicates that the most important input statistics to measure for queueing analysis is the power spectrum. Two basic concepts were discovered in [4] by describing the input traffic in the frequency domain. First, the effective bandwidth in a zero-loss finite-buffer system is essentially determined by the input traffic characteristics in a certain low frequency band. Second, the low frequency traffic flow basically stays intact as it travels through the finite-buffer system.

The major difference of our approach from most existing techniques is that we introduce the concept of dynamic bandwidth allocation. Instead of allocating a static effective bandwidth, we propose to adaptively change the transmission bandwidth using on-line measurement of video demand. In experimental study, we choose a data sequence coded from the movie "Star Wars" by using a JPEG-like compression technique [5]. Such a full-motion entertainment movie possibly represents a class of the most difficult video services to support in ATM networks. Our statistical analysis shows that the video traffic is well separated into two frequency regions: the low frequency signal captures the slow time-variation of consecutive scene changes; the high frequency signal exhibits the feature of frame autocorrelation. It is the low frequency signal that essentially determines the on-line demand of video transmission bandwidth. In other words, once the transmission bandwidth is adaptively changed with the low frequency signal, the whole video signal can be transmitted via the finite-buffer system with no information loss and negligible queueing delay.

There is a trade-off between transmission efficiency and processing efficiency in the design of dynamic bandwidth allocation. On the one hand, a higher transmission efficiency can always be achieved by more frequent adaptation of the

[1] The research reported here was supported by NSF under grant NCR-9015757 and Texas Advanced Research Program under grant TARP-129.

bandwidth to its low frequency input. On the other hand, since the decision of dynamic bandwidth allocation is often made at the network layer based on global traffic measurement, the frequency of the bandwidth adaptation is limited by the network protocol processing time. Our study indicates that the low frequency video signal typically stays in a well-founded region $\omega < 2 \times 2\pi$ radians. The corresponding time scale of scene changes will be in the range of longer than several hundred milliseconds. Hence our analysis shows that the video transmission bandwidth only needs to be adapted at the interval of several hundred milliseconds, which is feasible in practical network design. Note that although here we choose to tune the transmission bandwidth for a given input traffic, similar effect can be achieved by tuning the input traffic for a given transmission bandwidth (e.g., using dynamic routing schemes to redirect the individual traffic streams). Recently, for the transmission of an MPEG video source, a technique of frame-by-frame dynamic bandwidth allocation has been studied [6]. While its on-line bandwidth estimation scheme is simple, its application to control of network-wide video traffic flow is limited by the frequent adaptation of bandwidth.

A key question then is how to effectively predict abrupt scene changes in the incoming video traffic using on-line low frequency traffic measurement. A better prediction scheme allows a relatively longer lead time to predict any abrupt scene changes, which otherwise will cause buffer congestion if the bandwidth is not properly adapted. Two prediction schemes are proposed and compared in this paper. One is based on the recursive least square (RLS) method [8]. The other scheme takes the artificial neural network (ANN) approach. In particular, we choose a low-complexity high-order architecture, called "Pi-Sigma network (PSN)", for the construction of a time delay neural network (TDNN) [10].

Each scheme has its own strength and weakness in prediction design. In contrast to the RLS method, the ANN approach requires a training stage with off-line computation time and its solution may not be generally applied. However, the ANN approach has the advantage of much less on-line computation time and no initial transient state for convergence. Also, the PSN-TDNN scheme is a high-degree polynomial prediction method whereas the RLS scheme is a linear prediction method. In our application we find that the PSN-TDNN scheme, trained in one video segment, can generally be applied to other video segments which are collected from different scenes. Furthermore, the prediction lead time of the PSN-TDNN scheme is found to be longer than that of the RLS scheme to achieve virtually identical queueing performance.

Several examples of using the ANN approach for on-line prediction can be found in the literature, including financial forecasting in the stock market, electric load forecasting in power networks, traffic prediction in transportation networks and fault prediction in process control [11]-[13]. The ANN approach was also proposed for call admission control and link capacity allocation in ATM communication networks [14, 15]. This paper is the first attempt to use the ANN approach for bandwidth prediction in multimedia traffic communication environment. Our experimental study shows the efficiency and robustness of the PSN-TDNN scheme to predict abrupt scene changes in VBR video.

One can implement the dynamic bandwidth allocation in two different operations. In synchronous operation, the bandwidth is adapted periodically at a fixed adaptation interval based on the prediction of video demand. In asynchronous operation, the bandwidth will be adapted if and only if the demand exceeds a pre-assigned level. The asynchronous operation can significantly reduce the adaptation frequency at the cost of increasing transmission bandwidth.

The paper is organized as follows. In Section 2 we introduce the concept of dynamic bandwidth allocation and demonstrate its superior performance over the static allocation. Both RLS and PSN-TDNN prediction schemes are described and their complexities are examined in Section 3. Also in Section 3, we study the prediction performance of the two schemes in video application with emphasis on the PSN-TDNN scheme. The queueing performance is obtained in Section 4 as the two prediction schemes are used for the dynamic bandwidth allocation. The paper is concluded in Section 5.

2 Dynamic Bandwidth Allocation

No analytical models that are available today can adequately represent VBR video traffic. Here we choose a full-motion movie "Star Wars" released from Bellcore [5] as a testbed of our study. It is coded by 8×8 discrete cosine transform (DCT) and Huffman coding without motion compensation. The average bit rate is 5.3 Mbps. The original data are recorded in bytes per slice (1.4 milliseconds) for approximately two hours. There are 16 lines per slice, 30 slices per frame and 24 frames per second. The entire data sequence is divided into 60 consecutive pages for approximate 2 minutes per page. In ATM network application, bytes are converted into cells with each cell consisting of 44 bytes of video signal plus 9 bytes of protocol overhead.

Our interest here is in the dynamic behavior of the video cell sequence. Fig. 2a shows a typical 2-minute video traffic (page 56 of "Star Wars") measured in cells per slice, denoted by $x(t)$ at time t. The maximum number of cells in slice within this segment is 52.9 and the average is 24.3 cells per slice. Also shown in Fig. 2b is the corresponding power spectrum. Two key observations are made about the video power spectrum. First, the spectral spikes which appear at $24 \times 2\pi$ radians and its harmonics represent the frame correlations. Second, the rest of the video power, located in a very low frequency band typically less than $2 \times 2\pi$ radians, captures the strong correlation of scene changes.

The recent study in [4] indicates that the *effective bandwidth* in a zero-loss finite-buffer system is essentially determined by the input traffic characteristics in a certain low frequency band. Especially for video, as one will see, the effective bandwidth must be designed to cope with the worst scenario of consecutive scene changes, which is difficult to predict when the connection is initially set up. Note that VBR video traffic possesses highly bursty and nonstationary properties. An effective approach is to dynamically allocate transmission bandwidth using the on-line observation of video scene changes. Applying the above 2-minute signal $x(t)$ to a low-pass filter at the cutoff frequency $\omega_c = 2\pi$ radians, Fig. 3a shows the filtered signal $x_L(t)$ which characterizes the scene changes. The average and peak input rates of $x_L(t)$ are equal to 24.3 and 42.0 cells per slice, respectively. While a variety of low-pass filters are available, here we choose a class of finite impulse response filters with a Kaiser window. The time unit in the digital filtering process is one slice. As one will see, it is this $x_L(t)$ that essentially captures the on-line demand of video transmission bandwidth. For simplicity, we first con-

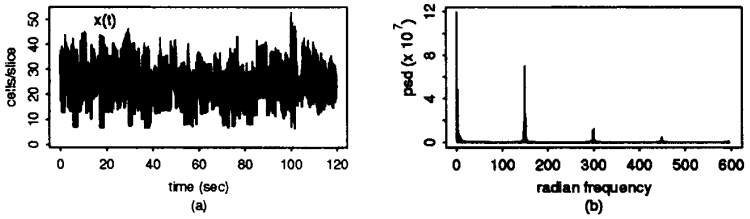

Figure 2: (a) A typical 2-minute video traffic where the mean rate and peak rate are respectively 24.3 and 52.9 cells per slice (page 56) (b) the corresponding power spectrum

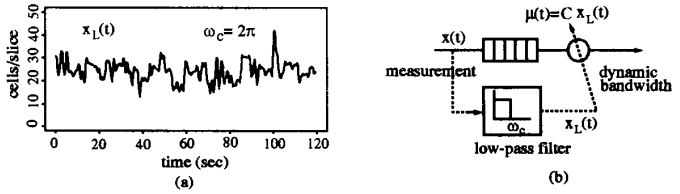

Figure 3: (a) low frequency video signal filtered at ω_c=2π radians where the mean rate and peak rate are respectively 24.3 and 42.0 cells per slice (page 56) (b) ideal dynamic bandwidth allocation by low frequency signal

sider an ideal situation where the transmission bandwidth is instantaneously changed with $x_L(t)$, as described in Fig. 3b. In other words, the transmission bandwidth at time t, denoted by $\mu(t)$, is defined as a function of $x_L(t)$. We further assume

$$\mu(t) = Cx_L(t) \quad (1)$$

where C is a control parameter. From the facts $E[\mu(t)] = CE[x_L(t)]$ and $E[x_L(t)] = E[x(t)]$, one can get $C = \rho^{-1}$ where ρ denotes the average link utilization. Since the bandwidth is instantaneously adapted by the filtered signal, the impact of video scene changes on the queueing process itself has now been removed. Note that we need $C > 1$ for $E[\mu(t)] > E[x(t)]$, i.e., some extra transmission bandwidth is required for the finite buffer to effectively smooth out the rest of the high frequency signal.

Let us now examine the effect of low frequency video signal on queueing performance. Consider a queueing system with infinite buffer capacity to transmit the above 2-minute video using the dynamic bandwidth allocation scheme at different cutoff frequencies. The control parameter C is fixed at 1.25, which is equivalent to the average link utilization $\rho = 0.8$. The simulation results are summarized in Table 1, where $\overline{q}$, σ_q and q_{max} represent the mean, standard deviation, and maximum of queue length in cell unit, respectively.

Note that while the video signal was recorded in the format of number of cells per slice, the queueing process must be evolved in the time unit of cell transmission slot. In converting the time unit from slice to slot, we consider two extreme scenarios. One is for the "worst scenario" where all the cells which are generated in each slice are assumed to arrive at the beginning of the first slot in that slice. The other is for the "best scenario" where the cell arrivals in each slice are assumed to be evenly distributed on all the slots of that slice. The queueing solutions in the two extremes provide us the upper and lower bounds of the exact queueing solution. When we choose $\omega_c = 2\pi\tau^{-1}$ where τ denotes one slice interval, the bandwidth will be adaptively changed with $Cx_L(t)$ in every slice interval. As a result, the queue will always be empty in the best scenario. In the worst scenario, we get $(\overline{q}, \sigma_q, q_{max}) = (9.0, 8.4, 52)$ due to the batch cell arrival at the first slot of each slice interval. When ω_c is reduced to $12 \times 2\pi$, the queueing solutions increase to $(\overline{q}, \sigma_q, q_{max}) = (10.2, 10.0, 129)$ in the worst scenario, which is contributed by the high frequency video signal (i.e., the frame correlation). In the extreme, one can take ω_c=0 to completely eliminate the dynamic bandwidth allocation. Then, due to the strong impact of low frequency video signal, the queue is drastically increased as measured by $(\overline{q}, \sigma_q, q_{max}) = (257.5, 1208.5, 9610)$ in the worst scenario. It is obvious that the low frequency video signal, which captures the slow time variation of consecutive scene changes, dominates the queueing performance without implementing the bandwidth adaptation.

ω_c (radians)	$\overline{q}$ (cells)	σ_q (cells)	q_{max} (cells)
$\tau^{-1} \times 2\pi$	9.0 (0)	8.4 (0)	52 (0)
$12 \times 2\pi$	10.2 (1.0)	10.0 (4.2)	129 (100)
$1 \times 2\pi$	**11.1(2.6)**	**13.1(9.1)**	**224(195)**
0	257.5 (247.7)	1208.5 (1207.5)	9610 (9580)

Table 1: Effect of cutoff frequency on queueing performance in the worst (best) scenario by dynamic bandwidth allocation at $C = 1.25$ (page 56)

In practice, the transmission bandwidth cannot be too frequently adapted but is limited by the network protocol processing time. In our case we choose $\omega_c = 2\pi$ at which the queueing performance is given by $(\overline{q}, \sigma_q, q_{max}) = (11.1, 13.1, 224)$. For the average input rate of the 2-minute video, equal to 24.3 cells per slice, the maximum queue length of 224 cells at $C = 1.25$ (or $\rho = 0.8$) is equivalent to the max-

	Static Alloc. peak rate (at $\omega_c = 2\pi\tau^{-1}$)	Static Alloc. filtered peak rate (at $\omega_c = 2\pi$)	Dynamic Alloc. (at $\omega_c = 2\pi$)
ρ	0.46	0.58	0.80
$\overline{q}$	5.2 (0)	6.9 (0.1)	11.1 (1.9)
σ_q	8.8 (0)	9.5 (1.8)	13.1 (9.1)
q_{max}	52 (0)	138 (97)	224 (195)

Table 2: Queueing performance in static and dynamic bandwidth allocations in the worst (best) scenario (page 56)

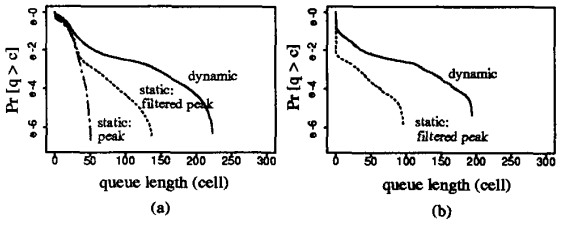

Figure 4: Queue distributions in static and dynamic bandwidth allocations (page 56); (a) worst scenario (b) best scenario

imum delay of 10 milliseconds, which is expected to be tolerable in video services. Notice that the time varying scale of the video signal in the low frequency band $\omega \leq 2\pi$ is at least one second long, which makes feasible the implementation of dynamic bandwidth allocation in network design.

Let us examine the performance improvement of dynamic bandwidth allocation over static allocation. The static allocation assigns a fixed transmission bandwidth to each connection during the entire service period. In order to prevent cell loss, the static bandwidth has to be designed to cope with the worst input scenario. The most conservative static allocation is to reserve bandwidth by the peak input rate, which would lead to the excessive use of transmission capacity with zero buffer size as in the circuit switched design. In our case, the video signal was originally collected at slice interval. If the bandwidth is statically assigned by the peak input rate of the above 2-minute video segment measured at slice interval, the link utilization will be as low as 0.46. The corresponding queueing solutions in the worst scenario are given by $(\overline{q}, \sigma_q, q_{max}) = (5.2, 8.8, 52)$ (see Table 2).

The study in [4] indicates that the effective transmission bandwidth in a zero-loss finite-buffer system is essentially determined by the peak of properly filtered input rate. Consider the filtered video input rate $x_L(t)$ at $\omega_c = 2\pi$. By the static allocation, the bandwidth will be assigned by the peak of $x_L(t)$, which leads to the link utilization $\rho = 0.58$ and queueing performance $(\overline{q}, \sigma_q, q_{max}) = (6.9, 9.5, 138)$. In contrast, when the bandwidth is dynamically adapted with $x_L(t)$ the link utilization can be as high as $\rho = 0.80$ with a moderate increase of buffer capacity. As the results are compared in Table 2, the dynamic bandwidth allocation, designed at $\omega_c = 2\pi$ for the low frequency input, can effectively improve the video transmission efficiency. Also compared in Fig. 4 are the queue distribution functions of the corresponding three bandwidth allocation schemes. Note that the improvement can be more significant by the dynamic allocation if a longer video segment is chosen. This is because the static allocation scheme needs to assign the bandwidth equal to the filtered peak input rate of the chosen video segment. In practice, it is always difficult to identify the filtered peak input rate of each individual video source at the call admission stage, which makes the static allocation scheme hardly implementable. By comparison, the dynamic bandwidth allocation is designed on the basis of on-line traffic measurement. Not only does it effectively improve the transmission efficiency, but also its implementation is highly feasible in practice.

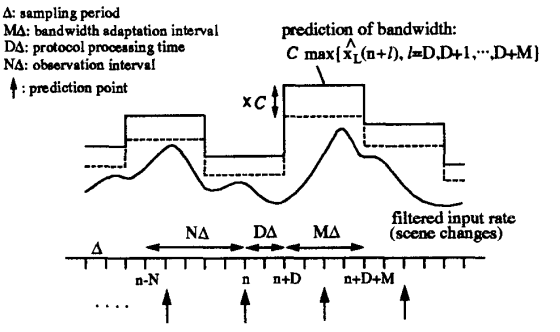

Figure 5: Bandwidth prediction in VBR video service

3 Bandwidth Prediction

In the above section we have assumed that the transmission bandwidth $\mu(t)$ is instantaneously adapted by the filtered input rate $x_L(t)$. In reality, the bandwidth can only be adapted intermittently using on-line observation and prediction of $x_L(t)$; the adaptation interval cannot be too short since it is limited by the protocol processing time required to allocate the desired bandwidth. Fig. 5 describes a realistic dynamic allocation scheme. Let the filtered input signal $x_L(t)$ be sampled at time unit Δ. Denote the sampled signal by $x_L(n)$ at the n-th Δ unit. The bandwidth is periodically adapted at the interval of $M\Delta$. There will be $D\Delta$ lead time for computation of the prediction algorithm and protocol processing of the bandwidth allocation. That is, if the prediction starts at the n-th unit, the bandwidth will be adapted at the $(n+D)$-th unit. Consequently, the next bandwidth adaptation will occur at the $(n+D+M)$-th unit. The prediction of the input rate at each consecutive unit of the adaptation interval, denoted by $\hat{x}_L(n+D+l)$ at $l = 0, 1, ..., M$, is made on the basis of $(N+1)$ consecutive observations collected at the n-th unit. In other words, the predictions $\{\hat{x}_L(n+D+0), \hat{x}_L(n+D+1), ..., \hat{x}_L(n+D+M)\}$ are made on the basis of $\{x_L(n-N), x_L(n-N+1), ..., x_L(n)\}$. In order to cope with the worst input rate in the adaptation interval, the transmission bandwidth is assigned by

$$C \max\{\hat{x}_L(n+D+0), \hat{x}_L(n+D+1), ..., \hat{x}_L(n+D+M)\} \quad (2)$$

The selection of Δ is dependent on the time variation of $x_L(t)$. In our application, $x_L(t)$ is defined at $\omega_c = 2\pi$ to represent the video scene changes as indicated in Figs. 2b,3a. The corresponding time varying scale of $x_L(t)$ is at least one second long. Here we take the over-sampling of $x_L(t)$ at $\Delta = 0.14$ seconds in order to capture any abrupt scene changes. This is because the underestimation of an abrupt scene change

can easily cause buffer congestion. Two prediction schemes are introduced in the following with comparison of prediction performance and computational complexity.

3.1 RLS-based Prediction

We first use the RLS algorithm to design an adaptive filter for traffic prediction [8]. The rate of convergence of the RLS algorithm is typically an order of magnitude faster than that of the least mean square (LMS) algorithm at the expense of increased computation. In contrast with the LMS algorithm, the rate of convergence of the RLS algorithm is insensitive to variations in the eigenvalue spread, defined as the ratio of the maximum to minimum eigenvalues of the correlation matrix of the input vector. The RLS algorithm also has to some extent the capability to track statistical variations in a nonstationary environment by setting the exponential forgetting factor less than unity [8]. As a result, here we choose the RLS algorithm for the on-line bandwidth prediction of video traffic.

Since each bandwidth adaptation requires computation of the predictions $\{\hat{x}_L(n+D+0), \hat{x}_L(n+D+1), ..., \hat{x}_L(n+D+M)\}$, the so called *indirect prediction* approach is used to avoid redundant computation [7]. That is, instead of directly constructing $(M+1)$ RLS prediction filters, we construct a single RLS filter to perform parameter estimation of a given autoregressive (AR) model of the time series. The $(M+1)$ predictions are then obtained by converting the model into the required predictor format. This is further described in the following.

Assume that the sampled time series of the filtered video is modeled by a deterministic AR process of order N, defined by

$$x_L(n) = \theta_1^\star x_L(n-1) + \theta_2^\star x_L(n-2) \cdots + \theta_N^\star x_L(n-N) \quad (3)$$

What we need to solve is the estimation of the unknown parameter vector $\vec{\theta}^\star = [\theta_1^\star, \theta_2^\star, \cdots, \theta_N^\star]^T$. At the n-th time unit, let us define the input vector $\vec{u}(n) = [x_L(n-1), x_L(n-2), \cdots, x_L(n-N)]^T$, the desired output $d(n) = x_L(n)$, and the estimated parameter vector $\hat{\vec{\theta}}(n) = [\hat{\theta}_1(n), \hat{\theta}_2(n), \cdots, \hat{\theta}_N(n)]^T$. The on-line RLS estimation of $\vec{\theta}^\star$ is then recursively expressed by

$$\vec{k}(n) = \frac{\lambda^{-1}\mathbf{P}(n-1)\vec{u}(n)}{1+\lambda^{-1}\vec{u}^T(n)\mathbf{P}(n-1)\vec{u}(n)} \quad (4)$$

$$\hat{\vec{\theta}}(n) = \hat{\vec{\theta}}(n-1) + \vec{k}(n)(d(n) - \hat{\vec{\theta}}^T(n-1)\vec{u}(n)) \quad (5)$$

$$\mathbf{P}(n) = \lambda^{-1}\mathbf{P}(n-1) - \lambda^{-1}\vec{k}(n)\vec{u}^T(n)\mathbf{P}(n-1) \quad (6)$$

where λ is the forgetting factor, $\mathbf{P}(n)$ denotes the inverse of the input correlation matrix, and $\vec{k}(n)$ represents the gain vector. From the given AR model in (3), the l-step ahead prediction at $l = D, D+1, \cdots, D+M$ is recursively obtained by

$$\hat{x}_L(n+l) = \hat{\vec{\theta}}^T(n)\vec{r}(n,l) \quad (7)$$

where $\vec{r}(n,l) = [\hat{x}_L(n+l-1), \hat{x}_L(n+l-2), \cdots, \hat{x}_L(n+1), x_L(n), x_L(n-1), \cdots, x_L(n+l-N)]^T$. Let $b(n)$ represent the transmission bandwidth required during the adaptation interval $[n+D, n+D+M)$. Its prediction, denoted by $\hat{b}(n)$, is then given by

$$\hat{b}(n) = C\max\{\hat{x}_L(n+l), \quad l = D, D+1, \cdots, D+M\} \quad (8)$$

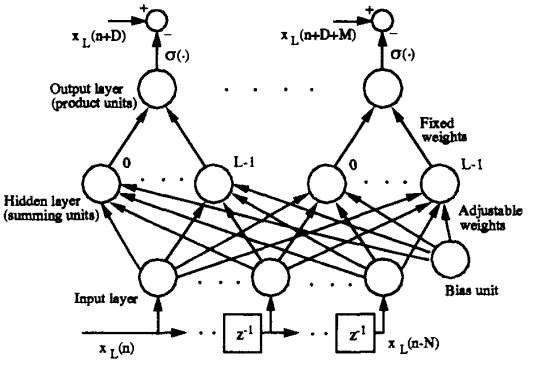

Figure 6: An L^{th}-degree PSN-TDNN

Using the *round-to-largest* rule, the real value of $\hat{b}(n)$ is quantized into an integer in cell unit. Such an adaptive bandwidth assignment is called *synchronous dynamic allocation* since it is adapted periodically at a fixed interval $M\Delta$. For low-complexity network management of bandwidth adaptation, one can also introduce an *asynchronous dynamic allocation* scheme, defined by

$$\hat{b}(n) = C\max\{\phi, \hat{x}_L(n+l), \quad l = D, D+1, \cdots, D+M\} \quad (9)$$

where ϕ denotes a pre-assigned *nominal bandwidth*. Since the bandwidth is adapted if and only if the video prediction exceeds the nominal bandwidth, the asynchronous operation can significantly reduce the bandwidth adaptation frequency at the expense of increased transmission bandwidth. As an example, we set $\phi = E[x(t)] + (Var[x(t)])^{\frac{1}{2}}$ in this paper.

3.2 TDNN-based Prediction

As an alternative approach we introduce a TDNN-based prediction scheme [16, 17]. ANNs have adaptation capability that can accommodate nonstationarity. ANNs have generalization capability which makes them flexible and robust when faced with new and/or noisy data patterns. Once the training is completed, an ANN can be computationally inexpensive even if it continues to adapt on-line. Recently a computationally efficient high-order neural network has been developed [10] that approximates the input-output relationship by a high-degree polynomial while avoiding an exponentially increasing computational and memory cost that affects ordinary high-order nets [19]. This architecture, called the Pi-Sigma network (PSN), is selected as the basis of our TDNN prediction scheme.

Fig. 6 shows a TDNN based on L^{th}-degree PSN with $N+1$ inputs and $M+1$ outputs. In conventional TDNNs, the architecture above tapped delay line in Fig. 6 is given by Multilayered Perceptron Networks (MLP) [16]-[18] which suffer slow training and relatively expensive on-line computation. The PSN architecture consists of a single hidden layer of $L \times (M+1)$ linear summing units (L summing units per output) and an output layer of $M+1$ product units. These product units make it possible to incorporate the capabilities of high-order networks while greatly reducing network complexity. The term "pi-sigma" comes from the fact that these networks use products of sums of input components, instead of sums of products in ordinary

high-order networks. The output from each product unit passes through the sigmoid activation function defined by $\sigma(x) = \frac{1}{1+e^{-x}}$. Unlike in MLP, the weights from the hidden layer to the outputs are fixed at 1. This property contributes to reducing training time substantially. The bias input is also fixed at 1. The purpose of this network is to find an approximate L^{th}-degree relationship between the inputs $x_L(n), x_L(n-1), \cdots, x_L(n-N)$ and the desired outputs $x_L(n+D), x_L(n+D+1), \cdots, x_L(n+D+M)$. For convenience, we define input vector, desired output vector and estimated output vector by $\vec{\xi}(n) = [\xi_0(n), \xi_1(n), \cdots, \xi_{N+1}(n)]^T$, $\vec{d}(n) = [d_0(n), d_1(n), \cdots, d_M(n)]^T$ and $\vec{y}(n) = [y_0(n), y_1(n), \cdots, y_M(n)]^T$ respectively. Correspondingly,

$$\begin{aligned}\vec{\xi}(n) &= [1, x_L(n), x_L(n-1), \cdots, x_L(n-N)]^T \\ \vec{d}(n) &= [x_L(n+D), x_L(n+D+1), \cdots, x_L(n+D+M)]^T \\ \vec{y}(n) &= [\hat{x}_L(n+D), \hat{x}_L(n+D+1), \cdots, \hat{x}_L(n+D+M)]^T\end{aligned} \tag{10}$$

Note that $\vec{\xi}(n)$ is augmented by a bias input and $\vec{y}(n)$ is from the array of sigmoid functions. By training, the relationship will be stored through the network in the form of strength of connectivity, namely, weights. Let $\vec{w}_j^l = [w_{0_j}^l, w_{1_j}^l, \cdots, w_{N+1_j}^l]^T$ be the weight vector for the j^{th} hidden unit of the l^{th} output where $l = 0, 1, \cdots, M$ and $j = 0, 1, \cdots, L-1$. The l^{th} output $y_l(n)$ at time n is then expressed by

$$y_l(n) = \sigma[\prod_{j=0}^{L-1} \vec{w}_j^{l\ T} \vec{\xi}(n)] = \sigma[\prod_{j=0}^{L-1} (\sum_{k=1}^{N+1} w_{k_j}^l \xi_k(n) + w_{0_j}^l)]. \tag{11}$$

It is noted that this regression model of the PSN-TDNN is more general than AR model in that it realizes an L^{th}-degree polynomial mapping from input to output. Furthermore, unlike in ordinary high-order nets, the high-degree approximation in a factorized form as in (11) greatly reduces the computational complexity.

The learning algorithm for the PSN is based on gradient descent on the estimated mean square error (MSE) surface in weight space. The MSE objective of l^{th} output is given by

$$J_l = \frac{1}{p} \sum_{n=0}^{p-1} [d_l(n) - y_l(n)]^2 \tag{12}$$

where p denotes number of training patterns. By taking LMS-type approach, the weight update rule is given by for $l = 0, 1, \cdots, M$,

$$\delta \vec{w}_s^l(n) = \eta [d_l(n) - y_l(n)][y_l(n)]' [\prod_{j \neq s} \vec{w}_j^{l\ T}(n) \vec{\xi}(n)] \vec{\xi}(n) \tag{13}$$

where $[y_l(n)]'$ is the first derivative of the sigmoid function and η is the learning rate. Here we have adopted an *asynchronous update rule*, which updates only a partial set of weights at a time instead of the overall weights [10]. There are total L sets of weights, defined by $\{\vec{w}_j^l(n), l = 0, 1, \cdots, M\}$ at $j = 0, 1, \cdots, L-1$, each of which is associated with a hidden unit. Only one set, $\{\vec{w}_s^l(n), l = 0, 1, \cdots, M\}$, is chosen at time n and updated by (13); the rest of the sets remain unchanged. This procedure is repeated in an asynchronous manner.

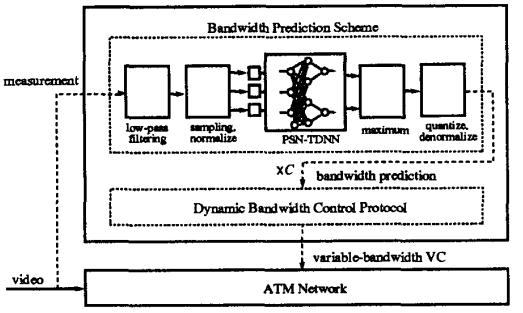

Figure 7: Structure of dynamic bandwidth allocation system using on-line bandwidth prediction

Similarly as in the RLS-based system, the transmission bandwidth required during the next adaptation interval $[n+D, n+D+M)$ is predicted by

$$\hat{b}(n) = C \max\{y_l(n), \quad l = 0, 1, \cdots, M\} \tag{14}$$

The asynchronous allocation scheme is given by

$$\hat{b}(n) = C \max\{\phi, y_l(n), \quad l = 0, 1, \cdots, M\} \tag{15}$$

Using the *round-to-largest* rule, the real value of $\hat{b}(n)$ is quantized into an integer in cell unit. The whole structure of dynamic bandwidth allocation system using the PSN-TDNN-based prediction scheme is shown in Fig. 7. In practice, such system is implemented at the network access point. The function of *dynamic bandwidth control protocol* is to guarantee the predicted bandwidth either locally at each node by adaptive link capacity assignment among individual virtual circuit (VC) connections or globally at the network layer by rerouting of active VC connections.

3.3 Computational Complexity

One disadvantage of the RLS prediction scheme is its computational complexity. The parameter estimation in (4)(5)(6) requires $2N^2 + 7N + 5$ multiplications and $N^2 + 4N + 3$ divisions per iteration. The bandwidth prediction in (7) needs $N \times (D+M)$ multiplications. Since the parameter estimation is computed at every Δ interval while the bandwidth prediction is computed at every $M\Delta$, the total complexity per $M\Delta$ is $M \times (2N^2 + 7N + 5) + N \times (D + M)$ multiplications and $M \times (N^2 + 4N + 3)$ divisions. The computational complexity can also be reduced by using the so called fast transversal filters (FTF) algorithm [9]. This algorithm attains the RLS solution with the same convergence properties as in the RLS algorithm but at a computational cost that is competitive with the LMS algorithm. As a result, the complexity in parameter estimation is reduced to $7N+12$ multiplications plus 4 divisions per iteration; the total complexity per $M\Delta$ then becomes $M \times (7N + 12) + N \times (D + M)$ multiplications and $4M$ divisions. In comparison, once it is trained, the PSN-TDNN scheme only requires $(M + 1) \times (N + 2) \times L$ multiplications per $M\Delta$. Although special computation for the sigmoid function is required, in practice the sigmoid function is usually replaced by a linear saturator or a lookup table.

3.4 Prediction Performance

In the experimental study, we choose $\omega_c = 2\pi$ for the low frequency signal $x_L(t)$ which represents the video scene

changes according to the power spectral distribution in Fig. 2b. The RLS scheme is designed by $(N, D, M) = (5, 1, 4)$ and the PSN-TDNN scheme is by $(N, D, M, L) = (5, 2, 4, 2)$, both of which are found to provide adequate prediction of abrupt scene changes in queueing performance. The prediction lead time is defined by $(D + M)\Delta$, which at $\Delta = 0.14$ seconds is equal to 0.7 seconds for the RLS scheme and 0.84 seconds for the PSN-TDNN scheme. The longer the lead time, the better the prediction scheme to achieve identical performance. Since the time varying scale of the low frequency video signal is at least one second long, we design the bandwidth adaptation interval at $M\Delta$=0.56 seconds close to the Nyquist sampling interval.

In the initial stage of our experimental study, we also considered the use of MLP-based TDNN for the prediction, but no significant performance advantage was observed over the PSN-based TDNN. In contrast with PSN-TDNN, MLP-TDNN suffers higher computational complexity and subsequent longer training period. For the training of the PSN-TDNN, we used a 2-minute video segment (page 56) in Fig. 3a, filtered at $\omega_c = 2\pi$. By scanning the filtered video segment along time, we collected 208 training examples. After adding a 2Δ-long offset, we again scanned the segment thereby updating the 208 examples. By repeating this procedure three times, we obtained a total of 624 examples. Due to the dynamic range of the sigmoid function, the input data needed to be normalized into $[0, 1]$. In the design of the PSN-TDNN, having the observation interval $N\Delta > 5\Delta$ or the degree $L > 2$ was found to simply add computational complexity with no significant performance improvement. It can be interpreted in that, in statistical estimation, increasing complexity of the model over some optimal point may degrade performance due to bias-variance dilemma. The learning rate was tuned at η=0.15. The training was carried out on a SPARC-10 workstation for 24 minutes (CPU time) through 5,000 epochs. In the design of the RLS scheme, the observation interval was also tuned at N=5 and the forgetting factor was set at λ=0.9.

The performance is measured by the prediction error statistics. In our application, the transmission bandwidth is adapted at every $M\Delta$ interval. Denote the maximum of the predictions in the adaptation interval $t \in [(n+D)\Delta, (n+D+M)\Delta)$ by $\hat{x}_{\max}(t)$, i.e.,

$$\hat{x}_{\max}(t) = \max\{\hat{x}_L(n+l), l = D, D+1, ..., D+M\} \quad (16)$$

The relative prediction error at time t is then defined by

$$\varepsilon(t) = [\hat{x}_{\max}(t) - x_L(t)]/x_L(t)$$

Positive and negative values of $\varepsilon(t)$ correspond to the overestimation and underestimation of the video scene changes. Note that it is the underestimation that may cause the buffer congestion while the overestimation can only result in the under-utilization of the transmission bandwidth. This is why in (2) we have taken the maximum of the predictions in each adaptation interval for the bandwidth allocation.

Table 3 shows the prediction error performance of the PSN-TDNN on the training video segment (page 56). The performance on the same segment achieved by the RLS scheme is also shown. While the lead time of the PSN-TDNN scheme is 0.14 seconds longer than that of the RLS scheme, the performance is basically identical. In the RLS prediction error statistics, initial transient performance for convergence was excluded. For comparison, also listed in Table 3 is the

	PSN-TDNN	RLS	
	$(D+M)\Delta$=0.84	$(D+M)\Delta$=0.7	$(D+M)\Delta$=0.84
$\bar{\varepsilon}$	0.044	0.051	0.050
σ_ε	0.071	0.092	0.114
ε_{max}	0.541	0.658	0.740
ε_{min}	-0.145	-0.144	-0.256

Table 3: Prediction error performance on training set (page 56) at $M\Delta$=0.56 seconds where $\bar{\varepsilon}$, σ_ε, ε_{max} and ε_{min} are respectively the mean, standard deviation, maximum and minimum of $\varepsilon(t)$

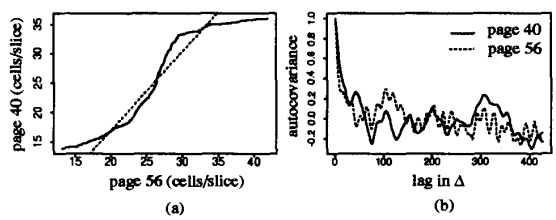

Figure 8: (a) Q-Q plot of page 40 versus page 56 (training set), at $\omega_c = 2\pi$ (b) autocovariance functions

performance of the RLS scheme at the lead time equal to 0.84 seconds. Obviously, this design of RLS scheme degrades the performance. It is observed that the RLS scheme has larger prediction error variance than the PSN-TDNN scheme.

Let us now examine the general applicability of the PSN-TDNN scheme, trained by one video segment, to other video segments. Six 2-minute filtered-video segments (pages 39-41, 55-57) were chosen as testing sets. Pages 39-41 were arbitrarily selected to represent statistically different scenes. For example, Fig. 8a shows the quantile-quantile (Q-Q) plot of page 40 versus page 56 (the training set), at $\omega_c = 2\pi$. Since the Q-Q plot is largely deviated from the linear reference line, the probability distribution of page 40 is substantially different from that of page 56. Also displayed in Fig. 8b are the autocovariance functions of the two segments, which are quite different. Hence, the scene statistics of page 40 must be substantially different from that of the training set. It is interesting to find that the PSN-TDNN scheme, trained on page 56, works very well on the other pages. Fig. 9a shows a part of the prediction curve to track the abrupt scene changes on page 40. Similar performance is observed on the other pages. Listed in Table 4 are the error statistics of the PSN-TDNN scheme on page 40 vs. page 56. The error statistics of the RLS scheme on page 40 are also included for comparison.

Furthermore, in Table 5 we compare the prediction performance of the two schemes for the overall 12 minutes. The generality of the PSN-TDNN scheme is also tested on the prediction of a 2-minute multiplexed video segment. Here we take the summation of five 2-minute video segments (pages 55-59) to represent the statistical multiplexing of five video sources. Similar performance is achieved as plotted in Fig. 9b. In summary, we argue that the PSN-TDNN scheme, properly trained on a 2-minute video segment, can be directly used to a certain extent on other video segments (which consist of statistically quite different scenes).

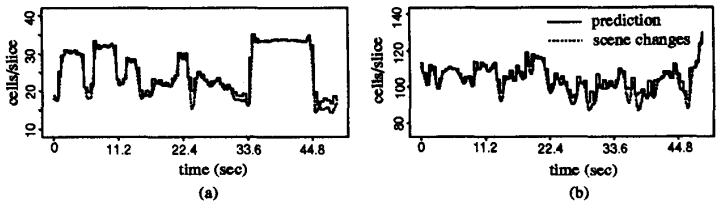

Figure 9: Testing of PSN-TDNN (a) on video segment with statistically different scenes(page 40) (b) on video multiplexed by pages 55-59

	on page 56	on page 40	
	PSN-TDNN $(D+M)\Delta$=0.84	PSN-TDNN $(D+M)\Delta$=0.84	RLS $(D+M)\Delta$=0.7
$\bar{\epsilon}$	0.044	0.045	0.048
σ_ϵ	0.071	0.086	0.128
ϵ_{max}	0.541	0.597	1.875
ϵ_{min}	-0.145	-0.152	-0.181

Table 4: Prediction error performance on a testing set at $M\Delta$=0.56 seconds

	PSN-TDNN $(D+M)\Delta$=0.84	RLS $(D+M)\Delta$=0.7
$\bar{\epsilon}$	0.044	0.045
σ_ϵ	0.072	0.098
ϵ_{max}	0.597	1.875
ϵ_{min}	-0.171	-0.188

Table 5: 12-minute prediction error performance (on pages 39-41, 55-57) at $M\Delta$=0.56 seconds

According to the analysis in Section 3.3, the complexity of the RLS scheme is 385 multiplications and 192 divisions while the complexity of the PSN-TDNN scheme is only 70 multiplications per 4Δ. When the FTF algorithm is applied [9], the complexity of the RLS scheme is reduced to 213 multiplications and 16 divisions. In contrast, the on-line computation of the PSN-TDNN scheme is less than one-eighth of the RLS scheme.

The study in this section indicates that, in our application of video bandwidth prediction, the PSN-TDNN scheme is superior to the RLS scheme in terms of both prediction performance and computational complexity. In the next section we evaluate the video queueing performance when the two schemes are applied to the dynamic adaptation of transmission bandwidth.

4 Performance Evaluation of Dynamic Bandwidth Allocation

First, let us design a zero-loss transmission system to deliver a 12-minute video segment (pages 39-41, 55-57). Assume that the buffer size should never exceed 300 cells. As described in Section 2, the transmission bandwidth is essentially captured by the video scene changes which are located in a well-founded low frequency band. For simplicity, we choose a fixed $\omega_c = 2\pi$ for the filtered video signal $x_L(t)$ to capture the scene changes. The transmission bandwidth is then allocated through the observation and/or prediction of $x_L(t)$, either dynamically or statically. In the static allocation, the bandwidth is assigned by $\max_t x_L(t)$, which is the maximum of $x_L(t)$ in the entire 12-minute period. In practice, however, $\max_t x_L(t)$ is unknown. In the ideal situation of dynamic allocation, the bandwidth is directly assigned by $Cx_L(t)$ as in (1), which is also unrealistic since the bandwidth cannot be instantaneously adapted by the filtered input rate. For the practical implementation of synchronous dynamic allocation, the bandwidth is periodically adapted by $C\hat{x}_{\max}(t)$, where $\hat{x}_{\max}(t)$ represents the maximum prediction of $x_L(t)$ in the next adaptation interval defined in (16). For the asynchronous dynamic allocation, the bandwidth is determined by $C\max\{\phi, \hat{x}_{\max}(t)\}$ where ϕ stands for a pre-assigned nominal video bandwidth. We assume ϕ=$E[x(t)] + (Var[x(t)])^{\frac{1}{2}}$. Both RLS and PSN-TDNN schemes, designed in Section 3 at the adaptation interval $M\Delta = 0.56$ seconds, are used to evaluate $\hat{x}_{\max}(t)$ in the synchronous/asynchronous dynamic allocation. We choose $C = 1.25$ for the dynamic allocation policies.

Listed in Table 6 are the transmission efficiency and queueing solutions with respect to each allocation policy. Obviously, the transmission efficiency ρ reaches its highest value at 0.80 by the ideal dynamic allocation while its lowest value occurs at 0.54 by the static allocation. We also get $\rho = 0.74$ for the synchronous dynamic allocation and $\rho = 0.62$ for the asynchronous dynamic allocation. Note that one can get $\rho = C^{-1}$ for the ideal dynamic allocation. The rest of the values of the transmission efficiency are measured by simulation. Although both RLS and PSN-TDNN prediction schemes have achieved the same transmission efficiency ($\rho = 0.74$ or 0.62), the queueing performance of the PSN-TDNN scheme is always better than that of the RLS scheme as shown in Table 6. This is consistent to the prediction performance comparison in Section 3. Displayed in Fig. 10 is the queue distribution with respect to each allocation policy. In Fig. 11 we show a sample path of the synchronous/asynchronous dynamic allocation using the PSN-TDNN scheme. As one can see, the asynchronous operation significantly reduces the frequency of bandwidth adaptation, which is desirable for low-complexity network management, but at the expense of increased transmission bandwidth. This study indicates the significant performance improvement of dynamic allocation and the feasibility of its implementation at a reasonably long adaptation interval such as 0.56 seconds for video transmission.

Next, we consider a single transmission trunk of fixed bandwidth μ to support five VC connections as shown in Fig. 12. Each connection is associated with a separate buffer.

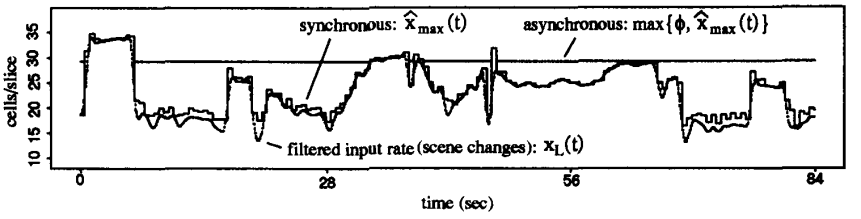

Figure 11: A sample path of synchronous/asynchronous dynamic allocation (on page 41) using the PSN-TDNN scheme

	Static	Dynamic				
	filtered peak'	ideal	RLS (syn.)	PSN (syn.)	RLS (asyn.)	PSN (asyn.)
ρ	0.54	0.80	0.74	0.74	0.62	0.62
$\overline{q}$	6.4	10.7	9.1	8.7	7.0	7.0
σ_q	8.7	13.3	11.7	10.2	8.9	8.8
q_{max}	138	283	269	231	202	186

Table 6: Performance comparison of different bandwidth allocation policies for a finite-buffer zero-loss system to transmit 12-minute video in the worst scenario

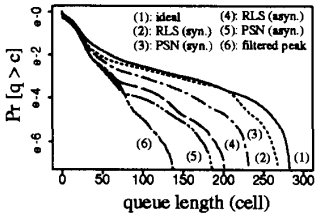

Figure 10: Queue distribution of 12-minute video transmission by different allocation policies in the worst scenario

The ATM traffic on each connection, denoted by $x_i(t)$ at $i = 1, 2, ...5$, is represented by a 2-minute video source. We use pages 39-41, 55 and 56 of the movie "Star Wars" to individually represent each source. Let the transmission bandwidth of each VC be dynamically adapted based on the filtered $x_i(t)$'s. As in the previous example, we choose $\omega_c = 2\pi$ for the input filter and denote the filtered $x_i(t)$ by $x_{iL}(t)$. For the ideal dynamic sharing, the bandwidth of each VC is instantaneously changed by

$$\mu_i(t) = \frac{x_{iL}(t)\mu}{\sum_{j=1}^{5} x_{jL}(t)} \qquad \text{for } i = 1, 2, ..., 5$$

For the synchronous dynamic sharing, similar to the definition of $\hat{x}_{\max}(t)$ in (16), we use $\hat{x}_{\max i}(t)$ to represent the maximum prediction of $x_{iL}(t)$ in each adaptation interval. The total bandwidth is then adaptively divided among the five VCs in every $M\Delta$ interval, according to

$$\mu_i(t) = \frac{\hat{x}_{\max i}(t)\mu}{\sum_{j=1}^{5} \hat{x}_{\max j}(t)}, \quad t \in [(n+D)\Delta, (n+D+M)\Delta) \quad (17)$$

Here we use the same RLS and PSN-TDNN schemes as used

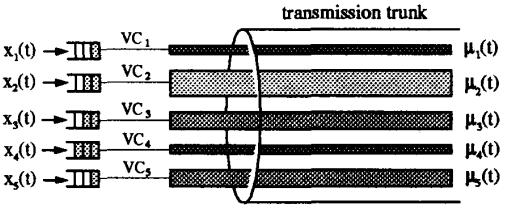

Figure 12: Transmission of five video sources by dynamic sharing on a single ATM trunk

in the previous example for the prediction, where the adaptation interval is fixed at 0.56 seconds.

Assume that the overall trunk utilization is $\rho = 0.7$. Since the mean of the aggregate video traffic is 114.1 cells per slice, we have the total trunk bandwidth equal to $\mu = 163.0$ cells per slice. One slice corresponds to 1.4 milliseconds. Listed in Table 7 are the queueing solutions of each VC connection using different dynamic sharing policies. As one can see, every dynamic sharing policy provides a fair service performance among the individual connections. It is also interesting to observe that the performance of the synchronous dynamic sharing is almost identical to that of the ideal one.

For comparison purposes, we also consider some static sharing policies where the total bandwidth is statically divided by

$$\mu_i(t) = \frac{e_i\mu}{\sum_{j=1}^{5} e_j}, \quad \forall\, t \qquad (18)$$

e_i denotes the *static bandwidth measure* of the i^{th} connection. For instance, e_i can be the peak of the input traffic ($\max_t x_i(t)$), the peak of the filtered input ($\max_t x_{iL}(t)$), or the average input ($E[x_i(t)]$). The assumption of infinite buffer size is made for each connection. As shown in Table 7, the queueing solutions of the static sharing are highly unbalanced among the individual connections. In contrast, the queueing performance of the static sharing is much worse than that of the dynamic one. In summary, the temporal bandwidth demand of each video connection is essentially characterized by the scene changes, which are highly predictable through the on-line traffic measurement. One can therefore implement the dynamic sharing to significantly reduce the transmission bandwidth and buffer capacity requirement. For the slow time-varying scale of scene changes, we also expect that the same technique can be applied for the control of network-wide video traffic flow. One feasible approach is to periodically reroute active VC connections to

	Static Alloc.			Dynamic Alloc.		
	peak	filtered peak	mean	ideal	RLS (syn.)	PSN (syn.)
$\overline{q}_1$	11.5	137.0	9.8	8.3	3.5	8.2
σ_{q_1}	17.7	371.8	13.4	9.7	14.0	9.6
q_{max1}	292	2179	214	171	445	177
$\overline{q}_2$	30.5	203.9	604.1	10.5	11.1	11.1
σ_{q_2}	122.0	672.7	1998.3	13.8	15.3	15.4
q_{max2}	1753	4129	11737	183	178	159
$\overline{q}_3$	247.0	138.3	247.0	8.5	8.1	8.5
σ_{q_3}	1206.4	682.6	1206.4	9.3	8.7	9.6
q_{max3}	9427	5656	9427	123	76	140
$\overline{q}_4$	10.2	19.9	10.2	9.0	9.3	9.2
σ_{q_4}	21.8	66.0	21.8	9.5	9.8	9.7
q_{max4}	672	1490	672	131	113	135
$\overline{q}_5$	106.9	16.3	78.6	9.2	9.0	10.0
σ_{q_5}	670.6	89.7	507.1	10.2	9.4	15.5
q_{max5}	6170	1282	5000	177	138	269

Table 7: Performance comparison of different sharing policies of transmission bandwidth

guarantee the predicted bandwidth.

5 Conclusion

This paper has presented a novel approach to dynamic bandwidth allocation for transport of real-time VBR video over ATM networks. Describe video traffic in the frequency domain: the low frequency signal captures the slow time-variation of consecutive scene changes; the high frequency signal exhibits the feature of strong frame correlations. Our study indicates that the video transmission bandwidth in a finite-buffer system is essentially characterized by the low frequency signal. Since the time scale of scene changes is usually in the range of a second or longer, the video low frequency signal is defined in a well-founded low frequency band. Hence, it is possible to implement dynamic allocation of video transmission capacity using on-line observation and prediction of scene changes. Two prediction schemes have been examined: recursive least square method vs. time delay neural network method. A time delay neural network with low-complexity high-order architecture, called Pi-Sigma Network, has been successfully used to predict scene changes. The proposed dynamic bandwidth allocation scheme is shown to be promising and practically feasible in obtaining efficient transmission of real-time video traffic with guaranteed quality of services.

References

[1] G. Guerin, H. Ahmadi and M. Naghshineh, "Equivalent Capacity and its Application to Bandwidth Allocation in High-Speed Networks," *IEEE J. Select. Areas in Communications*, Vol.9, No.7, pp.968-981, Sept. 1991.

[2] A. Elwalid and D. Mitra, "Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High Speed Networks", Proc. of *IEEE Infocom '93*, pp.256-265, Mar. 1993.

[3] S.Q. Li and C. Hwang, "Queue Response to Input Correlation Functions: Continuous Spectral Analysis," *IEEE/ACM Trans. on Networking*, Vol.1, No.6, pp.678-692, Dec. 1993.

[4] S.Q. Li, S. Chong, C. Hwang and X. Zhao, "Link Capacity Allocation and Network Control by Filtered Input Rate in High Speed Networks," Proc. of *IEEE Globecom '93*, pp.744-750, Dec. 1993.

[5] M. Garrett and M. Vetterli, "Congestion Control Strategies for Packet Video," presented in the *Fourth International Workshop on Packet Video*, Kyoto, Japan, Aug. 1991.

[6] P. Pancha and M. El Zarki, "Bandwidth Requirements of Variable Bit Rate MPEG Sources in ATM Networks," Proc. of *IEEE Infocom '93*, pp.902-909, Mar. 1993.

[7] G.C. Goodwin and K.S. Sin, "*Adaptive Filtering, Prediction and Control*," Prentice-Hall, Englewood Cliffs, N.J., 1984.

[8] S. Haykin, "*Adaptive Filter Theory*," Prentice-Hall, Englewood Cliffs, N.J., 1991.

[9] J.M. Cioffi and T. Kailath, "Fast, Recursive-least-squares Transversal Filters for Adaptive Filtering," *IEEE Trans. on Acoust., Speech, and Signal Processing*, Vol.ASSP-32, pp.304-337, 1984.

[10] J. Ghosh and Y. Shin, "Efficient High-order Neural Networks for Classification and Function Approximation," *Intl. Journal of Neural Systems*, Vol.3, No.4, pp.323-350, 1992.

[11] "Neural Networks at Work", *IEEE Spectrum*, pp.26-32, June 1993.

[12] D. Park et al., "Electric Load Forecasting Using An Artificial Neural Network", *IEEE Trans. on Power Systems*, Vol.6, No.2, pp.442-449, May 1991.

[13] H. Yang, T. Akiyama and T. Sasaki, "A Neural Network Approach to the Identification of Real-time Origin-destination Flows from Traffic Counts", Proc. of *International Conference on Artificial Intelligence Applications in Transportation Engineering*, pp. 253-269, June 1992.

[14] A. Hiramatsu, "ATM Communications Network Control by Neural Networks," *IEEE Trans. on Neural Networks*, Vol.1, No.1, pp.122-130, Mar. 1990.

[15] A. Hiramatsu, "Integration of ATM Call Admission Control and Link Capacity Control by Distributed Neural Networks," *IEEE J. Select. Areas in Communications*, Vol.9, No.7, pp.1131-1138, Sept. 1991.

[16] J. Hertz, A. Krogh and R. Palmer, *"Introduction to the Theory of Neural Computation"*, Addison-Wesley, CA, 1991.

[17] D. Hush and B. Horone, "Progress in Supervised Neural Networks: What's New since Lippmann?", *IEEE Signal Processing Magazine*, pp.8-38, Jan. 1993.

[18] K. Hornik, M. Stinchcombe and H. White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, Vol.2, No.23, pp.359-366, 1989.

[19] C. Giles and T. Maxwell, "Learning, Invariance, and Generalization in a High-order Neural Network," *Applied Optics*, Vol.26, No.23, pp. 4972-4978, 1987.